Newtonian Program Analysis ⋆

Javier Esparza, Stefan Kiefer, and Michael Luttenberger

Institut für Informatik, Technische Universität München, 85748 Garching, Germany
{esparza,kiefer,luttenbe}@model.in.tum.de

Abstract. This paper presents a novel generic technique for solving dataflow equations in interprocedural dataflow-analysis. The technique is obtained by generalizing Newton's method for computing a zero of a differentiable function to ω-continuous semirings. Complete semilattices, the common program analysis framework, are a special class of ω-continuous semirings. We show that our generalized method always converges to the solution, and requires at most as many iterations as current methods based on Kleene's fixed-point theorem. We also show that, contrary to Kleene's method, Newton's method always terminates for arbitrary idempotent and commutative semirings. Furthermore, the number of iterations required to solve a system of n equations is at most n.

1 Introduction

This paper presents a novel generic technique for solving dataflow equations in interprocedural dataflow-analysis. It is obtained by generalizing Newton's method, the 300-year-old technique for computing a zero of a differentiable function.

Our approach to interprocedural analysis is very similar to Sharir and Pnueli's functional approach [SP81,JM82,KS92,RHS95,SRH96,NNH99,RSJM05]. Sharir and Pnueli assume the following as given: a (join-) semilattice[1] of *values*, a mapping assigning to every program instruction a value, and a concatenation operator that, given the values of two sequences of instructions, returns the value corresponding to their concatenation. Sharir and Pnueli assume that the concatenation operator distributes over the lattice's join.[2] Sharir and Pnueli define a system of *abstract data flow equations*, containing one variable for each program point. They show that for every procedure P of the program and for every program point p of P, the least solution of the system is the join of the values of all valid program paths starting at the initial node of P and leading to p. Sharir and Pnueli's result was later extended by [KS92] to programs with local variables and to non-distributive concatenation operators, which allows to deal with certain non-distributive analyses [NNH99].

We slightly generalize Sharir and Pnueli's setting. Loosely speaking, we allow to replace the join operator with any operator satisfying the same algebraic properties but possibly idempotence. In algebraic terms, we extend the framework from the class of lattices considered in [SP81] to an ω-*continuous semiring* [Kui97], an algebraic structure with two operations, usually called sum and product. The interest of this otherwise simple extension is that our framework now encompasses equations over the semiring of the nonnegative reals with addition and multiplication. This allows us to compare the efficiency of generic solution methods for dataflow analysis when applied to the reals, with the efficiency of methods applied by numerical mathematics, in particular Newton's method.

It is well-known that Newton's method, when it converges to a solution, usually converges much faster than classical fixed-point iteration (see e.g. [OR70]). Furthermore, Etessami and Yannakakis have recently proved that Newton's method is guaranteed to converge for an analysis concerning the probability of termination of recursive programs [EY05]. These facts raise the question whether Newton's method can be

⋆ This work was partially supported by the DFG project *Algorithms for Software Model Checking*.

[1] For reasons that will be clear later, we use join-semilattices rather than meet-semilattices, deviating from the classical dataflow analysis literature such as [Kil73,KU77,SP81]. As a consequence, we also replace greatest fixed points by least fixed points, meet-over-all-paths by join-over-all-paths, etc. This change is purely notational.

[2] Actually, in [SP81] the value of a program instruction is the function describing its effect on program variables, and the extension operator is function composition. However, the extension to an arbitrary distributive concatenation operator is unproblematic.

generalized to the more abstract dataflow setting, where values are arbitrary entities, while preserving these good properties.

In the first part of the paper we show that the generalization is indeed possible. Inspired by work of Hopkins and Kozen on Kleene algebras [HK99], we show that the notion of a differential of a function lying at the heart of Newton's method, and the method itself, can be suitably generalized. This allows to apply Newton's method to, for instance, language equations.

In the second part of the paper we study the properties of Newton's method on idempotent semirings, the classical domain of program analysis. Recall that the method is iterative: it constructs better and better approximations to the solution of the equation system. We obtain a characterization of the approximants, and use it to derive two results, both showing that well-known concepts and procedures of the theory of context-free languages and Kleene algebras are particular instances of application of Newton's method. In the first instance we are interested in analysing the complete traces (terminating executions) of a program. In the simplest case of control-flow analysis, where data is abstracted away, the complete traces are a context-free language. We show that Newton's method corresponds to approximating a context-free language by context-free languages of *finite index*, a classical notion of language theory [Ynt67,GS68,Sal69,Gru71]. In the second instance we examine commutative idempotent semirings, previously studied by Hopkins and Kozen in a beautiful paper [HK99]. Hopkins and Kozen propose a generic solution method for the equations, and prove that it terminates after $\mathcal{O}(3^n)$ iterations, where n is the number of equations. We show that their method is in fact Newton's method. Applying our characterization of the approximants, we further prove that it terminates after at most n iterations.

Finally, in a short section we extend our framework to the non-distributive case. We show that Newton's method, like the classical fixed-point iteration, computes an overapproximation of the join of the values of all valid program paths.

In the rest of this introduction we go again through the paper's skeleton sketched above, but providing some more details.

1.1 A Summary of Sharir and Pnueli's Approach

[SP81] provide, for distributive analyses, a system of equations such that, for every procedure P of the program and for every program point p of P, the p-component of the least solution of the equation system equals the join of the values of all valid program paths starting at the initial node of P and leading to p. We show in this subsection how to construct this system of equations.

Consider a program with three procedures X, Y, Z, whose flow-graphs are shown in Figure 1. Nodes correspond to program points, and edges to program instructions. For instance, procedure X can execute b and terminate, or execute a, call itself recursively, and, after the recursive call has terminated, call Y.

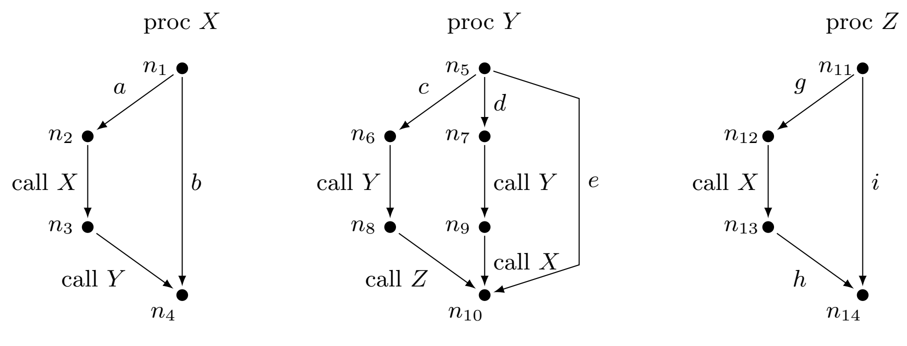


Fig. 1. Flowgraphs of three procedures

Sharir and Pnueli assume as given: a complete lattice[3] of values with a join operator $\vee$; a mapping ϕ assigning to each non-call edge (m, n) a lattice value $\phi(m, n)$, and a concatenation operator $\cdot$ that distributes over $\vee$ and has a neutral element 1. The system of equations contains a variable and an equation for each program node. If n is the initial node of a procedure then it contributes the equation $v_n = 1$, where v_n denotes n's variable. Otherwise, it contributes the equation

$$v_n = \bigvee_{m \in pred(n)} v_m \cdot h(m, n)$$

where $pred(n)$ denotes the set of immediate predecessors of n, and $h(m, n)$ is defined as follows: if (m, n) is a call edge calling, e.g., procedure X, then $h(m, n)$ is the variable for the return node of X; otherwise $h(m, n) = \phi(m, n)$.

The system of equations for Figure 1 can be more compactly represented if variables for all program points other than return points are eliminated by substitution. Only three equations remain, namely those for the return points n_4, n_{10}, and n_{14}. If moreover, and abusing language, we reuse X, Y, Z to denote the variables for these points, and $a, \ldots, i$ to denote the values $\phi(n_1, n_2), \ldots, \phi(n_{11}, n_{14})$, we obtain the system

$$\begin{aligned} X &= a \cdot X \cdot Y \ \vee \ b \\ Y &= c \cdot Y \cdot Z \ \vee \ d \cdot Y \cdot X \ \vee \ e \\ Z &= g \cdot X \cdot h \ \vee \ i \end{aligned} \tag{1}$$

which very closely resembles the structure of the flowgraphs. Since the right-hand-sides of the equations are monotonic mappings, and $\cdot$ distributes over $\vee$, the existence of the least fixed point is guaranteed by Kleene's fixed-point theorem.

1.2 A Slight Generalization: From Semilattices to Semirings

Let us examine the properties of the join operator $\vee$. First of all, since the lattice is complete, it is defined for arbitrary, finite or countably infinite, sets of lattice elements. Furthermore, it is associative, commutative, idempotent, and concatenation distributes over it. If we use the symbols 0 for the bottom element of the lattice (corresponding to an abort operation) and 1 for the element corresponding to a NOP instruction, then we have $0 \vee a = a \vee 0 = a$ and $1 \cdot a = a \cdot 1 = a$ for every a. It is argued in [SF00] that one can transform every program analysis to an essentially equivalent one that satisfies $0 \cdot a = a \cdot 0 = 0$. So the lattice, together with the two operations $\vee$ and $\cdot$ and the elements 0 and 1, constitutes an *idempotent semiring*. In the following we write '+' for '$\vee$' to conform with the standard semiring notation.

Idempotence of the join operator is not crucial for the existence of the least fixed point; it can be replaced by a weaker property. Consider the relation $\sqsubseteq$ on semiring elements defined as follows: $a \sqsubseteq a + b$ for all elements a, b. A semiring is *naturally ordered* if this relation is a partial order, and a naturally ordered semiring in which infinite sums exist and satisfy standard properties is called *ω-continuous*. Using Kleene's fixed-point theorem it is easy to show that systems of equations over ω-continuous semirings still have a least fixed point with respect to the partial order $\sqsubseteq$ (see for instance [Kui97]).

As an example of application of this more general setting, assume that the program of Figure 1 is probabilistic, and the values $a, \ldots, i$ are real numbers corresponding to the probabilities of taking the transitions. A particular case is shown in Figure 2. The semiring operations are addition and multiplication over the nonnegative reals. Notice that addition is not idempotent. The semiring is ω-continuous if a new element ∞ with the usual properties is added. It is not difficult to show [EKM04,EY05] that the least solution of the system

$$X = 0.4XY + 0.6$$

[3] More precisely, [SP81] initially considers semilattices with a least and a greatest element that satisfy the ascending-chain property (every non-decreasing chain eventually becomes stationary). However, the paper later concentrates on finite lattices, which are complete.

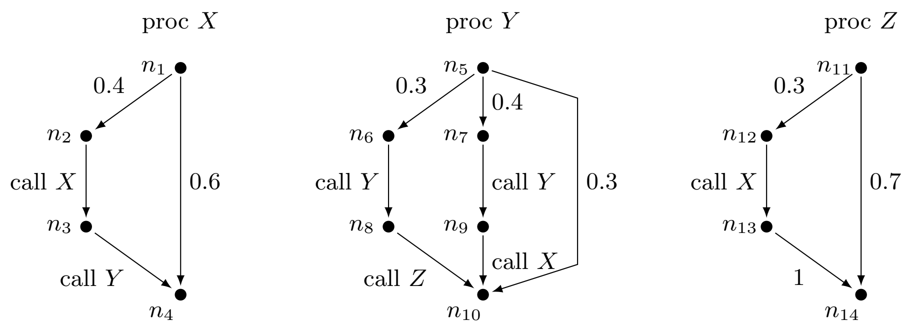


Fig. 2. Probabilistic flowgraphs

$$
\begin{aligned}
Y &= 0.3YZ + 0.4YX + 0.3 \\
Z &= 0.3X + 0.7
\end{aligned}
$$

yields the probability of termination of each procedure. (Incidentally, notice that, contrary to the intraprocedural case, this probability may be different from 1 even if every execution can be extended to a terminating execution.)

1.3 Solving Systems of Equations

Current generic algorithms for solving Sharir and Pnueli's equations (like the classical worklist algorithm of dataflow analysis) are based on variants of Kleene's fixed-point theorem [Kui97]. The theorem states that the least solution $\mu\boldsymbol{f}$ of a system of equations $\boldsymbol{X} = \boldsymbol{f}(\boldsymbol{X})$ over an ω-continuous semiring is equal to the supremum of the sequence $(\boldsymbol{\kappa}^{(i)})_{i\in\mathbb{N}}$ of *Kleene approximants* given by $\boldsymbol{\kappa}^{(0)} = \mathbf{0}$ and $\boldsymbol{\kappa}^{(i+1)} = \boldsymbol{f}(\boldsymbol{\kappa}^{(i)})$. This yields a procedure (let us call it *Kleene's method*) to compute or at least approximate $\mu\boldsymbol{f}$. If the domain satisfies the well-known *ascending chain condition* [NNH99], then the procedure terminates, because there exists an i such that $\boldsymbol{\kappa}^{(i)} = \boldsymbol{\kappa}^{(i+1)} = \mu\boldsymbol{f}$.

Kleene's method is generic and robust: it always converges when started at the vector $\mathbf{0}$ of 0-elements, for any ω-continuous semiring and for any system of equations. On the other hand, it often fails to terminate, and it can converge very slowly to the solution. We illustrate this point by means of two simple examples. Consider the equation $X = a \cdot X + b$ over the lattice of subsets of the language $\{a, b\}^*$. The least solution is the regular language a^*b, but we have $\boldsymbol{\kappa}^{(i)} = \{b, ab, \ldots, a^{i-1}b\}$, i.e., the solution is not reached in any finite number of steps. For our second example consider a very simple probabilistic procedure that can either terminate or call itself twice, both with probability $1/2$. The probability of termination of this program is given by the least solution of the equation $X = 1/2 + 1/2X^2$. It is easy to see that the least solution is equal to 1, but we have $\kappa^{(i)} \leq 1 - \frac{1}{i+1}$ for every $i \geq 0$, i.e., in order to approximate the solution within i bits of precision we have to compute about 2^i Kleene approximants. For instance, we have $\kappa^{(200)} = 0.9990$, i.e., 200 iterations produce only three digits of precision.

After our slight generalization of Sharir and Pnueli's framework, quantitative analyses like the probability of termination fall within the scope of the approach. So we can look at numerical mathematics for help with the inefficiencies of Kleene's method.

As could be expected, faster approximation techniques for equations over the reals have been known for a long time. In particular, Newton's method, suggested by Isaac Newton more than 300 years ago, is a standard efficient technique to approximate a zero of a differentiable function, and can be adapted to our problem. Since the least solution of $X = 1/2 + 1/2X^2$ is a zero of $1/2 + 1/2X^2 - X$, the method can be applied, and it yields $\nu^{(i)} = 1 - 2^{-i}$ for the i-th *Newton approximant*. So the i-th Newton approximant already has i bits of precision, instead of $\log i$ bits for the Kleene approximant.

However, Newton's method also has a number of disadvantages, at least at first sight. Newton's method on the real field is by far not as robust and well behaved as Kleene's method on semirings. The method may converge very slowly, converge only locally (only when started in a small neighborhood of the zero), or even not converge at all [OR70]. So we face the following situation. Kleene's method, a robust and general solution technique for arbitrary ω-continuous semirings, is inefficient in many cases. Newton's method is usually very efficient, but it is only defined for the real field, and it is not robust.

As part of their study of Recursive Markov Chains, [EY05] showed that a variant of Newton's method is robust for certain systems of equations over the real *semiring*: the method always converges when started at zero. In other words, moving from the real field to the real semiring (only nonnegative numbers) makes the instability problems disappear. Inspired by this work, in this paper we obtain a more general result. We show that Newton's method can be generalized to *arbitrary* ω-continuous semirings, and prove that on these structures it is as robust as Kleene's method. We then proceed to further analyse our generalized Newton's method. We provide a characterization of the Newton approximants, and apply it to idempotent semirings, the structures of classical program analysis. We first study the language semiring, where equation variables are interpreted over languages of finite words, sum is interpreted as union of languages, and product as concatenation. The least solutions of fixed-point equations are the context-free languages, and so our generalized Newton's method can be seen as a tool for approximating context-free languages. We show that the Newton approximants are the context-free languages *of finite index*, a well-known class studied since the 1960s in language theory [Ynt67,GS68,Sal69,Gru71]. We then proceed to study the case of commutative and idempotent semirings. Loosely speaking, these semirings correspond to *counting analysis*, in which one is interested on how often program points are visited, but not in which order. These semirings do not always satisfy the ascending chain condition, and Kleene's method may not terminate. We show that a very elegant iterative solution method for these semirings due to [HK99], is exactly Newton's method, and always terminates in a finite number of steps. As mentioned above, we further use our characterization of Newton approximants to show that the least fixed point is reached after at most n iterations, a tight bound, improving on the $\mathcal{O}(3^n)$ bound of [HK99].

In the final section of the paper we study the case in which $\cdot$ does not distribute over $+$, and only the inequality $a \cdot (b + c) \sqsupseteq a \cdot b + a \cdot c$ holds. It is well-known that in this case classical fixed-point iteration yields an overapproximation of the join-over-all-paths value, still useful for program analysis purposes (see e.g. [KS92,RHS95,SRH96,NNH99]). We show that the same property holds for Newton's method.

The paper is organized as follows. Section 2 introduces ω-continuous semirings, systems of fixed-point equations, and some semirings investigated in the rest of the paper. Section 3 recalls Newton's method, and generalizes it to arbitrary ω-continuous semirings. Section 4 characterizes the Newton approximants in terms of derivation trees, a generalization of the derivation trees of language theory. Section 5 considers the particular case of idempotent semirings and applies the characterization to the language semiring. Section 6 applies the characterization to idempotent and commutative semirings. Finally, Section 7 shows that Newton's method can also be applied to non-distributive program analyses.

2 ω-Continuous Semirings

Definition 2.1. *A* semiring *is a tuple $\langle S, +, \cdot, 0, 1\rangle$ where S is a set containing two distinguished elements 0 and 1, and the binary operations $+, \cdot \colon S \times S \to S$ satisfy the following conditions:*

(1) $\langle S, +, 0\rangle$ *is a commutative monoid.*
(2) $\langle S, \cdot, 1\rangle$ *is a monoid.*
(3) $0 \cdot a = a \cdot 0 = 0$ *for all* $a \in S$.
(4) $a \cdot (b + c) = a \cdot b + a \cdot c$ *and* $(a + b) \cdot c = a \cdot c + b \cdot c$ *for all* $a, b, c \in S$.

A semiring $\langle S, +, \cdot, 0, 1\rangle$ is ω-continuous if the following additional conditions hold:

(5) The relation $\sqsubseteq := \{(a, b) \in S \times S \mid \exists d \in S : a + d = b\}$ *is a partial order.*
*(6) Every ω-*chain *$(a_i)_{i\in\mathbb{N}}$ (i.e. $a_i \sqsubseteq a_{i+1}$ with $a_i \in S$) has a supremum w.r.t. $\sqsubseteq$ denoted by* $\sup_{i\in\mathbb{N}} a_i$.
(7) Given an arbitrary sequence $(b_i)_{i\in\mathbb{N}}$, define

$$\sum_{i\in\mathbb{N}} b_i := \sup\{b_0 + b_1 + \ldots + b_i \mid i \in \mathbb{N}\}$$

(the supremum exists by condition (6)). For every sequence $(a_i)_{i\in\mathbb{N}}$, for every $c \in S$, and for every partition $(I_j)_{j\in J}$ of $\mathbb{N}$:

$$c \cdot \left(\sum_{i\in\mathbb{N}} a_i\right) = \sum_{i\in\mathbb{N}}(c \cdot a_i), \quad \left(\sum_{i\in\mathbb{N}} a_i\right) \cdot c = \sum_{i\in\mathbb{N}}(a_i \cdot c), \quad \sum_{j\in J}\left(\sum_{i\in I_j} a_j\right) = \sum_{i\in\mathbb{N}} a_i \ .$$

An (ω-continuous) semiring is idempotent, *if $a + a = a$ holds for all $a \in S$. It is* commutative, *if $a \cdot b = b \cdot a$ for all $a, b \in S$. In an ω-continuous semiring we define the Kleene-star ${}^* : S \to S$ by*

$$a^* := \sum_{k\in\mathbb{N}} a^k = \sup\{1 + a + a \cdot a + \ldots + a^k | k \in \mathbb{N}\} \text{ for } a \in S.$$

We have the following important property:

Lemma 2.2. *In any ω-continuous semiring $\langle S, +, \cdot, 0, 1\rangle$ addition and multiplication are ω-continuous, i.e. for any ω-chain $(a_i)_{i\in\mathbb{N}}$ and any $c \in S$ we have*

$$c \cdot (\sup_{i\in\mathbb{N}} a_i) = \sup_{i\in\mathbb{N}}(c \cdot a_i), \quad (\sup_{i\in\mathbb{N}} a_i) \cdot c = \sup_{i\in\mathbb{N}}(a_i \cdot c), \quad c + (\sup_{i\in\mathbb{N}} a_i) = \sup_{i\in\mathbb{N}}(c + a_i).$$

Proof. By (5) and (6) in the definition above, for any ω-chain $(a_i)_{i\in\mathbb{N}}$, there exists a sequence $(d_i)_{i\in\mathbb{N}}$ such that $d_0 = a_0$ and $a_i + d_i = a_{i+1}$ (i.e. d_i is *a difference* of a_{i+1} and a_i), and so $\sup_{i\in\mathbb{N}} a_i = \sum_{i\in\mathbb{N}} d_i$. The result follows by applying (7) to this sequence.

Example 2.3. Common examples of ω-continuous semirings are the *real semiring*, i.e. non-negative real numbers extended by infinity $\langle \mathbb{R}_{\geq 0} \cup \{\infty\}, +, \cdot, 0, 1\rangle$, and the language semiring over some finite alphabet Σ, i.e. $\langle 2^{\Sigma^*}, \cup, \cdot, \emptyset, \{\varepsilon\}\rangle$ with $\cdot$ the canonical concatenation of languages, and ε the empty word. In both of these instances the natural order coincides with the canonical order on the respective carrier, i.e., $\sqsubseteq \equiv \leq$, resp. $\sqsubseteq \equiv \subseteq$ in the case of the real semiring, resp. the language semiring.

In the following we often write ab instead of $a \cdot b$.

2.1 Vectors, Polynomials and Power Series.

Let $\mathcal{S}$ be an ω-continuous semiring and let $\mathcal{X}$ be a finite set of variables. A *vector* is a mapping $\boldsymbol{v} : \mathcal{X} \to S$ which assigns every variable $X \in \mathcal{X}$ the value $\boldsymbol{v}(X)$. We usually write $\boldsymbol{v}_X$ for $\boldsymbol{v}(X)$. If there is some natural total order given on $\mathcal{X}$ like e.g. the lexicographic order in the case $\mathcal{X} = \{X, Y, Z\}$ or the total order on the indices in the case $\mathcal{X} = \{X_1, X_2, X_3\}$ we will also write a vector $\boldsymbol{v}$ as a column vector of dimension $|\mathcal{X}|$ enumerating the values starting with the lowest variable as the topmost value. The set of all vectors is denoted by V.

Given a countable set I and a vector $\boldsymbol{v}_i$ for every $i \in I$, we denote by $\sum_{i\in I} \boldsymbol{v}_i$ the vector given by $\left(\sum_{i\in I} \boldsymbol{v}_i\right)_X = \sum_{i\in I}(\boldsymbol{v}_i)_X$ for every $X \in \mathcal{X}$. Throughout the paper we use bold letters like '$\boldsymbol{v}$' or '$\boldsymbol{a}$' for vectors.

A *monomial* is a finite expression $a_1X_1a_2X_2 \cdots a_kX_ka_{k+1}$, where $k \geq 0$, $a_1, \ldots, a_{k+1} \in S$ and $X_1, \ldots, X_k \in \mathcal{X}$. Note that this more general definition of monomial is necessary as we do not require that multiplication is commutative. A *polynomial* is an expression of the form $m_1 + \ldots + m_k$ where $k \geq 0$ and $m_1, \ldots, m_k$ are monomials. A *power series* is an expression of the form $\sum_{i\in I} m_i$, where I is a countable set and m_i is a monomial for every $i \in I$.

Given a monomial $f = a_1X_1a_2X_2 \ldots a_kX_ka_{k+1}$ and a vector $\boldsymbol{v}$, we define $f(\boldsymbol{v})$, the *value of f at $\boldsymbol{v}$*, as $a_1v_{X_1}a_2v_{X_2} \cdots a_kv_{X_k}a_{k+1}$. We extend this to any power series $f = \sum_{i\in I} f_i$ by $f(\boldsymbol{v}) = \sum_{i\in I} f_i(\boldsymbol{v})$.

A *vector of power series* is a mapping $\boldsymbol{f}$ that assigns to each variable $X \in \mathcal{X}$ a power series $\boldsymbol{f}(X)$. Again we write $\boldsymbol{f}_X$ for $\boldsymbol{f}(X)$. Given a vector $\boldsymbol{v}$, we define $\boldsymbol{f}(\boldsymbol{v})$ as the vector satisfying $(\boldsymbol{f}(\boldsymbol{v}))_X = \boldsymbol{f}_X(\boldsymbol{v})$ for every $X \in \mathcal{X}$, i.e., $\boldsymbol{f}(\boldsymbol{v})$ is the vector that assigns to X the result of evaluating the power series $\boldsymbol{f}_X$ at $\boldsymbol{v}$. So, $\boldsymbol{f}$ naturally induces a mapping $\boldsymbol{f} \colon V \to V$.

2.2 Fixed-Point Equations and Kleene's Theorem.

The partial order $\sqsubseteq$ on the semiring $\mathcal{S}$ can be lifted to a partial order on vectors, also denoted by $\sqsubseteq$, and defined by $\boldsymbol{v} \sqsubseteq \boldsymbol{v}'$ if $\boldsymbol{v}_X \sqsubseteq \boldsymbol{v}'_X$ for every $X \in \mathcal{X}$.

Given a vector of power series $\boldsymbol{f}$, we are interested in the least fixed point of $\boldsymbol{f}$, i.e., the least vector $\boldsymbol{v}$ w.r.t. $\sqsubseteq$ satisfying $\boldsymbol{v} = \boldsymbol{f}(\boldsymbol{v})$. We briefly recall Kleene's theorem, which guarantees that the least fixed point exists.

A mapping $f\colon \mathcal{S} \to \mathcal{S}$ is *monotone* if $a \sqsubseteq b$ implies $f(a) \sqsubseteq f(b)$, and ω-*continuous* if for any infinite chain $a_0 \sqsubseteq a_1 \sqsubseteq a_2 \sqsubseteq \ldots$ we have $\sup\{f(a_i)\} = f(\sup\{a_i\})$. These definitions are extended to mappings $\boldsymbol{f}\colon V \to V$ from vectors to vectors by requiring them to hold in every component of $\boldsymbol{f}$. The following result is taken from [Kui97].

Proposition 2.4. *Let $\boldsymbol{f}$ be a vector of power series. The mapping induced by $\boldsymbol{f}$ is monotone and ω-continuous. Hence, by Kleene's theorem, $\boldsymbol{f}$ has a unique least fixed point $\mu\boldsymbol{f}$. Further, $\mu\boldsymbol{f}$ is the supremum (w.r.t. $\sqsubseteq$) of the* Kleene sequence *given by $\boldsymbol{\kappa}^{(0)} = \boldsymbol{f}(\mathbf{0})$, and $\boldsymbol{\kappa}^{(i+1)} = \boldsymbol{f}(\boldsymbol{\kappa}^{(i)})$.*[4]

2.3 Some Semiring Interpretations.

We recall that different interesting pieces of information about the program of Figure 1 correspond to the least solution of Equations (1) over different semirings.[5] For the rest of the section let $\Sigma = \{a, b, \ldots, i\}$ be the set of actions in the program, and let σ denote an arbitrary element of Σ.

Language interpretation Consider the following semiring. The carrier is 2^{Σ^*} (i.e., the set of languages over Σ). The semiring element σ is interpreted as the singleton language $\{\sigma\}$. The sum and product operations are union and concatenation of languages, respectively. We call it *language semiring* over Σ. Under this interpretation, Equations (1) are nothing but the following context-free grammar in Backus-Naur form:

$$X \to aXY \mid b \qquad Y \to cYZ \mid dYX \mid e \qquad Z \to gXh \mid i$$

The least solution of (1) is the triple $(L(X), L(Y), L(Z))$, where, for $U \in \{X, Y, Z\}$, $L(U)$ denotes the set of terminating executions of the program with U as main procedure, or, in language-theoretic terms, the language of the associated grammar with U as axiom.

Relational interpretation Assume that an action σ corresponds to a program instruction whose semantics is described by means of a relation $R_\sigma(V, V')$ over a set V of program variables (as usual, primed and unprimed variables correspond to the values before and after executing the instruction). Consider now the following semiring. The carrier is the set of all relations over (V, V'). The semiring element σ is interpreted as the relation R_σ. The sum and product operations are union and join of relations, respectively, i.e., $(R_1 \cdot R_2)(V, V') = \exists V'' R_1(V, V'') \wedge R_2(V'', V')$. Under this interpretation, the U-component of the least solution of (1) is the *summary* relation $R_U(V, V')$ containing the pairs V, V' such that if procedure U starts at valuation V, then it may terminate at valuation V'.

Counting interpretation Assume we wish to know how many as, bs, etc. we can observe in a (terminating) execution of the program, but we are not interested in the order in which they occur. In the terminology of abstract interpretation, we abstract an execution $w \in \Sigma^*$ by the vector $(n_a, \ldots, n_i) \in \mathbb{N}^{|\Sigma|}$ where $n_a, \ldots, n_i$ are the number of occurrences of $a, \ldots, i$ in w. We call $(n_a, \ldots, n_i)$ the *Parikh image* of w. The Parikh images of $L(X), L(Y), L(Z)$ are the least solution of (1) for the following semiring. The carrier is $2^{\mathbb{N}^{|\Sigma|}}$. The j-th action of Σ is interpreted as the singleton set $\{(0, \ldots, 0, 1, 0 \ldots, 0)\}$ with the "1" at the j-th position. The sum operation is set union, and the product operation is given by

$$S \cdot T = \{(s_a + t_a, \ldots, s_i + t_i) \mid (s_a, \ldots, s_i) \in S, (t_a, \ldots, t_i) \in T\} \; .$$

[4] Defining $\boldsymbol{\kappa}^{(0)} = \mathbf{0}$ would be more straightforward, but less convenient for this paper.

[5] This will be no surprise for the reader acquainted with abstract interpretation, but the examples will be used all throughout the paper.

Probabilistic interpretations Assume that the choices between actions are stochastic. For instance, actions a and b are chosen with probability p and $(1-p)$, respectively. The probability of termination is given by the least solution of (1) when interpreted over the following semiring (the *real semiring*) [EKM04,EY05]. The carrier is the set of non-negative real numbers, enriched with an additional element ∞. The semiring element σ is interpreted as the probability of choosing σ among all enabled actions. Sum and product are the standard operations on real numbers, suitably extended to ∞.

Assume now that actions are assigned not only a probability, but also a *duration*. Let d_σ denote the duration of σ. We are interested in the expected termination time of the program, under the condition that the program terminates (the *conditional expected time*). For this we consider the following semiring. The elements are the set of pairs (r_1, r_2), where r_1, r_2 are non-negative reals or ∞. We interpret σ as the pair (p_σ, d_σ), i.e., the probability and the duration of σ. The sum operation is defined as follows (where to simplify the notation we use $+_e$ and $\cdot_e$ for the operations of the semiring, and $+$ and $\cdot$ for sum and product of reals):

$$(p_1, d_1) +_e (p_2, d_2) = \left(p_1 + p_2, \frac{p_1 \cdot d_1 + p_2 \cdot d_2}{p_1 + p_2}\right)$$
$$(p_1, d_1) \cdot_e (p_2, d_2) = (p_1 \cdot p_2, d_1 + d_2)$$

The reader can easily check that this definition satisfies the semiring axioms. The U-component of the least solution of (1) is now the pair (t_U, e_U), where t_U is the probability that procedure U terminates, and e_U is its conditional expected time.

3 Newton's Method for ω-Continuous Semirings

We introduce our generalization of Newton's method for ω-continuous semirings. In Section 3.1 we consider the univariate case, i.e. the case of one equation in a single variable, which already allows us to introduce all important ideas. Here we first recall Newton's method as known from calculus, i.e., as a method for approximating a zero of a differentiable function. We then take a close look at the analytical definition, and identify the obstacles for a generalization to ω-continuous semirings. Finally, we propose a definition that overcomes the obstacles. In Section 3.2 we turn to the multivariate case. Sections 3.3 and 3.4 prove that our generalization of Newton's method is well-defined and converges to the least fixed point.

3.1 The Univariate Case

Given a differentiable function $g \colon \mathbb{R} \to \mathbb{R}$, Newton's method computes a zero of g, i.e., a solution of the equation $g(X) = 0$. The method starts at some value $\nu^{(0)}$ "close enough" to the zero, and proceeds iteratively: given $\nu^{(i)}$, it computes a value $\nu^{(i+1)}$ closer to the zero than $\nu^{(i)}$. For that, the method *linearizes* g at $\nu^{(i)}$, i.e., computes the tangent to g passing through the point $(\nu^{(i)}, g(\nu^{(i)}))$, and takes $\nu^{(i+1)}$ as the zero of the tangent (i.e., the x-coordinate of the point at which the tangent cuts the x-axis).

It is convenient for our purposes to formulate Newton's method in terms of the *differential* of g at a given point $v \in \mathbb{R}$. Recall that the differential of g is the mapping $Dg|_v : \mathbb{R} \to \mathbb{R}$ that assigns to each $v \in \mathbb{R}$ the linear function describing the tangent of g at the point $(v, g(v))$ in the coordinate system having $(v, g(v))$ as origin. If we denote the differential of g at v by $Dg|_v$, then we have $Dg|_v(X) = g'(v) \cdot X$ (for example, if $g(X) = X^2 + 3X + 1$, then $Dg|_3(X) = 9X$). In terms of differentials, Newton's method is formulated as follows. Starting at some $\nu^{(0)}$, compute iteratively $\nu^{(i+1)} = \nu^{(i)} + \Delta^{(i)}$, where $\Delta^{(i)}$ is the solution of the linear equation $Dg|_{\nu^{(i)}}(X) + g(\nu^{(i)}) = 0$ (assume for simplicity that the solution of the linear system is unique).

Computing the solution of a fixed-point equation, $f(X) = X$ amounts to computing a zero of $g(X) = f(X) - X$, and so we can apply Newton's method. Since for every real number v we have $Dg|_v(X) = Df|_v(X) - X$, the method looks as follows:

Starting at some $\nu^{(0)}$, compute iteratively

$$\nu^{(i+1)} = \nu^{(i)} + \Delta^{(i)} \tag{2}$$

where $\Delta^{(i)}$ is the solution of the linear equation

$$Df|_{\nu^{(i)}}(X) + f(\nu^{(i)}) - \nu^{(i)} = X \; . \tag{3}$$

So Newton's method "breaks down" the problem of finding a solution to a non-linear system $f(X) = X$ into finding solutions to the sequence (3) of linear systems.

Generalization Generalizing Newton's method to arbitrary ω-continuous semirings requires to overcome two obstacles. First, the notion of differential seems to require a richer algebraic structure than a semiring: differentials are usually defined in terms of derivatives, which are the limit of a quotient of differences, which requires both the sum and product operations to have inverses. Second, equation (3) contains the term $f(\nu^{(i)}) - \nu^{(i)}$, which again seems to be defined only if summation has an inverse.

The first obstacle Differentiable functions satisfy well known algebraic rules with respect to sums and products of functions. We take these rules as the definition of the differential of a power series f over an ω-continuous semiring $\mathcal{S}$. We remark that this definition of differential generalizes the usual algebraic definition of derivatives.

Definition 3.1. *Let f be a power series in one variable X over an ω-continuous semiring $\mathcal{S}$. The* differential *of f at the point v is the mapping $Df|_v : S \to S$ inductively defined as follows for every $b \in S$:*

$$Df|_v(b) = \begin{cases} 0 & \text{if } f \in S \\ b & \text{if } f = X \\ Dg|_v(b) \cdot h(v) + g(v) \cdot Dh|_v(b) & \text{if } f = g \cdot h \\ \sum_{i \in I} Df_i|_v(b) & \text{if } f = \sum_{i \in I} f_i(b) \ . \end{cases}$$

Example 3.2. First consider a polynomial f over some *commutative* ω-continuous semiring. Because of commutative multiplication, we may write any monomial as $a \cdot X^k$ for some $k \in \mathbb{N}$ and $a \in S$, and so $f = \sum_{k=0}^{n} a_k \cdot X^k$ for suitable $n \in \mathbb{N}$ and $a_k \in S$. Let f' denote the usual algebraic derivative of f w.r.t. X, i.e. $f' = \sum_{k=1}^{n} k \cdot a_k \cdot X^{k-1}$ where $k \cdot a_k$ is an abbreviation of $\sum_{i=1}^{k} a_k$. We then have

$$\begin{aligned} Df|_v(b) &= \textstyle\sum_{k=0}^{n} D(a_k \cdot X^k)|_v(b) \\ &= \textstyle\sum_{k=0}^{n} (Da_k|_v(b) \cdot (X^k)(v) + \sum_{j=0}^{k-1} a_k \cdot (X^j)(v) \cdot DX|_v(b) \cdot (X^{k-1-j})(v)) \\ &= \textstyle\sum_{k=0}^{n} \sum_{j=0}^{k-1} a_k \cdot v^j \cdot DX|_v(b) \cdot v^{k-1-j} \\ &= (\textstyle\sum_{k=1}^{n} k \cdot a_k \cdot v^{k-1}) \cdot b \\ &= f'(v) \cdot b. \end{aligned}$$

So, on commutative semirings, we have $Df|_v(b) = f'(v) \cdot b$ for all $v, b \in S$.

Now, assume that multiplication is not commutative, and consider the simple case of a quadratic monomial $m = a_0 X a_1 X a_2$. We then have

$$\begin{aligned} Dm|_v(b) &= a_0 \cdot DX|_v(b) \cdot a_1 \cdot v \cdot a_2 + a_0 \cdot v \cdot a_1 \cdot DX|_v(b) \cdot a_2 \\ &= a_0 \cdot b \cdot a_1 \cdot v \cdot a_2 + a_0 \cdot v \cdot a_1 \cdot b \cdot a_2. \end{aligned}$$

The important point here is that the differential "remembers" the position of the variables, and therefore not simply appends the value b. $\square$

The second obstacle Profiting from the fact that 0 is the unique minimal element of $\mathcal{S}$ with respect to $\sqsubseteq$, we fix $\nu^{(0)} = f(0)$, which guarantees $\nu^{(0)} \sqsubseteq f(\nu^{(0)})$. We *guess* that with this choice $\nu^{(i)} \sqsubseteq f(\nu^{(i)})$ will hold not only for $i = 0$, but for every $i \geq 0$ (the correctness of this guess is proved in Theorem 3.8). If the guess is correct, then, by the definition of $\sqsubseteq$, the semiring contains an element $\delta^{(i)}$ such that $f(\nu^{(i)}) = \nu^{(i)} + \delta^{(i)}$. We replace $f(\nu^{(i)}) - \nu^{(i)}$ by any such $\delta^{(i)}$. This leads to the following definition:

Definition 3.3. *Let f be a power series in one variable. A* Newton sequence *$(\nu^{(i)})_{i \in \mathbb{N}}$ is given by:*

$$\nu^{(0)} = f(0) \quad \text{and} \quad \nu^{(i+1)} = \nu^{(i)} + \Delta^{(i)} \tag{4}$$

where $\Delta^{(i)}$ is the least solution of

$$Df|_{\nu^{(i)}}(X) + \delta^{(i)} = X \tag{5}$$

and $\delta^{(i)}$ is any element satisfying $f(\nu^{(i)}) = \nu^{(i)} + \delta^{(i)}$.

In Section 3.3 we show that Newton sequences always exist (i.e., there is always at least one possible choice for $\delta^{(i)}$), and that they all converge at least as fast as the Kleene sequence. More precisely, we show that for every $i \geq 0$

$$\kappa^{(i)} \sqsubseteq \nu^{(i)} \sqsubseteq \nu^{(i+1)} \sqsubseteq \mu f \ .$$

Since we have $\mu f = \sup_{i \in \mathbb{N}} \kappa^{(i)}$ by Kleene's theorem, Newton sequences converge to μf.

In general, there can be more than one choice for $\delta^{(i)}$. In Section 3.4 we show, however, that the Newton sequence $(\nu^{(i)})_{i \geq 0}$ itself is uniquely determined by $\boldsymbol{f}$ (and $\mathcal{S}$). In other words, the choice of $\delta^{(i)}$ does not influence the Newton approximants $\nu^{(i)}$.

Before proving these results, let us consider some examples.

Examples We compute the Newton sequence for a program that can execute a and terminate, or execute b and then call itself twice, recursively (the abstract scheme of a divide-and-conquer procedure). The abstract equation of the program is

$$X = a + b \cdot X \cdot X \tag{6}$$

The real semiring Consider the case $a = b = 1/2$ (we can interpret a and b as probabilities). We have $Df|_v(X) = v \cdot X$, and one single possible choice for $\delta^{(i)}$, namely $\delta^{(i)} = f(\nu^{(i)}) - \nu^{(i)} = 1/2 + 1/2\,(\nu^{(i)})^2 - \nu^{(i)}$. Equation (5) becomes

$$\nu^{(i)}\,X + 1/2 + 1/2\,(\nu^{(i)})^2 - \nu^{(i)} = X$$

with $\Delta^{(i)} = (1 - \nu^{(i)})/2$ as unique solution. We get

$$\nu^{(0)} = 1/2 \qquad \nu^{(i+1)} = (1 + \nu^{(i)})/2$$

and therefore $\nu^{(i)} = 1 - 2^{(i+1)}$. So the Newton sequence converges to 1, and gains one bit of accuracy per iteration.

The language semiring Consider the language semiring with $\Sigma = \{a, b\}$. The product operation is concatenation of languages, and hence non-commutative. So we have $Df|_v(X) = bvX + bXv$. We show in Proposition 5.1 that when sum is idempotent (as in this case, where it is union of languages) the definition of the Newton sequence can be simplified to

$$\nu^{(0)} = f(0) \quad \text{ and } \quad \nu^{(i+1)} = \Delta^{(i)}, \tag{7}$$

where $\Delta^{(i)}$ is the least solution of

$$Df|_{\nu^{(i)}}(X) + f(\nu^{(i)}) = X \ . \tag{8}$$

With $f = a + b \cdot X \cdot X$ from Equation (6), Equation (8) becomes

$$\underbrace{b\nu^{(i)}X + bX\nu^{(i)}}_{Df|_{\nu^{(i)}}(X)} + \underbrace{a + b\nu^{(i)}\nu^{(i)}}_{f(\nu^{(i)})} = X \ . \tag{9}$$

Its least solution (which by (7) is equal to $(i+1)$-th Newton approximant) is a context-free language. Let $G^{(i)}$ be a grammar with axiom $S^{(i)}$ such that $\nu^{(i)} = L(G^{(i)})$. Since $\nu^{(0)} = f(0)$, the grammar $G^{(0)}$ contains one single production, namely $S^{(0)} \rightarrow a$. Equation (9) allows us to define $G^{(i+1)}$ in terms of $G^{(i)}$, and we get:

$$\begin{aligned} G^{(0)} &= \{S^{(0)} \rightarrow a\} \\ G^{(i+1)} &= G^{(i)} \cup \{S^{(i+1)} \rightarrow a \mid bS^{(i+1)}S^{(i)} \mid bS^{(i)}S^{(i+1)} \mid bS^{(i)}S^{(i)}\} \end{aligned}$$

Let $G = \{S \rightarrow a \mid bSS\}$ be the grammar derived from Equation (6). We have $L(G) = \bigcup_{i=1}^{n} L(G^{(i)})$. It is easy to see that $L(G^{(i)})$ contains the words of $L(G)$ *of index* $i+1$. Loosely speaking, the index of a word $w \in L(G)$ is the least number i such that some derivation of w contains no intermediate word having more than i occurrences of variables [Sal69]. Formally, the index of $w \in L(G)$ is the least number i for which a derivation $X = \alpha_0 \Rightarrow \cdots \Rightarrow \alpha_r = w$ exists such that for every $i \in \{0, \ldots, r\}$ the projection of α_i onto $\{X_1, \ldots, X_n\}$ has at most length i. In Section 5.1 we show that this characterization of the Newton approximants holds in general, i.e., the i-th Newton approximant of the language generated by a grammar G contains the words of $L(G)$ of index at most $i+1$.

The counting semiring Consider the counting semiring with $r_a = \{(1,0)\}$ and $r_b = \{(0,1)\}$. Since the sum operation is union of sets of vectors, it is idempotent and Equations (7) and (8) hold. Since the product operation is now commutative, we obtain for our example

$$b \cdot \nu^{(i)} \cdot X + a + b \cdot \nu^{(i)} \cdot \nu^{(i)} = X \tag{10}$$

Using Kleene's fixed-point theorem (Proposition 2.4), it is easy to see that the least solution of a linear equation $X = u \cdot X + v$ over a commutative ω-continuous semiring is $u^* \cdot v$, where $u^* = \sum_{i \in \mathbb{N}} u^i$. The least solution $\Delta^{(i)}$ of Equation (10) is then given by

$$\Delta^{(i)} = (r_b \cdot \nu^{(i)})^* \cdot (r_a + r_b \cdot \nu^{(i)} \cdot \nu^{(i)})$$

and we obtain:

$$\begin{aligned}
\nu^{(0)} &= r_a = \{(1,0)\} \\
\nu^{(1)} &= (r_b \cdot r_a)^* \cdot (r_a + r_b \cdot r_a \cdot r_a) = \{(n,n) \mid n \geq 0\} \cdot \{(1,0),(2,1)\} \\
&= \{(n+1,n) \mid n \geq 0\} \\
\nu^{(2)} &= (\{(n,n) \mid n \geq 1\})^* \cdot (\{(1,0)\} \cup \{(2n+2, 2n+1) \mid n \geq 0\}) \\
&= \{(n+1,n) \mid n \geq 0\}
\end{aligned}$$

So the Newton sequence reaches a fixed point after one iteration. In Section 6 we show that the Newton sequence of a system of n equations over *any commutative* and *idempotent* semiring converges after at most n iterations. Further note that the counting semiring does not satisfy the ascending-chain property, i.e. there are monotonically increasing sequences in the counting semiring which do not become stationary. Therefore, the Kleene sequence (and possible variations) does not reach $\mu \boldsymbol{f}$ after a finite number of steps in general.

3.2 The Multivariate Case

Newton's method can be easily generalized to the multivariate case. Given differentiable functions $g_1, \ldots, g_n \colon \mathbb{R}^n \to \mathbb{R}$, the method computes a solution of $\boldsymbol{g}(\boldsymbol{X}) = \boldsymbol{0}$, where $\boldsymbol{g} = (g_1, \ldots, g_n)$; starting at some $\boldsymbol{\nu}^{(0)}$, the method computes $\boldsymbol{\nu}^{(i+1)} = \boldsymbol{\nu}^{(i)} + \boldsymbol{\Delta}^{(i)}$, where $\boldsymbol{\Delta}^{(i)}$ is the solution of the *system* of linear equations

$$\begin{aligned}
Dg_1|_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{X}) + g_1(\boldsymbol{\nu}^{(i)}) &= 0 \\
&\vdots \\
Dg_n|_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{X}) + g_n(\boldsymbol{\nu}^{(i)}) &= 0
\end{aligned}$$

and $Dg_j|_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{X})$ is the differential of g_j at $\boldsymbol{\nu}^{(i)}$, i.e., the function corresponding to the tangent hyperplane of g_j at the point $(\boldsymbol{\nu}^{(i)}, g_j(\boldsymbol{\nu}^{(i)}))$.

Given a function $g \colon \mathbb{R}^n \to \mathbb{R}$ differentiable at a point $\boldsymbol{v}$, there exists a function $D_X g|_v$ for each variable $X \in \mathcal{X}$ such that $Dg|_{\boldsymbol{v}} = \sum_{X \in \mathcal{X}} D_X g|_{\boldsymbol{v}}$. These functions are closely related to the partial derivatives, more precisely we have $D_X g|_{\boldsymbol{v}}(\boldsymbol{X}) = \left.\frac{\partial g}{\partial X}\right|_{\boldsymbol{v}} \cdot X$.

We denote the system above by $D\boldsymbol{g}|_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{X}) + \boldsymbol{g}(\boldsymbol{\nu}^{(i)}) = \boldsymbol{0}$. For the problem of computing a solution of a system of fixed-point equations, the method looks as follows:

starting at some $\boldsymbol{\nu}^{(0)}$, compute iteratively

$$\boldsymbol{\nu}^{(i+1)} = \boldsymbol{\nu}^{(i)} + \boldsymbol{\Delta}^{(i)} \tag{11}$$

where $\boldsymbol{\Delta}^{(i)}$ is the least solution of the linear system of fixed-point equations

$$D\boldsymbol{f}|_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{X}) + \boldsymbol{f}(\boldsymbol{\nu}^{(i)}) - \boldsymbol{\nu}^{(i)} = \boldsymbol{X} \; . \tag{12}$$

Generalization Again, we use the algebraic definition of differential:

Definition 3.4. *Let f be a power series over an ω-continuous semiring $\mathcal{S}$ and let $X \in \mathcal{X}$ be a variable. The* differential of f w.r.t. X *at the point $\boldsymbol{v}$ is the mapping $D_X f|_{\boldsymbol{v}} : V \to S$ inductively defined as follows:*

$$D_X f|_{\boldsymbol{v}}(\boldsymbol{b}) = \begin{cases} 0 & \text{if } f \in S \text{ or } f \in \mathcal{X} \setminus \{X\} \\ \boldsymbol{b}_X & \text{if } f = X \\ D_X g|_{\boldsymbol{v}}(\boldsymbol{b}) \cdot h(\boldsymbol{v}) + g(\boldsymbol{v}) \cdot D_X h|_{\boldsymbol{v}}(\boldsymbol{b}) & \text{if } f = g \cdot h \\ \sum_{i \in I} D_X f_i|_{\boldsymbol{v}}(\boldsymbol{b}) & \text{if } f = \sum_{i \in I} f_i \,. \end{cases}$$

Further, we define the differential of f *at $\boldsymbol{v}$ as the function*

$$Df|_{\boldsymbol{v}} := \sum_{X \in \mathcal{X}} D_X f|_v .$$

Finally, the differential of a vector of power series $\boldsymbol{f}$ at $\boldsymbol{v}$ is defined as the function $D\boldsymbol{f}|_{\boldsymbol{v}} : V \to V$ with

$$\left(D\boldsymbol{f}|_{\boldsymbol{v}}(\boldsymbol{b})\right)_X := D\boldsymbol{f}_X|_{\boldsymbol{v}}(\boldsymbol{b}) \,.$$

As in the univariate case we guess that $\boldsymbol{\nu}^{(i)} \sqsubseteq \boldsymbol{f}(\boldsymbol{\nu}^{(i)})$ will hold for every $i \geq 0$. If the guess is correct, then the semiring contains an element $\boldsymbol{\delta}^{(i)}$ such that $\boldsymbol{f}(\boldsymbol{\nu}^{(i)}) = \boldsymbol{\nu}^{(i)} + \boldsymbol{\delta}^{(i)}$, and Equation (12) becomes

$$D\boldsymbol{f}|_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{X}) + \boldsymbol{\delta}^{(i)} = \boldsymbol{X} \,. \tag{13}$$

This leads to the following definition:

Definition 3.5. *Let $\boldsymbol{f} \colon V \to V$ be a vector of power series.*

- *Let $i \in \mathbb{N}$. An i-th* Newton approximant $\boldsymbol{\nu}^{(i)}$ *is inductively defined by*

$$\boldsymbol{\nu}^{(0)} = \boldsymbol{f}(\boldsymbol{0}) \quad \text{and} \quad \boldsymbol{\nu}^{(i+1)} = \boldsymbol{\nu}^{(i)} + \boldsymbol{\Delta}^{(i)} \,,$$

  where $\boldsymbol{\Delta}^{(i)}$ is the least solution of Equation (13) *and $\boldsymbol{\delta}^{(i)}$ is any vector satisfying $\boldsymbol{f}(\boldsymbol{\nu}^{(i)}) = \boldsymbol{\nu}^{(i)} + \boldsymbol{\delta}^{(i)}$.*
- *A sequence $(\boldsymbol{\nu}^{(i)})_{i \in \mathbb{N}}$ of Newton approximants is called* Newton sequence.

Remark 3.6. One can easily show by induction that for any $\boldsymbol{v}, \boldsymbol{a}, \boldsymbol{a}' \in V$, and any vector of power series $\boldsymbol{f}$ we have

$$D\boldsymbol{f}|_{\boldsymbol{v}}(\boldsymbol{b} + \boldsymbol{b}') = D\boldsymbol{f}|_{\boldsymbol{v}}(\boldsymbol{b}) + D\boldsymbol{f}|_{\boldsymbol{v}}(\boldsymbol{b}') \,.$$

Remark 3.7. If the product operation of the semiring is commutative, the differential $D_X f|_{\boldsymbol{v}}(\boldsymbol{a})$ can be written as $\frac{\partial f}{\partial X}|_{\boldsymbol{v}} \cdot \boldsymbol{a}_X$, where $\frac{\partial f}{\partial X}|_{\boldsymbol{v}}$ denotes the usual partial derivative of the power series f w.r.t. X, taken at $\boldsymbol{v}$, as known from algebra:

$$\left.\frac{\partial f}{\partial X}\right|_{\boldsymbol{v}} = \begin{cases} 0 & \text{if } f \in S \text{ or } f \in \mathcal{X} \setminus \{X\} \\ 1 & \text{if } f = X \\ \frac{\partial g}{\partial x}|_{\boldsymbol{v}} \cdot h(\boldsymbol{v}) + g(\boldsymbol{v}) \cdot \frac{\partial h}{\partial X}|_{\boldsymbol{v}} & \text{if } f = g \cdot h \\ \sum_{i \in I} \frac{\partial f_i}{\partial X}|_{\boldsymbol{v}} & \text{if } f = \sum_{i \in I} f_i \,. \end{cases}$$

So, in commutative semirings we may use the usual representation of the differential by means of the gradient of a power series f, or more generally, by the Jacobian of a vector $\boldsymbol{f}$ of power series.

3.3 Fundamental Properties of the Newton Sequences

In the rest of the section we prove the following theorem, showing that there exists exactly one Newton sequence, that it converges to the least fixed point, and that it does so at least as fast as the Kleene sequence.

Theorem 3.8. *Let $\boldsymbol{f} \colon V \to V$ be a vector of power series.*

– *There is exactly one Newton sequence* $(\boldsymbol{\nu}^{(i)})_{i\in\mathbb{N}}$.
– *The Newton sequence is monotonically increasing, converges to the least fixed point and does so at least as fast as the Kleene sequence. More precisely, it satisfies*

$$\boldsymbol{\kappa}^{(i)} \sqsubseteq \boldsymbol{\nu}^{(i)} \sqsubseteq \boldsymbol{f}(\boldsymbol{\nu}^{(i)}) \sqsubseteq \boldsymbol{\nu}^{(i+1)} \sqsubseteq \mu\boldsymbol{f} = \sup_{j\in\mathbb{N}} \boldsymbol{\kappa}^{(j)} \text{ for all } i \in \mathbb{N}.$$

We split the proof Theorem 3.8 in two propositions. Proposition 3.14 in Section 3.4 states that there is only one Newton sequence. The following proposition covers the rest of Theorem 3.8:

Proposition 3.9. *Let* $\boldsymbol{f}\colon V \to V$ *be a vector of power series.*

– *For every Newton approximant* $\boldsymbol{\nu}^{(i)}$ *there exists a vector* $\boldsymbol{\delta}^{(i)}$ *such that* $\boldsymbol{f}(\boldsymbol{\nu}^{(i)}) = \boldsymbol{\nu}^{(i)} + \boldsymbol{\delta}^{(i)}$. *So there is at least one Newton sequence.*
– *It satisfies* $\boldsymbol{\kappa}^{(i)} \sqsubseteq \boldsymbol{\nu}^{(i)} \sqsubseteq \boldsymbol{f}(\boldsymbol{\nu}^{(i)}) \sqsubseteq \boldsymbol{\nu}^{(i+1)} \sqsubseteq \mu\boldsymbol{f} = \sup_{j\in\mathbb{N}} \boldsymbol{\kappa}^{(j)}$ *for all* $i \in \mathbb{N}$.

The proof of Proposition 3.9 is based on two lemmata. The first one, an easy consequence of Kleene's theorem, provides a closed form for the least solution of a linear system of fixed-point equations in terms of the Kleene star operator, defined as follows:

Definition 3.10. *Let* $\boldsymbol{g}\colon V \to V$ *be a monotone map. The map* $\boldsymbol{g}^*\colon V \to V$ *is defined as* $\boldsymbol{g}^*(\boldsymbol{v}) := \sum_{i\in\mathbb{N}} \boldsymbol{g}^i(\boldsymbol{v})$, *where* $\boldsymbol{g}^0(\boldsymbol{v}) := \boldsymbol{v}$, $\boldsymbol{g}^{i+1}(\boldsymbol{v}) := \boldsymbol{g}(\boldsymbol{g}^i(\boldsymbol{v}))$ *for every* $i \geq 0$. *Similarly, we set for all* $j \in \mathbb{N}$: $\boldsymbol{g}^{\leq j} := \sum_{0\leq i\leq j} \boldsymbol{g}^i(\boldsymbol{v})$.

The existence of $\sum_{i\in\mathbb{N}} \boldsymbol{g}^i(\boldsymbol{v})$ is guaranteed by the properties of ω-continuous semirings. Observe that $\boldsymbol{v} \sqsubseteq \boldsymbol{g}^*(\boldsymbol{v})$ and $\boldsymbol{g}^*(\boldsymbol{v}) = \boldsymbol{v} + \boldsymbol{g}(\boldsymbol{g}^*(\boldsymbol{v}))$ hold.

Lemma 3.11. *Let* $\boldsymbol{f}\colon V \to V$ *be a vector of power series, and* $\boldsymbol{u}, \boldsymbol{v} \in V$. *Then the least solution of* $D\boldsymbol{f}|_{\boldsymbol{u}}(\boldsymbol{X}) + \boldsymbol{v} = \boldsymbol{X}$ *is* $D\boldsymbol{f}|_{\boldsymbol{u}}^*(\boldsymbol{v})$. *In particular, a Newton sequence from Definition 3.5 can be equivalently defined by setting* $\boldsymbol{\nu}^{(0)} = \boldsymbol{f}(\boldsymbol{0})$ *and* $\boldsymbol{\nu}^{(i+1)} = \boldsymbol{\nu}^{(i)} + D\boldsymbol{f}|_{\boldsymbol{\nu}^{(i)}}^*(\boldsymbol{\delta}^{(i)})$.

Proof. Set $\boldsymbol{g}(\boldsymbol{X}) := D\boldsymbol{f}|_{\boldsymbol{u}}(\boldsymbol{X}) + \boldsymbol{v}$. The vector $\boldsymbol{g}$ is a power series in every component and thus a monotone map from V to V. By Kleene's fixed-point theorem, the least solution of $\boldsymbol{g}(\boldsymbol{X}) = \boldsymbol{X}$ is given by $\sup\{\boldsymbol{g}^i(\boldsymbol{0}) \mid i \in \mathbb{N}\} = \sup\{D\boldsymbol{f}|_{\boldsymbol{u}}^{\leq i}(\boldsymbol{v}) \mid i \in \mathbb{N}\} = D\boldsymbol{f}|_{\boldsymbol{u}}^*(\boldsymbol{v})$. ☐

The second lemma, which is interesting by itself, is a generalization of Taylor's theorem to arbitrary ω-continuous semirings.

Lemma 3.12. *Let* $\boldsymbol{f}\colon V \to V$ *be a vector of power series and let* $\boldsymbol{u}, \boldsymbol{v}$ *be two vectors. We have*

$$\boldsymbol{f}(\boldsymbol{u}) + D\boldsymbol{f}|_{\boldsymbol{u}}(\boldsymbol{v}) \sqsubseteq \boldsymbol{f}(\boldsymbol{u} + \boldsymbol{v}) \sqsubseteq \boldsymbol{f}(\boldsymbol{u}) + D\boldsymbol{f}|_{\boldsymbol{u}+\boldsymbol{v}}(\boldsymbol{v}) .$$

Proof. It suffices to show those inequalities for each component separately, so let w.l.o.g. $\boldsymbol{f} = f\colon V \to S$ be a power series. We proceed by induction on the construction of f. The base case (where f is a constant) and the case where f is a sum of polynomials are easy, and so it suffices to consider the case in which f is a monomial. So let

$$f = g \cdot X \cdot a$$

for a monomial g, a variable $X \in \mathcal{X}$ and a constant a. We have

$$f(\boldsymbol{u}) = g(\boldsymbol{u}) \cdot \boldsymbol{u}_X \cdot a \quad \text{and} \quad Df|_{\boldsymbol{u}}(\boldsymbol{v}) = g(\boldsymbol{u}) \cdot \boldsymbol{v}_X \cdot a + Dg|_{\boldsymbol{u}}(\boldsymbol{v}) \cdot \boldsymbol{u}_X \cdot a .$$

By induction we obtain:

$$\begin{aligned} f(\boldsymbol{u}+\boldsymbol{v}) &= g(\boldsymbol{u}+\boldsymbol{v}) \cdot (\boldsymbol{u}_X + \boldsymbol{v}_X) \cdot a \\ &\sqsupseteq \big(g(\boldsymbol{u}) + Dg|_{\boldsymbol{u}}(\boldsymbol{v})\big) \cdot (\boldsymbol{u}_X + \boldsymbol{v}_X) \cdot a \\ &= g(\boldsymbol{u}) \cdot \boldsymbol{u}_X \cdot a + g(\boldsymbol{u}) \cdot \boldsymbol{v}_X \cdot a + Dg|_{\boldsymbol{u}}(\boldsymbol{v}) \cdot (\boldsymbol{u}_X + \boldsymbol{v}_X) \cdot a \\ &\sqsupseteq f(\boldsymbol{u}) + g(\boldsymbol{u}) \cdot \boldsymbol{v}_X \cdot a + Dg|_{\boldsymbol{u}}(\boldsymbol{v}) \cdot \boldsymbol{u}_X \cdot a \\ &= f(\boldsymbol{u}) + Df|_{\boldsymbol{u}}(\boldsymbol{v}) \end{aligned}$$

and

$$\begin{aligned}
f(\boldsymbol{u}+\boldsymbol{v}) &= g(\boldsymbol{u}+\boldsymbol{v}) \cdot (\boldsymbol{u}_X + \boldsymbol{v}_X) \cdot a \\
&\sqsubseteq \big(g(\boldsymbol{u}) + Dg|_{\boldsymbol{u}+\boldsymbol{v}}(\boldsymbol{v})\big) \cdot (\boldsymbol{u}_X + \boldsymbol{v}_X) \cdot a \\
&= g(\boldsymbol{u}) \cdot \boldsymbol{u}_X \cdot a + g(\boldsymbol{u}) \cdot \boldsymbol{v}_X \cdot a + Dg|_{\boldsymbol{u}+\boldsymbol{v}}(\boldsymbol{v}) \cdot (\boldsymbol{u}_X + \boldsymbol{v}_X) \cdot a \\
&\sqsubseteq f(\boldsymbol{u}) + g(\boldsymbol{u}+\boldsymbol{v}) \cdot \boldsymbol{v}_X \cdot a + Dg|_{\boldsymbol{u}+\boldsymbol{v}}(\boldsymbol{v}) \cdot (\boldsymbol{u}_X + \boldsymbol{v}_X) \cdot a \\
&= f(\boldsymbol{u}) + Df|_{\boldsymbol{u}+\boldsymbol{v}}(\boldsymbol{v})
\end{aligned}$$

□

We can now proceed to prove Proposition 3.9.

Proof (of Proposition 3.9). First we prove for all $i \in \mathbb{N}$ that a suitable $\boldsymbol{\delta}^{(i)}$ exists and, at the same time, that the inequality $\boldsymbol{\kappa}^{(i)} \sqsubseteq \boldsymbol{\nu}^{(i)} \sqsubseteq \boldsymbol{f}(\boldsymbol{\nu}^{(i)})$ holds. We proceed by induction on i. The base case $i = 0$ is easy. For the step, let $i \geq 0$.

$$\begin{aligned}
\boldsymbol{\kappa}^{(i+1)} &= \boldsymbol{f}(\boldsymbol{\kappa}^{(i)}) && \text{(definition of } \boldsymbol{\kappa}^{(i)}\text{)} \\
&\sqsubseteq \boldsymbol{f}(\boldsymbol{\nu}^{(i)}) && \text{(induction: } \boldsymbol{\kappa}^{(i)} \sqsubseteq \boldsymbol{\nu}^{(i)}\text{)} \\
&= \boldsymbol{\nu}^{(i)} + \boldsymbol{\delta}^{(i)} \text{ for some } \boldsymbol{\delta}^{(i)} && \text{(induction)} \\
&\sqsubseteq \boldsymbol{\nu}^{(i)} + D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{\delta}^{(i)}) && (\boldsymbol{v} \sqsubseteq \boldsymbol{g}^*(\boldsymbol{v})) \\
&= \boldsymbol{\nu}^{(i+1)} && \text{(Lemma 3.11)} \\
&= \boldsymbol{\nu}^{(i)} + \boldsymbol{\delta}^{(i)} + D\boldsymbol{f}|_{\boldsymbol{\nu}^{(i)}}(D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{\delta}^{(i)})) && (\boldsymbol{g}^*(\boldsymbol{v}) = \boldsymbol{v} + \boldsymbol{g}(\boldsymbol{g}^*(\boldsymbol{v}))\,) \\
&= \boldsymbol{f}(\boldsymbol{\nu}^{(i)}) + D\boldsymbol{f}|_{\boldsymbol{\nu}^{(i)}}(D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{\delta}^{(i)})) && \text{(definition of } \boldsymbol{\delta}^{(i)}\text{)} \\
&\sqsubseteq \boldsymbol{f}(\boldsymbol{\nu}^{(i)} + D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{\delta}^{(i)})) && \text{(Lemma 3.12)} \\
&= \boldsymbol{f}(\boldsymbol{\nu}^{(i+1)}) && \text{(Lemma 3.11)}
\end{aligned}$$

Since $\boldsymbol{\nu}^{(i+1)} \sqsubseteq \boldsymbol{f}(\boldsymbol{\nu}^{(i+1)})$, there exists a $\boldsymbol{\delta}^{(i+1)}$ such that $\boldsymbol{\nu}^{(i+1)} + \boldsymbol{\delta}^{(i+1)} \sqsubseteq \boldsymbol{f}(\boldsymbol{\nu}^{(i+1)})$. Next we prove $\boldsymbol{f}(\boldsymbol{\nu}^{(i)}) \sqsubseteq \boldsymbol{\nu}^{(i+1)}$:

$$\begin{aligned}
\boldsymbol{f}(\boldsymbol{\nu}^{(i)}) &= \boldsymbol{\nu}^{(i)} + \boldsymbol{\delta}^{(i)} && \text{(as proved above)} \\
&\sqsubseteq \boldsymbol{\nu}^{(i)} + D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{\delta}^{(i)}) && (\boldsymbol{v} \sqsubseteq \boldsymbol{g}^*(\boldsymbol{v})) \\
&= \boldsymbol{\nu}^{(i+1)} && \text{(Lemma 3.11)}
\end{aligned}$$

It remains to prove $\sup_{j\in\mathbb{N}} \boldsymbol{\kappa}^{(j)} = \mu\boldsymbol{f}$ and $\boldsymbol{\nu}^{(i)} \sqsubseteq \mu\boldsymbol{f}$ for all i. The equation $\sup_{j\in\mathbb{N}} \boldsymbol{\kappa}^{(j)} = \mu\boldsymbol{f}$ holds by Kleene's theorem (Proposition 2.4). To prove $\boldsymbol{\nu}^{(i)} \sqsubseteq \mu\boldsymbol{f}$ for all i we need a lemma.

Lemma 3.13. *Let $\boldsymbol{f}(\boldsymbol{x}) \sqsupseteq \boldsymbol{x}$. For all $d \geq 0$ there exists a vector $\boldsymbol{e}^{(d)}(\boldsymbol{x})$ such that*

$$\begin{aligned}
\boldsymbol{f}^d(\boldsymbol{x}) + \boldsymbol{e}^{(d)}(\boldsymbol{x}) = \boldsymbol{f}^{d+1}(\boldsymbol{x}) \quad \textit{and} \\
\boldsymbol{e}^{(d)}(\boldsymbol{x}) &\sqsupseteq D\boldsymbol{f}|_{\boldsymbol{f}^{d-1}(\boldsymbol{x})}(D\boldsymbol{f}|_{\boldsymbol{f}^{d-2}(\boldsymbol{x})}(\ldots D\boldsymbol{f}|_{\boldsymbol{x}}(\boldsymbol{e}^{(0)}(\boldsymbol{x}))\ldots)) \\
&\sqsupseteq D\boldsymbol{f}|^d_{\boldsymbol{x}}(\boldsymbol{e}^{(0)}(\boldsymbol{x}))\ .
\end{aligned}$$

Proof of the lemma. By induction on d. For $d = 0$ there is an appropriate $\boldsymbol{e}^{(0)}(\boldsymbol{x})$ by assumption. Let $d \geq 0$.

$$\begin{aligned}
\boldsymbol{f}^{d+2}(\boldsymbol{x}) &= \boldsymbol{f}(\boldsymbol{f}^d(\boldsymbol{x}) + \boldsymbol{e}^{(d)}(\boldsymbol{x})) && \text{(induction)} \\
&\sqsupseteq \boldsymbol{f}^{d+1}(\boldsymbol{x}) + D\boldsymbol{f}|_{\boldsymbol{f}^d(\boldsymbol{x})}(\boldsymbol{e}^{(d)}(\boldsymbol{x})) && \text{(Lemma 3.12)} \\
&\sqsupseteq \boldsymbol{f}^{d+1}(\boldsymbol{x}) + D\boldsymbol{f}|_{\boldsymbol{f}^d(\boldsymbol{x})}(\ldots D\boldsymbol{f}|_{\boldsymbol{x}}(\boldsymbol{e}^{(0)}(\boldsymbol{x}))\ldots) && \text{(induction)}
\end{aligned}$$

Therefore, there exists an $\boldsymbol{e}^{(d+1)}(\boldsymbol{x}) \sqsupseteq D\boldsymbol{f}|_{\boldsymbol{f}^d(\boldsymbol{x})}(\ldots D\boldsymbol{f}|_{\boldsymbol{x}}(\boldsymbol{e}^{(0)}(\boldsymbol{x}))\ldots)$. Since $D\boldsymbol{f}|_{\boldsymbol{y}}$ is monotone in $\boldsymbol{y}$ and $\boldsymbol{x} \sqsubseteq \boldsymbol{f}(\boldsymbol{x}) \sqsubseteq \boldsymbol{f}^2(\boldsymbol{x}) \sqsubseteq \ldots$, the second inequality also holds. This completes the proof of the lemma. $\square$

Notice that Lemma 3.13 holds for $\boldsymbol{x} = \boldsymbol{\nu}^{(i)}$ and $\boldsymbol{e}^{(0)}(\boldsymbol{\nu}^{(i)}) = \boldsymbol{\delta}^{(i)}$, because we have already shown $\boldsymbol{\nu}^{(i)} \sqsubseteq \boldsymbol{f}(\boldsymbol{\nu}^{(i)})$. Now we can prove $\boldsymbol{\nu}^{(i)} \sqsubseteq \mu\boldsymbol{f}$ by induction on i. The case $i = 0$ is trivial. Let $i \geq 0$. We have:

$$
\begin{aligned}
\boldsymbol{\nu}^{(i+1)} &= \boldsymbol{\nu}^{(i)} + D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{\delta}^{(i)}) && \text{(Lemma 3.11)} \\
&= \boldsymbol{\nu}^{(i)} + \sum_{d \in \mathbb{N}} D\boldsymbol{f}|^d_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{\delta}^{(i)}) && \text{(definition of } D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}) \\
&\sqsubseteq \boldsymbol{\nu}^{(i)} + \sum_{d \in \mathbb{N}} \boldsymbol{e}^{(d)}(\boldsymbol{\nu}^{(i)}) && \text{(Lemma 3.13)} \\
&= \sup_{d \in \mathbb{N}} \boldsymbol{f}^d(\boldsymbol{\nu}^{(i)}) && (\omega\text{-continuity)} \\
&\sqsubseteq \mu\boldsymbol{f} && \text{(induction:} \\
& && \quad \boldsymbol{\nu}^{(i)} \sqsubseteq \boldsymbol{f}(\boldsymbol{\nu}^{(i)}) \sqsubseteq \boldsymbol{f}(\boldsymbol{f}(\boldsymbol{\nu}^{(i)})) \sqsubseteq \ldots \sqsubseteq \mu\boldsymbol{f})
\end{aligned}
$$

This completes the proof of Proposition 3.9. $\square$

3.4 Uniqueness

In Definition 3.5 the Newton approximant $\boldsymbol{\nu}^{(i)}$ is defined in terms of a vector $\boldsymbol{\delta}^{(i)}$ satisfying $\boldsymbol{\nu}^{(i)} + \boldsymbol{\delta}^{(i)} = \boldsymbol{f}(\boldsymbol{\nu}^{(i)})$. In the previous section we have shown that such a vector always exists. However, in a semiring there there may be multiple such $\boldsymbol{\delta}^{(i)}$'s, and so in principle there could be multiple Newton sequences. We show now that this is *not* the case, i.e., there is only one Newton sequence $(\boldsymbol{\nu}^{(i)})_{i \in \mathbb{N}}$, independent of the choice of $\boldsymbol{\delta}^{(i)}$:

Proposition 3.14. *Let $\boldsymbol{f} : V \to V$ be a vector of power series. There is exactly one Newton sequence $(\boldsymbol{\nu}^{(i)})_{i \in \mathbb{N}}$.*

Theorem 3.8 follows directly by combining Proposition 3.9 and Proposition 3.14. So for Theorem 3.8 it remains to prove Proposition 3.14, which we do in the rest of this section.

It is convenient for the proof to introduce *substitutionals*, a notion related to differentials, see Definition 3.4.

Definition 3.15. *Let f be a power series over an ω-continuous semiring $\mathcal{S}$ and let $s \in \mathbb{N}_+$. The* substitutional of f w.r.t. s *at the point $\boldsymbol{v}$ is the mapping $\$_s f|_{\boldsymbol{v}} : V \to S$ defined as follows:*
If f is a monomial, i.e., of the form $f = a_1 X_1 \cdots a_k X_k a_{k+1}$, then

$$
\$_s f|_{\boldsymbol{v}}(\boldsymbol{b}) = \begin{cases} a_1 \boldsymbol{v}_{X_1} \cdots a_{s-1} \boldsymbol{v}_{X_{s-1}} a_s \boldsymbol{b}_{X_s} a_{s+1} \boldsymbol{v}_{X_{s+1}} \cdots a_k \boldsymbol{v}_{X_k} a_{k+1} & \text{if } 1 \leq s \leq k \\ 0 & \text{otherwise.} \end{cases}
$$

If f is a power series, i.e., of the form $f = \sum_{i \in I} f_i$, then

$$
\$_s f|_{\boldsymbol{v}}(\boldsymbol{b}) = \sum_{i \in I} \$_s f_i|_{\boldsymbol{v}}(\boldsymbol{b}).
$$

In words: if f is a monomial with at least s variables then $\$_s f|_{\boldsymbol{v}}(\boldsymbol{b})$ is obtained from f by replacing the s-th variable X_s by $\boldsymbol{b}_{X_s}$ and all other variables by the corresponding component of $\boldsymbol{v}$. If f is a monomial with less than s variables then $\$_s f|_{\boldsymbol{v}}(\boldsymbol{b}) = 0$. If f is a power series then the substitutional of f is the sum of the substitutionals of f's monomials.

Analogously to differentials, we extend the definition of substitutionals to vectors of power series by applying the substitution componentwise. Formally, we define the substitutional of a vector of power series $\boldsymbol{f}$ at $\boldsymbol{v}$ as the function $\$_s \boldsymbol{f}|_{\boldsymbol{v}} : V \to V$ with

$$
(\$_s \boldsymbol{f}|_{\boldsymbol{v}}(\boldsymbol{b}))_X := \$_s \boldsymbol{f}_X|_{\boldsymbol{v}}(\boldsymbol{b}) \, .
$$

Observe that, like the differential (see Remark 3.6), the substitutional is "linear", i.e., $\$_s \boldsymbol{f}|_{\boldsymbol{v}}(\boldsymbol{b}+\boldsymbol{b}') = \$_s \boldsymbol{f}|_{\boldsymbol{v}}(\boldsymbol{b}) + \$_s \boldsymbol{f}|_{\boldsymbol{v}}(\boldsymbol{b}')$.

Notation 1. *For any $j \in \mathbb{N}$ and any sequence $s = (s_1, \dots s_j) \in \mathbb{N}_+^j$ we write $\$_s \boldsymbol{f}|_{\boldsymbol{v}}(\boldsymbol{b})$ for $\$_{s_1} \boldsymbol{f}|_{\boldsymbol{v}}(\$_{s_2} \boldsymbol{f}|_{\boldsymbol{v}}(\cdots \$_{s_j} \boldsymbol{f}|_{\boldsymbol{v}}(\boldsymbol{b}) \cdots))$, and $\$_s \boldsymbol{f}|_{\boldsymbol{v}}(\boldsymbol{b}) = \boldsymbol{b}$ if $j = 0$.*

The following facts are immediate from the definitions.

Proposition 3.16. *Let f be a monomial. Then*

$$D_X f|_{\boldsymbol{v}}(\boldsymbol{b}) = \sum \{\$_s f|_{\boldsymbol{v}}(\boldsymbol{b}) \mid X \text{ is the } s\text{-th variable in } f\}.$$

Let $\boldsymbol{f}$ be a vector of power series. Then:

1. $D\boldsymbol{f}|_{\boldsymbol{v}}(\boldsymbol{b}) = \sum_{s \in \mathbb{N}_+} \$_s \boldsymbol{f}|_{\boldsymbol{v}}(\boldsymbol{b})$.
2. $D\boldsymbol{f}|_{\boldsymbol{v}}^j(\boldsymbol{b}) = \sum_{s \in \mathbb{N}_+^j} \$_s \boldsymbol{f}|_{\boldsymbol{v}}(\boldsymbol{b})$.
3. *For all $s \in \mathbb{N}_+$ we have $\boldsymbol{f}(\boldsymbol{v}) \sqsupseteq \$_s \boldsymbol{f}|_{\boldsymbol{v}}(\boldsymbol{v})$.*

Example 3.17. Consider the polynomial $f = aXYX + cY$. Then

$$\begin{aligned}
\$_1 f|_{\boldsymbol{v}}(\boldsymbol{b}) &= a\boldsymbol{b}_X \boldsymbol{v}_Y \boldsymbol{v}_X + c\boldsymbol{b}_Y \\
\$_2 f|_{\boldsymbol{v}}(\boldsymbol{b}) &= a\boldsymbol{v}_X \boldsymbol{b}_Y \boldsymbol{v}_X \\
\$_3 f|_{\boldsymbol{v}}(\boldsymbol{b}) &= a\boldsymbol{v}_X \boldsymbol{v}_Y \boldsymbol{b}_X \\
D_X f|_{\boldsymbol{v}}(\boldsymbol{b}) &= a\boldsymbol{b}_X \boldsymbol{v}_Y \boldsymbol{v}_X + a\boldsymbol{v}_X \boldsymbol{v}_Y \boldsymbol{b}_X \\
D_Y f|_{\boldsymbol{v}}(\boldsymbol{b}) &= a\boldsymbol{v}_X \boldsymbol{b}_Y \boldsymbol{v}_X + c\boldsymbol{b}_Y \ .
\end{aligned}$$

Observe that $Df|_{\boldsymbol{v}}(\boldsymbol{b}) = D_X f|_{\boldsymbol{v}}(\boldsymbol{b}) + D_Y f|_{\boldsymbol{v}}(\boldsymbol{b}) = \$_1 f|_{\boldsymbol{v}}(\boldsymbol{b}) + \$_2 f|_{\boldsymbol{v}}(\boldsymbol{b}) + \$_3 f|_{\boldsymbol{v}}(\boldsymbol{b})$ and that $f(\boldsymbol{v}) = a\boldsymbol{v}_X \boldsymbol{v}_Y \boldsymbol{v}_X + c\boldsymbol{v}_Y \sqsupseteq \$_s f|_{\boldsymbol{v}}(\boldsymbol{v})$ holds for all $s \in \mathbb{N}_+$. ◻

For the proof of Proposition 3.14 we need the following two lemmata.

Lemma 3.18. *Let $\boldsymbol{f}$ be a vector of power series. Let $\boldsymbol{\nu} + \boldsymbol{\delta} = \boldsymbol{f}(\boldsymbol{\nu})$. Let $j \in \mathbb{N}$ and $(s_1, \dots, s_{j+1}) \in \mathbb{N}_+^{j+1}$. Then $\boldsymbol{\nu} + D\boldsymbol{f}|_{\boldsymbol{\nu}}^{\leq j}(\boldsymbol{\delta}) \sqsupseteq \$_{(s_1,\dots,s_{j+1})} \boldsymbol{f}|_{\boldsymbol{\nu}}(\boldsymbol{\nu})$.*

Proof. By induction on j. For $j = 0$ we have $\boldsymbol{\nu} + D\boldsymbol{f}|_{\boldsymbol{\nu}}^{\leq 0}(\boldsymbol{\delta}) = \boldsymbol{\nu} + \boldsymbol{\delta} = \boldsymbol{f}(\boldsymbol{\nu}) \sqsupseteq \$_{s_1} \boldsymbol{f}|_{\boldsymbol{\nu}}(\boldsymbol{\nu})$ by Proposition 3.16.3. Let $j \geq 0$. We have:

$$\begin{aligned}
\boldsymbol{\nu} + D\boldsymbol{f}|_{\boldsymbol{\nu}}^{\leq j+1}(\boldsymbol{\delta}) &= \boldsymbol{\nu} + D\boldsymbol{f}|_{\boldsymbol{\nu}}^{\leq j}(\boldsymbol{\delta}) + D\boldsymbol{f}|_{\boldsymbol{\nu}}^{j+1}(\boldsymbol{\delta}) \\
&\sqsupseteq \$_{(s_1,\dots,s_{j+1})} \boldsymbol{f}|_{\boldsymbol{\nu}}(\boldsymbol{\nu}) + D\boldsymbol{f}|_{\boldsymbol{\nu}}^{j+1}(\boldsymbol{\delta}) && \text{(induction)} \\
&\sqsupseteq \$_{(s_1,\dots,s_{j+1})} \boldsymbol{f}|_{\boldsymbol{\nu}}(\boldsymbol{\nu}) + \$_{(s_1,\dots,s_{j+1})} \boldsymbol{f}|_{\boldsymbol{\nu}}(\boldsymbol{\delta}) && \text{(Prop. 3.16.2.)} \\
&= \$_{(s_1,\dots,s_{j+1})} \boldsymbol{f}|_{\boldsymbol{\nu}}(\boldsymbol{f}(\boldsymbol{\nu})) && (\boldsymbol{\nu} + \boldsymbol{\delta} = \boldsymbol{f}(\boldsymbol{\nu})) \\
&\sqsupseteq \$_{(s_1,\dots,s_{j+1})} \boldsymbol{f}|_{\boldsymbol{\nu}}(\$_{s_{j+2}} \boldsymbol{f}|_{\boldsymbol{\nu}}(\boldsymbol{\nu})) && \text{(Prop. 3.16.3.)} \\
&= \$_{(s_1,\dots,s_{j+2})} \boldsymbol{f}|_{\boldsymbol{\nu}}(\boldsymbol{\nu})
\end{aligned}$$

◻

Lemma 3.19. *Let $\boldsymbol{f}$ be a vector of power series. Let $\boldsymbol{\nu} + \boldsymbol{\delta} = \boldsymbol{\nu} + \boldsymbol{\delta}' = \boldsymbol{f}(\boldsymbol{\nu})$. Then $\boldsymbol{\nu} + D\boldsymbol{f}|_{\boldsymbol{\nu}}^*(\boldsymbol{\delta}) = \boldsymbol{\nu} + D\boldsymbol{f}|_{\boldsymbol{\nu}}^*(\boldsymbol{\delta}')$.*

Proof. We show $\boldsymbol{\nu} + D\boldsymbol{f}|_{\boldsymbol{\nu}}^{\leq j}(\boldsymbol{\delta}) = \boldsymbol{\nu} + D\boldsymbol{f}|_{\boldsymbol{\nu}}^{\leq j}(\boldsymbol{\delta}')$ for all $j \in \mathbb{N}$. Then the lemma follows by ω-continuity. We proceed by induction on j. The induction base ($j = 0$) is clear. Let $j \geq 0$. We have:

$$\begin{aligned}
\boldsymbol{\nu} + D\boldsymbol{f}|_{\boldsymbol{\nu}}^{\leq j+1}(\boldsymbol{\delta}) &= \boldsymbol{\nu} + D\boldsymbol{f}|_{\boldsymbol{\nu}}^{\leq j}(\boldsymbol{\delta}) + D\boldsymbol{f}|_{\boldsymbol{\nu}}^{j+1}(\boldsymbol{\delta}) \\
&= \boldsymbol{\nu} + D\boldsymbol{f}|_{\boldsymbol{\nu}}^{\leq j}(\boldsymbol{\delta}') + D\boldsymbol{f}|_{\boldsymbol{\nu}}^{j+1}(\boldsymbol{\delta}) && \text{(induction)} \\
&= \underbrace{\boldsymbol{\nu} + D\boldsymbol{f}|_{\boldsymbol{\nu}}^{\leq j}(\boldsymbol{\delta}')}_{=:\boldsymbol{u}} + \sum_{s \in \mathbb{N}_+^{j+1}} \$_s \boldsymbol{f}|_{\boldsymbol{\nu}}(\boldsymbol{\delta}) && \text{(Prop. 3.16.2.)}
\end{aligned}$$

By Lemma 3.18, we have $\boldsymbol{u} \sqsupseteq \$_s \boldsymbol{f}|_{\boldsymbol{\nu}}(\boldsymbol{\nu})$ for all $s \in \mathbb{N}_+^{j+1}$. In other words, for all $s \in \mathbb{N}_+^{j+1}$ there is a $\boldsymbol{u}'$ such that $\boldsymbol{u} = \boldsymbol{u}' + \$_s \boldsymbol{f}|_{\boldsymbol{\nu}}(\boldsymbol{\nu})$. Hence, for all $s \in \mathbb{N}_+^{j+1}$, we have $\boldsymbol{u} + \$_s \boldsymbol{f}|_{\boldsymbol{\nu}}(\boldsymbol{\delta}) = \boldsymbol{u}' + \$_s \boldsymbol{f}|_{\boldsymbol{\nu}}(\boldsymbol{\nu}) + \$_s \boldsymbol{f}|_{\boldsymbol{\nu}}(\boldsymbol{\delta}) = \boldsymbol{u}' + \$_s \boldsymbol{f}|_{\boldsymbol{\nu}}(\boldsymbol{f}(\boldsymbol{\nu})) = \boldsymbol{u} + \$_s \boldsymbol{f}|_{\boldsymbol{\nu}}(\boldsymbol{\delta}')$. Therefore, in the above equation, we can replace $\boldsymbol{\delta}$ by $\boldsymbol{\delta}'$ due to the "presence" of $\boldsymbol{u}$:

$$
\begin{aligned}
&= \boldsymbol{\nu} + D\boldsymbol{f}|_{\boldsymbol{\nu}}^{\leq j}(\boldsymbol{\delta}') + \sum_{s \in \mathbb{N}_+^{j+1}} \$_s \boldsymbol{f}|_{\boldsymbol{\nu}}(\boldsymbol{\delta}') && \text{(as argued above)} \\
&= \boldsymbol{\nu} + D\boldsymbol{f}|_{\boldsymbol{\nu}}^{\leq j}(\boldsymbol{\delta}') + D\boldsymbol{f}|_{\boldsymbol{\nu}}^{j+1}(\boldsymbol{\delta}') && \text{(Prop. 3.16.2.)} \\
&= \boldsymbol{\nu} + D\boldsymbol{f}|_{\boldsymbol{\nu}}^{\leq j+1}(\boldsymbol{\delta}') && \square
\end{aligned}
$$

Now Proposition 3.14 follows immediately from Lemma 3.19 by a straightforward inductive proof. □

4 Derivation Trees and the Newton Approximants

In this section we reinterpret a system of power-series as a context-free grammar, and assign it a set of *derivation trees*. We then characterize the Kleene and Newton approximants of the system in terms of subsets of this set of trees.

We assume that the reader is familiar with the notion of derivation tree of a context-free grammar. Recall that the yield of a derivation tree (obtained by reading the leaves from left to right) is a word generated by the grammar, and every word generated by the grammar is the yield of one or more derivation trees. In our reinterpretation the non-terminals will be the variables of the system of power series, and the terminals will be its coefficients.

We show that the Kleene approximants $\boldsymbol{\kappa}^{(i)}$ are equal to the sum of the yields of the derivation trees having a certain height. Similarly, we show that the Newton approximants $\boldsymbol{\nu}^{(i)}$ are equal to the sum of the yields of the trees having a certain *dimension*, a notion introduced in Definition 4.6 below.

For the rest of the section we fix a vector $\boldsymbol{f}$ of power series over a fixed but arbitrary ω-continuous semiring. Without loss of generality, we assume that $\boldsymbol{f}_X = \sum_{j \in J} m_{X,j}$ holds for every variable $X \in \mathcal{X}$, i.e., we assume that for all variables the sum is over the same countable set J of indices.

Consider the set of ordered trees whose nodes are labelled by pairs (X, j), where $X \in \mathcal{X}$ and $j \in J$. Sometimes we identify a tree and its root. In particular, we say that a tree t is labelled by (X, j) if its root is labelled by (X, j). The mappings λ, λ_v and λ_m are defined by $\lambda(t) := (X, j)$, $\lambda_v(t) := X$, and $\lambda_m(t) := j$. Given a set T of trees, we denote by T_X the set of trees $t \in T$ such that $\lambda_v(t) = X$.

We define the set of derivation trees of $\boldsymbol{f}$, and show how to assign to each tree a semiring element called the yield of the tree. For technical reasons our definition differs slightly from the straightforward generalization of derivation trees for grammars.

Definition 4.1 ((derivation tree, yield)). *The* derivation trees *of* $\boldsymbol{f}$ *and their* yields *are inductively defined as follows:*

- *For every monomial $m_{X,j}$ of $\boldsymbol{f}_X$, if no variable occurs in $m_{X,j}$, then the tree t consisting of one single node labelled by (X, j) is a derivation tree of $\boldsymbol{f}$. Its yield $Y(t)$ is equal to $m_{X,j}$.*
- *Let $m_{X,j} = a_1 X_1 a_2 X_2 \ldots a_k X_k a_{k+1}$ for some $k \geq 1$, and let $t_1, \ldots, t_k$ be derivation trees of $\boldsymbol{f}$ such that $\lambda_v(t_i) = X_i$ for $1 \leq i \leq k$. Then the tree t labelled by (X, j) and having $t_1, \ldots, t_k$ as (ordered) children is also a derivation tree of $\boldsymbol{f}$, and its yield $Y(t)$ is equal to $a_1 Y(t_1) \ldots a_k Y(t_k) a_{k+1}$.*

The yield $Y(T)$ *of a countable set T of derivation trees is defined by $Y(T) = \sum_{t \in T} Y(t)$. In the following, we mean* derivation tree *whenever we say* tree.

Figure 3 shows a system of equations (system (1) from the introduction, on the left) and a derivation tree (in the middle). Consider the node labelled by $(Y, 1)$ (the right child of the root). Since the first monomial of the equation for Y is cYZ, the node has two children, say c_1, c_2 with $\lambda_v(c_1) = Y$ and $\lambda_v(c_2) = Z$. As $\lambda_m(c_2) = 2$, the children of c_2 are determined by the second monomial of the equation for Z. Since this monomial is h, which contains no variables, c_2 has no children. The right part of the figure shows the result of labelling each node of the tree with the yield of the subtree rooted at it.

$$
\begin{aligned}
X &= aXY + b \\
Y &= cYZ + dYX + e \\
Z &= gXh + i
\end{aligned}
$$

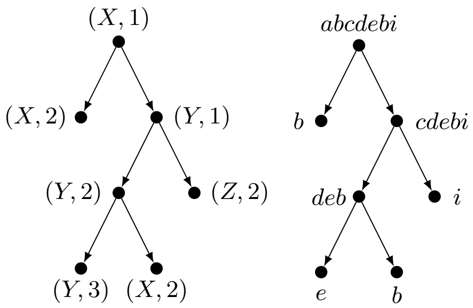


Fig. 3. A system of equations, a derivation tree, and its yield

4.1 Kleene Sequence and Height

As a warm-up for the Newton case, we characterize the Kleene sequence $(\boldsymbol{\kappa}^{(i)})_{i\in\mathbb{N}}$ in terms of the derivation trees of a certain height.

Definition 4.2 ((height)). *Let t be a derivation tree. The* height *of t, denoted by $h(t)$, is the length (number of edges)of a longest path from the root to some leaf. We denote by $\mathcal{H}^i$ the set of derivation trees of height at most i.*

Proposition 4.3. $\left(\boldsymbol{\kappa}^{(i)}\right)_X = Y(\mathcal{H}^i_X)$*, i.e., the X-component of the i-th Kleene approximant $\boldsymbol{\kappa}^{(i)}$ is equal to the yield of $\mathcal{H}^i_X$.*

The proof can be found in Appendix A.

Notice that Proposition 4.3 no longer holds if nodes are only labelled with a variable, and not with a pair. Consider for instance the equation $X = a + a$, for which $\kappa^{(0)} = a + a$. There are two derivation trees t_1, t_2 of height 0, both consisting of one single node: t_1 is labelled by $(X, 1)$, and t_2 by $(X, 2)$. We get $Y(t_1) + Y(t_2) = a + a = \kappa^{(0)}$. If we labelled nodes only with variables, then there would be one single derivation tree t, and we would get $Y(t) = a$, which in general is different from $a + a$.

Example 4.4. Consider again the equation $X = 1/2 \cdot X^2 + 1/2$ over the real semiring. We have $\kappa^{(2)} = 89/128$. Figure 4 shows the five derivation trees of height at most 2. It is easy to see that their yields are $1/2, 1/8, 1/32, 1/32, 1/128$, which add up to $89/128$.

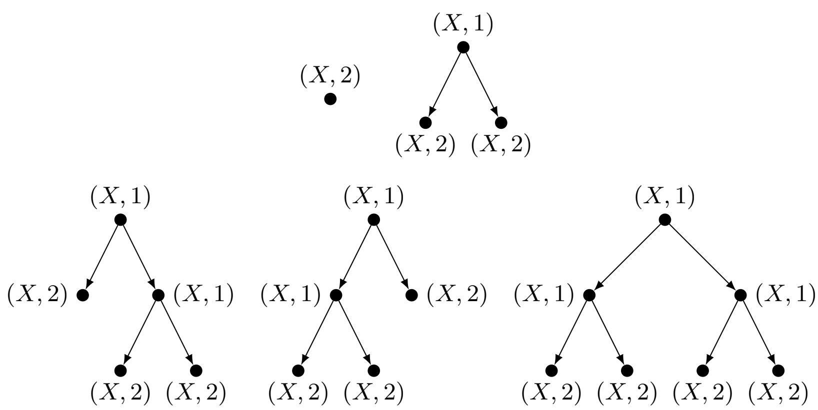


Fig. 4. Trees of height at most 2 for the equation $X = 1/2 \cdot X^2 + 1/2$.

By Kleene's theorem we obtain that the least solution of the equation system is equal to the yield of the set of all trees.

Corollary 4.5. *Let $\mathcal{T}$ be the set of all derivation trees of $\boldsymbol{f}$. For all $X \in \mathcal{X}$: $(\mu \boldsymbol{f})_X = Y(\mathcal{T}_X)$.*

Proof. By Kleene's Theorem (Proposition 2.4) we have $(\mu \boldsymbol{f})_X = \sup_{i\in\mathbb{N}}(\boldsymbol{\kappa}^{(i)})_X$. The result follows from Proposition 4.3. □

4.2 Newton Sequence and Dimension

We introduce a second parameter of a tree, namely its *dimension*. Like the height, it depends only on the tree structure, and not on the labels of its nodes. Loosely speaking, a tree has dimension 0 if it consists of just one node; a tree has dimension i if there is a path from its root to some node which has at least 2 children with dimension $i-1$ and all subtrees of the path that are not themselves on the path have dimension at most $i-1$. The path is called the *backbone* of the tree. Figure 5 illustrates this idea.

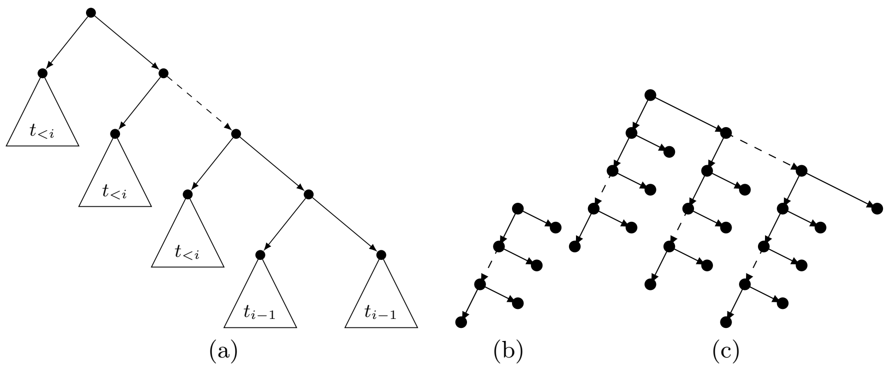


Fig. 5. (a) shows the general structure of a tree of dimension i, where $t_{<i}$ (resp. t_{i-1}) represents any tree of dimension $< i$ (resp. $= i-1$). (b) and (c) give some idea of the topology of one-, resp. two-dimensional trees.

Formally, we use an inductive definition of dimension that is more convenient for proofs.

Definition 4.6 ((dimension)). *The* dimension $d(t)$ *of a tree t is inductively defined as follows:*

1. *If t has no children, then $d(t) = 0$.*
2. *If t has exactly one child t_1, then $d(t) = d(t_1)$.*
3. *If t has at least two children, let t_1, t_2 be two distinct children of t such that $d(t_1) \geq d(t_2)$ and $d(t_2) \geq d(t')$ for every child $t' \neq t_1$. Let $d_1 = d(t_1)$ and $d_2 = d(t_2)$. Then*

$$d(t) = \begin{cases} d_1 + 1 & \text{if } d_1 = d_2 \\ d_1 & \text{if } d_1 > d_2. \end{cases}$$

We denote by $\mathcal{D}^i$ the set of derivation trees of dimension at most i.

Remark: It is easy to prove by induction that $h(t) \geq d(t)$ holds for every derivation tree t.

In the rest of the section we show that the i-th Newton approximant $\boldsymbol{\nu}^{(i)}$ is equal to the yield of the derivation trees of dimension at most i:

Theorem 4.7 (Tree Characterization of the Newton Sequence). *Let $(\boldsymbol{\nu}^{(i)})_{i\in\mathbb{N}}$ be the Newton sequence of $\boldsymbol{f}$. For every $X \in \mathcal{X}$ and every $i \geq 0$ we have $\left(\boldsymbol{\nu}^{(i)}\right)_X = Y(\mathcal{D}^i_X)$, i.e., the X-component of the i-th Newton approximant is equal to the yield of $\mathcal{D}^i_X$.*

The proof is as follows. We define, in terms of trees, a sequence $(\boldsymbol{\tau}^{(i)})_{i\in\mathbb{N}}$ satisfying $\boldsymbol{\tau}^{(i)}_X = Y(\mathcal{D}^i_X)$ (Lemma 4.9), and we prove that it is a Newton sequence (Lemma 4.10). As the Newton sequence is unique by Proposition 3.14, we have $\boldsymbol{\tau}^{(i)} = \boldsymbol{\nu}^{(i)}$ and Theorem 4.7 follows.

We need the following definition.

Definition 4.8. *A tree t is* proper *if $d(t) > d(t')$ for every child t' of t. For every $i \geq 0$, let P^i be the set of proper trees of dimension i. Define the sequence $(\boldsymbol{\tau}^{(i)})_{i\in\mathbb{N}}$ as follows:*

$$\begin{aligned}\boldsymbol{\tau}^{(0)} &= \boldsymbol{f}(\mathbf{0})\\ \boldsymbol{\tau}^{(i+1)} &= \boldsymbol{\tau}^{(i)} + D\boldsymbol{f}|^*_{\boldsymbol{\tau}^{(i)}}(\boldsymbol{\delta}^{(i)})\,,\end{aligned}$$

where $\boldsymbol{\delta}^{(i)}_X = Y(P^{i+1}_X)$ for all $X \in \mathcal{X}$.

Lemma 4.9. *For every variable $X \in \mathcal{X}$ and every $i \geq 0$: $\boldsymbol{\tau}^{(i)}_X = Y(\mathcal{D}^i_X)$.*

Lemma 4.10. *The sequence $(\boldsymbol{\tau}^{(i)})_{i\in\mathbb{N}}$ is a Newton sequence as defined in Definition 3.5, i.e., the $\boldsymbol{\delta}^{(i)}$ of Definition 4.8 satisfy $\boldsymbol{f}(\boldsymbol{\tau}^{(i)}) = \boldsymbol{\tau}^{(i)} + \boldsymbol{\delta}^{(i)}$.*

The proofs of Lemma 4.9 and Lemma 4.10 can be found in Appendix A.

Example 4.11. Let us recall our example from the introduction (cf. Fig. 1) with the equations

$$\begin{aligned}X &= a \cdot X \cdot Y + b\\ Y &= c \cdot Y \cdot Z + d \cdot Y \cdot X + e\\ Z &= g \cdot X \cdot h + i.\end{aligned}$$

Using our characterizations of $\boldsymbol{\kappa}^{(i)}$ and $\boldsymbol{\nu}^{(i)}$ by means of derivation trees we see that (a) every derivation tree t represents a terminating run of the procedure $\lambda(t)$, and, thus, (b) while $\boldsymbol{\kappa}^{(i)}$ only corresponds to a finite set of trees (runs), for $i > 0$ every $\boldsymbol{\nu}^{(i)}$ corresponds to an infinite set of runs. Hence, it is not very surprising that in general the Newton approximants give a better approximation of the (abstract) semantics of a program than the Kleene approximants. □

5 Idempotent Semirings

In this and the next section we focus on ω-continuous semirings whose summation operator is idempotent. Such semirings are called *idempotent* ω-continuous semirings, or just idempotent semirings. In idempotent semirings, the natural order can be characterized as follows: $a \sqsubseteq b$ holds if and only if $a+b = b$. This is because $a \sqsubseteq b$ means by definition that there is a c such that $a + c = b$. Then we have $a + b = a + a + c = a + c = b$. This extends analogously to vectors.

The following proposition shows that the definition of the Newton sequence $(\boldsymbol{\nu}^{(i)})_{i\in\mathbb{N}}$ can be simplified in the idempotent case.

Proposition 5.1. *Let $\boldsymbol{f}$ be a vector of power series over an idempotent semiring. Let $(\boldsymbol{\nu}^{(i)})_{i\in\mathbb{N}}$ denote the Newton sequence of $\boldsymbol{f}$. It satisfies the following equations for all $i \in \mathbb{N}$:*

(a) $\boldsymbol{\nu}^{(i+1)} = D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{f}(\boldsymbol{\nu}^{(i)}))$
(b) $\boldsymbol{\nu}^{(i+1)} = D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{\nu}^{(i)})$
(c) $\boldsymbol{\nu}^{(i+1)} = D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{f}(\mathbf{0}))$

Proof. We first show (a). By Theorem 3.8 we have $\boldsymbol{\nu}^{(i)} \sqsubseteq \boldsymbol{f}(\boldsymbol{\nu}^{(i)})$, hence with idempotence $\boldsymbol{\nu}^{(i)} + \boldsymbol{f}(\boldsymbol{\nu}^{(i)}) = \boldsymbol{f}(\boldsymbol{\nu}^{(i)})$. So we can choose $\boldsymbol{\delta}^{(i)} = \boldsymbol{f}(\boldsymbol{\nu}^{(i)})$ and have $\boldsymbol{\nu}^{(i+1)} = \boldsymbol{\nu}^{(i)} + D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{f}(\boldsymbol{\nu}^{(i)})) = D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{f}(\boldsymbol{\nu}^{(i)}))$, because $\boldsymbol{\nu}^{(i)} \sqsubseteq \boldsymbol{f}(\boldsymbol{\nu}^{(i)}) \sqsubseteq D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{f}(\boldsymbol{\nu}^{(i)}))$. So (a) is shown.

Again by Theorem 3.8 we have $\boldsymbol{f}(\mathbf{0}) = \boldsymbol{\nu}^{(0)} \sqsubseteq \boldsymbol{\nu}^{(i)} \sqsubseteq \boldsymbol{f}(\boldsymbol{\nu}^{(i)})$. So we have $D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{f}(\mathbf{0})) \sqsubseteq D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{\nu}^{(i)}) \sqsubseteq D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{f}(\boldsymbol{\nu}^{(i)}))$. Hence, for (b) and (c), it remains to show $D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{f}(\boldsymbol{\nu}^{(i)})) \sqsubseteq D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{\nu}^{(i)})$ and $D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{\nu}^{(i)}) \sqsubseteq D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{f}(\mathbf{0}))$, respectively. For (b) we have:

$$D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{f}(\boldsymbol{\nu}^{(i)}))$$

$$
\begin{aligned}
&\sqsubseteq D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{f}(\mathbf{0}) + D\boldsymbol{f}|_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{\nu}^{(i)})) && \text{(Lemma 3.12)}\\
&= D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{f}(\mathbf{0})) + D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(D\boldsymbol{f}|_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{\nu}^{(i)})) && \\
&\sqsubseteq D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{\nu}^{(i)}) + D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(D\boldsymbol{f}|_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{\nu}^{(i)})) && (\boldsymbol{f}(\mathbf{0}) \sqsubseteq \boldsymbol{\nu}^{(i)})\\
&\sqsubseteq D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{\nu}^{(i)}) + D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{\nu}^{(i)}) && \text{(Lemma 3.11)}\\
&= D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{\nu}^{(i)}) && \text{(idempotence)}
\end{aligned}
$$

So (b) is shown.

For (c) it remains to show $D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{\nu}^{(i)}) \sqsubseteq D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{f}(\mathbf{0}))$. We proceed by induction on i. The base case $i = 0$ is easy because $\boldsymbol{\nu}^{(0)} = \boldsymbol{f}(\mathbf{0})$. Let $i \geq 1$. We have:

$$
\begin{aligned}
&D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{\nu}^{(i)}) && \\
&= D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i-1)}}(\boldsymbol{\nu}^{(i-1)})) && \text{(by (b))}\\
&\sqsubseteq D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i-1)}}(\boldsymbol{f}(\mathbf{0}))) && \text{(by induction)}\\
&\sqsubseteq D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{f}(\mathbf{0}))) && \text{(Theorem 3.8: } \boldsymbol{\nu}^{(i-1)} \sqsubseteq \boldsymbol{\nu}^{(i)})\\
&= D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{f}(\mathbf{0})) && \text{(see explanation below)}
\end{aligned}
$$

For the last step we used that in the idempotent case we have $\boldsymbol{g}^*(\boldsymbol{g}^*(\boldsymbol{x})) = \boldsymbol{g}^*(\boldsymbol{x})$ for any linear map $\boldsymbol{g} : V \to V$. Recall that Remark 3.6 states that $D\boldsymbol{f}|_{\boldsymbol{\nu}^{(i)}}$ is linear.

$$
\begin{aligned}
\boldsymbol{g}^*(\boldsymbol{g}^*(\boldsymbol{x})) &= \sum_{j\in\mathbb{N}} \boldsymbol{g}^j\left(\sum_{k\in\mathbb{N}} \boldsymbol{g}^k(\boldsymbol{x})\right) && \text{(Definition 3.10)}\\
&= \sum_{j\in\mathbb{N}}\sum_{k\in\mathbb{N}} \boldsymbol{g}^j(\boldsymbol{g}^k(\boldsymbol{x})) && \text{(linearity)}\\
&= \sum_{l\in\mathbb{N}} \boldsymbol{g}^l(\boldsymbol{x}) && \text{(idempotence)}\\
&= \boldsymbol{g}^*(\boldsymbol{x}) && \text{(Definition 3.10)}
\end{aligned}
$$

This concludes the proof. □

5.1 Language Semirings

Now we consider language semirings, the typical example of idempotent semirings. Let $\mathcal{S}_\Sigma$ be the language semiring over a finite alphabet Σ. Let $\boldsymbol{f}$ be a vector of polynomials over $\mathcal{X}$ whose coefficients are elements of Σ. Then, for each $X_0 \in \mathcal{X}$, there is a naturally associated context-free grammar $G_{\boldsymbol{f},X_0} = (\mathcal{X}, \Sigma, P, X_0)$, where the set of productions is $P = \{(X \to m_{X,j}) \mid m_{X,j} \text{ is a monomial of } \boldsymbol{f}_X\}$. It is well-known that $L(G_{\boldsymbol{f},X_0}) = \big(\mu\boldsymbol{f}\big)_{X_0}$ (see e.g. [Kui97]). Analogously, each grammar is naturally associated with a vector of polynomials. In the following we use grammars and vectors of polynomials interchangeably.

We show in this section that the Newton approximants $\boldsymbol{\nu}^{(i)}$ are strongly linked with the *finite-index* approximations of $L(G)$. Finite-index languages have been extensively investigated under different names by Salomaa, Gruska, Yntema, Ginsburg and Spanier, among others [Sal69,Gru71,Ynt67,GS68] (see [FH97] for historical background).

Definition 5.2. *Let G be a grammar, and let D be a derivation $X_0 = \alpha_0 \Rightarrow \cdots \Rightarrow \alpha_r = w$ of $w \in L(G)$, and for every $i \in \{0, \ldots, r\}$ let β_i be the projection of α_i onto the variables of G. The* index *of D is the maximum of $\{|\beta_0|, \ldots, |\beta_r|\}$. The* index-$i$ approximation *of $L(G)$, denoted by $L_i(G)$, contains the words derivable by some derivation of G of index at most i.*

We show that for a context-free grammar G in Chomsky normal form (CNF), the Newton approximations to $L(G)$ coincide with the finite-index approximations.

Theorem 5.3. *Let $G = (\mathcal{X}, \Sigma, P, X_0)$ be a context-free grammar in CNF and let $(\boldsymbol{\nu}^{(i)})_{i\in\mathbb{N}}$ be the Newton sequence associated with G. Then $\big(\boldsymbol{\nu}^{(i)}\big)_{X_0} = L_{i+1}(G)$ for every $i \geq 0$.*

Proof sketch (full proof in Appendix B). The proof builds on the tree-dimension characterization of the Newton approximants (Theorem 4.7). It can be shown that a tree of dimension i can be "flattened" to a derivation of index at most $i+1$. For the other direction it can be similarly shown that a derivation of index $i+1$ corresponds to a derivation tree of dimension at most i. $\square$

In particular, it follows from Theorem 5.3 that the (X_0-component of the) Newton sequence for a context-free grammar G converges in finitely many steps if and only if $L(G) = L_i(G)$ for some $i \in \mathbb{N}$.

6 Commutative Idempotent Semirings

In this section we study Newton's method in the case where the ω-continuous semiring does not only have an idempotent addition (as in the previous section), but also a *commutative* multiplication. We will use the abbreviation *ci-semirings* for such semirings in the following. Commutative language semirings are a prominent example of ci-semirings.

An instance of the Newton sequence in a ci-semiring has already been presented in the counting semiring example on page 11. We show another one here.

Example 6.1. Let $\langle 2^{\{a\}^*}, +, \cdot, 0, 1\rangle$ denote the ci-semiring $\langle 2^{\{a\}^*}, \cup, \cdot, \emptyset, \{\varepsilon\}\rangle$. The multiplication $\cdot$ is meant to be commutative. For simplicity, we write a^i instead of $\{a^i\}$. Consider $\boldsymbol{f}(X_1, X_2) = (X_2^2 + a,\ X_1^2)$. We have:

$$D\boldsymbol{f}|_{(v_1,v_2)}(X_1, X_2) = \big(v_2 X_2,\ v_1 X_1\big)$$

and

$$D\boldsymbol{f}|^*_{(v_1,v_2)}(X_1, X_2) = (v_1 v_2)^* \big(X_1 + v_2 X_2,\ v_1 X_1 + X_2\big)\ .$$

The first three elements of the Newton sequence are:

$$\boldsymbol{\nu}^{(0)} = (a, 0), \quad \boldsymbol{\nu}^{(1)} = (a, a^2), \quad \boldsymbol{\nu}^{(2)} = (a^3)^*(a, a^2)\ .$$

It is easy to check that $\boldsymbol{\nu}^{(2)}$ is a fixed point of $\boldsymbol{f}$. Hence we have $\boldsymbol{\nu}^{(2)} = \mu\boldsymbol{f}$, as $\boldsymbol{\nu}^{(2)} \sqsubseteq \mu\boldsymbol{f}$ by Theorem 3.8. $\square$

In the previous section we have seen that, even though the Newton sequence accelerates the Kleene sequence, it does not generally converge in finitely many steps: The language semirings are examples of idempotent ω-continuous semirings, but the Newton sequence of a context-free grammar G with start symbol X_0 does not reach $(\mu\boldsymbol{f})_{X_0} = L(G)$ after finitely many steps, unless $L(G)$ coincides with some finite-index approximation $L_i(G)$.

In the case of ci-semirings the behaviours of the Kleene and Newton sequence differ very much: while the Kleene sequence may still need infinitely many steps, the Newton sequence always reaches $\mu\boldsymbol{f}$ after finitely many. This was first shown by Hopkins and Kozen in 6.2. Hopkins and Kozen defined the sequence $(\boldsymbol{\nu}^{(i)})_{i\in\mathbb{N}}$ directly through the equations $\boldsymbol{\nu}^{(0)} = \boldsymbol{f}(\boldsymbol{0})$ and $\boldsymbol{\nu}^{(i+1)} = D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{\nu}^{(i)})$ from Proposition 5.1 (b), without noticing the connection to Newton's method (which is not surprising, since in the idempotent case the original equations get masked). They proved the following result, which gives a $O(3^n)$ upper bound for the number of Newton iterations required for a system of n equations:

Theorem 6.2 ([HK99]). *Let $\boldsymbol{f}$ be a vector of power series over a ci-semiring and a set $\mathcal{X}$ of variables with $|\mathcal{X}| = n$. There is a function $P : \mathbb{N} \to \mathbb{N}$ with $P(n) \in \mathcal{O}(3^n)$ such that $\boldsymbol{\nu}^{(P(n))} = \mu\boldsymbol{f}$.*

In Section 6.1 we improve Theorem 6.2 by showing that it holds with $P(n) = n$. This is achieved through our characterisation of the Newton approximants in terms of derivation trees. In Section 6.2 we generalize our result to commutative Kleene algebras, thereby improving the result of [HK99] which was not stated in terms of ci-semirings as in Theorem 6.2, but in terms of commutative Kleene algebras whose axioms are weaker.

6.1 Analysis of the Convergence Speed

We analyze how many steps the Newton iteration and, equivalently, the Hopkins-Kozen iteration need to reach $\mu\boldsymbol{f}$ when we consider ci-semirings.

Recall from Section 4 the concept of derivation trees (short: trees). A tree t has a height $h(t)$, a dimension $d(t)$, and a yield $Y(t)$. We define yet another tree property.

Definition 6.3. *A tree t is* compact *if $d(t) \leq L(t)$, where $L(t)$ denotes the number of distinct λ_v-labels in t.*

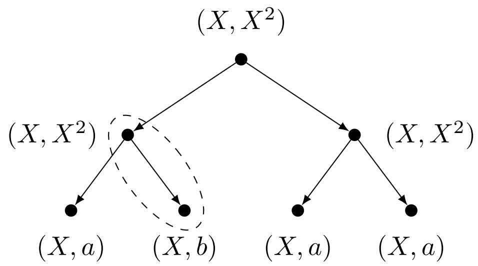
Now we are ready to prove the key lemma of this section, which states that any tree can be made compact.

Lemma 6.4. *For each tree t there is a compact tree t' with $\lambda_v(t) = \lambda_v(t')$ and $Y(t) = Y(t')$.*

Example 6.5. We first sketch the proof of the lemma by means of an example. Consider the following univariate polynomial equation system:

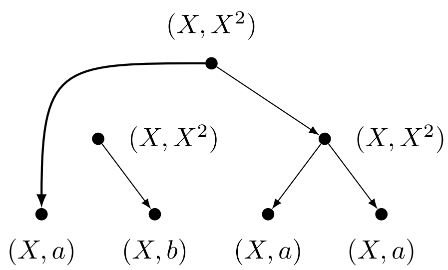
$$X = f(X) := X^2 + a + b.$$

Consider now the following tree $t \in \mathcal{T}_X$.[6]



This tree has dimension 2 and is therefore not compact by definition. In order to make it compact, we have to transform it into a derivation tree of $\boldsymbol{f}$ which is of dimension 1 without changing its yield nor the variable-label of the root.

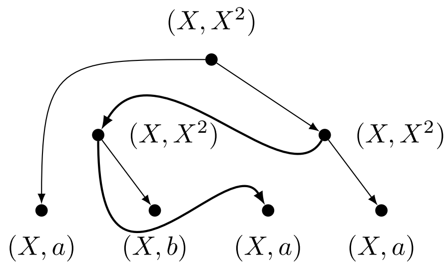
The idea is to reduce the left subtree to a tree of dimension 0 by reallocating "pump trees" (encircled in the above figure) into the right subtree; after that, we deal recursively with the right subtree.[7] We first remove such a pump tree from the rest of the tree by deleting the connecting edges and connecting the remaining parts as depicted here:



Note that we can introduce the new edge because the roots of the pump tree and the remaining subtree, in our example the left-most leaf, are labeled by the same variable. Next, we reallocate the detached pump tree into the right subtree, e.g. as shown here:

[6] To improve readability in the following illustrations, we replace the node labels $(X, 1)$, $(X, 2)$, $(X, 3)$ by (X, X^2), (X, a), (X, b), respectively.

[7] Here, with "pump tree" we refer to partial derivation trees one adds or removes in the proof of the pumping lemma for context-free grammars.



It is easy to check that this new tree is indeed a derivation tree of $\boldsymbol{f}$, and has the same yield as the original one. Further this tree is already compact. In general, we would have to proceed recursively in order to make the right subtree compact.

Note that, as we assume multiplication to be commutative, it is not important where we insert the pump tree into the right subtree. In the following proof we show that we can always find such pump trees and relocate them, i.e. find insertion points, if the tree under consideration is not compact. □

We now give a formal proof of Lemma 6.4:

Proof. We write $t = t_1 \cdot t_2$ to denote that t is combined from t_1 and t_2 in the following way: The tree t_1 is a "partial" derivation tree, i.e., a regular derivation tree except for one leaf l missing its children. The tree t_2 is a derivation tree with $\lambda_v(t_2) = \lambda_v(l)$. The tree t is obtained from t_1 and t_2 by replacing the leaf l of t_1 by the tree t_2.

We proceed by induction on the number of nodes. In the base case, t has just one node, so $d(t) = 0$, hence t is compact, and we are done. In the following, assume that t has more than one node and $d(t) > L(t)$ holds. We show how to construct a compact tree from t.

Let w.l.o.g. $s_1, s_2, \ldots, s_r$ be the children of t with $d(t) \geq d(s_1) \geq d(s_2) \geq \ldots \geq d(s_r)$. By induction we can make every child compact, i.e. $d(s_i) \leq L(s_i)$. We then have by definition of dimension

$$L(t) + 1 \leq d(t) \leq d(s_1) + 1 \leq L(s_1) + 1 \leq L(t) + 1.$$

Hence, we have $d(t) = d(s_1) + 1$ which, by definition of dimension and compactness, implies $d(s_1) = d(s_2) = L(t) = L(s_1) = L(s_2)$. As $h(s_2) \geq d(s_2) = L(s_2)$ by the remark after Definition 4.6, we find a path in s_2 from the root to a leaf which passes through at least two nodes with the same λ_v-label, say X_j. In other words, we may factor s_2 into $t_1^b \cdot (t_2^b \cdot t_3^b)$ such that $\lambda_v(t_2^b) = \lambda_v(t_3^b) = X_j$. As $L(t) = L(s_1) = L(s_2)$, we also find a node of s_1 labelled by X_j which allows us to write $s_1 = t_1^a \cdot t_2^a$ with $\lambda_v(t_2^a) = X_j$.

Now we move the middle part of s_2 to s_1, i.e., let $s_1' = t_1^a \cdot (t_2^b \cdot t_3^a)$ and let $s_2' = t_1^b \cdot t_3^b$. We then have $L(s_1') = L(s_1) = L(s_2) \geq L(s_2')$. By induction, s_1' and s_2' can be made compact, so $d(s_1') \leq d(s_1) = d(s_2) \geq d(s_2')$. Consider the tree t' obtained from t by replacing s_1 by s_1' and s_2 by s_2'. By commutativity, t and t' have the same yield. If $d(s_2') < d(s_2)$ then $d(t') \leq d(t) - 1 = L(t) = L(t')$ and we are done. Otherwise we iterate the described procedure.

This procedure terminates, because the number of nodes of (the current) s_2 strictly decreases in every iteration, and the number of nodes is an upper bound for $h(s_2)$ and, therefore, for $d(s_2)$. □

Now we can prove the main theorem of this section.

Theorem 6.6. *Let $\boldsymbol{f}$ be a vector of power series over a ci-semiring $\mathcal{S}$ given in the set $\mathcal{X}$ of variables with $|\mathcal{X}| = n$. Then $\boldsymbol{\nu}^{(n)} = \mu \boldsymbol{f}$.*

Proof. We have for all $X \in \mathcal{X}$:

$$\begin{aligned}
(\mu \boldsymbol{f})_X &= \sum_{\text{trees } t \text{ with } \lambda_v(t)=X} Y(t) && \text{(Corollary 4.5)} \\
&= \sum_{\substack{\text{trees } t \text{ with } \lambda_v(t)=X \\ \text{and } d(t)\leq n}} Y(t) && \text{(Lemma 6.4)} \\
&= (\boldsymbol{\nu}^{(n)})_X && \text{(Theorem 4.7)}
\end{aligned}$$

□

Remark 6.7. The bound of this theorem is tight, as shown by the following example: If $\boldsymbol{f}(X_1,\dots,X_n) = (X_2^2 + a, X_3^2, \dots, X_n^2, X_1^2)$, then $(\boldsymbol{\nu}^{(k)})_{X_1} = a$ for $k < n$, but $a^{2^n} \le (\boldsymbol{\nu}^{(n)})_{X_1} = (\mu \boldsymbol{f})_{X_1}$.

In terms of languages, Theorem 6.2 can be understood in the following way, using Theorem 5.3.

Corollary 6.8. *Let $G = (\mathcal{X}, \Sigma, P, X_0)$ be a context-free grammar in CNF. Let $|\mathcal{X}| = n$. Then the commutative image of the index-$(n+1)$ approximation $L_{n+1}(G)$ is equal to the commutative image of $L(G)$.*

6.2 Generalization to Commutative Kleene Algebras

In this subsection we generalize Theorem 6.6 to commutative Kleene algebras. A *commutative Kleene algebra* $\langle K, +, \cdot, ^*, 0, 1\rangle$ is an idempotent commutative semiring $\langle K, +, \cdot, 0, 1\rangle$ where the *-operator is only required to satisfy these two equations for all $a, b, c \in K$:

$$1 + aa^* \le a^* \;\text{ and }\; a + bc \le c \to b^*a \le c\,.$$

Notice that for a Kleene algebra there may not exist a notion of countable summation, as the *-operator is defined axiomatically. Thus, the axioms of commutative Kleene algebras are weaker than those of ci-semirings. In particular, the following example from [Koz90] shows there are commutative Kleene algebras which are not ci-semirings:

Example 6.9. Consider the Kleene algebra with carrier $\omega^2 := \mathbb{N}^2 \cup \{\bot, \top\}$, i.e. the set of ordered pairs of natural numbers extended by a bottom and a top element. We assume that ω^2 is totally ordered by $\prec$ with $\bot$ the minimum element, $\top$ the maximum element, and the lexicographic order on $\mathbb{N}^2$. Addition is defined to be the supremum of the elements w.r.t. $\prec$. Thus the additive neutral element is $\bot$. Note that this also gives us a notion of countable summation on ω^2. Multiplication is defined by

$$\begin{array}{l} x \cdot \bot = \bot \cdot x = \bot \\ x \cdot \top = \top \cdot x = \top \quad (x \ne \bot) \\ (a,b)\cdot(c,d) = (a+c, b+d) \end{array}$$

with neutral element $(0,0)$. Finally, the Kleene-star is defined by

$$a^* = \begin{cases} \bot & \text{if } a = \bot \vee a = (0,0) \\ \top & \text{else.} \end{cases}$$

This definition satisfies the axioms stated above. But obviously, we do not have a ci-semiring as

$$\sum_{i \in \mathbb{N}} (0,1)^n = \sup\{(0,1)^i | i \in \mathbb{N}\} = (1,0) \prec \top = (0,1)^*.$$

□

Notation 2. *Let M be any set. Then RExp_M denotes the set of* regular expressions *generated by the elements of M. We write $R_M : \mathsf{RExp}_M \to 2^{M^*}$ for their* canonical interpretation *as languages.*

In the rest of the section we prove the following theorem which improves the result of [HK99] from $\mathcal{O}(3^n)$ to n.

Theorem 6.10. *Let $\boldsymbol{f} \in \mathsf{RExp}_{K\cup\mathcal{X}}^{\mathcal{X}}$ be a vector of regular expressions over a commutative Kleene algebra $\langle K, +, \cdot, ^*, 0, 1\rangle$. Let $|\mathcal{X}| = n$. Then $\boldsymbol{\nu}^{(n)} = \mu \boldsymbol{f}$.*

We have not yet defined $\boldsymbol{\nu}^{(i)}$ over a commutative Kleene algebra. We take the equations $\boldsymbol{\nu}^{(0)} = \boldsymbol{f}(\mathbf{0})$ and $\boldsymbol{\nu}^{(i+1)} = D\boldsymbol{f}|^*_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{\nu}^{(i)})$ (cf. Proposition 5.1) as definition. For convenience, we define the *Hopkins-Kozen operator* $\mathsf{H}_{\boldsymbol{f}}$ by

$$\mathsf{H}_{\boldsymbol{f}}(\boldsymbol{X}) = D\boldsymbol{f}|^*_{\boldsymbol{X}}(\boldsymbol{X}) \ .$$

Then $\boldsymbol{\nu}^{(i)}$ is obtained by i times applying $\mathsf{H}_{\boldsymbol{f}}$ to $\boldsymbol{f}(\mathbf{0})$:

$$\boldsymbol{\nu}^{(i)} = \mathsf{H}^i_{\boldsymbol{f}}(\boldsymbol{f}(\mathbf{0})) \ .$$

However, we still need to adapt some definitions for ω-continuous semirings to commutative Kleene algebras. In Kleene algebras, the Kleene-star operator replaces the infinite summation operator. So we modify the definition of differentials (see Remark 3.7) by replacing the equation for the $\sum$-operator by the definition of [HK99]:

$$\left.\frac{\partial g^*}{\partial X}\right|_{\boldsymbol{v}} = g^*(\boldsymbol{v}) \cdot \left.\frac{\partial g}{\partial X}\right|_{\boldsymbol{v}} \ . \tag{14}$$

Further, [HK99] gives, implicitly, a definition of $D\boldsymbol{f}|^*_{\boldsymbol{u}}(\boldsymbol{v})$ in commutative Kleene algebra, i.e., without expressing * using $\sum$.

With those notations, and using the fact that [HK99] shows $\boldsymbol{\nu}^{(n)} \sqsubseteq \mu\boldsymbol{f}$, proving Theorem 6.10 amounts to showing the equation

$$\boldsymbol{f}(\mathsf{H}^n_{\boldsymbol{f}}(\boldsymbol{f}(\mathbf{0}))) = \mathsf{H}^n_{\boldsymbol{f}}(\boldsymbol{f}(\mathbf{0})) \ . \tag{15}$$

In order to prove (15) we appeal to Redko's theorem (see [Con71]) that essentially states that an equation of terms over any commutative Kleene algebra holds if it holds under the *canonical commutative interpretation.* See Appendix C for a technical justification of this fact. Let Σ be the finite set of elements of K appearing in $\boldsymbol{f}$. The canonical commutative interpretation $\mathsf{c}_\Sigma : \mathsf{RExp}_\Sigma \to 2^{\mathbb{N}^\Sigma}$ is defined by

$$\mathsf{c}_\Sigma(\alpha) = \{\#w \mid w \in R_\Sigma(\alpha)\} \ ,$$

where $\#w$ is the Parikh-vector of $w \in \Sigma^*$, i.e. $a \in \Sigma$ appears exactly $(\#w)_a$-times in w. We omit the subscript of c_Σ in the following. The ci-semiring of sets of Parikh-vectors $\mathcal{C}_\Sigma$ is defined by $\mathcal{C}_\Sigma = \langle 2^{\mathbb{N}^\Sigma}, \cup, \cdot, \emptyset, \{\mathbf{0}\}\rangle$ with $A \cdot B = \{a + b \mid a \in A, b \in B\}$ for all $A, B \subseteq \mathbb{N}^\Sigma$ and $\sum S = \bigcup S$ for all $S \subseteq 2^{\mathbb{N}^\Sigma}$. In particular we have $\mathsf{c}(\alpha^*) = \bigcup_{i\in\mathbb{N}} \mathsf{c}(\alpha)^i$. By Redko's theorem, we can prove (15) by showing $\mathsf{c}(\boldsymbol{f}(\mathsf{H}^n_{\boldsymbol{f}}(\boldsymbol{f}(\mathbf{0})))) = \mathsf{c}(\mathsf{H}^n_{\boldsymbol{f}}(\boldsymbol{f}(\mathbf{0})))$ over $\mathcal{C}_\Sigma$.

For any function $g : \mathsf{RExp}_\Sigma \to \mathsf{RExp}_\Sigma$, let g^{c} denote the commutative interpretation of g as a map over $\mathcal{C}_\Sigma$, i.e., $\mathsf{c}(g(\alpha)) = g^{\mathsf{c}}(\mathsf{c}(\alpha))$ for all $\alpha \in \mathsf{RExp}_\Sigma$.

Assume $(\mathsf{H}_{\boldsymbol{f}})^{\mathsf{c}} = \mathsf{H}_{\boldsymbol{f}^{\mathsf{c}}}$. By Theorem 6.6, $\mathsf{H}^n_{\boldsymbol{f}^{\mathsf{c}}}(\boldsymbol{f}^{\mathsf{c}}(\emptyset))$ solves the equation system $\boldsymbol{X} = \boldsymbol{f}^{\mathsf{c}}(\boldsymbol{X})$ over $\mathcal{C}_\Sigma$. Combining this, we get:

$$\begin{aligned}\mathsf{c}(\boldsymbol{f}(\mathsf{H}^n_{\boldsymbol{f}}(\boldsymbol{f}(\mathbf{0})))) &= \boldsymbol{f}^{\mathsf{c}}((\mathsf{H}^n_{\boldsymbol{f}})^{\mathsf{c}}(\boldsymbol{f}^{\mathsf{c}}(\emptyset))) = \boldsymbol{f}^{\mathsf{c}}(\mathsf{H}^n_{\boldsymbol{f}^{\mathsf{c}}}(\boldsymbol{f}^{\mathsf{c}}(\emptyset))) = \mathsf{H}^n_{\boldsymbol{f}^{\mathsf{c}}}(\boldsymbol{f}^{\mathsf{c}}(\emptyset)) \\ &= \mathsf{c}(\mathsf{H}^n_{\boldsymbol{f}}(\boldsymbol{f}(\mathbf{0}))) \ .\end{aligned}$$

Then (15) follows by Redko's theorem.

So it remains to show that $(\mathsf{H}_{\boldsymbol{f}})^{\mathsf{c}} = \mathsf{H}_{\boldsymbol{f}^{\mathsf{c}}}$ indeed holds, which is equivalent to

$$\mathsf{c}(D\boldsymbol{f}|^*_{\boldsymbol{X}}(\boldsymbol{X})) = D\boldsymbol{f}^{\mathsf{c}}|^*_{\mathsf{c}(\boldsymbol{X})}(\mathsf{c}(\boldsymbol{X})) \ . \tag{16}$$

First we show the following lemma.

Lemma 6.11. *The following equation holds for all $\boldsymbol{u}, \boldsymbol{v} \in \mathsf{RExp}^{\mathcal{X}}_\Sigma$:*

$$\mathsf{c}(D\boldsymbol{f}|_{\boldsymbol{u}}(\boldsymbol{v})) = D\boldsymbol{f}^{\mathsf{c}}|_{\mathsf{c}(\boldsymbol{u})}(\mathsf{c}(\boldsymbol{v})) \ .$$

Proof. One can prove this vector equation for each component separately, so we can assume $\boldsymbol{f} = f \in \mathsf{RExp}_{\Sigma\cup\mathcal{X}}$. Moreover, it suffices to show $\mathsf{c}(D_X f|_{\boldsymbol{u}}(\boldsymbol{v})) = D_X f^{\mathsf{c}}|_{\mathsf{c}(\boldsymbol{u})}(\mathsf{c}(\boldsymbol{v}))$ for all $X \in \mathcal{X}$. By Remark 3.7 it is equivalent to prove

$$\mathsf{c}\left(\left.\frac{\partial f}{\partial X}\right|_{\boldsymbol{u}}\right) = \left.\frac{\partial f^{\mathsf{c}}}{\partial X}\right|_{\mathsf{c}(\boldsymbol{u})} \ .$$

We proceed by induction on the structure of f. Only the case $f = g^*$ is interesting. We have:

$$\begin{aligned}
\mathsf{c}\left(\left.\frac{\partial g^*}{\partial X}\right|_{\boldsymbol{u}}\right) &= \mathsf{c}\left(g^*(\boldsymbol{u}) \cdot \left.\frac{\partial g}{\partial X}\right|_{\boldsymbol{u}}\right) && \text{(Equation (14))}\\
&= \bigcup_{i\in\mathbb{N}} \mathsf{c}(g(\boldsymbol{u}))^i \cdot \mathsf{c}\left(\left.\frac{\partial g}{\partial X}\right|_{\boldsymbol{u}}\right) && \text{(definition of } \mathsf{c}\text{)}\\
&= \bigcup_{i\in\mathbb{N}} \mathsf{c}(g(\boldsymbol{u}))^i \cdot \left.\frac{\partial g^{\mathsf{c}}}{\partial X}\right|_{\mathsf{c}(\boldsymbol{u})} && \text{(induction)}\\
&= \bigcup_{i\geq 1} \left(g^{\mathsf{c}}(\mathsf{c}(\boldsymbol{u}))\right)^{i-1} \cdot \left.\frac{\partial g^{\mathsf{c}}}{\partial X}\right|_{\mathsf{c}(\boldsymbol{u})} && \text{(definition of } g^{\mathsf{c}}\text{)}\\
&= \bigcup_{i\in\mathbb{N}} \left(\left.\frac{\partial (g^{\mathsf{c}})^i}{\partial X}\right|_{\mathsf{c}(\boldsymbol{u})}\right) && \begin{array}{l}\text{(idempotence of } \cup\text{,}\\ \text{Remark 3.7: equation for } \cdot \text{)}\end{array}\\
&= \left.\frac{\partial \bigcup_{i\in\mathbb{N}} (g^{\mathsf{c}})^i}{\partial X}\right|_{\mathsf{c}(\boldsymbol{u})} && \text{(Remark 3.7: equation for +)}\\
&= \left.\frac{\partial (g^*)^{\mathsf{c}}}{\partial X}\right|_{\mathsf{c}(\boldsymbol{u})} && \text{(definition of } \mathsf{c}\text{)} \qquad \square
\end{aligned}$$

As mentioned above, [HK99] implicitly defines $D\boldsymbol{f}|^*_{\boldsymbol{u}}(\boldsymbol{v})$ in commutative Kleene algebra. In particular, their definition satisfies

$$\mathsf{c}\left(D\boldsymbol{f}|^*_{\boldsymbol{u}}(\boldsymbol{v})\right) = \bigcup_{i\in\mathbb{N}} \mathsf{c}\left(D\boldsymbol{f}|^i_{\boldsymbol{u}}(\boldsymbol{v})\right) \ . \tag{17}$$

Now we can prove (16):

$$\begin{aligned}
\mathsf{c}\left(D\boldsymbol{f}|^*_{\boldsymbol{X}}(\boldsymbol{X})\right) &= \bigcup_{i\in\mathbb{N}} \mathsf{c}\left(D\boldsymbol{f}|^i_{\boldsymbol{X}}(\boldsymbol{X})\right) && \text{(Equation (17))}\\
&= \bigcup_{i\in\mathbb{N}} D\boldsymbol{f}^{\mathsf{c}}\Big|^i_{\mathsf{c}(\boldsymbol{X})}(\mathsf{c}(\boldsymbol{X})) && \text{(Lemma 6.11)}\\
&= D\boldsymbol{f}^{\mathsf{c}}\Big|^*_{\mathsf{c}(\boldsymbol{X})}(\mathsf{c}(\boldsymbol{X})) && \text{(Lemma 3.11)}
\end{aligned}$$

This concludes the proof of Theorem 6.10.

6.3 Comparison with Previous Proofs of Parikh's Theorem

Theorem 6.6 and Theorem 6.10 imply Parikh's theorem: for every context-free language there is a regular language with the same commutative image. We briefly sketch how our proof relates to those by [Par66], [HK99], and [AEI01].

Given a context-free grammar G, let $L'(G)$ be the set of words generated by derivation trees in which every variable (non-terminal) of G appears at least once. Parikh reduces the problem of calculating the commutative image of $L(G)$ to the same problem for $L'(G)$, and then proceeds to solving the latter by analysing the structure of the derivation trees associated with $L'(G)$. The proof by [HK99] relies completely on the axiomatic definitions of commutative Kleene algebras, and combines these with generalisations of results known from (vector) calculus, in particular the notion of partial derivative as used in this paper. Finally, [AEI01] identify a set of axioms describing the properties of the Kleene star[8], and derive Parikh's theorem from them. The axioms are purely equational, while the axioms used in [HK99] involve inequalities and implications (see the beginning of Subsection 6.2).

[8] More precisely, in [AEI01] least fixed-point expressions (μ-terms), like $\mu z.xz + y$, are considered, generalizing the Kleene star.

Our proof combines both transformation of derivation trees and algebraic methods, and so it lies between Parikh's proof and those in [HK99,AEI01]. The main contribution of our proof is the study of the relation between derivatives and derivation trees.

Finding a purely algebraic proof of $\boldsymbol{\nu}^{(n)} = \mu\boldsymbol{f}$ is still an open problem.

7 Non-Distributive Program Analyses

In this paper we have focused on *distributive* program analyses, which allows us to use semirings as algebraic structure. Recall that semirings are distributive, i.e., all semiring elements a, b, c satisfy $a \cdot (b+c) = a \cdot b + a \cdot c$ and $(a+b) \cdot c = a \cdot c + b \cdot c$.

Distributive intraprocedural analyses (i.e., for programs without procedures) were considered first in [Kil73]. This seminal paper showed that, given a program and the distributive transfer functions of a program analysis, one can construct a vector $\boldsymbol{f}$ of polynomials such that, for every program point p,the p-component of the least fixed point $\mu\boldsymbol{f}$ coincides with the so-called *MOP-value*[9] of p, the sum of the dataflow values of all program paths leading to p.

The framework of [Kil73] was generalized to non-distributive transfer functions in [KU77]. Non-distributivity means, in our terms, that only *subdistributivity* holds: $a \cdot (b + c) \sqsupseteq a \cdot b + a \cdot c$ and $(a + b) \cdot c \sqsupseteq a \cdot c + b \cdot c$.[10] There are interesting program analyses, such as constant propagation, which are non-distributive, see e.g. [KU77,NNH99]. In those cases, the least fixed point does not necessarily coincide with the MOP-value, but rather safely approximates ("overapproximates") it.

[SP81] extended the work of [Kil73] to the interprocedural case. The generalization to non-distributive analyses was done by [KS92], who proved that, as in the intraprocedural case, the least fixed point is an overapproximation of the MOP-value.

We define the MOP-value as the vector $\boldsymbol{M}$ with $\boldsymbol{M}_p = Y(\mathcal{T}_p)$, where $\mathcal{T}_p$ is the set of trees labeled with p. Notice that a depth-first traversal of a tree labeled with p precisely corresponds to an interprocedural path from the beginning of the procedure of p to the program point p, i.e., the MOP-value $\boldsymbol{M}_p = Y(\mathcal{T}_p)$ is indeed the sum of the dataflow values of all paths to p. Corollary 4.5 states that $\boldsymbol{M} = \mu\boldsymbol{f}$ holds in the distributive case. Proposition 2.4 and Theorem 3.8 show that the Kleene and Newton sequences converge to this value.

For the non-distributive case, the least fixed point overapproximates the MOP-value, i.e., $\boldsymbol{M} \sqsubseteq \mu\boldsymbol{f}$, cf. [KS92].

In the following we show that Newton's method is still well-defined in "sub-distributive semirings", and that the Kleene and Newton sequences both converge to overapproximations of $\boldsymbol{M}$, more precisely, we show $\boldsymbol{M} \sqsubseteq \sup_{i\in\mathbb{N}} \boldsymbol{\kappa}^{(i)} \sqsubseteq \sup_{i\in\mathbb{N}} \boldsymbol{\nu}^{(i)}$.

For this we first define *subdistributive (ω-complete) semirings*[11]:

Definition 7.1. *A subdistributive semiring is a tuple $\langle S, +, \cdot, 0, 1\rangle$ satisfying the following properties:*

1. *$\langle S, +, 0\rangle$ is a commutative monoid.*
2. *$\langle S, \cdot, 1\rangle$ is a monoid.*
3. *$0 \cdot a = a \cdot 0 = 0$ for all $a \in S$.*
4. *$a \cdot (b + c) \sqsupseteq a \cdot b + a \cdot c$ and $(a + b) \cdot c \sqsupseteq a \cdot c + b \cdot c$ for all $a, b, c \in S$.*
5. *The relation $\sqsubseteq := \{(a, b) \in S \times S \mid \exists d \in S : a + d = b\}$ is a partial order.*
6. *For all ω-chains $(a_i)_{i\in\mathbb{N}}$ (i.e. $a_0 \sqsubseteq a_1 \sqsubseteq a_2 \sqsubseteq \ldots$ with $a_i \in S$) $\sup^{\sqsubseteq}_{i\in\mathbb{N}} a_i$ exists. For any sequence $(b_i)_{i\in\mathbb{N}}$ define $\sum_{i\in\mathbb{N}} b_i := \sup^{\sqsubseteq}\{a_0 + a_1 + \ldots + a_i \mid i \in \mathbb{N}\}$.*

Remark 7.2. We obtain the definition of subdistributive semiring from the definition of ω-continuous semiring by removing (7), and replacing distributivity with subdistributivity (see (4)).

[9] We keep the term MOP-value for historical reasons.

[10] If addition is idempotent (as for lattice joins) this condition is equivalent to the monotonicity of multiplication, or, in traditional terms, to the monotonicity of the transfer functions [KU77]. The stricter distributivity condition, on the other hand, amounts to requiring the transfer functions to be homomorphisms.

[11] We drop *ω-complete* in the following.

In the rest of the section $\langle S, +, \cdot, 0, 1\rangle$ denotes a subdistributive semiring. Polynomials, vectors, differential, etc. are defined as in the distributive setting.

Note that the following inequalities still hold for all sequences $(a_i)_{i\in\mathbb{N}}$, $c \in S$, and partitions $(I_j)_{j\in J}$ of $\mathbb{N}$:

$$c \cdot \left(\sum_{i\in\mathbb{N}} a_i\right) \sqsupseteq \sum_{i\in\mathbb{N}}(c \cdot a_i), \quad \left(\sum_{i\in\mathbb{N}} a_i\right) \cdot c \sqsupseteq \sum_{i\in\mathbb{N}}(a_i \cdot c), \quad \sum_{j\in J}\left(\sum_{i\in I_j} a_j\right) \sqsupseteq \sum_{i\in\mathbb{N}} a_i \ .$$

Thus, any polynomial p is still monotone, although not necessarily ω-continuous. For any sequence $(\boldsymbol{v}_i)_{i\in\mathbb{N}}$ (of vectors) we still have $p(\sum_{i\in\mathbb{N}} \boldsymbol{v}_i) \sqsupseteq \sum_{i\in\mathbb{N}} p(\boldsymbol{v}_i)$. Hence, the Kleene sequence of a polynomial system $\boldsymbol{f}$ still converges, but not necessarily to the least fixed point of $\boldsymbol{f}$:

Corollary 7.3. *For any system $\boldsymbol{f}$ of polynomials, the Kleene sequence $(\boldsymbol{\kappa}^{(i)})_{i\in\mathbb{N}}$ is an ω-chain. Moreover, if $\boldsymbol{f}$ has a least solution $\mu\boldsymbol{f}$, then $\sup_{i\in\mathbb{N}} \boldsymbol{\kappa}^{(i)} \sqsubseteq \mu\boldsymbol{f}$.*

Since the Kleene sequence is still an ω-chain, its limit exists and is a safe approximation of the MOP-value:

Proposition 7.4. *For any polynomial system $\boldsymbol{f}$ we have $\left(\boldsymbol{\kappa}^{(i)}\right)_X \sqsupseteq Y(\mathcal{H}^i_X)$, and, hence, $\left(\sup_{i\in\mathbb{N}} \boldsymbol{\kappa}^{(i)}\right)_X \sqsupseteq Y(\mathcal{T}_X)$ where $\mathcal{T}_X$ is the set of trees labeled with X.*

We skip the proof of this proposition as it is almost identical to the one of Proposition 4.3. The only difference is that when expanding the components of $\boldsymbol{\kappa}^{(i)}$ into a sum of products of coefficients, subdistributivity only guarantees that $\boldsymbol{\kappa}^{(i)}$ is an upper bound, but not equality anymore. Similarly, subdistributivity only allows us to generalize the lower bound from Lemma 3.12, i.e. we have

$$\boldsymbol{f}(\boldsymbol{u}) + D\boldsymbol{f}|_{\boldsymbol{u}}(\boldsymbol{v}) \sqsubseteq \boldsymbol{f}(\boldsymbol{u} + \boldsymbol{v})$$

for a polynomial system $\boldsymbol{f}$ and vectors $\boldsymbol{u}, \boldsymbol{v}$.

We now turn to the definition of *Newton sequence*.

Definition 7.5. *For $\boldsymbol{f}$ a polynomial system in the variables $\boldsymbol{X}$, and $\boldsymbol{a}, \boldsymbol{b}$ vectors we set*

$$L_{\boldsymbol{f};\boldsymbol{a};\boldsymbol{b}}(\boldsymbol{X}) := \boldsymbol{b} + D\boldsymbol{f}|_{\boldsymbol{a}}(\boldsymbol{X}).$$

Definition 7.6. *Let $\boldsymbol{f}$ be a polynomial system.*

- *Let $i \in \mathbb{N}$. An i-th* Newton approximant *$\boldsymbol{\nu}^{(i)}$ is inductively defined by*

$$\boldsymbol{\nu}^{(0)} = \boldsymbol{f}(\boldsymbol{0}) \quad \text{and} \quad \boldsymbol{\nu}^{(i+1)} = \boldsymbol{\nu}^{(i)} + \boldsymbol{\Delta}^{(i)} \ ,$$

  where $\boldsymbol{\Delta}^{(i)}$ has to satisfy $\sum_{k\in\mathbb{N}} D\boldsymbol{f}|^k_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{\delta}^{(i)}) \sqsubseteq \boldsymbol{\Delta}^{(i)} \sqsubseteq L_{\boldsymbol{f};\boldsymbol{\nu}^{(i)};\boldsymbol{\delta}^{(i)}}\left(\boldsymbol{\Delta}^{(i)}\right)$.
- *Any such sequence $(\boldsymbol{\nu}^{(i)})_{i\in\mathbb{N}}$ of Newton approximants is called* Newton sequence*.*

Remark 7.7. If $\boldsymbol{\delta}^{(i)}$ exists, then possible choices for $\boldsymbol{\Delta}^{(i)}$ are

$$\sum_{k\in\mathbb{N}} D\boldsymbol{f}|^k_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{\delta}^{(i)}),\ \sup_{k\in\mathbb{N}} L^k_{\boldsymbol{f};\boldsymbol{\nu}^{(i)};\boldsymbol{\delta}^{(i)}}(\boldsymbol{0}) \text{ or (if it exists) } \mu L_{\boldsymbol{f};\boldsymbol{\nu}^{(i)};\boldsymbol{\delta}^{(i)}}.$$

Note that in the distributive setting all three values coincide.

Proposition 7.8. *Let $\boldsymbol{f} \colon V \to V$ be a vector of power series.*

- *For every Newton approximant $\boldsymbol{\nu}^{(i)}$ there exists a vector $\boldsymbol{\delta}^{(i)}$ such that $\boldsymbol{f}(\boldsymbol{\nu}^{(i)}) = \boldsymbol{\nu}^{(i)} + \boldsymbol{\delta}^{(i)}$. So there is at least one Newton sequence.*
- *Every Newton sequence $\boldsymbol{\nu}^{(i)}$ satisfies $\boldsymbol{\kappa}^{(i)} \sqsubseteq \boldsymbol{\nu}^{(i)} \sqsubseteq \boldsymbol{f}(\boldsymbol{\nu}^{(i)}) \sqsubseteq \boldsymbol{\nu}^{(i+1)}$ for all $i \in \mathbb{N}$.*

Proof. First we prove for all $i \in \mathbb{N}$ that a suitable $\boldsymbol{\delta}^{(i)}$ exists and, at the same time, that the inequality $\boldsymbol{\kappa}^{(i)} \sqsubseteq \boldsymbol{\nu}^{(i)} \sqsubseteq \boldsymbol{f}(\boldsymbol{\nu}^{(i)})$ holds. We proceed by induction on i. For the base case $i = 0$ we have:

$$\boldsymbol{\nu}^{(0)} = \boldsymbol{f}(\mathbf{0}) = \boldsymbol{\kappa}^{(0)} \sqsubseteq \boldsymbol{\kappa}^{(1)} = \boldsymbol{f}(\boldsymbol{\kappa}^{(0)}) = \boldsymbol{f}(\boldsymbol{\nu}^{(0)}).$$

So, there exists a $\boldsymbol{\delta}^{(0)}$ with $\boldsymbol{\nu}^{(0)} + \boldsymbol{\delta}^{(0)} = \boldsymbol{f}(\boldsymbol{\nu}^{(0)})$, and hence we have:

$$\boldsymbol{\nu}^{(1)} = \boldsymbol{\nu}^{(0)} + \boldsymbol{\Delta}^{(0)} \sqsupseteq \boldsymbol{\nu}^{(0)} + \sum_{k \in \mathbb{N}} D\boldsymbol{f}|^{k}_{\boldsymbol{\nu}^{(0)}}(\boldsymbol{\delta}^{(0)}) \sqsupseteq \boldsymbol{\nu}^{(0)} + \boldsymbol{\delta}^{(0)} = \boldsymbol{f}(\boldsymbol{\nu}^{(0)}).$$

For the induction step, let $i \geq 0$.

$$\boldsymbol{\kappa}^{(i+1)} = \boldsymbol{f}(\boldsymbol{\kappa}^{(i)}) \sqsubseteq \boldsymbol{f}(\boldsymbol{\nu}^{(i)}) = \boldsymbol{\nu}^{(i)} + \boldsymbol{\delta}^{(i)} \sqsubseteq \boldsymbol{\nu}^{(i)} + \sum_{k \in \mathbb{N}} D\boldsymbol{f}|^{k}_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{\delta}^{(i)}).$$

As we require that $\sum_{k \in \mathbb{N}} D\boldsymbol{f}|^{k}_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{\delta}^{(i)}) \sqsubseteq \boldsymbol{\Delta}^{(i)}$, it now immediately follows that

$$\boldsymbol{\kappa}^{(i+1)} \sqsubseteq \boldsymbol{\nu}^{(i)} + \boldsymbol{\Delta}^{(i)} = \boldsymbol{\nu}^{(i+1)}.$$

By definition of $\boldsymbol{\Delta}^{(i)}$ we have $\boldsymbol{\Delta}^{(i)} \sqsubseteq L_{\boldsymbol{f};\boldsymbol{\nu}^{(i)};\boldsymbol{\delta}^{(i)}}(\boldsymbol{\Delta}^{(i)}))$, it therefore follows:

$$\begin{aligned}\boldsymbol{\nu}^{(i+1)} = \boldsymbol{\nu}^{(i)} + \boldsymbol{\Delta}^{(i)} &\sqsubseteq \boldsymbol{\nu}^{(i)} + \boldsymbol{\delta}^{(i)} + D\boldsymbol{f}|_{\boldsymbol{\nu}^{(i)}}\left(\boldsymbol{\Delta}^{(i)}\right) \\ &= \boldsymbol{f}(\boldsymbol{\nu}^{(i)}) + D\boldsymbol{f}|_{\boldsymbol{\nu}^{(i)}}\left(\boldsymbol{\Delta}^{(i)}\right) \sqsubseteq \boldsymbol{f}\left(\boldsymbol{\nu}^{(i)} + \boldsymbol{\Delta}^{(i)}\right) = \boldsymbol{f}(\boldsymbol{\nu}^{(i+1)}).\end{aligned}$$

We complete our proof by

$$\begin{aligned}\boldsymbol{f}(\boldsymbol{\nu}^{(i+1)}) = \boldsymbol{\nu}^{(i+1)} + \boldsymbol{\delta}^{(i+1)} &\sqsubseteq \boldsymbol{\nu}^{(i+1)} + \sum_{k \in \mathbb{N}} D\boldsymbol{f}|^{k}_{\boldsymbol{\nu}^{(i+1)}}(\boldsymbol{\delta}^{(i+1)}) \\ &\sqsubseteq \boldsymbol{\nu}^{(i+1)} + \boldsymbol{\Delta}^{(i+1)} = \boldsymbol{\nu}^{(i+2)}.\end{aligned}$$

Proposition 7.9. *Let $\boldsymbol{M}$ be the MOP-value, i.e., the vector $\boldsymbol{M}$ with $\boldsymbol{M}_X = Y(\mathcal{T}_X)$. Then $\boldsymbol{M} \sqsubseteq \sup_{i \in \mathbb{N}} \boldsymbol{\kappa}^{(i)} \sqsubseteq \sup_{i \in \mathbb{N}} \boldsymbol{\nu}^{(i)}$.*

Proof. Follows directly from Propositions 7.4 and 7.8. □

Proposition 7.10. *For $\boldsymbol{\Delta}^{(i)} = \sum_{k \in \mathbb{N}} D\boldsymbol{f}|^{k}_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{\delta}^{(i)})$ we have $\sup_{i \in \mathbb{N}} \boldsymbol{\nu}^{(i)} \sqsubseteq \mu\boldsymbol{f}$, if $\mu\boldsymbol{f}$ exists.*

Proof. The proof is almost identical to the one of Proposition 3.9. Note that the proof of Lemma 3.13 does not use distributivity.

Theorem 7.11 (Tree Characterization of the Newton Sequence). *Let $(\boldsymbol{\nu}^{(i)})_{i \in \mathbb{N}}$ be a Newton sequence of $\boldsymbol{f}$. For every $X \in \mathcal{X}$ and every $i \geq 0$ we have $\left(\boldsymbol{\nu}^{(i)}\right)_X \sqsupseteq Y(\mathcal{D}^i_X)$, i.e., the X-component of the i-th Newton approximant is a safe approximation of the yield of $\mathcal{D}^i_X$.*

Proof. In the distributive setting we proved this theorem via induction where we expanded the the terms we obtained using distributivity. In the subdistributive case the same proof still guarantees that $\left(\boldsymbol{\nu}^{(i)}\right)_X \sqsupseteq Y(\mathcal{D}^i_X)$.

8 Conclusions

In this paper we have presented a contribution to the mathematical foundations of program analysis. Since its inception, the theory of program analysis has been based on two fundamental observations:

- Analysis problems can be reduced (using abstract interpretation [CC77]) to the mathematical problem of computing the least solution of a system of equations over a semilattice;
- Such systems of equations can be solved using Kleene's fixed-point theorem as basic algorithm scheme.

We have generalized the algebraic framework from semilattices to arbitrary semirings (a generalization to idempotent semirings was already present in the work of Reps et al. on pushdown systems for program analysis [RSJM05]). This otherwise simple step has an interesting consequence: it leads to a common algebraic setting for "qualitative" analyses, which, loosely speaking, explore the existence of execution paths satisfying a given property, and "quantitative" analyses, in which paths are assigned a numerical weight, and one is interested in the sum of the weights of all paths leading to a program point. Classical examples of qualitative analyses are live variables, reaching definitions, or constant propagation, while examples of quantitative analysis arise in the study of probabilistic programs: probability of termination, expected execution time or, in the interprocedural case, expected stack height (for the latter, see [EKM05,BEK05]). The common setting allows to compare the algorithmic schemes used in the qualitative and quantitative case, and examine if a transfer of techniques is possible. We have shown that Newton's method, the classical technique of numerical mathematics for systems of equations over the reals, can be generalized to the abstract setting. In particular, it can be applied to qualitative analysis problems.

We have explored Newton's method for idempotent semirings, i.e., for the semirings corresponding to qualitative analyses. We have shown that notions and techniques of the theory of context-free languages (the languages corresponding to the control-flow of interprocedural programs) and of the theory of Kleene algebras (extensively studied by Kozen et al. as a mathematical formalism for control flow, see for instance [Koz91,Koz97,Koz00,Koz08]) can be naturally formulated in terms of Newton's method. More precisely, the context-free languages of finite index, already studied by Yntema, Salomaa, and Gruska, among others [Ynt67,Sal69,Gru71] turn out to be the Newton approximants of the context-free languages. Finally, we have shown that the beautiful algebraic algorithm of [HK99] for solving systems of equations over commutative Kleene algebras is a particular instance of Newton's method. Moreover, we have proved that the algorithm requires at most n iterations for a system of n equations, a tight bound that improves on the $\mathcal{O}(3^n)$ bound presented in [HK99].

While this paper imports notions of calculus and numerical mathematics into program analysis, our work also has some consequences pointing in the opposite direction. Quantitative analyses lead to systems of equations over the real semiring, a particular case of the systems over the real field. Surprisingly, the performance of Newton's method on this special case seems not to have received much attention from numerical mathematicians. The method turns out to have much better properties than in the general case. A consequence of our main result (which was already proved, in a slightly more restricted form, by [EY05]), is that on the real semiring Newton's method always converges to the least fixed point starting from zero. This is not so in the real field, where it may not converge or converge only locally, i.e., when started sufficiently close to the zero (see e.g. [Ort72,OR70]). In related work we have shown that the convergence order of the method is at least linear, meaning that the number of accurate bits of the Newton approximants grows at least linearly with the number of iterations [KLE07,EKL08].

APPENDIX

A Proofs of Section 4

To avoid typographical clutter in the following proofs, we use the following notation. Given some class of objects (e.g. derivation trees t) and a predicate $P(t)$, we write

$$\sum_t Y(t) : P(t)$$

instead of

$$\sum_{t \text{ such that } P(t) \text{ holds}} Y(t) \, .$$

Proposition 4.3. *$\left(\boldsymbol{\kappa}^{(i)}\right)_X = Y(\mathcal{H}_X^i)$, i.e., the X-component of the i-th Kleene approximant $\boldsymbol{\kappa}^{(i)}$ is equal to the yield of $\mathcal{H}_X^i$.*

Proof. By induction on i. The base case $i = 0$ is easy. Induction step ($i \geq 0$):

$$
\begin{aligned}
&\left(\boldsymbol{\kappa}^{(i+1)}\right)_X \\
&\quad = \boldsymbol{f}_X(\boldsymbol{\kappa}^{(i)}) \\
&\quad = \sum_{j \in J} m_{X,j}(\boldsymbol{\kappa}^{(i)}) \\
&\quad = \sum_{j \in J} y : \begin{cases} m_{X,j} = a_1 X_1 \cdots X_k a_{k+1} \\ y = a_1 \boldsymbol{\kappa}^{(i)}_{X_1} \cdots \boldsymbol{\kappa}^{(i)}_{X_k} a_{k+1} \end{cases}
\end{aligned}
$$

by induction:

$$
\begin{aligned}
&= \sum_{j \in J} y : \begin{cases} m_{X,j} = a_1 X_1 \cdots X_k a_{k+1} \\ y = a_1 Y(\mathcal{H}^i_{X_1}) \cdots Y(\mathcal{H}^i_{X_k}) a_{k+1} \end{cases} \\
&= \sum_{\substack{j \in J \\ t_1, \ldots, t_k}} y : \begin{cases} m_{X,j} = a_1 X_1 \cdots X_k a_{k+1} \\ t_1, \ldots, t_k \text{ trees with } h(t_r) \leq i, \lambda_v(t_r) = X_r \quad (1 \leq r \leq k) \\ y = a_1 Y(t_1) \cdots Y(t_k) a_{k+1} \end{cases} \\
&= \sum_{j \in J, t} Y(t) : t \text{ is a tree with } h(t) \leq i + 1,\ \lambda(t) = (X, j) \\
&= Y(\mathcal{H}^i_X)
\end{aligned}
$$

□

The following definition of *fine dimension* is analogous to Definition 4.6, but adds a second component, which measures the length of the path from the root to the lowest node with the same dimension as the root:

Definition A.1 ((fine dimension)). *The* fine dimension $dl(t) = (d(t), l(t))$ *of a tree* t *is inductively defined as follows:*

1. *If* t *has no children, then* $dl(t) = (0, 0)$.
2. *If* t *has exactly one child* t_1, *then* $dl(t) = (d(t_1), l(t_1) + 1)$.
3. *If* t *has at least two children, let* t_1, t_2 *be two distinct children of* t *such that* $d(t_1) \geq d(t_2)$ *and* $d(t_2) \geq d(t')$ *for every child* $t' \neq t_1$. *Let* $d_1 = d(t_1)$ *and* $d_2 = d(t_2)$. *Then*

$$
dl(t) = \begin{cases} (d_1 + 1, 0) & \text{if } d_1 = d_2 \\ (d_1, l(t_1) + 1) & \text{if } d_1 > d_2. \end{cases}
$$

Remark A.2. Notice that, by Definition 4.8, a tree t is proper if and only if $l(t) = 0$. So we have:

$$
Y(P^i_X) = \sum_t Y(t) : t \text{ tree with } \lambda_v(t) = X,\ dl(t) = (i, 0)
$$

Now we can prove the remaining lemmata from Section 4.

Lemma 4.9. *For every variable* $X \in \mathcal{X}$ *and every* $i \geq 0$: $\boldsymbol{\tau}^{(i)}_X = Y(\mathcal{D}^i_X)$.

Proof. By induction on i. Induction base ($i = 0$):

$$
\begin{aligned}
\boldsymbol{\tau}^{(0)}_X = \boldsymbol{f}_X(\mathbf{0}) &= \sum_t Y(t) : \lambda_v(t) = X, h(t) = 0 \\
&= \sum_t Y(t) : \lambda_v(t) = X, d(t) = 0 \\
&= Y(\mathcal{D}^0_X)
\end{aligned}
$$

Induction step $(i+1>0)$:
We need to show that $D\boldsymbol{f}|^*_{\boldsymbol{\tau}^{(i)}}(\boldsymbol{\delta}^{(i)})$ equals exactly the yield of all trees of dimension $i+1$, i.e., that for all $X \in \mathcal{X}$

$$\left(D\boldsymbol{f}|^*_{\boldsymbol{\tau}^{(i)}}(\boldsymbol{\delta}^{(i)})\right)_X = \sum_t Y(t) : \lambda_v(t) = X,\ d(t) = i+1\ .$$

We prove the following stronger claim by induction on p:

$$\left(D\boldsymbol{f}|^p_{\boldsymbol{\tau}^{(i)}}(\boldsymbol{\delta}^{(i)})\right)_X = \sum_t Y(t) : \lambda_v(t) = X,\ dl(t) = (i+1,p)$$

The claim holds for $p=0$ by Remark A.2. For the induction step, let $p \geq 0$. Then we have for all $X \in \mathcal{X}$:

$$\begin{aligned}
&\left(D\boldsymbol{f}|^{p+1}_{\boldsymbol{\tau}^{(i)}}(\boldsymbol{\delta}^{(i)})\right)_X \\
&= \left(D\boldsymbol{f}|_{\boldsymbol{\tau}^{(i)}} \circ\ D\boldsymbol{f}|^p_{\boldsymbol{\tau}^{(i)}}(\boldsymbol{\delta}^{(i)})\right)_X \\
&= D\boldsymbol{f}_X|_{\boldsymbol{\tau}^{(i)}} \circ\ D\boldsymbol{f}|^p_{\boldsymbol{\tau}^{(i)}}(\boldsymbol{\delta}^{(i)})
\end{aligned}$$

Define the vector $\widetilde{\boldsymbol{Y}}$ by $\widetilde{\boldsymbol{Y}}_{X_0} = \sum_t Y(t) : \lambda_v(t) = X_0, dl(t) = (i+1,p)$. Then, by induction hypothesis (on p), above expression equals

$$\begin{aligned}
&= D\boldsymbol{f}_X|_{\boldsymbol{\tau}^{(i)}}(\widetilde{\boldsymbol{Y}}) \\
&= \sum_{j\in J} Dm_{X,j}|_{\boldsymbol{\tau}^{(i)}}(\widetilde{\boldsymbol{Y}}) : m_{X,j} = a_1 X_1 \cdots a_k X_k a_{k+1} \\
&= \sum_{j\in J,r} y : \begin{cases} m_{X,j} = a_1 X_1 \cdots a_k X_k a_{k+1} \\ 1 \leq r \leq k \\ y = a_1 \boldsymbol{\tau}^{(i)}_{X_1} \cdots a_r \widetilde{\boldsymbol{Y}}_{X_r} a_{r+1} \boldsymbol{\tau}^{(i)}_{X_{r+1}} \cdots a_k \boldsymbol{\tau}^{(i)}_{X_k} a_{k+1} \end{cases}
\end{aligned}$$

by induction on i:

$$\begin{aligned}
&= \sum_{\substack{j\in J,r,\\ t_1,\ldots,t_k}} y : \begin{cases} m_{X,j} = a_1 X_1 \cdots a_k X_k a_{k+1} \\ 1 \leq r \leq k \\ t_1,\ldots,t_k \text{ trees with } \lambda_v(t_s) = X_s \quad (1 \leq s \leq k) \\ \qquad dl(t_r) = (i+1,p), \\ \qquad d(t_s) \leq i \quad (1 \leq s \leq k,\ s \neq r) \\ y = a_1 Y(t_1) \cdots a_r Y(t_r) \cdots a_k Y(t_k) a_{k+1} \end{cases} \\
&= \sum_{j \in J,t} Y(t) : t \text{ tree with } \lambda(t) = (X,j),\ dl(t) = (i+1,p+1) \\
&= \sum_t Y(t) : t \text{ tree with } \lambda_v(t) = X,\ dl(t) = (i+1,p+1) \qquad \square
\end{aligned}$$

LEMMA 4.10. *The sequence $(\boldsymbol{\tau}^{(i)})_{i\in\mathbb{N}}$ is a Newton sequence as defined in Definition 3.5, i.e., the $\boldsymbol{\delta}^{(i)}$ of Definition 4.8 satisfy $\boldsymbol{f}(\boldsymbol{\tau}^{(i)}) = \boldsymbol{\tau}^{(i)} + \boldsymbol{\delta}^{(i)}$.*

Proof.

$$\begin{aligned}
\boldsymbol{f}_X(\boldsymbol{\tau}^{(i)}) &= \sum_{j\in J} m_{X,j}(\boldsymbol{\tau}^{(i)}) \\
&= \sum_{j\in J} y : \begin{cases} m_{X,j} = a_1 X_1 \cdots a_k X_k a_{k+1} \\ y = a_1 \boldsymbol{\tau}^{(i)}_{X_1} \cdots a_k \boldsymbol{\tau}^{(i)}_{X_k} a_{k+1} \end{cases}
\end{aligned}$$

by Lemma 4.9:

$$= \sum_{\substack{j \in J \\ t_1,\dots,t_k}} y : \begin{cases} m_{X,j} = a_1 X_1 \cdots a_k X_k a_{k+1} \\ t_1,\dots,t_k \text{ trees with } \lambda_v(t_r) = X_r,\ d(t_r) \leq i, \quad (1 \leq r \leq k) \\ y = a_1 Y(t_1) \cdots a_k Y(t_k) a_{k+1} \end{cases}$$

$$= \sum_{\substack{j \in J \\ t_1,\dots,t_k}} y : \begin{cases} m_{X,j} = a_1 X_1 \cdots a_k X_k a_{k+1} \\ t_1,\dots,t_k \text{ trees with } \lambda_v(t_r) = X_r,\ d(t_r) \leq i, \quad (1 \leq r \leq k) \\ \qquad \text{such that at most one of the } t_r \text{ with } d(t_r) = i \\ y = a_1 Y(t_1) \cdots a_k Y(t_k) a_{k+1} \end{cases}$$

$$+ \sum_{\substack{j \in J \\ t_1,\dots,t_k}} y : \begin{cases} m_{X,j} = a_1 X_1 \cdots a_k X_k a_{k+1} \\ t_1,\dots,t_k \text{ trees with } \lambda_v(t_r) = X_r,\ d(t_r) \leq i, \quad (1 \leq r \leq k) \\ \qquad \text{such that at least two of the } t_r \text{ with } d(t_r) = i \\ y = a_1 Y(t_1) \cdots a_k Y(t_k) a_{k+1} \end{cases}$$

$$= \sum_t Y(t) : t \text{ tree with } \lambda_v(t) = X,\ d(t) \leq i$$

$$+ \sum_t Y(t) : t \text{ tree with } \lambda_v(t) = X,\ dl(t) = (i+1, 0)$$

by Lemma 4.9 resp. Remark A.2:

$$= \boldsymbol{\tau}_X^{(i)} + Y(P_X^{i+1})$$
$$= \boldsymbol{\tau}_X^{(i)} + \boldsymbol{\delta}_X^{(i)} \qquad \square$$

B Proofs of Section 5.1

Theorem 5.3. *Let $G = (\mathcal{X}, \Sigma, P, X_0)$ be a context-free grammar in CNF and let $(\boldsymbol{\nu}^{(i)})_{i \in \mathbb{N}}$ be the Newton sequence associated with G. Then $\big(\boldsymbol{\nu}^{(i)}\big)_{X_0} = L_{i+1}(G)$ for every $i \geq 0$.*

The proof of Theorem 5.3 follows from Theorem 4.7 and the following two lemmata.

Lemma B.1. *Let $G = (\mathcal{X}, \Sigma, P, X_0)$ be a context-free grammar in CNF. Let $w \in \Sigma^*$ be derivable from X by an index-i derivation. Then there is a derivation tree t with $\lambda_v(t) = X, Y(t) = w$ and $d(t) < i$.*

Proof. Let D be a derivation of w. One can associate a derivation tree t to D in the obvious way. We show by induction on i and on the height of t that $d(t) \geq i$ implies $ind(D) > i$, where $ind(D)$ denotes the index of D. The base case $i = 0$ is trivial, because any derivation has index at least 1. The other base case $i = 1$ implies that t has two children, hence $ind(D) \geq 2$. Let $i > 1$ and $d(t) \geq i$. Then t has two children t_1, t_2. By definition of dimension, either $d(t_1) \geq i-1$ and $d(t_2) \geq i-1$ or $d(t_1) \geq i$.

- In the first case, the very first step of D already produces two variables $\lambda_v(t_1)$ and $\lambda_v(t_2)$. Since $d(t_1) \geq 1$ and $d(t_2) \geq 1$, neither of those two variables can be derived to a terminal word immediately. So the most "economical" way to continue the derivation is to finish the derivation of $\lambda_v(t_1)$ or $\lambda_v(t_2)$ before touching the other variable. But, by induction on i, any subderivations of D that "flatten" t_1 and t_2 have indices at least i. Hence $ind(D) > i$.
- In the second case, any subderivation of D that "flattens" t_1 has, by induction on the height, index greater than i. So, D itself cannot have a smaller index. $\square$

Lemma B.2. *Let $G = (\mathcal{X}, \Sigma, P, X_0)$ be a context-free grammar. Let m be the largest number of nonterminals in the right-hand sides of P. Let t be a derivation tree with $\lambda_v(t) = X$, $Y(t) = w$ and $d(t) = i$. Then there is a derivation of w from X with index at most $i \cdot (m-1) + 1$.*

Proof. The sought derivation D can be constructed by "flattening" the derivation tree t according to a certain strategy. The first step of D is $\lambda_v(t) \Rightarrow \lambda_m(t)$. After that, the strategy is to completely flatten each subtree of t in the order of increasing dimension. We prove by induction on i and on the height of t that this yields $ind(D) \leq i \cdot (m-1) + 1$. The base case $i = 0$ is clear. Let $i > 0$ and $t_1, \ldots, t_k$ ($k \leq m$) be the subtrees of t ordered by increasing dimension. During the flattening of t_j, at most $m-1$ nonterminals, namely $\lambda_v(t_{j+1}), \ldots, \lambda_v(t_k)$, stick around. The trees $t_1, \ldots, t_{k-1}$ have dimension at most $i-1$. By induction on i, they can be flattened to derivations with index at most $(i-1) \cdot (m-1) + 1$. So, during the flattening of $t_1, \ldots, t_{k-1}$ the index of D grows to at most $(i-1) \cdot (m-1) + 1 + (m-1) = i \cdot (m-1) + 1$. The tree t_k has dimension at most i. By induction on the height, t_k can be flattened to a derivation with index at most $i \cdot (m-1) + 1$. During the flattening of t_k, no other nonterminals stick around. So, the index of D does not grow over $i \cdot (m-1) + 1$. □

C Redko's Theorem and Commutative Kleene Algebras

There is a number of inequivalent definitions of Kleene algebras. This includes *C-algebras* and *Kleene algebras in the sense of Kozen* the latter of which we simply refer to as *Kleene algebras.*

Both definitions require an algebraic structure $(K, +, \cdot, {}^*, 0, 1)$ that is an idempotent semiring under $+, \cdot, 0, 1$. In addition, different sets of axioms are required.

A C-algebra [Con71] must satisfy the following axioms:

$$\begin{array}{ll} C11 & (a+b)^* = (a^*b)^*a^* \\ C12 & (ab)^* = 1 + a(ba)^*b \\ C13 & (a^*)^* = a^* \\ C14.n & a^* = a^{n*}a^{<n} \quad (n > 0). \end{array}$$

A Kleene algebra [Koz91] on the other hand must satisfy the following axioms:

$$\begin{array}{llcl} K1 & 1 + aa^* \leq a^* & & \\ K2 & 1 + a^*a \leq a^* & & \\ K3 & a + bc \leq c & \rightarrow & b^*a \leq c \\ K4 & a + cb \leq c & \rightarrow & ab^* \leq c, \end{array}$$

where $\leq$ refers to the natural partial order on K.

It was shown in [Koz91] that the axioms of Kleene algebra are *complete* for the algebra of regular languages. That means, if an equation $\alpha = \beta$ between regular expressions holds under the canonical interpretation over the regular languages, then it holds in any Kleene algebra. It is easy to see that equations $C11 - C14$ hold under the canonical interpretation. Therefore any Kleene algebra is a C-algebra.

The axioms of C-algebra are not complete, i.e., they are too weak to derive some equation valid under the canonical interpretation [Con71]. However, if two more axioms (C^+1 and C^+2, see below) describing commutativity are added, the resulting system of axioms (defining *commutative C-algebras*) becomes complete for the algebra of commutative regular languages. In other words, if the Parikh images of languages $L(\alpha)$ and $L(\beta)$ are equal, then $\alpha = \beta$ can be proved using only the axioms of commutative C-algebras. The additional axioms are:

$$\begin{array}{ll} C^+1 & ab = ba \\ C^+2 & a^*b^* = (ab)^*(a^* + b^*). \end{array}$$

The completeness of commutative C-algebras is called *Redko's theorem.* Conway's monograph [Con71] contains a proof of this theorem.

We want to show that the system of axioms of Kleene algebra plus the commutativity axiom $ab = ba$ (defining *commutative Kleene algebras*) is complete for commutative regular languages as well. Appealing to Redko's theorem, we only have to show that equation C^+2 is a theorem of commutative Kleene algebra.

We use the identity $a^*b^* = (a+b)^*$ which is a theorem of commutative Kleene algebra [HK99]. Since $(a+b)^* \geq (ab)^*(a^* + b^*)$ holds in any Kleene algebra, we only need to show $(a+b)^* \leq (ab)^*(a^* + b^*)$. With $K3$ it suffices to show

$$1 + (a+b)(ab)^*(a^* + b^*) \leq (ab)^*(a^* + b^*).$$

We show this inequality for each term of the sum at the left hand side. For 1 it obviously holds. We also have $a(ab)^*a^* = (ab)^*aa^* \leq (ab)^*a^*$ using commutativity and $K1$. Similarly, $a(ab)^*b^* = (ab)^*ab^* = (ab)^*a + (ab)^*abb^* \leq (ab)^*a + (ab)^*b^* \leq (ab)^*(a^* + b^*)$. Here we used that $b^* = 1 + bb^*$ is a theorem of Kleene algebra. The other inequalities follow symmetrically.

Acknowledgment

We like to thank Helmut Seidl for the helpful suggestions.

References

[AEI01] Luca Aceto, Zoltán Ésik, and Anna Ingólfsdóttir. A fully equational proof of parikhs theorem. *RAIRO, Theoretical Informatics and Applications*, 36:200–2, 2001.

[BEK05] Tomás Brázdil, Javier Esparza, and Antonín Kucera. Analysis and prediction of the long-run behavior of probabilistic sequential programs with recursion. In *FOCS*, pages 521–530. IEEE Computer Society, 2005.

[CC77] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of POPL*, pages 238–252. ACM, 1977.

[Con71] J.H. Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, 1971.

[EKL08] Javier Esparza, Stefan Kiefer, and Michael Luttenberger. Convergence thresholds of Newton's method for monotone polynomial equations. In *Proceedings of STACS*, pages 289–300, 2008.

[EKM04] J. Esparza, A. Kučera, and R. Mayr. Model checking probabilistic pushdown automata. In *LICS 2004*. IEEE Computer Society, 2004.

[EKM05] J. Esparza, A. Kučera, and R. Mayr. Quantitative analysis of probabilistic pushdown automata: Expectations and variances. In *Proceedings of LICS 2005*, pages 117–126. IEEE Computer Society Press, 2005.

[EY05] K. Etessami and M. Yannakakis. Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations. In *STACS*, pages 340–352, 2005.

[FH97] H. Fernau and M. Holzer. Conditional context-free languages of finite index. In *New Trends in Formal Languages*, pages 10–26, 1997.

[Gru71] J. Gruska. A few remarks on the index of context-free grammars and languages. *Information and Control*, 19:216–223, 1971.

[GS68] S. Ginsburg and E. Spanier. Derivation-bounded languages. *Journal of Computer and System Sciences*, 2:228–250, 1968.

[HK99] M. W. Hopkins and D. Kozen. Parikh's theorem in commutative Kleene algebra. In *Logic in Computer Science*, pages 394–401, 1999.

[JM82] N. Jones and S. Muchnick. A flexible approach to interprocedural data flow analysis and programs with recursive data structures. In *Proceedings of POPL*, pages 66–74. ACM, 1982.

[Kil73] G. A. Kildall. A unified approach to global program optimization. In *POPL*, pages 194–206. ACM, 1973.

[KLE07] S. Kiefer, M. Luttenberger, and J. Esparza. On the convergence of Newton's method for monotone systems of polynomial equations. In *Proceedings of STOC*, pages 217–226. ACM, 2007.

[Koz90] D. Kozen. On Kleene algebras and closed semirings. In B. Rovan, editor, *Proc. Math. Found. Comput. Sci.*, volume 452 of *Lecture Notes in Computer Science*, pages 26–47. Springer-Verlag, 1990.

[Koz91] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. In *Logic in Computer Science*, pages 214–225, 1991.

[Koz97] Dexter Kozen. Kleene algebra with tests. *ACM Trans. Program. Lang. Syst.*, 19(3):427–443, 1997.

[Koz00] Dexter Kozen. On Hoare logic and Kleene algebra with tests. *ACM Trans. Comput. Log.*, 1(1):60–76, 2000.

[Koz08] Dexter Kozen. Nonlocal flow of control and Kleene algebra with tests. In *LICS*, pages 105–117. IEEE Computer Society, 2008.

[KS92] J. Knoop and B. Steffen. The interprocedural coincidence theorem. In *International Conference on Compiler Construction*, volume 641 of *LNCS*, pages 125–140. Springer-Verlag, 1992.

[KU77] J. B. Kam and J. D. Ullman. Monotone data flow analysis frameworks. *Acta Inf.*, 7:305–317, 1977.

[Kui97] W. Kuich. *Handbook of Formal Languages*, volume 1, chapter 9: Semirings and Formal Power Series: Their Relevance to Formal Languages and Automata, pages 609 – 677. Springer, 1997.

[NNH99] F. Nielson, H.R. Nielson, and C. Hankin. *Principles of Program Analysis*. Springer, 1999.

[OR70] J.M. Ortega and W.C. Rheinboldt. *Iterative solution of nonlinear equations in several variables*. Academic Press, 1970.

[Ort72] J.M. Ortega. *Numerical Analysis: A Second Course*. Academic Press, New York, 1972.

[Par66] Rohit J. Parikh. On context-free languages. *J. ACM*, 13(4):570–581, 1966.

[RHS95] T. Reps, S. Horwitz, and M. Sagiv. Precise interprocedural dataflow analysis via graph reachability. In *Proceedings of POPL*, pages 49–61. ACM, 1995.

[RSJM05] T. Reps, S. Schwoon, S. Jha, and D. Melski. Weighted pushdown systems and their application to interprocedural dataflow analysis. *Science of Computer Programming*, 58(1–2):206–263, October 2005. Special Issue on the Static Analysis Symposium 2003.

[Sal69] A. Salomaa. On the index of a context-free grammar and language. *Information and Control*, 14:474–477, 1969.

[SF00] H. Seidl and C. Fecht. Interprocedural analyses: A comparison. *Journal of Logic Programming (JLP*, 43:123–156, 2000.

[SP81] M. Sharir and A. Pnueli. *Program Flow Analysis: Theory and Applications*, chapter 7: Two Approaches to Interprocedural Data Flow Analysis, pages 189–233. Prentice-Hall, 1981.

[SRH96] S. Sagiv, T. W. Reps, and S. Horwitz. Precise interprocedural dataflow analysis with applications to constant propagation. *Theoretical Computer Science*, 167(1&2):131–170, 1996.

[Ynt67] M.K. Yntema. Inclusion relations among families of context-free languages. *Information and Control*, 10:572–597, 1967.