Convergence of Newton's Method over Commutative Semirings*

Michael Luttenberger and Maximilian Schlund

Institut für Informatik, Technische Universität München,
Boltzmannstr. 3, 85748 Garching, Germany
{luttenbe,schlund}@model.in.tum.de

Abstract. We give a lower bound on the speed at which Newton's method (as defined in [5, 6]) converges over arbitrary ω-continuous commutative semirings. From this result, we deduce that Newton's method converges within a finite number of iterations over any semiring which is "collapsed at some $k \in \mathbb{N}$" (i.e. $k = k + 1$ holds) in the sense of [1]. We apply these results to (1) obtain a generalization of Parikh's theorem, (2) to compute the provenance of Datalog queries, and (3) to analyze weighted pushdown systems. We further show how to compute Newton's method over any ω-continuous semiring.

1 Introduction

Fixed-point iteration is a standard approach for solving equation systems of the form $\boldsymbol{X} = F(\boldsymbol{X})$: The naive approach is to compute the sequence $\boldsymbol{X}_{i+1} = F(\boldsymbol{X}_i)$ given some suitable initial approximation $\boldsymbol{X}_0$. In calculus Banach's fixed-point theorem guarantees that the constructed sequence converges to a solution if F is a contraction over a complete metric space. In computer science, Kleene's fixed-point theorem[1] guarantees convergence if F is an ω-continuous map over a complete partial order. In reference to Kleene's fixed-point theorem, we will call the naive application of fixed-point iteration "Kleene's method" in the following. It is well-known that Kleene's method converges only very slowly in general. Consider the equation $X = 1/2X^2+1/2$ over the reals. Kleene's method $\boldsymbol{\kappa}^{(h+1)} = 1/2(\boldsymbol{\kappa}^{(h)})^2 + 1/2$ converges from below to the only solution $x = 1$ starting from the initial approximation $\boldsymbol{\kappa}^{(0)} = 0$. However, it takes 2^{h-3} iterations to gain h bits of precision, i.e. $1 - \boldsymbol{\kappa}^{(2^{h-3})} \leq 2^{-h}$ [8].

Therefore, many approximation schemes do not apply Kleene's method, instead they construct from F a new map G to which fixed-point iteration is then applied: Newton's method, for instance, obtains G from a nonlinear function F by linearization. In above example, $F(X) = 1/2X^2 + 1/2$ is replaced by $G(X) = 1/2X + 1/2$ yielding the sequence $\boldsymbol{\nu}^{(h+1)} = G(\boldsymbol{\nu}^{(h)}) = 1 - 2^{-h}$ for $\boldsymbol{\nu}^{(0)} = 0$, i.e. we get one bit of precision with each iteration.

* This work was partially funded by the DFG project "Polynomial Systems on Semirings: Foundations, Algorithms, Applications"

[1] Depending on literature, this result is also attributed to Tarski.

A system $\boldsymbol{X} = F(\boldsymbol{X})$ where F is given in terms of polynomials over a semiring is called algebraic. In computer science, algebraic systems arise e.g. in the analysis of procedural programs where their least solution describes the set of runs of the program (possibly evaluated under a suitable abstraction). Motivated by the fast convergence of Newton's method over the reals, in [5, 6] (see [7] for an updated version) Newton's method was extended to algebraic systems over ω-continuous semirings: It was shown there that Newton's method always converges monotonically from below to the least solution at least as fast as Kleene's method. In particular, there are semirings where Newton's method converges within a finite number of iterations while Kleene's method does not. This extension of Newton's method found several applications in verification (see e.g. [7, 4, 11]). Independent of the mentioned work, the same extension of Newton's method has been proposed in [17] in the setting of combinatorics which led to new efficient algorithms for random generation of objects.

In this article we give a lower bound on the speed at which Newton's method converges over arbitrary *commutative* ω-continuous semirings. We measure the speed by essentially looking at the number of terms evaluated by Newton's method. To make this more precise, consider the equation $X = aX^2 + c$ in the formal parameters a, c (e.g. over the semiring of formal power series). Its least solution is the series $B = \sum_{n\in\mathbb{N}} C_n a^n c^{n+1}$ with C_n the n-th Catalan number.

The Kleene approximations $\boldsymbol{\kappa}^{(h+1)} := a\boldsymbol{\kappa}^{(h)}\boldsymbol{\kappa}^{(h)} + c$ of B are always polynomials and one can show that the number of correctly computed coefficients increases by one in each iteration, e.g. $\boldsymbol{\kappa}^{(3)} = c + ac^2 + 2a^2c^3 + a^3c^4$. By contrast, the Newton approximations $\boldsymbol{\nu}^{(h)}$ are (infinite) power series. It follows easily from the characterization [5] of the Newton approximations by "tree-dimension" (see Sec. 3), that the coefficient of $a^n c^{n+1}$ in $\boldsymbol{\nu}^{(h)}$ has converged to C_n if and only if $n + 1 < 2^h$, i.e. the number of coefficients which have converged is now roughly doubled in each iteration. In [17] this property is called *quadratic convergence* (see also Ex. 5) and is used there to argue that Newton's method allows to efficiently compute a finite number of coefficients of the formal power series representing a generating function.

In programs analysis, monomials correspond to runs of a program and we are in general not only interested in the coefficients of a finite number of monomials. We show in Theorem 6 for *any* monomial m that either its coefficient in $\boldsymbol{\nu}^{(n+k+1)}$ has already converged or it is bounded from below by 2^{2^k} (where n is the number of variables of the given algebraic system). In particular, if the coefficient of m is less than 2^{2^k} in $\boldsymbol{\nu}^{(n+k+1)}$, then we know that it has converged. Using this theorem, we extend Parikh's theorem[2] to multiplicities bounded by a given $k \in \mathbb{N}$ (see Sec. 5.1). From this it follows that the set of monomials whose coefficients have converged in the h-th Newton approximation is Presburger definable. In Sec. 5.2 we apply these results to the problem of computing the provenance of a Datalog query improving on the algorithms proposed in [12]. As a further application of our results, we show in Sec. 5.3 how Newton's method

[2] Parikh's theorem states that the commutative image of a context-free grammar is a semilinear set, i.e. definable by means of a Presburger formula

by virtue of Theorem 6 can be used to speed up the computation of predecessors and successors in weighted pushdown-systems [18] which has applications e.g. in the analysis of procedural programs or generalized authorization problems in SPKI/SDSI. As a side result, we also show how to compute Newton's method for algebraic systems over arbitrary, also noncommutative, ω-continuous semirings (Sec. 3, Definition 2). Due to the page limit, we refer the reader to the technical report [13] for the missing proofs.

2 Preliminaries

$\mathbb{N}$ denotes the nonnegative integers (natural numbers). $\mathbb{N}_\infty$ are the natural numbers extended by a greatest element ∞. For $k \in \mathbb{N}$ let $\mathbb{N}_k = \{0, 1, \ldots, k\}$. A^* ($\mathsf{A}^\oplus$) denotes the free (commutative) monoid generated by A. Elements of $\mathsf{A}^\oplus$ are usually written as monomials (in the variables A). $\mathbb{N}_\infty\langle\langle \mathsf{A}^* \rangle\rangle$ denotes the set of all total functions from A^* to $\mathbb{N}_\infty$. These functions are commonly represented a formal power series (in noncommuting variables A and coefficients in $\mathbb{N}_\infty$). Analogously for $\mathbb{N}_\infty\langle\langle \mathsf{A}^\oplus \rangle\rangle$ with now commuting variables.

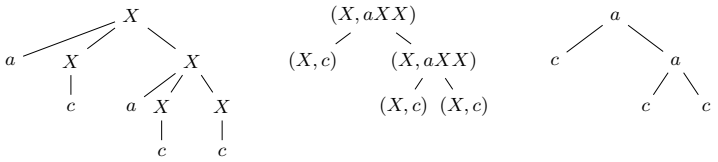
A context-free grammar is a triple $G = (\mathcal{X}, \mathsf{A}, R)$ with variables (nonterminals) $\mathcal{X}$, alphabet (formal parameters) A, and (rewrite) rules R. We do not assume a specific start symbol. G is nonexpansive if no variable $X \in \mathcal{X}$ can be rewritten into a sentential form in which X occurs at least twice (see e.g. [19]). G is in quadratic normal form if any rule $X \to u_0 X_1 u_1 \ldots u_{r-1} X_r u_r$ of G satisfies $u_0 u_1 \ldots u_r \in \mathsf{A}^+$, $X_1 X_2 \ldots X_r \in \mathcal{X}^+$, and $r \in \{0, 2\}$.

We slightly deviate from the standard representation of derivation trees: We label the nodes of a derivation tree directly by the corresponding rule (see Example 1). For $X \in \mathcal{X}$, a derivation tree of G is an X-tree if its root is labeled by a rule rewriting X. The word represented by a derivation tree is called its yield. The ambiguity of a context-free grammar G w.r.t. to $X \in \mathcal{X}$ is the map $\mathsf{amb}_X \in \mathbb{N}_\infty\langle\langle \mathsf{A}^* \rangle\rangle$ which assigns to a word $w \in \mathsf{A}^*$ the number of X-trees of G which yield w. Analogously, we define the *commutative* ambiguity $\mathsf{camb}_X \in \mathbb{N}_\infty\langle\langle \mathsf{A}^\oplus \rangle\rangle$ which assigns to each monomial $m \in \mathsf{A}^\oplus$ the number of X-trees of G which yield a permutation of m. G is unambiguous w.r.t. X if every word has a unique X-tree, i.e. if amb_X takes only values in $\{0, 1\}$.

The family $\mathsf{amb} = (\mathsf{amb}_X \mid X \in \mathcal{X})$ can equally be characterized as the least solution of the algebraic system $\boldsymbol{X} = F_G(\boldsymbol{X})$ over $\mathbb{N}_\infty\langle\langle \mathsf{A}^* \rangle\rangle$ consisting of the equations $X = \sum_{(X,\gamma)\in P} \gamma$. In particular, for any interpretation $\iota\colon \mathsf{A} \to S$ of the alphabet symbols as elements of some ω-continuous semiring $\langle S, +, \cdot \rangle$ it is known [3, 7] that amb evaluates under (the ω-continuous homomorphism induced by) ι to the least solution of the algebraic system $\boldsymbol{X} = F^\iota(\boldsymbol{X})$ over S where F^ι is obtained from F by substituting every occurrence of $a \in \mathsf{A}$ by $\iota(a)$. Similarly, any approximation scheme for amb translates to an approximation scheme for $\iota(\mathsf{amb})$ over S. As we can associate with any algebraic system $\boldsymbol{X} = F(\boldsymbol{X})$ over $\langle S, +, \cdot \rangle$ a context-free grammar (in the restricted from defined above) such that $\boldsymbol{X} = F_G(\boldsymbol{X})$ has the same least solution, it suffices to study how to approximate amb. Analogously for a commutative semiring $\langle S, +, \cdot \rangle$ and camb. We therefore

do not introduce ω-continuous semirings and algebraic systems formally, but refer the reader to e.g. [19].

Example 1. Consider the grammar $G_L\colon X \to aXX \mid c$. The language $L(G_L)$ generated by G_L is known as Lukasiewicz language of all proper[3] binary trees with binary nodes labeled by a, and leaves labeled by c represented as a word using Polish notation. Below on the left the common depiction of the derivation tree of *acacc* is shown; the middle tree is the representation used in the following which is isomorphic to the binary tree represented by *acacc* shown on the right:



As G_L is unambiguous, amb enumerates all proper binary trees. camb on the other hand is the generating function of proper binary trees, i.e. $\mathsf{camb}(a^n c^{n+1})$ is the n-th Catalan number C_n.

$$\mathsf{camb} = c+ac^2+2a^2c^3+5a^3c^4+14a^4c^5+42a^5c^6+132a^6c^7+429a^7c^8+1430a^8c^9+\dots$$

3 Newton's Method for Context-Free Grammars

The Kleene approximation $\boldsymbol{\kappa}^{(h)}$ of amb ($\boldsymbol{\kappa}^{(h+1)} = F_G(\boldsymbol{\kappa}^{(h)})$ with $\boldsymbol{\kappa}^{(0)} = 0$) can be characterized by means of the derivation trees evaluated by them (see e.g. [5]): The X-component $\boldsymbol{\kappa}_X^{(h)}$ of $\boldsymbol{\kappa}^{(h)}$ assigns to $w \in \mathsf{A}^*$ the number of X-trees of *height less than* h which yield w. In [6, 5] the notion of dimension was introduced to give a similar characterization of the Newton approximations $\boldsymbol{\nu}^{(h)}$: The *dimension* of a (rooted) tree t is the maximal height of any perfect[4] binary tree which is a minor of t. The dimension is also known as Horton-Strahler number or register number [9]. Then $\boldsymbol{\nu}_X^{(h)}$ assigns to $w \in \mathsf{A}^*$ the number of X-trees *of dimension less than* h which yield w. Analogously for camb. We use this result to unfold any context-free grammar G w.r.t. to the dimension into a new context-free grammar $G^{(h)}$ so that the (commutative) ambiguity of $G^{(h)}$ is exactly the h-th Newton approximation of the (commutative) ambiguity of G. One advantage of this new definition is that it allows to effectively compute Newton's method over any ω-continuous semiring for which we can compute the semiring operations and the Kleene star. By contrast, the algebraic definition in [6, 5] requires the user to find in every iteration step a certain semiring element. There, only for particular semirings, e.g. when addition is idempotent, it was shown how to construct these elements. For the unfolding we assume that G is in quadratic normal form. This is no real restriction but simplifies the presentation.[5]

[3] A binary tree is proper if every node is either binary or nullary.
[4] A proper binary tree is perfect if every leaf has the same distance to the root.
[5] See the technical report [13] for how to unfold arbitrary context-free grammars.

Definition 2. *Let G be a context-free grammar $G = (\mathcal{X}, \mathsf{A}, R)$. Set $\mathcal{X}^\nu := \{X^{(d)}, \hat{X}^{(d)} \mid X \in \mathcal{X}, d \in \mathbb{N}\}$. The unfolding $G^\nu = (\mathcal{X}^\nu, \mathsf{A}, R^\nu)$ of G is:*

- *$X^{(d)} \to \hat{X}^{(e)}$ for every $d \in \mathbb{N}$, and every $0 \leq e < d$.*
- *If $X \to u_0$ in R, then $\hat{X}^{(0)} \to u_0$.*
- *If $X \to_G u_0 X_1 u_1 X_2 u_2$ in R, then for every $d \geq 1$:*

$$\begin{aligned}\hat{X}^{(d)} &\to u_0 X^{(d)} u_1 \hat{X}^{(d)} u_2\\ \hat{X}^{(d)} &\to u_0 \hat{X}^{(d)} u_1 X^{(d)} u_2\\ \hat{X}^{(d)} &\to u_0 \hat{X}^{(d-1)} u_1 \hat{X}^{(d-1)} u_2.\end{aligned}$$

For any given $h \in \mathbb{N}$ let $G^{(h)} = (\mathcal{X}^{(h)}, \mathsf{A}, R^{(h)})$ be the context-free grammar induced by the variables $\{X^{(h)} \mid X \in \mathcal{X}\}$. The h-th Newton approximation $\boldsymbol{\nu}_X^{(h)}$ of the (commutative) ambiguity of G w.r.t. X is the (commutative) ambiguity of $G^{(h)}$ w.r.t. $X^{(h)}$.

Lemma 3. *Every $\hat{X}^{(d)}$-tree ($X^{(d)}$-tree) has dimension exactly (less than) d. There is a yield-preserving bijection between the $\hat{X}^{(d)}$-trees ($X^{(d)}$-trees) and the X-trees of dimension exactly (less than) d.*

Newton's method is closely related to nonexpansive grammars and related notions like quasi-rational languages:

Theorem 4. *Let $G = (\mathcal{X}, \mathsf{A}, R)$ be a context-free grammar.*

1. *All Newton approximations of camb are rational in $\mathbb{N}_\infty\langle\langle \mathsf{A}^\oplus \rangle\rangle$.*
2. *Newton's method converges to amb (camb) of G within a finite number of iterations if and only if G is nonexpansive. If G is nonexpansive, then Newton's method converges within $|\mathcal{X}|$ iterations.*

If G is expansive, not much can be said regarding convergence speed in the noncommutative setting as illustrated by any unambiguous grammar G: For a given $w \in L(G)$, the least h with $\boldsymbol{\nu}_X^{(h)}(w) = \mathsf{amb}_X(w)$ is simply the dimension of the unique X-tree yielding w. Thus, in the following section we focus on the commutative setting and study the speed at which Newton's method converges to camb by means of a lower bound on all coefficients which have not yet converged.

Example 5. Unfolding G_L (see Ex. 5) w.r.t. the dimension gives us $\hat{X}^{(0)} \to c$, $X^{(1)} \to \hat{X}^{(0)}$ and for $d > 0$

$$\begin{aligned}X^{(d)} &\to \hat{X}^{(0)} \mid \hat{X}^{(1)} \mid \ldots \mid \hat{X}^{(d-1)}\\ \hat{X}^{(d)} &\to aX^{(d)}\hat{X}^{(d)} \mid a\hat{X}^{(d)}X^{(d)} \mid a\hat{X}^{(d-1)}\hat{X}^{(d-1)}\end{aligned}$$

Modulo commutativity, we can deduce from this the following rational expressions for the first few approximations of camb: $\boldsymbol{\nu}^{(0)} = 0$, $\boldsymbol{\nu}^{(1)} = c$,

$$\begin{aligned}\boldsymbol{\nu}^{(2)} &= (2ac)^*ac^2 + c\\ &= c + ac^2 + 2a^2c^3 + 4a^3c^4 + \ldots\\ \boldsymbol{\nu}^{(3)} &= (2a((2ac)^*ac^2 + c))^*a((2ac)^*ac^2)^2\\ &= c + ac^2 + 2a^2c^3 + 5a^3c^4 + 14a^4c^5 + 42a^5c^6 + 132a^6c^7 + 428a^7c^8 + \ldots\end{aligned}$$

We have expanded the series until the first coefficient which differs from camb (see Ex. 1) to exemplify the notion of quadratic convergence introduced in [17]: $\boldsymbol{\nu}^{(h)}$ differs from camb in the coefficient of $a^n c^{n+1}$ if and only if $n+1 \geq 2^h$ as any tree with less than 2^h leaves can only have dimension at most $h-1$. This also shows that Newton's method cannot converge faster than quadratic in this sense. Note that although Newton's method converges quadratically w.r.t. camb, it only converges linearly over the reals: Consider G_L interpreted as an algebraic system over $\mathbb{R}$ with $\iota(a) = \iota(c) = 1/2$ yielding $X = 1/2X^2 + 1/2$. By also reading the unfolded grammar as an algebraic system and interpreting the alphabet by the same ι we recover the Newton approximations over $\mathbb{R}$: $X^{(0)} = 0$, $\hat{X}^{(0)} = 1/2$, and for $d > 0$:

$$X^{(d)} = X^{(d-1)} + \hat{X}^{(d-1)} \text{ and } \hat{X}^{(d)} = (1 - X^{(d)})^{-1} \cdot 1/2 \left(\hat{X}^{(d-1)}\right)^2$$

Induction shows that indeed $\iota(\boldsymbol{\nu}^{(h)}) = X^{(h)} = 1 - 2^{-h}$.

4 Rate of Convergence Modulo Commutativity

Let $G = (\mathcal{X}, \mathsf{A}, R)$ be a context-free grammar. In the following n denotes $|\mathcal{X}|$ and $\boldsymbol{\nu}^{(h)}$ denotes the h-th Newton approximation of camb of G, i.e. $\boldsymbol{\nu}_X^{(h)} = \mathsf{camb}_{X^{(h-1)}}$. We say that two X-trees (w.r.t. G) are *Parikh-equivalent* if they yield the same word up to commutativity. We show that after $n+1$ iterations all coefficients which have not converged yet are bounded from below by 2^{2^k}.

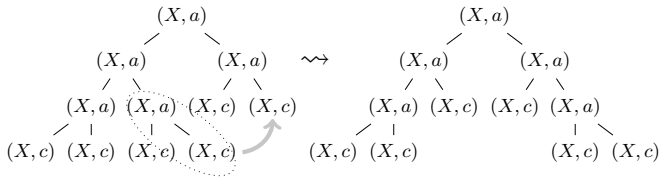
Theorem 6. *For all $k \geq 0$ and $\boldsymbol{v} \in \mathsf{A}^{\oplus}$: $\boldsymbol{\nu}_X^{(n+k+1)}(\boldsymbol{v}) \geq \min(\mathsf{camb}_X(\boldsymbol{v}), 2^{2^k})$.*

Proof (sketch). Assume there is $\boldsymbol{v} \in \mathsf{A}^{\oplus}$ with $\boldsymbol{\nu}_X^{(n+k)}(\boldsymbol{v}) < \mathsf{camb}_X(\boldsymbol{v})$. This means there exists some derivation tree t with dimension $\mathsf{dim}(t) \geq n+k+1$ and yield $\boldsymbol{v}$ modulo commutativity. Essentially we show that t witnesses the existence of at least 2^{2^k} different, but Parikh-equivalent trees of lower dimension.

To make this more precise, we need to introduce $l(t)$: Recall that we labeled the nodes of derivation trees by rules of G. A variable Y is a label of t if there is at least one node which is labeled by a rule rewriting Y. Then $l(t)$ is the number of variables labeling t. We prove by induction on the number of vertices of t that if $\mathsf{dim}(t) \geq l(t) + k + 1$, then there exist at least 2^{2^k} Parikh-equivalent trees of dimension at most $l(t) + k$.

Assume that t has dimension $l(t) + k + 1$ and exactly two subtrees t_1, t_2 having dimension exactly $l(t) + k$ and furthermore $l(t_1) = l(t_2) = l(t)$ (all other cases reduce to this one, or follow from the induction hypothesis). Since t_1 has dimension $l(t) + k$ it contains a perfect binary tree of height $l(t) + k$ as a minor. The set of nodes of this minor on level k define 2^k (independent) subtrees of t_1. Each of these 2^k subtrees has height at least $l(t)$, and thus by the Pigeonhole principle contains a path with two variables repeating. We call the partial derivation tree defined by these two repeating variables a *pump-tree*. We relocate any subset of these 2^k pump-trees to t_2 which is possible since $l(t_2) = l(t) = l(t_1)$.

See the following picture for an illustration of the relocation process (we have two choices for the pump-tree on the left, yielding four possible "remainders").



Each of these 2^{2^k} choices produces a different tree $\tilde{t}$—the trees differ in the subtree $\tilde{t_1}$. We now apply the following result from [6]: For every derivation tree t there is a Parikh-equivalent tree $\tilde{t}$ of dimension at most $l(t)$. Applying this result to $\tilde{t_2}$ allows us to reduce the dimension of each $\tilde{t}$ to at most $\mathsf{dim}(t_1) = l(t) + k$. This way we obtain at least 2^{2^k} different Parikh-equivalent trees of dimension at most $\mathsf{dim}(t_1) = l(t) + k$.

Remark 7. As we can also choose t_2 as the source and t_1 as the destination of the relocation process, we obtain in fact a lower bound of 2^{1+2^k}, which is best possible (in this form): Looking at Ex. 5 for $k = 0$ we obtain a lower bound of $\boldsymbol{\nu}^{(2)}(v) \geq 4$ for all coefficients that have not converged yet – and indeed $\boldsymbol{\nu}^{(2)}(a^3c^4) = 4$.

It would be nice to have a non-uniform global bound on the coefficients $\boldsymbol{\nu}^{(n+1+k)}(v)$ (i.e. some bound that depends on k and $|v|$). However, the following grammar H shows that this cannot be done without taking into account the structure of the grammar: $H : Y \to BY \mid BX, B \to b, X \to aXX \mid c$. This grammar contains G_L, but any word produced by Y can have an arbitrarily long prefix of b's and each such prefix has a unique derivation. Thus $\mathsf{camb}_Y(b^m a^n c^{n+1}) = \mathsf{camb}_X(a^n c^{n+1}) = C_n$.

We say that a ω-continuous semiring S is *collapsed* at some positive integer k if in S the identity $k = k + 1$ holds (see e.g. [1]). For instance, the semirings $\mathbb{N}_k\langle\langle \mathsf{A}^* \rangle\rangle$ and $\mathbb{N}_k\langle\langle \mathsf{A}^{\oplus} \rangle\rangle$ are collapsed at k. For $k = 1$ the semiring is idempotent.

Corollary 8. *Newton's method converges within $n + \log\log k$ iterations for any algebraic system with n variables over a commutative semiring collapsed at k.*

5 Applications

5.1 Parikh's Theorem for Bounded Multiplicities

Petre [15] defines a hierarchy of power series over $\mathbb{N}_\infty\langle\langle \mathsf{A}^{\oplus} \rangle\rangle$ and showed that this hierarchy is strict. In particular he shows that Parikh's Theorem does not hold if multiplicities are considered. Here we combine our convergence result and some identities for weighted rational expressions over commutative k-collapsed semirings to show that moving from $\mathbb{N}_\infty\langle\langle \mathsf{A}^{\oplus} \rangle\rangle$ to $\mathbb{N}_k\langle\langle \mathsf{A}^{\oplus} \rangle\rangle$ allows us to prove a Parikh-like theorem, i.e. we give a semilinear characterization of camb_G.

In the following, let k denote a fixed positive integer. By Theorem 4 and Corollary 8 we know that camb_G is rational modulo $k = k+1$. In the idempotent setting ($k = 1$), see e.g. [16] the identities (i) $(x^*)^* = x^*$, (ii) $(x+y)^* = x^*y^*$, and (iii) $(xy^*)^* = 1 + xx^*y^*$ can be used to transform any regular expression into a regular expression in "semilinear normal form" $\sum_{i=1}^{r} w_{i,0}w_{i,1}^* \dots w_{i,l_r}^*$ with $w_{i,j} \in \mathsf{A}^*$. It is not hard to deduce the following identities over $\mathbb{N}_k\langle\langle \mathsf{A}^{\oplus} \rangle\rangle$ where $x^{<r}$ abbreviates the sum $\sum_{i=0}^{r-1} x^i$. By $\mathsf{supp}(x)$ we denote the characteristic series of the support of x:

Lemma 9. *The following identities hold over* $\mathbb{N}_k\langle\langle \mathsf{A}^{\oplus} \rangle\rangle$*:*

$$
\begin{aligned}
&(\mathsf{I1}) && kx && = k\,\mathsf{supp}(x)\\
&(\mathsf{I2}) && (\gamma x)^* && = (\gamma x)^{<\lceil \log_\gamma k \rceil} + kx^{\lceil \log_\gamma k \rceil}x^*\\
&(\mathsf{I3}) && (x^*)^* && = kx^*\\
&(\mathsf{I4}) && (x+y)^* && = (x+y)^{<k} + x^kx^* + y^ky^* + kxy(x+y)^{\max(k-2,0)}x^*y^*\\
&(\mathsf{I5}) && (xy^*)^* && = 1 + xy^* + x^2x^* + x^2y\textstyle\sum_{0\le m,j<k-2}\binom{2+m+j}{1+j}x^my^j\\
& && && \quad + kx^2y(x^{\max(k-2,0)} + y^{\max(k-2,0)})x^*y^*
\end{aligned}
$$

for γ any integer greater than one.

Consider a rational series $\mathfrak{r} \in \mathbb{N}_k\langle\langle \mathsf{A}^{\oplus} \rangle\rangle$ represented by the rational expression ρ. The above identities, where $(\mathsf{I3}), (\mathsf{I4}), (\mathsf{I5})$ generalizes (i), (ii), (iii), respectively, allow us to reduce the star height of ρ to at most one by distributing the Kleene stars over sums and products yielding a rational expression ρ' of the form $\rho' = \sum_{i=1}^{s} \gamma_i w_{i,0}w_{i,1}^* \dots w_{i,l_i}^*$ ($w_{i,j} \in \mathsf{A}^*, \gamma_i \in \mathbb{N}_k$) which still represents $\mathfrak{r}$ over $\mathbb{N}_k\langle\langle \mathsf{A}^{\oplus} \rangle\rangle$. By $(\mathsf{I1})$ we know that, if $\gamma_{i,0} = k$, we may replace $w_{i,0}w_{i,1}^* \dots w_{i,l_i}^*$ by its support which is a linear set in $\mathbb{N}^{\mathsf{A}}$.

Theorem 10. *Every rational $\mathfrak{r} \in \mathbb{N}_k\langle\langle \mathsf{A}^{\oplus} \rangle\rangle$ can be represented as a finite sum of weighted linear sets, i.e. $\mathfrak{r} = \sum_{i\in[s]} \gamma_i\,\mathsf{supp}(w_{i,0}w_{i,1}^* \dots w_{i,l}^*)$ with $w_{i,j} \in \mathsf{A}^*$ and $\gamma_i \in \mathbb{N}_k$.*

Example 11. The rational expression $\rho = (a+2b)^*$ represents the power series $\sum_{i,j\in\mathbb{N}} 2^j a^i b^j$ in $\mathbb{N}_\infty\langle\langle \mathsf{A}^{\oplus} \rangle\rangle$. Computing over $\mathbb{N}_2\langle\langle \mathsf{A}^{\oplus} \rangle\rangle$ we may transform ρ as follows:

$$(a+2b)^* \overset{(\mathsf{I4})}{=} (a+2b)^{<2} + a^2a^* + (2b)^2(2b)^* + 2a(2b)a^*(2b)^* = a^* + 2(bb^* + aba^*b^*) = a^* + 2(bb^*a^*) \overset{(\mathsf{I1})}{=} 1\,\mathsf{supp}(a^*) + 2\,\mathsf{supp}(bb^*a^*)$$

Corollary 12. *For every $k \in \mathbb{N}_\infty$ we can construct a formula of Presburger arithmetic that represents the set $\{\boldsymbol{v} \in \mathbb{N}^{\mathsf{A}} \mid \mathsf{camb}_{G,X}(\boldsymbol{v}) = k\}$.*

This corollary can be applied to inclusion testing between two rational series over $\mathbb{N}_k\langle\langle \mathsf{A}^{\oplus} \rangle\rangle$ which is relevant e.g. for detecting early convergence of Newton's method, i.e. if $\boldsymbol{\nu}^{(h+1)} = \boldsymbol{\nu}^{(h)}$. Although we know that after $n + \log\log k$ steps the method has converged, in applications (see Sec. 5.2) n could be quite large and the $n + \log\log k$ bound might be too pessimistic.

5.2 Provenance Computation for Datalog

Roughly speaking, provenance is additional information attached to the results of a database query explaining how said results were obtained from the current facts in the database. Provenance information is important e.g. to implement updatable views [10]. Recently, commutative ω-continuous semirings were proposed as provenance annotations where the provenance of unions or projections is modelled by addition of the annotation and joins yield multiplications. Tagging the tuples from the facts in the database allows us to trace back the provenance of the results by solving an algebraic system [12].

For an example consider the binary relation E depicted below (first table). The Datalog query $T(x,y)$:- $E(x,y)$; $T(x,y)$:- $E(x,z), E(z,y)$ computes its transitive closure $T = E^*$ (second table).

X	Y	
a	b	e_1
b	b	e_2
b	c	e_3
c	d	e_4

X	Y	
a	b	$X_1 = e_1 + X_1X_4$
a	c	$X_2 = X_1X_5$
a	d	$X_3 = X_1X_6 + X_2X_7$
b	b	$X_4 = e_2 + X_4X_4$
b	c	$X_5 = e_3 + X_4X_5$
b	d	$X_6 = X_4X_6 + X_5X_7$
c	d	$X_7 = e_4$

$$
\left.\begin{array}{ll}
X_1 = X_4^* e_1 & e_1 e_2^* \\
X_2 = (X_4^*)^2 e_1 e_3 & e_1 e_2^* e_3 \\
X_3 = [(X_4^*)^2 + (X_4^*)^3] e_1 e_3 e_4 & e_1 e_2^* e_3 e_4 \\
X_4 = \sum_{n\geq 0} C_n (e_2)^{n+1} & \overset{(1=1+1)}{=} e_2 e_2^* \\
X_5 = X_4^* e_3 & e_3 e_2^* \\
X_6 = (X_4^*)^2 e_3 e_4 & e_2^* e_3 e_4 \\
X_7 = e_4 & e_4
\end{array}\right\}
$$

To capture the so called "how-provenance" we tag every tuple in E by a letter from $\Sigma = \{e_1, e_2, e_3, e_4\}$. The provenance of the k-th tuple in T is the value of X_k in the (least) solution (over a suitable semiring) of the algebraic system representing the query. In our example the solution over $\mathbb{N}\langle\langle \mathsf{A}^{\oplus}\rangle\rangle$ can be computed by hand and we can also give a very short representation as rational expressions if we assume idempotence of addition ($1 = 1 + 1$). From the result we can see that the tuple (b,d) can be obtained by a join of (b,c) and (c,d), preceded by any number of joins of (b,b) with itself [6].

Depending on our choice of the semiring we obtain a coarser or finer view on the provenance. As $\mathbb{N}_\infty\langle\langle \mathsf{A}^{\oplus}\rangle\rangle$ is the commutative semiring, freely generated by A, we can regard it as the universal provenance semiring [12]. However, $\mathbb{N}_\infty\langle\langle \mathsf{A}^{\oplus}\rangle\rangle$ is in some sense a bad choice for representing solutions, as we cannot do this finitely. Green et al. [12] therefore resort to compute the complete provenance series only if they finite by enumerating all derivation trees using Kleene's method essentially; if the power series is an infinite sum they only compute the coefficient for a given monomial.

For many applications, idempotent semirings suffice to capture interesting provenance information. Useful examples are the tropical semiring $\langle \mathbb{N}_\infty, \min, + \rangle$, or the Viterbi-semiring $\langle [0,1], \max, \cdot \rangle$ for probabilistic settings. [12] raised the open question how to compute provenance over the tropical semiring, which can be done by Newton's method as already described in [6]. A useful generalization which is not idempotent is the k-tropical semiring $\mathcal{T}_k$ [14] which was used there for general k-shortest distance computations. This semiring satisfies the identity

[6] More precisely, the operations are joins followed by projections.

$k = k+1$, so by our results Newton's method can be used to calculate provenance series over $\mathcal{T}_k$ in $n + \log\log k$ steps.

As already remarked in [12], idempotent semirings are often too coarse an abstraction in a database context where one often considers the so called *bag-semantics* (i.e. we also care about the multiplicities of query results or provenance information). The k-collapsed semirings $\mathbb{N}_k\langle\langle \mathsf{A}^{\oplus}\rangle\rangle$ are a possible way out of the dilemma that we want to capture the bag-semantics to some extent but cannot use the most general semiring $\mathbb{N}_\infty\langle\langle \mathsf{A}^{\oplus}\rangle\rangle$ since its elements are not finitely representable in general. Suppose, we want to compute provenance for a recursive query and are satisfied with a power series having coefficients less than $k = 2^{64}$ (i.e. standard 64-bit integers). By Theorem 6 we know that Newton's method converges after at most $n + 6$ steps.

5.3 Analysis of Weighted Pushdown Systems

A Pushdown system (PDS) $\langle Q, \Gamma, \Delta\rangle$ consists of a finite set of control states Q, a finite set of stack symbols Γ, and a set of rewrite rules $\Delta \subset Q\Gamma \to Q\Gamma^{\leq 2}$. A PDS induces an infinite graph over the $Q\Gamma^*$ of configurations: there is an edge from $q\gamma$ to $q'\gamma'$ if there is a rule $qA \to q'\rho \in \Delta$ such that $\gamma' = A\gamma''$ and $\gamma' = \rho\gamma''$. In a weighted PDS each rule carries also as weight an element of a semiring $\langle S, +, \cdot\rangle$. The semiring multiplication is used to *extend* weights from single rules to paths, while addition is used to *combine* the weight of several paths. Such weighted graphs arise e.g. in the analysis of procedural programs [18] or in authorization problems [20]. A central problem is: given a configuration c of the graph, determine for any other configuration c' the weight of all finite paths leading from c to c'.

To solve this problem for arbitrary configurations, one builds a weighted finite automaton whose transitions corresponds to particular runs starting in a configuration pA with a single stack symbol and ending in a configuration $q\varepsilon$ with empty stack. The total weight of these paths is the least solution of an algebraic system over the given semiring S. In the standard approach [18] this algebraic system is solved on the fly while constructing the automaton. For this a work list variant of Kleene's method is used. This approach therefore only works for certain semirings and its running time is directly proportional to the number of iterations needed by Kleene's method to converge which depends on the given semiring. Alternatively, as discussed in [2], one can first build the unweighted automaton, and then solve the algebraic system explicitly. We give an example how Newton's method in combination with Theorem 6 allows to speed this up:

Consider the PDS $pA \xrightarrow{a} pAA$, $pA \xrightarrow{b} q$, and $qA \xrightarrow{c} p$ where we have assigned a unique label (weight) to each rule. The PDS encodes a program which always starts in the configuration pA, and we expect it to terminate in $p\varepsilon$. Termination in configuration $q\varepsilon$ is considered to be an error. To simplify debugging, we would like to have, say the k paths from pa to $q\varepsilon$, in particular, these paths should be short. All paths from pa to $p\varepsilon$ resp. $q\varepsilon$ are described by the grammar

$$X \to aXX \mid aYc \text{ and } Y \to aXY \mid b.$$

We first determine the length of the k shortest paths. To this end, we can collapse the alphabet to a singleton, say $\iota(a) = \iota(b) = \iota(c) = z$, and compute the commutative ambiguity of the resulting grammar modulo $k = k + 1$. The coefficient of z^i in camb_X resp. camb_Y then tells us, how many paths (up to k) of length i lead from pA to $p\varepsilon$ resp. $q\varepsilon$. For simplicity, assume $k = 4$. By virtue of Theorem 6 we know that at most $n + 1 + \log\log k = 4$ Newton iterations suffice to compute camb modulo $k = k+1$. (For comparison, Kleene's method can take up to $\mathcal{O}(k)$ iterations, consider e.g. $pA \to pAA, pA \to qA, qA \to q\varepsilon$.) This gives us: $\mathsf{camb}_X = z^3 + 2z^7 + 2z^{11} + \mathcal{O}(z^{12})$ and $\mathsf{camb}_Y = z + z^5 + 3z^9 + \mathcal{O}(z^{10})$. The partial expansion of camb_Y tells us the four shortest paths from pA to $q\varepsilon$ consist of one path of length 1, one path of length 5, and two paths of length 9 each. For constructing the actual paths, these lengths allows us to early discard paths which cannot contribute to the k shortest paths. For instance, we can now apply Kleene's method and discard after each iteration any path of length at least 10. This will take 5 iterations until we have discovered enough paths.

On the other hand, by virtue of Theorem 6 we know that we discover a sufficient number of paths of any given length l when considering only derivation trees of low dimension. Consider e.g. the restriction of the grammar to derivation trees of dimension at most one (see Def. 2). Dimension 0 gives us the shortest path b from pA to $q\varepsilon$. The unfolding of the grammar to dimension exactly 1 is:

$$\begin{aligned}\hat{X}^{(1)} &\to a\hat{X}^{(1)}abc \mid aabc\hat{X}^{(1)} \mid aabcabc \mid a\hat{Y}^{(1)}c\\ \hat{Y}^{(1)} &\to a\hat{X}^{(1)}b \mid aabc\hat{Y}^{(1)} \mid aabcb\end{aligned}$$

Applying Kleene iteration now to this unfolded grammar, we only enumerate trees of dimension 1 with at most 9 leaves. Within two iterations we obtain enough paths, namely $aabcb$, $(aabc)^2b$, $aaaabcbcb$, and $aaabcabcb$, to answer the query. Note that a path of the form $(aabc)^hb$ has a derivation tree of dimension 1, but of height $h+1$, i.e. it takes $h+1$ Kleene iterations on the original grammar to discover this path. By increasing k, the gap between Newton's method and Kleene's method can thus be made arbitrarily large.

6 Future Work

For proper binary trees, [9] provide a closed form for the number of trees with n leaves and dimension less than h, i.e. for $\boldsymbol{\nu}^{(h)}(a^{n-1}c^n)$. They show that the expected dimension of a random binary tree with n leaves is tightly concentrated around $1/2\log_2 n$. This implies a much faster convergence of Newton's method in the case of G_L. We conjecture that a similar result can also be derived for arbitrary context-free grammars.

In the idempotent case, we can use a result of [21] to obtain for a given context-free grammar G a Presburger formula of size linear in $|G|$ defining its Parikh image. It would be interesting, if one could generalize this procedure to semirings collapsed at k as the result of Sec. 5.1 in general leads to very large expressions.

References

1. Bloom, S.L., Ésik, Z.: Axiomatizing rational power series over natural numbers. Inf. Comput. 207(7), 793–811 (2009)
2. Bouajjani, A., Esparza, J., Touili, T.: A generic approach to the static analysis of concurrent programs with procedures. Int. J. Found. Comput. Sci. 14(4), 551– (2003)
3. Bozapalidis, S.: Equational elements in additive algebras. Theory Comput. Syst. 32(1), 1–33 (1999)
4. Esparza, J., Ganty, P., Kiefer, S., Luttenberger, M.: Parikh's theorem: A simple and direct automaton construction. Inf. Process. Lett. 111(12), 614–619 (2011)
5. Esparza, J., Kiefer, S., Luttenberger, M.: An extension of Newton's method to ω-continuous semirings. In: DLT. LNCS, vol. 4588, pp. 157–168. Springer (2007)
6. Esparza, J., Kiefer, S., Luttenberger, M.: On fixed point equations over commutative semirings. In: STACS. LNCS, vol. 4393, pp. 296–307. Springer (2007)
7. Esparza, J., Kiefer, S., Luttenberger, M.: Newtonian program analysis. J. ACM 57(6), 33 (2010)
8. Etessami, K., Yannakakis, M.: Recursive markov chains, stochastic grammars, and monotone systems of nonlinear equations. J. ACM 56(1) (2009)
9. Flajolet, P., Raoult, J.C., Vuillemin, J.: The number of registers required for evaluating arithmetic expressions. Theor. Comput. Sci. 9, 99–125 (1979)
10. Foster, J.N., Karvounarakis, G.: Provenance and data synchronization. IEEE Data Eng. Bull. 30(4), 13–21 (2007)
11. Ganty, P., Majumdar, R., Monmege, B.: Bounded underapproximations. In: CAV. pp. 600–614 (2010)
12. Green, T.J., Karvounarakis, G., Tannen, V.: Provenance semirings. In: PODS. pp. 31–40 (2007)
13. Luttenberger, M., Schlund, M.: An extension of Parikh's theorem beyond idempotence. Tech. rep., TU München (2011), (http://arxiv.org/abs/1112.2864)
14. Mohri, M.: Semiring frameworks and algorithms for shortest-distance problems. J. Autom. Lang. Comb. 7(3), 321–350 (2002)
15. Petre, I.: Parikh's theorem does not hold for multiplicities. J. Autom. Lang. Comb. 4(1), 17–30 (1999)
16. Pilling, D.L.: Commutative regular equations and Parikh's theorem. J. London Math. Soc. pp. 663–666 (1973)
17. Pivoteau, C., Salvy, B., Soria, M.: Algorithms for combinatorial structures: Well-founded systems and newton iterations. J. Comb. Theory, Ser. A 119(8), 1711–1773 (2012)
18. Reps, T.W., Schwoon, S., Jha, S., Melski, D.: Weighted pushdown systems and their application to interprocedural dataflow analysis. Sci. Comput. Program. 58(1-2), 206–263 (2005)
19. Rozenberg, G.: Handbook of formal languages: Word, language, grammar, vol. 1. Springer Verlag (1997)
20. Schwoon, S., Jha, S., Reps, T.W., Stubblebine, S.G.: On generalized authorization problems. In: CSFW. pp. 202–218 (2003)
21. Verma, K.N., Seidl, H., Schwentick, T.: On the complexity of equational horn clauses. In: CADE. LNCS, vol. 3632, pp. 337–352 (2005)