Newton's Method for ω-Continuous Semirings ⋆

Javier Esparza, Stefan Kiefer, Michael Luttenberger

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

Abstract. Fixed point equations $\boldsymbol{X} = \boldsymbol{f}(\boldsymbol{X})$ over ω-continuous semirings are a natural mathematical foundation of interprocedural program analysis. Generic algorithms for solving these equations are based on Kleene's theorem, which states that the sequence $\mathbf{0}, \boldsymbol{f}(\mathbf{0}), \boldsymbol{f}(\boldsymbol{f}(\mathbf{0})), \ldots$ converges to the least fixed point. However, this approach is often inefficient. We report on recent work in which we extend Newton's method, the well-known technique from numerical mathematics, to arbitrary ω-continuous semirings, and analyze its convergence speed in the real semiring.

1 Introduction

In the last two years we have investigated generic algorithms for solving systems of fixed point equations over *ω-continuous semirings* [15]. These semirings provide a nice mathematical foundation for program analysis. A program can be translated (in a syntax-driven way) into a system of $O(n)$ equations over an abstract semiring, where n is the number of program points. Depending on the information about the program one wants to compute, the carrier of the semiring and its abstract sum and product operations can be instantiated so that the desired information is the least solution of the equations. Roughly speaking, the translation maps choice and sequential composition at program level into the sum and product operators of the semiring. Procedures, even recursive ones, are first order citizens and can be easily translated. The translation is very similar to the one that maps a program into a monotone framework [16].

Kleene's fixed point theorem applies to ω-continuous semirings. It shows that the least solution $\mu\boldsymbol{f}$ of a system of equations $\boldsymbol{X} = \boldsymbol{f}(\boldsymbol{X})$ 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's call it *Kleene's method*) to compute or at least approximate $\mu\boldsymbol{f}$. If the domain satisfies what is usually known as the *ascending chain condition*, 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 $\mathbf{0}$, for any ω-continuous semiring, and whatever the shape of $\boldsymbol{f}$ is. On the other hand, its efficiency can be very unsatisfactory. If the ascending chain condition fails, then the sequence of Kleene approximants hardly ever reaches the solution after a finite number of steps. Another problem of the Kleene sequence arises in the area of quantitative program analysis. Quantitative information, like average runtime and probability of termination (for programs with a stochastic component) can also be computed as the least

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

solution of a system of equations, in this case over the semiring of the non-negative real numbers plus infinity. While in these analyses one cannot expect to compute the exact solution by any iterative method (it may be irrational and not even representable by radicals), it is very important to find approximation techniques that converge fast to the solution. However, the convergence of the Kleene approximants can be extremely slow. An artificial but illustrative example is the case of a 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 $\kappa^{(200)} = 0.9990$.

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. 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 i bits of precision require to compute only i approximants, i.e., Newton's method converges exponentially faster than Kleene's in this case. However, 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 when started at a point very close to the zero, or even not converge at all [17].

So there is a puzzling mismatch between the current states of semantics and program analysis on the one side, and numerical mathematics on the other. On ω-continuous semirings, the natural domain of semantics and program analysis, Kleene's method is robust and generally applicable, but inefficient in many cases, in particular for quantitative analyses. On the real field, the natural domain of numerical mathematics, Newton's method can be very efficient, but it is not robust.

We became aware of this mismatch two years ago through the the work of Etessami and Yannakakis on Recursive Markov Chains and our work on Probabilistic Pushdown Automata. Both are abstract models of probabilistic programs with procedures, and their analysis reduces to or at least involves solving systems of fixed point equations. The mismatch led us to investigate the following questions:

- Can Newton's method be generalized to arbitrary ω-continuous semirings?
  I.e., could it be the case that Newton's method is in fact as generally applicable as Kleene's, but nobody has noticed yet?
- Is Newton's method robust when restricted to the real semiring?
  I.e., could it be the case that the difficult behaviour of Newton's method disappears when we restrict its application to the non-negative reals, but nobody has noticed yet?

The answer to both questions is essentially affirmative, and has led to a number of papers [5, 4, 14, 6]. In this note we present the results, devoting some attention to those examples and intuitions that hardly ever reach the final version of a conference paper due to the page limit.

2 From Programs to Fixed Point Equations on Semirings

Recall that a semiring is a set of *values* together with two binary operations, usually called sum and product. Sum and product are associative and have neutral elements 0 and 1, respectively. Moreover, sum is commutative, and product distributes over sum. The *natural order* relation $\sqsubseteq$ on a semiring is defined by setting $a \sqsubseteq a + d$ for every d. A semiring is *naturally ordered* if $\sqsubseteq$ is a partial order.

An ω-*continuous* semiring is a naturally ordered semiring extended by an infinite summation-operator $\sum$ that satisfies some natural properties. In particular, for every sequence $(a_i)_{i\geq 0}$ the supremum $\sup\{\sum_{0\leq i\leq k} a_i \mid k \in \mathbb{N}\}$ w.r.t. $\sqsubseteq$ exists, and is equal to $\sum_{i\in\mathbb{N}} a_i$ [15].

We show how to assign to a procedural program a set of abstract equations by means of an example. Consider the (very abstractly defined) program consisting of three procedures X_1, X_2, X_3, and associate to it a system of equations. For our discussion it is not relevant which is the main procedure. The flow graphs of the procedures are shown in Figure 1. For instance, procedure X_1 can either execute the abstract action b and terminate, or execute a, call itself recursively, and, after the call has terminated, call procedure X_2.

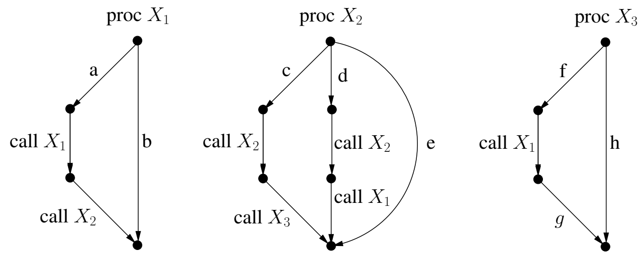

Fig. 1. Flowgraphs of three procedures

We associate to the program the following three abstract equations [1]

$$
\begin{aligned}
X_1 &= a \cdot X_1 \cdot X_2 + b \\
X_2 &= c \cdot X_2 \cdot X_3 + d \cdot X_2 \cdot X_1 + e \\
X_3 &= f \cdot X_1 \cdot g + h
\end{aligned}
\tag{1}
$$

where $+$ and $\cdot$ are the abstract semiring operations, and $\{a, b, \ldots, h\}$ are semiring values. Notice that we slightly abuse language and use the same symbol for a program action and its associated value.

[1] One for each procedure. A systematic translation from programs to equations yields one variable and one equation for each program point. We have not done it in order to keep the number of equations small.

2.1 Some Semiring Interpretations

Many interesting pieces of information about our program correspond to the least solution of the system of equations over different semirings.[2] For the rest of the section let $\Sigma = \{a, b, \ldots, h\}$ 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). A program action $\sigma \in \Sigma$ 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, the system of equations (1) is nothing but the following context-free grammar:

$$\begin{aligned} X_1 &\to aX_1X_2 \mid b \\ X_2 &\to cX_2X_3 \mid dX_2X_1 \mid e \\ X_3 &\to fX_1g \mid h \end{aligned}$$

The least solution of (1) is the triple $(L(X_1), L(X_2), L(X_3))$, where $L(X_i)$ denotes the set of terminating executions of the program with X_i as main procedure, or, in language-theoretic terms, the language of the associated grammar with X_i 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 i-th component of the least solution of (1) is the *summary* relation $R_i(V, V')$ containing the pairs V, V' such that if procedure X_i 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 [2], we abstract an execution $w \in \Sigma^*$ by the vector $(n_a, \ldots, n_h) \in \mathbb{N}^{|\Sigma|}$, where $n_a, \ldots, n_h$ are the number of occurrences of $a, \ldots, h$ in w. We call this vector the *Parikh image* of w. We wish to compute the vector $(P(X_1), P(X_2), P(X_3))$ where $P(X_i)$ contains the Parikh images of the words of $L(X_i)$. It is easy to see that this is the least solution of (1) for the following semiring. The carrier is $2^{\mathbb{N}^{|\Sigma|}}$. The i-th action of Σ is interpreted as the singleton set $\{(0, \ldots, 0, 1, 0 \ldots, 0)\}$ with the "1" at the i-th position. The sum operation is set union, and the product operation is given[3] by

$$U \cdot V = \{(u_a + v_a, \ldots, u_h + v_h) \mid (u_a, \ldots, u_h) \in U, (v_a, \ldots, v_h) \in V\} .$$

[2] This will be no surprise for the reader acquainted with monotone frameworks or abstract interpretation, but the examples will be used throughout the paper.

[3] Abstract interpretation provides a general recipe to define these operators.

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*) [8, 9]. 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 ∞ – if we are instead interested in the probability of the most likely execution, we just have to reinterpret the sum operator as maximum.

As a last example, assume 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)

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

One can easily check that this definition satisfies the semiring axioms. The i-th component of the least solution of (1) is now the pair (t_i, e_i), where t_i is the probability that procedure X_i terminates, and e_i is its conditional expected time.

3 Fixed Point Equations

Fix an arbitrary ω-continuous semiring with a set S of values. We define systems of fixed point equations and present Kleene's fixed point theorem.

Given a finite set $\mathcal{X}$ of variables, a *monomial* is a finite expression

$$a_1 X_1 a_2 \cdots a_k X_k a_{k+1}$$

where $k \geq 0$, $a_1, \ldots, a_{k+1} \in S$ and $X_1, \ldots, X_k \in \mathcal{X}$. 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 *vector* is a mapping $\boldsymbol{v}$ that assigns to every variable $X \in \mathcal{X}$ a value denoted by $\boldsymbol{v}_X$ or v_X, called the X-component of $\boldsymbol{v}$. The value of a monomial $m = a_1 X_1 a_2 \cdots a_k X_k a_{k+1}$ at $\boldsymbol{v}$ is $m(\boldsymbol{v}) = a_1 v_{X_1} a_2 \cdots a_k v_{X_k} a_{k+1}$. The value of a polynomial at $\boldsymbol{v}$ is the sum of the values of its monomials at $\boldsymbol{v}$. A polynomial induces a mapping from vectors to values that assigns to $\boldsymbol{v}$ the vector $f(\boldsymbol{v})$. A vector of polynomials is a mapping $\boldsymbol{f}$ that assigns a polynomial f_X to each variable $X \in \mathcal{X}$; it induces a mapping from vectors to vectors that assigns to a vector $\boldsymbol{v}$ the vector $\boldsymbol{f}(\boldsymbol{v})$ whose X-component is $f_X(\boldsymbol{v})$. A *fixed point of* $\boldsymbol{f}$ is a solution of the equation $\boldsymbol{X} = \boldsymbol{f}(\boldsymbol{X})$.

It is easy to see that polynomials are monotone and continuous mappings w.r.t. $\sqsubseteq$. Kleene's theorem can then be applied (see e.g. [15]), which leads to this proposition:

Proposition 3.1. *A vector $\boldsymbol{f}$ of polynomials has a unique least fixed point $\mu\boldsymbol{f}$ which is the $\sqsubseteq$-supremum of the* Kleene sequence *given by $\boldsymbol{\kappa}^{(0)} = \mathbf{0}$, and $\boldsymbol{\kappa}^{(i+1)} = \boldsymbol{f}(\boldsymbol{\kappa}^{(i)})$.*

4 Newton's Method for ω-Continuous Semirings

We recall Newton's method for approximating a zero of a differentiable function, and apply it to find the least solution of a system of fixed point equations over the reals. Then, we present the generalization of Newton's method to arbitrary ω-continuous semirings we obtained in [5]. We focus on the univariate case (one single equation in one variable), because it already introduces all the basic ideas of the general 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).

We formulate the method in terms of the *differential* of g at a given point v. This is is the mapping $Dg|_v : \mathbb{R} \to \mathbb{R}$ that assigns to each $x \in \mathbb{R}$ a linear function, namely the one corresponding to the tangent of g at v, but represented in the coordinate system having the point $(v, g(v))$ 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 starts at some $\nu^{(0)}$, and computes 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 a 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 for computing the least solution of $f(X) = X$ 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 solving a non-linear system $f(X) = X$ into solving the sequence (3) of linear systems.

4.1 Generalizing Newton's Method

In order to generalize Newton's method to arbitrary ω-continuous semirings we have to overcome two obstacles. First, differentials are defined in terms of derivatives, which are the limit of a quotient of differences. This requires both the sum and product operations to have inverses, which is not the case in general semirings. Second, Equation (3) contains the term $f(\nu^{(i)}) - \nu^{(i)}$, which again seems to be defined only if the sum operation 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 polynomial f over an ω-continuous semiring.

Definition 4.1. *Let f be a polynomial in one variable X over an ω-continuous semiring with carrier S. The* differential of f *at the point v is the mapping $Df|_v : S \rightarrow S$ inductively defined as follows for every $a \in S$:*

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

On commutative semirings, like the real semiring, we have $Df|_v(a) = f'(v) \cdot a$ for all $v, a \in S$, where $f'(v)$ is the derivative of f. This no longer holds when product is not commutative. For a function $f(X) = a_0 X a_1 X a_2$ we have

$$Df|_v(a) = a_0 \cdot a \cdot a_1 \cdot v \cdot a_2 + a_0 \cdot v \cdot a_1 \cdot a \cdot a_2 .$$

The second obstacle. It turns out that the Newton sequence is well-defined if we choose $\nu^{(0)} = f(0)$. More precisely, in [5] we *guess* that this choice will solve the problem, define the Newton sequence, and then *prove* that the guess is correct. The precise guess is that this choice implies $\nu^{(i)} \sqsubseteq f(\nu^{(i)})$ for every $i \geq 0$. By the definition of $\sqsubseteq$, the semiring then contains a value $\delta^{(i)}$ such that $f(\nu^{(i)}) = \nu^{(i)} + \delta^{(i)}$. We can replace $f(\nu^{(i)}) - \nu^{(i)}$ by any such $\delta^{(i)}$. This leads to the following definition:

Definition 4.2. *Let f be a polynomial in one variable over an ω-continuous semiring. The* Newton sequence $(\nu^{(i)})_{i \in \mathbb{N}}$ *is given by:*

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

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

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

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

Notice that for arbitrary semirings the Newton sequence is not unique, since we may have different choices for $\delta^{(i)}$.

The definition can be easily generalized to the multivariate case. Fix a set $\mathcal{X} = \{X_1, \ldots, X_n\}$ of variables. Given a multivariate polynomial f, we define the differential of f at the vector $\boldsymbol{v}$ *with respect to the variable* X by almost the same equations as above:

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

Then the differential of f at the vector $\boldsymbol{v}$ is defined as $Df|_{\boldsymbol{v}} = D_{X_1} f|_{\boldsymbol{v}} + \cdots + D_{X_n} f|_{\boldsymbol{v}}$. Finally, for a vector of polynomials $\boldsymbol{f}$ we set $D\boldsymbol{f}|_{\boldsymbol{v}} = (Df_{X_1}|_{\boldsymbol{v}}, \ldots, Df_{X_n}|_{\boldsymbol{v}})$.

Definition 4.3. *Let $\boldsymbol{f}: V \to V$ be a vector of polynomials. The* Newton sequence *$(\boldsymbol{\nu}^{(i)})_{i\in\mathbb{N}}$ is given by:*

$$\boldsymbol{\nu}^{(0)} = \boldsymbol{f}(\mathbf{0}) \quad \textit{and} \quad \boldsymbol{\nu}^{(i+1)} = \boldsymbol{\nu}^{(i)} + \boldsymbol{\Delta}^{(i)}, \tag{6}$$

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

$$D\boldsymbol{f}|_{\boldsymbol{\nu}^{(i)}}(\boldsymbol{X}) + \boldsymbol{\delta}^{(i)} = \boldsymbol{X}\ . \tag{7}$$

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

Theorem 4.4. *Let $\boldsymbol{f}: V \to V$ be a vector of polynomials. For every ω-continuous semiring and every $i \in \mathbb{N}$:*

- *There exists at least one Newton sequence, i.e., there exists a vector $\boldsymbol{\delta}^{(i)}$ such that $\boldsymbol{f}(\boldsymbol{\nu}^{(i)}) = \boldsymbol{\nu}^{(i)} + \boldsymbol{\delta}^{(i)}$;*
- *$\boldsymbol{\kappa}^{(i)} \sqsubseteq \boldsymbol{\nu}^{(i)} \sqsubseteq \boldsymbol{f}(\boldsymbol{\nu}^{(i)}) \sqsubseteq \mu\boldsymbol{f} = \sup_j \boldsymbol{\kappa}^{(j)}$.*

5 Newton's method on different semirings

In this section we introduce the main results of our study of Newton's method [5, 4, 14, 6] by focusing on three representative semirings: the language, the counting, and the real semiring. We first show that the Newton approximants of the language semiring are the context-free languages of *finite index*, a notion extensively studied in the 60s [20, 11, 19, 12]. We then explain how the algebraic technique for solving fixed point equations over the counting semiring presented by Hopkins and Kozen in [13] is again nothing but a special case of Newton's method. Finally, we show that in the real semiring Newton's method is just as robust as Kleene's.

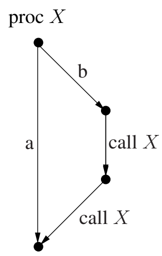

Fig. 2. Flowgraph of a recursive program with one procedure

We present the results for the three semirings with the help of an example. Consider the recursive program from the introduction that can execute action a and terminate, or action b, after which it recursively calls itself twice, see Figure 2. Its corresponding abstract equation is

$$X = a + b \cdot X \cdot X \tag{8}$$

We solve this equation in the three semirings, point out some of its peculiarities, and then introduce the general results.

5.1 The Language Semiring

Consider the language semiring with $\Sigma = \{a, b\}$. Recall that the product operation is concatenation of languages, and hence non-commutative. So we have $Df|_v(X) =$

$bvX + bXv$. It is easy to show that when sum is idempotent the definition of the Newton sequence can be simplified to

$$\nu^{(0)} = f(0) \quad \text{and} \quad \nu^{(i+1)} = \Delta^{(i)}, \tag{9}$$

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

$$Df|_{\nu^{(i)}}(X) + f(\nu^{(i)}) = X \; . \tag{10}$$

For the program of Figure 2 Equation (10) becomes

$$\underbrace{b\nu^{(i)}X + bX\nu^{(i)}}_{Df|_{\nu^{(i)}}(X)} + \underbrace{a + b\nu^{(i)}\nu^{(i)}}_{f(\nu^{(i)})} = X \; . \tag{11}$$

Its least solution, and by (9) the $i{+}1$-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 $S^{(0)} \to a$. Equation (11) allows us to define $G^{(i+1)}$ in terms of $G^{(i)}$, and we get:

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

and it is easy to see that in this case $L(G^{(i)}) \neq L(G^{(i+1)})$ for every $i \geq 0$.

It is well known that in a language semiring, context-free grammars and vectors of polynomials are essentially the same, so we identify them in the following.

We can characterize the Newton approximants of a context-free grammar by the notion of *index*, a well-known concept from the theory of context-free languages [20, 11, 19, 12]. Loosely speaking, a word of $L(G)$ has index i if it can be derived in such a way that no intermediate word contains more than i occurrences of variables.

Definition 5.1. *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_r be the projection of α_r 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.*

Finite-index languages have been extensively investigated under different names by Salomaa, Gruska, Yntema, Ginsburg and Spanier, among others [19, 12, 20, 11](see [10] for historical background). In [4] we show that for a context-free grammar in Chomsky normal form, the Newton approximants coincide with the finite-index approximations:

Theorem 5.2. *Let G be a context-free grammar in CNF with axiom S and let $(\boldsymbol{\nu}^{(i)})_{i\in\mathbb{N}}$ be the Newton sequence associated with G. Then $\big(\boldsymbol{\nu}^{(i)}\big)_S = L_{i+1}(G)$ for every $i \geq 0$.*

In particular, it follows from Theorem 5.2 that the (S-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 $i \in \mathbb{N}$.

5.2 The Counting Semiring

Consider the counting semiring with $a = \{(1,0)\}$ and $b = \{(0,1)\}$. Since the sum operation is union of sets of vectors, it is idempotent and Equations (9) and (10) hold. Since the product operation is now commutative, Equation (10) becomes

$$b \cdot \nu^{(i)} \cdot X + a + b \cdot \nu^{(i)} \cdot \nu^{(i)} = X \ . \tag{12}$$

By virtue of Kleene's fixed point theorem the least solution of a *linear* equation $X = u \cdot X + v$ over an ω-continuous semiring is given by the supremum of the sequence

$$v, \quad v + uv, \quad v + uv + uuv, \ldots$$

i.e. by $(\sum_{i \in \mathbb{N}} u^i) \cdot v = u^* \cdot v$, where $*$ is Kleene's iteration operator. The least solution $\Delta^{(i)}$ of Equation (12) is then given by

$$\Delta^{(i)} = (b \cdot \nu^{(i)})^* \cdot (a + b \cdot \nu^{(i)} \cdot \nu^{(i)})$$

and we obtain:

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

So the Newton sequence reaches a fixed point after only one iteration.

It turns out that the Newton sequence *always* reaches a fixed point in the counting semiring. This immediately generalizes to any finitely generated commutative idempotent ω-continuous semiring as we simply can opt not to evaluate the products and sums. More surprisingly, this is even the case for all semirings where sum is idempotent and product is commutative. This was first shown by Hopkins and Kozen in [13], who introduced the sequence without knowing that it was Newton's sequence (see [5] for the details). Hopkins and Kozen also gave an $O(3^n)$ upper bound for the number of iterations needed to reach the fixed point of a system of n equations. In [5] we reduced this upper bound from $O(3^n)$ to n, which is easily shown to be tight.

Theorem 5.3. *Let $\boldsymbol{f}$ be a vector of n polynomials over a commutative idempotent ω-continuous semiring. Then $\mu\boldsymbol{f} = \boldsymbol{\nu}^{(n)}$, i.e., Newton's method reaches the least fixed point after n iterations.*

We have mentioned above that the least solution of $X = u \cdot X + v$ is $u^* \cdot v$. Using this fact it is easy to show that the Newton approximants of equations over commutative semirings can be described by regular expressions. A corollary of this result is Parikh's theorem, stating that the Parikh image of a context-free language is equal to the Parikh image of some regular language [18]. To see why this is the case, notice that a context-free language is the least solution of a system of fixed point equations over the language semiring. Its Parikh image is the least solution of the same system over the counting semiring. Since Newton's method terminates over the counting semiring, and Newton approximants can be described by regular expressions, the result follows.

Notice that we are by no means the first to provide an algebraic proof of Parikh's theorem. A first proof was obtained by Aceto et al. in [1], and in fact the motivation of Hopkins and Kozen for the results of [13] was again to give a proof of the theorem. Our results in [5] make two contributions: first, the aesthetically appealing connection between Newton and Parikh, and, second, an algebraic algorithm for computing the Parikh image with a tight bound on the the number of iterations.

We conclude the section with a final remark. The counting semiring is a simple example of a semiring that does not satisfy the ascending chain condition. Kleene's method does not terminate for any program containing at least one loop. However, Newton's method always terminates!

5.3 The Real Semiring

Consider again Equation (8), but this time over the real semiring (non-negative real numbers enriched with ∞) and with $a = b = 1/2$. We get the equation

$$X = 1/2 + 1/2 \cdot X^2 \tag{13}$$

which was already briefly discussed in the introduction. We have $Df|_v(X) = v \cdot X$, and a 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 its only solution. So we get

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

and therefore $\nu^{(i)} = 1 - 2^{(i+1)}$. The Newton sequence converges to 1, and gains one bit of accuracy per iteration, i.e., the relative error is halved at each iteration.

In [14, 6] we have analyzed in detail the convergence behaviour of Newton's method. Loosely speaking, our results say that Equation (13) is an example of the worst-case behaviour of the method.

To characterize it, we use the term *linear convergence*, a notion from numerical analysis that states that the number of bits obtained after i iterations depends linearly on i. If $\|\mu \boldsymbol{f} - \boldsymbol{v}\| \;/\; \|\mu \boldsymbol{f}\| \le 2^{-i}$ (in the maximum-norm), we say that the approximation $\boldsymbol{v}$ of $\mu \boldsymbol{f}$ has (at least) i bits of accuracy . Newton's method converges linearly provided that $\boldsymbol{f}$ has a finite least fixed point and is in an easily achievable normal form (the polynomials have degree at most 2, and $\mu \boldsymbol{f}$ is nonzero in all components). More precisely [6]:

Theorem 5.4. *Let $\boldsymbol{f}$ be a vector of n polynomials over the real semiring in the above mentioned normal form. Then Newton's method converges linearly: there exists a $t_{\boldsymbol{f}} \in \mathbb{N}$ such that the Newton approximant $\boldsymbol{\nu}^{(t_{\boldsymbol{f}} + i \cdot (n+1) \cdot 2^n)}$ has at least i bits of accuracy.*

Theorem 5.4 is essentially tight. Consider the following family of equation systems.

$$\begin{aligned} X_1 &= 1/2 + 1/2 \cdot X_1^2 \\ X_2 &= 1/4 \cdot X_1^2 + 1/2 \cdot X_1 X_2 + 1/4 \cdot X_2^2 \\ &\vdots \\ X_n &= 1/4 \cdot X_{n-1}^2 + 1/2 \cdot X_{n-1} X_n + 1/4 \cdot X_n^2 \end{aligned} \tag{14}$$

Its least solution is $(1, \dots, 1)$. We show in [14, 6] that at least $i \cdot 2^{n-1}$ iterations of Newton's method are needed to obtain i bits. More precisely, we show that after $i \cdot 2^{n-1}$ iterations no more than $i \cdot 2^{n-1}$ bits of accuracy have been obtained for the first component (cf. the convergence behaviour of (13) above) and that the number of accurate bits of the $(k+1)$-th component is at most one half of the number of accurate bits of the k-th component, for all $k < n$. This implies that for the n-th component we have obtained at most i bits of accuracy.

This example exploits the fact that X_k depends only on the X_l for $l \leq k \leq n$. In fact, Theorem 5.4 can be substantially strengthened if $\boldsymbol{f}$ is *strongly connected*. More formally, let a variable X *depend* on Y if Y appears in $\boldsymbol{f}_X$. Then, $\boldsymbol{f}$ is said to be strongly connected if every variable depends transitively on every variable. For those systems we show that Newton's method gains 1 bit of accuracy per iteration after the "threshold" $t_{\boldsymbol{f}}$ has been reached. In addition (and even more importantly from a computational point of view) we can give bounds on $t_{\boldsymbol{f}}$ [6]:

Theorem 5.5. *Let $\boldsymbol{f}$ be as in Theorem 5.4, and, additionally, strongly connected. Further, let m be the size of $\boldsymbol{f}$ (coefficients in binary). Then i bits of accuracy are attained by $\boldsymbol{\nu}^{(n2^{n+2}m+i)}$. This improves to $\boldsymbol{\nu}^{(5n^2m+i)}$, if $\boldsymbol{f}(\mathbf{0})$ is positive in all components.*

In [7], a recent invited paper, we discuss equation systems over the real semiring, the motivation and complexity of computing their least solutions, and our results [14, 6] on Newton's method for the real semiring in more detail. We present an extension of Newton's method on polynomials with $\min$ and $\max$ operators in [3].

6 Conclusion

We have shown that the two questions we asked in the introduction have an affirmative answer. Newton's method, a 300 years old technique for approximating the solution of a system of equations over the reals, can be extended to arbitrary ω-continuous semirings. And, when restricted to the real semiring, the pathologies of Newton's method—no convergence, or only local and slow convergence—disappear: the method always exhibits at least linear convergence.

We like to look at our results as bridges between numerical mathematics and the foundations of program semantics and program analysis. On the one hand, while numerical mathematics has studied Newton's method in large detail, it has not payed much attention to its restriction to the real semiring. Our results indicate that this is an interesting case certainly deserving further research.

On the other hand, program analysis relies on computational engines for solving systems of equations over a large variety of domains, and these engines are based, in one way or another, on Kleene's iterative technique. This technique is very slow when working on the reals, and numerical mathematics has developed much faster ones, Newton's method being one of the most basic. The generalization of these techniques to the more general domains of semantics and program analysis is an exciting research program.

References

1. L. Aceto, Z. Ésik, and A. Ingólfsdóttir. A fully equational proof of Parikh's theorem. *Informatique Théorique et Applications*, 36(2):129–153, 2002.
2. P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *POPL*, pages 238–252, 1977.
3. J. Esparza, T. Gawlitza, S. Kiefer, and H. Seidl. Approximative methods for monotone systems of min-max-polynomial equations. In *this volume*, 2008.
4. J. Esparza, S. Kiefer, and M. Luttenberger. An extension of Newton's method to ω-continuous semirings. In *Proceedings of DLT (Developments in Language Theory)*, LNCS 4588, pages 157–168. Springer, 2007.
5. J. Esparza, S. Kiefer, and M. Luttenberger. On fixed point equations over commutative semirings. In *Proceedings of STACS*, LNCS 4397, pages 296–307. Springer, 2007.
6. J. Esparza, S. Kiefer, and M. Luttenberger. Convergence thresholds of Newton's method for monotone polynomial equations. In *Proceedings of STACS*, pages 289–300, 2008.
7. J. Esparza, S. Kiefer, and M. Luttenberger. Solving monotone polynomial equations. In *Proceedings of IFIP TCS 2008*. Springer, to appear. Invited paper.
8. J. Esparza, A. Kučera, and R. Mayr. Model checking probabilistic pushdown automata. In *LICS 2004*. IEEE Computer Society, 2004.
9. K. Etessami and M. Yannakakis. Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations. In *STACS*, pages 340–352, 2005.
10. H. Fernau and M. Holzer. Conditional context-free languages of finite index. In *New Trends in Formal Languages*, pages 10–26, 1997.
11. S. Ginsburg and E. Spanier. Derivation-bounded languages. *Journal of Computer and System Sciences*, 2:228–250, 1968.
12. J. Gruska. A few remarks on the index of context-free grammars and languages. *Information and Control*, 19:216–223, 1971.
13. M. W. Hopkins and D. Kozen. Parikh's theorem in commutative Kleene algebra. In *Logic in Computer Science*, pages 394–401, 1999.
14. 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.
15. 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.
16. F. Nielson, H.R. Nielson, and C. Hankin. *Principles of Program Analysis*. Springer, 1999.
17. J.M. Ortega and W.C. Rheinboldt. *Iterative solution of nonlinear equations in several variables*. Academic Press, 1970.
18. R. J. Parikh. On context-free languages. *J. Assoc. Comput. Mach.*, 13(4):570–581, 1966.
19. A. Salomaa. On the index of a context-free grammar and language. *Information and Control*, 14:474–477, 1969.
20. M.K. Yntema. Inclusion relations among families of context-free languages. *Information and Control*, 10:572–597, 1967.