Derivation Tree Analysis for Accelerated Fixed-Point Computation

Javier Esparza

Institut für Informatik, Technische Universität München, 85748 Garching, Germany

Stefan Kiefer

Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford OX1 3QD, Great Britain

Michael Luttenberger

Institut für Informatik, Technische Universität München, 85748 Garching, Germany

Abstract

We show that for several classes of idempotent semirings the least fixed-point of a polynomial system of equations $\boldsymbol{X} = \boldsymbol{f}(\boldsymbol{X})$ is equal to the least fixed-point of a *linear* system obtained by "linearizing" the polynomials of $\boldsymbol{f}$ in a certain way. Our proofs rely on derivation tree analysis, a proof principle that combines methods from algebra, calculus, and formal language theory, and was first used in [10] to show that Newton's method over commutative and idempotent semirings converges in a linear number of steps. Our results lead to efficient generic algorithms for computing the least fixed-point. We use these algorithms to derive several consequences, including an $O(N^3)$ algorithm for computing the throughput of a context-free grammar (obtained by speeding up the $O(N^4)$ algorithm of [7]), and a generalization of Courcelle's result stating that the downward-closed image of a context-free language is regular [8].

Keywords: Fixed-point equations, semirings, derivation trees.

1. Introduction

Systems $\boldsymbol{X} = \boldsymbol{f}(\boldsymbol{X})$ of fixed-point equations, where $\boldsymbol{f}$ is a system of polynomials, appear naturally in program analysis [22, 24, 6, 23], language theory [25], semantics of programming languages and process algebras [18, 21, 2] and in the study of probabilistic systems [16, 11, 13, 12]. In many of these applications, the system of equations is easily derived from a program or process by syntactic means; the sum and product operations in the polynomials standing for choice and sequential composition,

Email addresses: esparza@model.in.tum.de (Javier Esparza), stekie@comlab.ox.ac.uk (Stefan Kiefer), luttenbe@model.in.tum.de (Michael Luttenberger)

A previous version of this work has been published in the proceedings of DLT 2008.

respectively. The system is solved over different domains corresponding to different *abstractions* or *overapproximations* of the system [6, 23]. While the abstractions lose information about the behaviour of the system, they are necessary in order to efficiently compute a solution of the equations.

In this paper we study how to solve the equations for different classes of domains. We follow an axiomatic approach, i.e., we investigate generic algorithms that can be applied to equations over any domain satisfying a given set of axioms. Our work can be seen as an algorithmic-oriented version of the approach to process algebra pioneered by Bergstra and Klop [3, 4], and further developed by the "Dutch school" (see e.g. [2, 14, 5]), in which the semantic models of a process are also classified in terms of axioms. In fact, as shown in Section 2.1, our systems of equations can be interpreted as recursive definitions of processes in Bergstra and Klop's Basic Process Algebra (BPA), but over models satisfying not only the right-distributivity axiom $(x+y)\cdot z = x\cdot z+y\cdot z$, but also the left-distributivity axiom $x \cdot (y + z) = x \cdot z + y \cdot z$).

The starting point of our work is Kleene's classical fixed-point theorem [20]. The theorem shows that for any system $\boldsymbol{X} = \boldsymbol{f}(\boldsymbol{X})$ of polynomial equations and for any domain satisfying the axioms of a complete semilattice or, more generally, of an ω-continuous semiring, there exists a least solution $\mu\boldsymbol{f}$ that can be approximated by iteratively computing $\boldsymbol{f}(\mathbf{0}), \boldsymbol{f}(\boldsymbol{f}(\mathbf{0})), \boldsymbol{f}(\boldsymbol{f}(\boldsymbol{f}(\mathbf{0}))), \ldots$. The theorem is extremely useful and finds many applications in distributive program analysis, analysis of context-free grammars, trace semantics for process algebras, and probabilistic verification. However, it is often unsatisfactory from an algorithmic point of view, because it fails to terminate in general. Consider for instance 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 $\boldsymbol{f}^{(i)}(\mathbf{0}) = \{b, ab, \ldots, a^{i-1}b\}$ for $i \geq 1$, i.e., the solution is not reached in any finite number of steps.

In [10, 9] we have shown that Newton's method—the well-known method from numerical mathematics for approximating a zero of a differentiable function— can be generalized to arbitrary ω-continuous semirings. Like Kleene's, Newton's method proceeds by iteratively computing approximations to $\mu\boldsymbol{f}$. We were able to show that for *idempotent* (w.r.t. addition) and *commutative* (w.r.t. multiplication) semirings the method terminates[2], and in fact terminates after a small number of iterations: if the system has n equations, then the n-th Newton approximant is already equal to $\mu\boldsymbol{f}$.

Our proof of this result uses a (to the best of our knowledge) novel technique, which we call *derivation tree analysis*. The system $\boldsymbol{f}$ induces a set $\mathcal{T}$ of *derivation trees*, a generalization of the well-known derivation trees of context-free grammars. Each tree can be naturally assigned a semiring element, called the *yield* of the tree. It is easy to show that $\mu\boldsymbol{f}$ is equal to the sum of the yields of all derivation trees. Derivation tree analysis first identifies a subset T' of derivation trees whose total yield $\mathsf{Y}(T')$ is easy to compute in some sense, and then proves that T' satisfies the *embedding property*: $\mathsf{Y}(t) \sqsubseteq \mathsf{Y}(T')$ for every derivation tree t. If the semiring is idempotent, the embedding property implies $\mathsf{Y}(\mathcal{T}) = \mathsf{Y}(T')$, and so $\mu\boldsymbol{f} = \mathsf{Y}(T')$. In [10], the set T' was chosen so that $\mathsf{Y}(T')$ is equal to the n-th Newton approximant, and the embedding property

[2] A similar result was already proved in [19].

was proved using some tree surgery and exploiting the commutativity of the semiring.

The computation of the n-th Newton approximant can still require considerable resources. In this paper we present a further application of derivation tree analysis to idempotent semirings, leading to more efficient algorithms for computing the least fixed point. For this, we define the set $\mathcal{B}$ of *bamboos* of a system $\boldsymbol{f}$. Loosely speaking, bamboos are derivation trees with an arbitrarily long stem but only short branches. We first show that $\mathsf{Y}(\mathcal{B})$ is the solution of a linear system of equations whose functions are similar (but not identical) to the straightforward linearisation of $\boldsymbol{f}$. Then, we prove that the following three classes of semirings satisfy the embedding property:

Star-distributive semirings are idempotent and commutative semirings satisfying the additional axiom $(a+b)^* = a^* + b^*$ (where * is the well-known Kleene iteration operator). The so-called "tropical" $(\min, +)$-semiring over the reals (extended with $+\infty$ and $-\infty$) is star-distributive. Our tree analysis leads to an algorithm for computing $\mu\boldsymbol{f}$ very similar to the generalized Bellman-Ford algorithm of Gawlitza and Seidl [15]. We use it to derive a new algorithm for computing the throughput of a context-free grammar, a problem introduced and analyzed by Caucal et al. in [7]. Our algorithm runs in $O(N^3)$, a factor N faster than the algorithm presented in [7].

Lossy semirings are idempotent semirings satisfying the additional axiom $a+1 = a$ where 1 is the neutral element of multiplication. A natural model are downward-closed languages with union and concatenation as operations. Lossy semirings find application in the verification of lossy channel systems, a model of computation thoroughly investigated by Abdulla et al. (see e.g. [1]). Our tree analysis leads to an algebraic proof of Courcelle's theorem stating that the downward closure of a context-free language is effectively regular [8].

1-*bounded semirings* are idempotent semirings where the equation $a+1 = 1$ holds. A natural example is the "maximum probability" semiring with the interval $[0, 1]$ as carrier, maximum as addition, and standard multiplication over the reals. Using derivation tree analysis it is very easy to show that the least fixed-point $\mu\boldsymbol{f}$ of a polynomial system $\boldsymbol{f}$ with n variables is given by $\boldsymbol{f}^n(\mathbf{0})$, the n-fold application of $\boldsymbol{f}$ to $\mathbf{0}$.

The rest of the paper is organized as follows. After the preliminaries in Section 2 we introduce derivation tree analysis in Section 3. Bamboos are defined in Section 4. In the Sections 5, 6 and 7 we apply derivation tree analysis to the semiring classes mentioned above.

2. Preliminaries

As usual, $\mathbb{N}$ denotes the set of natural numbers including 0.

An *idempotent semiring* $\mathcal{S} = \langle S, +, \cdot, 0, 1\rangle$ consists of a commutative, idempotent additive monoid $\langle S, +, 0\rangle$, and a multiplicative monoid $\langle S, \cdot, 1\rangle$. In the following we often omit the dot $\cdot$ in products. Both algebraic structures are connected by left- and right-distributivity, e.g. $a(b+c) = ab+ac$, and by the requirement that $0 \cdot a = a \cdot 0 = 0$ for all $a \in S$. The *natural order* $\sqsubseteq$ on S is defined defined by $a \sqsubseteq b \Leftrightarrow a + b = b$.

An *io-semiring* is an idempotent semiring which is further ω-*continuous*, i.e., on S countable summation $\sum_{i\in\mathbb{N}} a_i \in S$ is defined (with $a_i \in S$), and satisfies the following requirements: (i) summation is continuous, i.e., $\sup^{\sqsubseteq}\{a_0 + a_1 + \ldots + a_k \mid k \in \mathbb{N}\} =$

$\sum_{i\in\mathbb{N}} a_i$ for all sequences $a : N \to S$; (ii) distributivity extends in the natural way to countable summation; and (iii) $\sum_{j\in J}\sum_{i\in I_j} a_i = \sum_{i\in\mathbb{N}} a_i$ holds for all partitions $(I_j)_{j\in J}$ of $\mathbb{N}$. Note that for io-semiring we have $\sum_{i\in\mathbb{N}} a_i \sqsubseteq b$ if and only if $a_i \sqsubseteq b$ for all $i \in \mathbb{N}$. In every such io-semiring the Kleene-star operator $^* : S \to S$ is well-defined by $a^* := \sum_{k\in\mathbb{N}} a^k$ for all $a \in S$. In the following we consider only io-semirings $\mathcal{S}$.

We fix a finite, non-empty set $\mathcal{X}$ of *variables* for the rest of the section, and use n to denote $|\mathcal{X}|$ in the following. A map from $\mathcal{X}$ to S is called a *vector*. The set of all vectors is denoted by V with $\mathbf{0} \in V$ the vector which maps every $X \in \mathcal{X}$ to $0 \in S$. We write both $\boldsymbol{v}(X)$ and $\boldsymbol{v}_X$ for the value of a vector $\boldsymbol{v}$ at $X \in \mathcal{X}$, also called the X-component of $\boldsymbol{v}$. Sum of vectors is defined componentwise: 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 $X \in \mathcal{X}$.

A *monomial of degree* k is a finite expression $a_1X_1a_2\cdots a_kX_ka_{k+1}$ where $k \geq 0$, $a_1, \ldots, a_{k+1} \in S \setminus \{0\}$ and $X_1, \ldots, X_k \in \mathcal{X}$. A *polynomial* is an expression of the form $m_1 + \cdots + m_k$ where $k \geq 0$ and $m_1, \ldots, m_k$ are monomials. Since $\mathcal{S}$ is idempotent, we assume w.l.o.g. that all monomials of a polynomial are distinct. The degree of a polynomial is the largest degree of its monomials. We let $\mathcal{S}[\mathcal{X}]$ denote the set of all polynomials.

Let $f = \alpha_1X_1\alpha_2\ldots X_k\alpha_{k+1}$ be a monomial and let $\boldsymbol{v}$ be a vector. The *evaluation of f at $\boldsymbol{v}$*, denoted by $f(\boldsymbol{v})$, is the product $\alpha_1\boldsymbol{v}_{X_1}\alpha_2\cdots\alpha_k\boldsymbol{v}_{X_k}\alpha_{k+1}$. We extend this to any polynomial: if $f = \sum_{i=1}^{k} m_i$, then $f(\boldsymbol{v}) = \sum_{i=1}^{k} m_i(\boldsymbol{v})$.

A *system of polynomials* or polynomial system $\boldsymbol{f} : \mathcal{X} \to \mathcal{S}[\mathcal{X}]$ assigns to every variable $X \in \mathcal{X}$ a polynomial $\boldsymbol{f}_X$ and induces a map from V to V by componentwise evaluation of the polynomials: $\boldsymbol{f}(\boldsymbol{v})_X := \boldsymbol{f}_X(\boldsymbol{v})$ for all $\boldsymbol{v} \in V$, and $X \in \mathcal{X}$. We write $\boldsymbol{X} = \boldsymbol{f}(\boldsymbol{X})$ for the equation system $\bigwedge_{X\in\mathcal{X}} X = \boldsymbol{f}_X$ (over the respective io-semiring S). The following proposition, which follows easily from Kleene's theorem and the fact that $\boldsymbol{f}$ is a monotone and continuous mapping, shows that any polynomial system $\boldsymbol{f}$ has a least fixed-point $\mu\boldsymbol{f}$, i.e., the corresponding equation system $\boldsymbol{X} = \boldsymbol{f}(\boldsymbol{X})$ has a least solution $\mu\boldsymbol{f}$.

Proposition 1. *A polynomial system $\boldsymbol{f}$ has a unique least fixed-point $\mu\boldsymbol{f}$, i.e., $\mu\boldsymbol{f} = \boldsymbol{f}(\mu\boldsymbol{f})$, and $\mu\boldsymbol{f} \sqsubseteq \boldsymbol{v}$ holds for all $\boldsymbol{v}$ with $\boldsymbol{v} = \boldsymbol{f}(\boldsymbol{v})$. Further, $\mu\boldsymbol{f}$ is the supremum (w.r.t. $\sqsubseteq$) of the* Kleene sequence $(\boldsymbol{f}^i(\mathbf{0}))_{i\in\mathbb{N}}$, *where $\boldsymbol{f}^i$ denotes the i-fold application of $\boldsymbol{f}$.*

2.1. Connection to Basic Process Algebra

Idempotent semirings are closely related to Basic Process Algebra (BPA), the fragment of Bergstra and Klop's process algebra involving only sequential composition and choice [3]. Recall the axioms of BPA with deadlock–special constant δ–and immediate

termination–special constant ε (see Chapter 2 of [2]):

$$\begin{array}{rcll} x+y & = & y+x & \text{A1} \\ (x+y)+z & = & x+(y+z) & \text{A2} \\ x+x & = & x & \text{A3} \\ (x+y)\cdot z & = & x\cdot z+y\cdot z & \text{A4} \\ (x\cdot y)\cdot z & = & x\cdot(y\cdot z) & \text{A5} \\ x+\delta & = & x & \text{A6} \\ \delta\cdot x & = & \delta & \text{A7} \\ x\cdot\varepsilon & = & x & \text{A8} \\ \varepsilon\cdot x & = & x & \text{A9} \end{array}$$

It follows immediately from this list that every semiring $\mathcal{S} = \langle S, +, \cdot, 0, 1\rangle$ yields a model of BPA when δ and ε are interpreted as 0 and 1, respectively. The converse does not hold, because semirings also satisfy the left-distributivity axiom

$$x\cdot(y+z) = x\cdot y + x\cdot z\ .$$

Loosely speaking, this is the axiom that distinguishes linear from branching time models: in linear time models, the semantics of a process is a set of traces, and processes that only differ on the point at which nondeterminism is resolved are considered equal; in branching time models (like bisimulation or testing equivalence), the axiom does not hold. Our work on semirings can therefore be seen as the algorithmic study of linear time models of BPA.

3. Derivation Trees

We generalize the notion of derivation tree, as known from formal languages and grammars. We identify a node u of a (ordered) tree t with the subtree of t rooted at u. In particular, we identify a tree with its root.

Let $\boldsymbol{f}$ be a polynomial system over a set $\mathcal{X}$ of variables. A *derivation tree* t of $\boldsymbol{f}$ is an ordered (finite) tree whose nodes are labelled with both a variable X and a monomial m of $\boldsymbol{f}_X$. We write λ_v, resp. λ_m for the corresponding labelling-functions. Moreover, if the monomial labelling of a node u is $\lambda_m(u) = a_1X_1a_2\ldots X_sa_{s+1}$ for some $s \geq 0$, then u has exactly s children $u_1, \ldots, u_s$, ordered from left to right, with $\lambda_v(u_i) = X_i$ for all $i = 1, \ldots, s$. A derivation tree t is an *X-tree* if $\lambda_v(t) = X$. The set of all X-trees of $\boldsymbol{f}$ is denoted by $\mathcal{T}_{\boldsymbol{f},X}$, or just by $\mathcal{T}_X$ if $\boldsymbol{f}$ is clear from the context.

The left part of Figure 1 shows a derivation tree of the system $\boldsymbol{f}$ over the variables X and Y given by $\boldsymbol{f}_X = aXYb + c$ and $\boldsymbol{f}_Y = dX + Ye$. The derivation trees of $\boldsymbol{f}$ are very similar to the derivation trees of the context-free grammar with productions $X \to aXYb|c$ and $Y \to dX|Ye$. Note that the nodes of "our" trees are labeled by "productions" (for instance, the label $(X, aXYb)$ corresponds to the production $X \to aXYb$). This simplifies notation in the following proofs. On the right of Figure 1 the corresponding tree according to the standard definition is shown. The height $h(t)$ of a derivation tree t is the length of a longest path from the root to a leaf. The set of X-trees (of $\boldsymbol{f}$) of height *at most* h is denoted by $\mathcal{T}_X^{(h)}$. The yield $\mathsf{Y}(t)$ of a

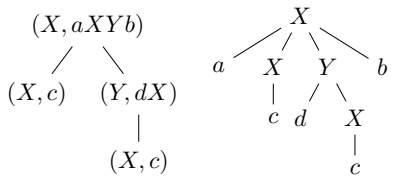


Figure 1: A derivation tree on the left, and its standard representation on the right

derivation tree t with $\lambda_m(t) = a_1 X_1 a_2 \cdots X_s a_{s+1}$ is inductively defined to be $\mathsf{Y}(t) = a_1 \mathsf{Y}(t_1) a_2 \cdots \mathsf{Y}(t_s) a_{s+1}$. We extend the definition of Y to sets $T \subseteq \mathcal{T}_X$ by setting $\mathsf{Y}(T) := \sum_{t \in T} \mathsf{Y}(t)$. E.g., the system $\boldsymbol{f}$ defined above has exactly two X-trees of height at most 2: the tree consisting of a single node labeled by (X, c), and the left tree of Figure 1. Their yields are c and $acdcb$, respectively, and so $\mathsf{Y}(\mathcal{T}_X^{(2)}) = c + acdcb$. It follows $\mathsf{Y}(\mathcal{T}_X^{(2)}) = \boldsymbol{f}^3(\mathbf{0})_X$, i.e., the yield of the X-trees of height at most 2 is equal to the "Kleene approximant" $\boldsymbol{f}^3(\mathbf{0})_X$ from Proposition 2. The following proposition, easy to prove [9], shows that this is not a coincidence.

Proposition 2. *For all $h \in \mathbb{N}$ and $X \in \mathcal{X}$, we have $\mathsf{Y}(\mathcal{T}_X^{(h)}) = \big(\boldsymbol{f}^{h+1}(\mathbf{0})\big)_X$.*

Together with Proposition 2 we get:

Corollary 3. $\mu \boldsymbol{f}_X = \mathsf{Y}(\mathcal{T}_X)$.

3.1. Derivation Tree Analysis

We say that a set T_X of X-trees satisfies the *embedding property* if $\mathsf{Y}(t) \sqsubseteq \mathsf{Y}(T_X)$ for every X-tree $t \in \mathcal{T}_X$ Loosely speaking, this means that for a tree $t \in \mathcal{T}_X \setminus T_X$, its yield $\mathsf{Y}(t)$ is already taken into account when only considering T_X. For our setting of io-semirings, the embedding property is of course equivalent to $\mathsf{Y}(\mathcal{T}_X) \sqsubseteq \mathsf{Y}(T_X)$ resp. $\mathsf{Y}(\mathcal{T}_X) = \mathsf{Y}(T_X)$ as $T_X \subseteq \mathcal{T}_X$. In combination with Corollary 3 we obtain:

Proposition 4. *Let $\boldsymbol{f}$ be a system of polynomials over an io-semiring, and let X be a variable of $\boldsymbol{f}$. If a set T_X of X-trees of $\boldsymbol{f}$ satisfies the embedding property, then $\mu \boldsymbol{f} = \mathsf{Y}(T_X)$.*

This proposition suggests a technique for the design of efficient algorithms computing $\mu \boldsymbol{f}$: (1) define a set T_X of derivation trees whose yield is "easy to compute" in some io-semiring, and (2) identify "relevant" classes of io-semirings for which T_X satisfies the embedding property. By Proposition 4, $\mu \boldsymbol{f}$ is "easy to compute" for these classes. We call this technique *derivation tree analysis*.

4. Bamboos and their Yield

The difficulty of derivation tree analysis lies in finding a set T_X exhibiting a good balance between the contradictory requirements "easy to compute" and "relevant": if $T_X = \emptyset$ then the yield is trivial to compute, but T_X does not satisfy the embedding

property in any interesting case. Conversely, $T_X = \mathcal{T}_X$ trivially satisfies the embedding property for every io-semiring, but is not easy to compute. The main contribution of this paper is the identification of a class of derivation trees, *bamboos*, exhibiting this balance. In this section we define bamboos and show that their yield is the least solution of a system of *linear* equations easily derivable from $\boldsymbol{f}$. The "easy to compute" part is justified by the fact that in most semirings used in practice linear equations are far easier to solve than polynomial equations (e.g., in the real semiring or the language semiring with union and concatenation as operations). The "relevance" of bamboos is justified in the next three sections.

Definition 5. Let $\boldsymbol{f}$ be a system of polynomials. A tree $t \in \mathcal{T}_{\boldsymbol{f},X}$ is an X-*bamboo* if there is a path leading from the root of t to some leaf of t, the *stem*, such that the height of every subtree of t not containing a node of the stem is at most $n-1$. The set of all X-bamboos of $\boldsymbol{f}$ is denoted by $\mathcal{B}_{f,X}$, or just by $\mathcal{B}_X$ if $\boldsymbol{f}$ is clear from the context.

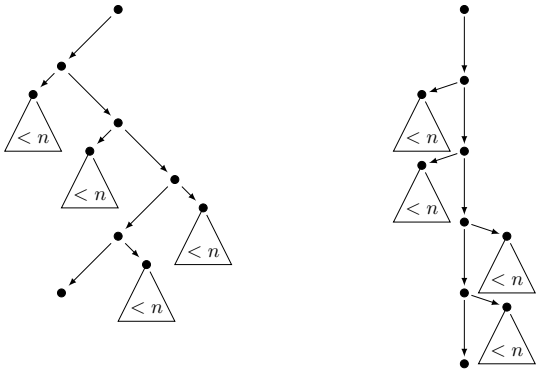


Figure 2: An example of the structure of a bamboo: it consists of a stem of unbounded length from which subtrees of height less than n sprout; on the right it is shown with its stem straightened.

The linear system of equations mentioned above can be obtained by suitably generalizing the notion of *differential* of a function. Recall from elementary calculus that the differential of a function $f(X)$ at value v is the linear function (more precisely, differential 1-form) $D_f|_v(X)$ which describes the differential change of f in a neighborhood of v. Over the real numbers, differential and derivative of a function are closely related, indeed we have $D_f|_v(X) = f'(v)X$ which, neglecting the constant offset, is the best linear approximation to $f(X)$ in the neighborhood of v. Differentials can be suitably extended to functions over several variables (where there is similar connection with the partial derivatives).

This suggests to use differentials to "linearize" polynomial equations over semirings, i.e., for finding linear equations whose solutions are good aproximations or even equal to the solutions of the polynomial systems. For this, however, we must first come up with a suitable definition of differential! In calculus the differential is defined in terms of the derivative (see baove), which in turn is defined as a limit of quotients. Since multiplication in a semiring does not necessarily have an inverse, there seems to be no

simple of way of generalizing the differential. To overcome this problem, we recall that a polynomial function can be computed using simple rules very similar to the rules for computing derivatives. For instance, the rule $(fg)'(X) = f'(X)g(X) + f(X)g'(X)$ for the computation of derivatives yields the rule

$$D_(fg)|_v(X) = D_f|_v(X)h(v) + f(v)D_g|_v(X)$$

for differentials. Since these rules only involve addition and multiplication, they can be used to define the differential of a polynomial function not only for functions over the reals, but for functions over an arbitrary semiring. This is what we do in Definition 6 below. Notice that, in principle, this definition is purely formal: nothing guarantees that the differential of a polynomial over a semiring will be in any sense "a linear approximation" to the polynomial. However, this turns out to be the case: Definition 6 defines a linearization of polynomial equations using this formal definition of differentals, and Theorem establishes the relation between the elast solution of a system and the least solution of its linearization.

Definition 6. Let $f \in \mathcal{S}[\mathcal{X}]$ be a polynomial and let $\boldsymbol{v} \in V$ be a vector. The *differential of f at $\boldsymbol{v}$ w.r.t. a variable X* is the map $D_X f|_{\boldsymbol{v}} \colon V \to S$ inductively defined as follows:

$$D_X f|_{\boldsymbol{v}}(\boldsymbol{a}) = \begin{cases} 0 & \text{if } f \in S \text{ or } f \in \mathcal{X} \setminus \{X\} \\ \boldsymbol{a}_X & \text{if } f = X \\ D_X g|_{\boldsymbol{v}}(\boldsymbol{a}) \cdot h(\boldsymbol{v}) + g(\boldsymbol{v}) \cdot D_X h|_{\boldsymbol{v}}(\boldsymbol{a}) & \text{if } f = g \cdot h \\ \sum_{i=1}^{k} D_X m_i|_{\boldsymbol{v}}(\boldsymbol{a}) & \text{if } f = \sum_{i=1}^{k} m_i. \end{cases}$$

Further, we define the *differential of f* at $\boldsymbol{v}$ by $Df|_{\boldsymbol{v}}(\boldsymbol{a}) := \sum_{X \in \mathcal{X}} D_X f|_{\boldsymbol{v}}(\boldsymbol{a})$. The differential of a system of polynomials $\boldsymbol{f}$ at $\boldsymbol{v}$ is defined componentwise by $(D\boldsymbol{f}|_{\boldsymbol{v}}(\boldsymbol{a}))_X := D(\boldsymbol{f}_X)|_{\boldsymbol{v}}(\boldsymbol{a})$ for all $X \in \mathcal{X}$.

Example 7. *For $f(X,Y) = a \cdot X \cdot X \cdot Y \cdot b$, $\boldsymbol{v} = (v_X, v_Y)$, $\boldsymbol{c} = (c_X, c_Y)$ we have:*

$$\begin{aligned} D_X f|_{\boldsymbol{v}}(\boldsymbol{c}) &= a \cdot c_X \cdot v_X \cdot v_Y \cdot b + a \cdot v_X \cdot c_X \cdot v_Y \cdot b \\ D_Y f|_{\boldsymbol{v}}(\boldsymbol{c}) &= a \cdot v_X \cdot v_X \cdot c_Y \cdot b \end{aligned}$$

Remark. At this point the reader may wonder why we use differentials instead of derivatives, whose rules are simpler than the ones of Definition 6. The reason is that the connection $D_f|_v(X) = f'(v)X$ only holds because multiplication over the reals is commutative, which is not necessarily the case in semirings. Consider the polynomial $f(X) = aXb + a$. Taking the derivative yields $f'(X) = ab$ and, subsequently, $l(X) = abX + a$ for the tangent at the point $X = 0$. Clearly, if multiplication is commutative, then $l(X)$ approximates $f(X)$ from below, i.e., we have $l(v) \sqsubseteq f(v)$ for all $v \in S$ as $l(X) = f(X)$. But on the language semiring generated by $\Sigma = \{a, b\}$, where multiplication is notcommutative, this does not hold anymore, e.g., $l(a) \neq f(a)$. Since the connection between derivative and differential no longer holds, we are forced to work directly with differentials.

Using differentials we define a particular linearization of a polynomial system. The idea is to replace a polynomial system of equations $\boldsymbol{X} = \boldsymbol{f}(\boldsymbol{X})$ by a linear system $\boldsymbol{X} = \boldsymbol{f}_{\mathcal{B}}(\boldsymbol{X})$ whose least solution is a "good approximation" to the least solution of $\boldsymbol{X} = \boldsymbol{f}(\boldsymbol{X})$.

Definition 8. Let $\boldsymbol{f}$ be a system of n polynomials. The *bamboo system* $\boldsymbol{f}_{\mathcal{B}}$ associated to $\boldsymbol{f}$ is the linear system $\boldsymbol{f}_{\mathcal{B}}(\boldsymbol{X}) = D\boldsymbol{f}|_{\boldsymbol{f}^n(\mathbf{0})}(\boldsymbol{X}) + \boldsymbol{f}(\mathbf{0})$. The least solution of the system of equations $\boldsymbol{X} = \boldsymbol{f}_{\mathcal{B}}(\boldsymbol{X})$ is denoted by $\mu\boldsymbol{f}_{\mathcal{B}}$.

The following theorem establishes the relation between the least solutions of $\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{X})$ and $\boldsymbol{x} = \boldsymbol{f}_{\mathcal{B}}(\boldsymbol{X})$.

Theorem 9. *Let $\boldsymbol{f}$ be a system of polynomials over an io-semiring. For every variable X of $\boldsymbol{f}$ we have $\mathsf{Y}(\mathcal{B}_X) = (\mu\boldsymbol{f}_{\mathcal{B}})_X$, i.e., the yield of the X-bamboos is equal to the X-component of the least solution of the bamboo system.*

PROOF. In the following, tet $\mathcal{B}_X^{(h)} := \mathcal{B}_X \cap \mathcal{T}_X^{(h)}$ be the set of X-bamboos (w.r.t. $\boldsymbol{f}$) of height *at most* h. Note that $\mathcal{B}_X^{(n-1)} = \mathcal{T}_X^{(n-1)}$.

Further note that by Kleene's fixed point theorem, we have

$$\mu\boldsymbol{f}_{\mathcal{B}} = \sum_{k\in\mathbb{N}} D\boldsymbol{f}|^k_{\boldsymbol{f}^n(\mathbf{0})}\big(\boldsymbol{f}(\mathbf{0})\big),$$

where $D\boldsymbol{f}|^k_{\boldsymbol{f}^n(\mathbf{0})}$ denotes the k-fold application of $D\boldsymbol{f}|_{\boldsymbol{f}^n(\mathbf{0})}$. (In particular, we have $D\boldsymbol{f}|^0_{\boldsymbol{f}^n(\mathbf{0})}\big(\boldsymbol{f}(\mathbf{0})\big) = \boldsymbol{f}(\mathbf{0})$.) By definition, we have

$$\boldsymbol{f}_{\mathcal{B}}(\boldsymbol{X}) = \boldsymbol{f}(\mathbf{0}) + D\boldsymbol{f}|_{\boldsymbol{f}^n(\mathbf{0})}\big(\boldsymbol{X}\big).$$

We now turn to the actual proof. We fist show that for all $h \geq 0$

$$\mathsf{Y}(\mathcal{B}_X^{(h+n-1)}) \sqsupseteq \Big(D\boldsymbol{f}|^h_{\boldsymbol{f}^n(\mathbf{0})}\big(\boldsymbol{f}^n(\mathbf{0})\big)\Big)_X.$$

Note that $D\boldsymbol{f}|^h_{\boldsymbol{f}^n(\mathbf{0})}\big(\boldsymbol{f}^n(\mathbf{0})\big) \sqsupseteq D\boldsymbol{f}|^h_{\boldsymbol{f}^n(\mathbf{0})}\big(\boldsymbol{f}(\mathbf{0})\big)$ as $\boldsymbol{f}^n(\mathbf{0}) \sqsupseteq \boldsymbol{f}(\mathbf{0})$ so that $\mathsf{Y}(\mathcal{B}_X) \sqsupseteq \big(\mu\boldsymbol{f}_{\mathcal{B}}\big)_X$ follows by ω-continuity. We proceed by induction on h: For $h = 0$, we have $\mathcal{B}_X^{(n-1)} = \mathcal{T}_X^{(n-1)}$, and $\mathsf{Y}(\mathcal{T}_X^{(n-1)}) = \boldsymbol{f}^n(\mathbf{0})_X$ (cf. prop. 2). Thus,

$$\mathsf{Y}(\mathcal{B}_X^{(n-1)}) = \mathsf{Y}(\mathcal{T}_X^{(n-1)}) = \boldsymbol{f}^n(\mathbf{0})_X = \Big(D\boldsymbol{f}|^0_{\boldsymbol{f}^n(\mathbf{0})}\big(\boldsymbol{f}^n(\mathbf{0})\big)\Big)_X$$

follows immediately.

Consider therefore $\Big(D\boldsymbol{f}|^{h+1}_{\boldsymbol{f}^n(\mathbf{0})}\big(\boldsymbol{f}^n(\mathbf{0})\big)\Big)_X$ for $h \geq 0$, and let $\mathsf{Y}(\mathcal{B}^{(h)})$ denote the vector defined by $\mathsf{Y}(\mathcal{B}^{(h)})_X := \mathsf{Y}(\mathcal{B}_X^{(h)})$. We then have by induction on h and by monotonicty of $D\boldsymbol{f}|_{\boldsymbol{f}^n(\mathbf{0})}$ that

$$\begin{aligned}\Big(D\boldsymbol{f}|^{h+1}_{\boldsymbol{f}^n(\mathbf{0})}\big(\boldsymbol{f}^n(\mathbf{0})\big)\Big)_X &= \Big(D\boldsymbol{f}|_{\boldsymbol{f}^n(\mathbf{0})}\big(\big(D\boldsymbol{f}|^h_{\boldsymbol{f}^n(\mathbf{0})}\big(\boldsymbol{f}^n(\mathbf{0})\big)\big)\big)\Big)_X \\ &\sqsubseteq \Big(D\boldsymbol{f}|_{\boldsymbol{f}^n(\mathbf{0})}\big(\mathsf{Y}(\mathcal{B}^{(h+n-1)})\big)\Big)_X \\ &= D\boldsymbol{f}_X|_{\boldsymbol{f}^n(\mathbf{0})}\big(\mathsf{Y}(\mathcal{B}^{(h+n-1)})\big)\end{aligned}$$

Assume that $\boldsymbol{f}_X = \sum_{i=1}^k m_i$ where $m_1, \ldots, m_k$ are monomials. As addition is idempotent, we may assume that these monomials are pairwise different. By linearity of the differential, we obtain

$$D\boldsymbol{f}_X|_{\boldsymbol{f}^n(\mathbf{0})}\big(\mathsf{Y}(\mathcal{B}^{(h+n-1)})\big) = \sum_{i=1}^k Dm_i|_{\boldsymbol{f}^n(\mathbf{0})}\big(\mathsf{Y}(\mathcal{B}^{(h+n-1)})\big).$$

We only need to consider the monomials m_i of degree at least one as for every constant monomial its differential is always null. In particular, if $\boldsymbol{f}_X$ is constant, then $D(\boldsymbol{f}_X)|_{\boldsymbol{v}}(\boldsymbol{a}) = \mathbf{0}$ for all $\boldsymbol{v}, \boldsymbol{a}$ and $\mathcal{B}^{(h)} = \mathcal{T}_X^{(0)}$, and we are done. Hence, assume that $\boldsymbol{f}_X$ is not constant, and consider any monomial $m = a_1 X_1 a_2 \dots X_l a_{l+1}$ of degree $l \geq 1$ in $\{m_1, \dots, m_k\}$. We then have

$$Dm|_{\boldsymbol{f}^n(\mathbf{0})}\big(\mathsf{Y}(\mathcal{B}^{(h+n-1)})\big) = \sum_{Y \in \mathcal{X}} D_Y m|_{\boldsymbol{f}^n(\mathbf{0})}\big(\mathsf{Y}(\mathcal{B}^{(h+n-1)})\big)$$

by definition. Consider any $Y \in \{X_1, \dots, X_l\}$, i.e. a variable appearing in m (for $Y \notin \{X_1, \dots, X_n\}$ the differential $D_Y m$ is again 0), and let $\mathrm{pos}_Y(m) = \{i \mid X_i = Y\}$ be the set of "positions of Y in m". We then may write

$$\begin{aligned} & D_Y m|_{\boldsymbol{f}^n(\mathbf{0})}\big(\mathsf{Y}(\mathcal{B}^{(h+n-1)})\big) \\ = & \sum_{p \in \mathrm{pos}_Y(m)} \left(\prod_{q=1}^{p-1} a_q \cdot \boldsymbol{f}^n(\mathbf{0})_{X_q} \right) \cdot a_p \cdot \mathsf{Y}(\mathcal{B}^{(h+n-1)})_{X_p} \cdot \left(\prod_{q=p+1}^{l} a_q \cdot \boldsymbol{f}^n(\mathbf{0})_{X_q} \right) \cdot a_{l+1} \end{aligned}$$

As $f^n(0) = \mathsf{Y}(\mathcal{T}^{(n-1)})$ we can rewrite the first product as follows:

$$\prod_{q=1}^{p-1} a_q \cdot \boldsymbol{f}^n(\mathbf{0})_{X_q} = \prod_{q=1}^{p-1} a_q \cdot \mathsf{Y}(\mathcal{T}^{(n-1)})_{X_q} = \prod_{q=1}^{p-1} a_q \cdot \mathsf{Y}(\mathcal{T}_{X_q}^{(n-1)}) = \prod_{q=1}^{p-1} a_q \cdot \sum_{t \in \mathcal{T}_{X_q}^{(n-1)}} \mathsf{Y}(t).$$

A similar calculation shows:

$$\prod_{q=p+1}^{l} a_q \cdot \boldsymbol{f}^n(\mathbf{0})_{X_q} = \prod_{q=p+1}^{l} a_q \cdot \sum_{t \in \mathcal{T}_{X_q}^{(n-1)}} \mathsf{Y}(t).$$

We therefore obtain:

$$\begin{aligned} & D_Y m|_{\boldsymbol{f}^n(\mathbf{0})}\big(\mathsf{Y}(\mathcal{B}^{(h+n-1)})\big) \\ = & \sum_{p \in \mathrm{pos}_Y(m)} \left(\prod_{q=1}^{p-1} a_q \cdot \boldsymbol{f}^n(\mathbf{0})_{X_q} \right) \cdot a_p \cdot \mathsf{Y}(\mathcal{B}^{(h+n-1)})_{X_p} \cdot \left(\prod_{q=p+1}^{l} a_q \cdot \boldsymbol{f}^n(\mathbf{0})_{X_q} \right) \cdot a_{l+1} \\ = & \sum_{p \in \mathrm{pos}_Y(m)} \left(\prod_{q=1}^{p-1} a_q \cdot \sum_{t \in \mathcal{T}_{X_q}^{(n-1)}} \mathsf{Y}(t) \right) \cdot a_p \cdot \left(\sum_{t \in \mathcal{B}_{X_p}^{(h+n-1)}} \mathsf{Y}(t) \right) \\ & \cdot \left(\prod_{q=p+1}^{l} a_q \cdot \sum_{t \in \mathcal{T}_{X_q}^{(n-1)}} \mathsf{Y}(t) \right) \cdot a_{l+1} \end{aligned}$$

But this last sum is simply the yield of all X-bamboos $t \in \mathcal{B}_X^{(h+n)}$ with $\lambda_2(t) = m$ having height at least 1, and at most $h + n$. As for every $t \in \mathcal{B}_X^{(h)}$ its root is labeled by a monomial $\lambda_2(t)$ in $\{m_1, \dots, m_k\}$, we get by idempotence

$$\Big(D\boldsymbol{f}|_{\boldsymbol{f}^n(\mathbf{0})}\big(\mathsf{Y}(\mathcal{B}^{(h+n-1)})\big)\Big)_X \sqsubseteq \mathsf{Y}(\mathcal{B}_X^{(h+n)}).$$

For the other direction we show that for every $t \in \mathcal{B}_X$ we have

$$\mathsf{Y}(t) \sqsubseteq \big(\mu \boldsymbol{f}_{\mathcal{B}}\big)_X.$$

As already mentioned, for every io-semiring, this is equivalent to $\mathsf{Y}(\mathcal{B}_X) \sqsubseteq \big(\mu \boldsymbol{f}_{\mathcal{B}}\big)_X$. We proceed by induction on the number of nodes in t. If t has just one node then $\mathsf{Y}(t) \sqsubseteq (\boldsymbol{f}(\mathbf{0}))_X \sqsubseteq \big(\mu \boldsymbol{f}_{\mathcal{B}}\big)_X$. For the induction step, t has children. So assume w.l.o.g. that $\lambda_2(t) = a_1 X_1 \cdots X_s a_{s+1}$ for some $s \geq 1$. Denote the children of t by $t_1, \ldots, t_s$. Furthermore we assume w.l.o.g. that the backbone of t goes through t_1. Hence, t_1 is itself a bamboo having less nodes than t. By induction we have $\mathsf{Y}(t_1) \sqsubseteq \big(\mu \boldsymbol{f}_{\mathcal{B}}\big)_{X_1}$. As t is a bamboo, the other children $t_2, \ldots, t_s$ have a height of at most $n-1$. By Proposition [**?**] we know that

$$\mathsf{Y}(t_r) \sqsubseteq (\boldsymbol{f}^n(\mathbf{0}))_{X_r} \text{ for all } 2 \leq r \leq s\,. \tag{1}$$

Now we have:

$$\begin{aligned}
\mathsf{Y}(t) &= a_1 \mathsf{Y}(t_1) \cdots \mathsf{Y}(t_s) a_{s+1} && \text{(def. of yield } \mathsf{Y}) \\
&\sqsubseteq a_1 \big(\mu \boldsymbol{f}_{\mathcal{B}}\big)_{X_1} a_2 \mathsf{Y}(t_2) \cdots \mathsf{Y}(t_s) a_{s+1} && \text{(by induction)} \\
&\sqsubseteq a_1 \big(\mu \boldsymbol{f}_{\mathcal{B}}\big)_{X_1} a_2 (\boldsymbol{f}^n(\mathbf{0}))_{X_2} \cdots (\boldsymbol{f}^n(\mathbf{0}))_{X_s} a_{s+1} && \text{(Equation (1))} \\
&\sqsubseteq D_{X_1}(a_1 X_1 \cdots X_s a_{s+1})|_{\boldsymbol{f}^n(\mathbf{0})}\big(\mu \boldsymbol{f}_{\mathcal{B}}\big) && \text{(def. of differentials)} \\
&\sqsubseteq D_{X_1} \boldsymbol{f}_X|_{\boldsymbol{f}^n(\mathbf{0})}\big(\mu \boldsymbol{f}_{\mathcal{B}}\big) && (t \in \mathcal{B}_X) \\
&\sqsubseteq D\boldsymbol{f}_X|_{\boldsymbol{f}^n(\mathbf{0})}\big(\mu \boldsymbol{f}_{\mathcal{B}}\big) && \text{(def. of differentials)} \\
&= \big(D\boldsymbol{f}|_{\boldsymbol{f}^n(\mathbf{0})}\big(\mu \boldsymbol{f}_{\mathcal{B}}\big)\big)_X && \text{(def. of differentials)} \\
&\sqsubseteq \big(\mu \boldsymbol{f}_{\mathcal{B}}\big)_X && \text{(def. of fixed point)}
\end{aligned}$$

□

Together with Proposition 4 we get the following corollary.

Corollary 10 (derivation tree analysis for bamboos). *Let $\boldsymbol{f}$ be a system of polynomials over an io-semiring. If $\mathcal{B}_X$ satisfies the embedding property for all X, i.e., for all X-trees t it holds that $\mathsf{Y}(t) \sqsubseteq \mathsf{Y}(\mathcal{B}_X)$, then $\mu\boldsymbol{f} = \mu\boldsymbol{f}_{\mathcal{B}}$.*

5. Star-Distributive Semirings

Definition 11. A commutative (w.r.t. multiplication) io-semiring $\mathcal{S}$ is *star-distributive* if $(a+b)^* = a^* + b^*$ holds for all $a, b \in S$.

A commutative io-semiring is star-distributive whenever the natural order $\sqsubseteq$ is total:

Proposition 12. *Any totally ordered commutative io-semiring is star-distributive.*

PROOF. Let w.l.o.g. $a \sqsubseteq b$. Then $(a+b)^* = b^* \sqsubseteq a^* + b^* \sqsubseteq (a+b)^*$. □

In particular, the $(\min, +)$-semiring over the integers or reals is star-distributive.

We have already considered commutative idempotent semirings in [10] where we showed that $\mu\boldsymbol{f}$ can be computed by solving n *linear* equation systems by means of a Newton-like method, improving the $\mathcal{O}(3^n)$ bound of Hopkins and Kozen [19]. In this section we improve this result even further for star-distributive semirings: One single linear system, the bamboo system $\boldsymbol{f}_{\mathcal{B}}$, needs to be solved. This leads to an efficient algorithm for computing $\mu\boldsymbol{f}$ in arbitrary star-distributive semirings. In Section 5.1 we instantiate this algorithm for the $(\min, +)$-semiring; in Section 5.2 we use it to improve the algorithm of [7] for computing the throughput of a context-free grammar.
We start by stating two useful properties of star-distributive semirings.

Proposition 13. *In any star-distributive semiring the following equations hold: (1)* $a^*b^* = a^* + b^*$*, and (2)* $(ab^*)^* = a^* + ab^*$.

PROOF.

(1) The equation $a^*b^* = (a + b)^*$ holds in any commutative idempotent semiring. By star-distributivity, $(a + b)^* = a^* + b^*$.

(2) In any commutative io-semiring, we have $(ab^*)^* = 1 + aa^*b^*$ (see e.g. [19]). By (1), we have $1 + aa^*b^* = 1 + aa^* + ab^* = a^* + ab^*$. □

We can now state and prove our result:

Theorem 14. $\mu\boldsymbol{f} = \mu\boldsymbol{f}_{\mathcal{B}}$ *holds for polynomial systems* $\boldsymbol{f}$ *over star-distributive semirings.*

The proof is technical. We therefore first sketch the main idea:
Proof Sketch. The proof is by derivation tree analysis. So it suffices to discharge the precondition of Corollary 10. More precisely we show for any X-tree t that $\mathsf{Y}(t) \sqsubseteq \mathsf{Y}(\mathcal{B}_X)$ holds. It suffices to consider the case where t is not an X-bamboo. Then the height of t is at least n, and so t is "pumpable", i.e., one can choose a path p in t from the root to a leaf such that two different nodes on the path share the same variable-label. So, by commutativity, t can be decomposed into three (partial) trees with yields a, b, c, respectively, such that $\mathsf{Y}(t) = abc$, see Figure 3(a).

Notice that, by commutativity of product, ab^*c is the yield of a set of trees obtained by "pumping" t. We show $ab^*c \sqsubseteq \mathsf{Y}(\mathcal{B}_X)$ which implies $\mathsf{Y}(t) \sqsubseteq \mathsf{Y}(\mathcal{B}_X)$. As t is not an X-bamboo, t has a pumpable subtree disjoint from p. In this sketch we assume that it is a subtree of that part of t whose yield is a, see Figure 3(b). Now we have $a = a_1a_2a_3$, and so $ab^*c = a_1a_2a_3b^*c \sqsubseteq a_1a_2^*a_3b^*c = a_1a_3b^*c + a_1a_2^*a_3c$, where we used commutativity and Proposition 13(1) in the last step. Both summands in the sum above are yields of sets of trees obtained by pumping pumpable trees smaller than t, see Figure 3(c+d). By an inductive argument those yields are both included in $\mathsf{Y}(\mathcal{B}_X)$.
□

We now present the complete proof of Theorem 14:

PROOF. We will need the following notation: If t' is a subtree of a derivation tree t, we write $t = \widehat{t} \cdot t'$ where $\widehat{t}$ is the partial derivation tree obtained from t by removing

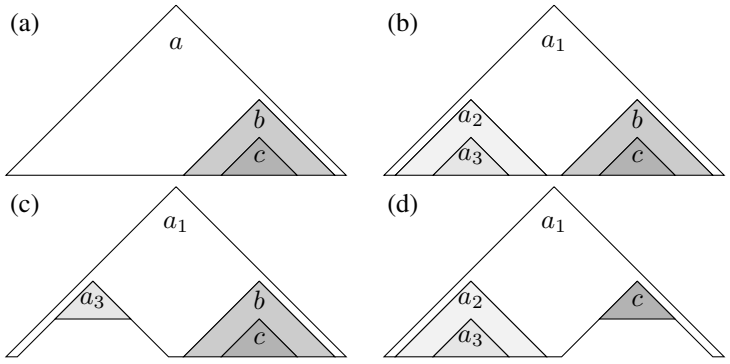


Figure 3: "Unpumping" trees to make them bamboos

t'. If, in addition, $t' = \widehat{t'} \cdot t''$, and t' and t'' have the same variable-label. we say the decomposition $t = \widehat{t} \cdot \widehat{t'} \cdot t''$ is *pumpable*, because $\widehat{t} \cdot (\widehat{t'})^i \cdot t''$ is a valid tree for all $i \geq 0$. We define $\widehat{t} \cdot (\widehat{t'})^* \cdot t'' = \{\widehat{t} \cdot (\widehat{t'})^i \cdot t'' \mid i \geq 0\}$. Notice that, due to commutativity of product, it holds that $\mathsf{Y}(\widehat{t} \cdot (\widehat{t'})^* \cdot t'') = \mathsf{Y}(\widehat{t}) \cdot \mathsf{Y}(\widehat{t'})^* \cdot \mathsf{Y}(t'')$. We call this yield the *pumping yield* of the decomposition $t = \widehat{t} \cdot \widehat{t'} \cdot t''$.

The proof is by derivation tree analysis. So it suffices to discharge the precondition of Corollary 10. More precisely we need to show that, for any X-tree t, we have $\mathsf{Y}(t) \sqsubseteq \mathsf{Y}(\mathcal{B}_X)$. If t does not have a pumpable decomposition, then t has a height of at most $n-1$, hence $t \in \mathcal{B}_X$ and so $\mathsf{Y}(t) \sqsubseteq \mathsf{Y}(\mathcal{B}_X)$. It remains to show: if t has a pumpable decomposition $t = \widehat{t} \cdot \widehat{t_1} \cdot t_1'$, then $\mathsf{Y}(t) \sqsubseteq \mathcal{B}_X$. In fact, we show $\mathsf{Y}(\widehat{t} \cdot (\widehat{t_1})^* \cdot t_1') \sqsubseteq \mathsf{Y}(\mathcal{B}_X)$, which is stronger because $\mathsf{Y}(t) \sqsubseteq \mathsf{Y}(\widehat{t} \cdot (\widehat{t_1})^* \cdot t_1')$.

Denote by $\#(t)$ the number of nodes in a tree t. We assign to a pumpable decomposition $t = \widehat{t} \cdot \widehat{t_1} \cdot t_1'$ a *size* by setting $\mathit{size}(t = \widehat{t} \cdot \widehat{t_1} \cdot t_1') = (\#(t), \#(\widehat{t_1} \cdot t_1'))$. If t has no pumpable decomposition, set $\mathit{size}(t) = (\#(t), 0)$. We order these sizes lexicographically, i.e., we set $(i,j) \lhd (i',j')$ if either $i < i'$ or $i = i'$ and $j < j'$. We use this order to prove by induction that for any size (i,j), if there is a pumpable decomposition $t = \widehat{t} \cdot \widehat{t_1} \cdot t_1'$ of size (i,j), then $\mathsf{Y}(\widehat{t} \cdot (\widehat{t_1})^* \cdot t_1') \sqsubseteq \mathsf{Y}(\mathcal{B}_X)$.

The induction base is trivial because trees t with $\#(t) = 1$ do not have a decomposition. For the induction step, let t be an X-tree and let $t = \widehat{t} \cdot \widehat{t_1} \cdot t_1'$ be pumpable. Choose a path p in t from the root to a leaf through t_1'. If p is a valid stem of an X-bamboo, then all trees in $\widehat{t} \cdot (\widehat{t_1})^* \cdot t_1'$ are X-bamboos, so $\mathsf{Y}(\widehat{t} \cdot (\widehat{t_1})^* \cdot t_1') \sqsubseteq \mathcal{B}_X$. Hence, assume that p is not a valid stem, i.e., there is some subtree of t, disjoint from p, with height at least n. So this tree has a subtree $t_2 = \widehat{t_2} \cdot t_2'$ such that t_2 and t_2' have the same variable-label. We distinguish two cases.

(a) Let t_2 *not* be a subtree of $\widehat{t_1}$. Then $\widehat{t_1}$ and $\widehat{t_2}$ are disjoint and so there exists a $\widetilde{y}$ such that $\mathsf{Y}(t) = \widetilde{y} \cdot \mathsf{Y}(\widehat{t_1}) \cdot \mathsf{Y}(\widehat{t_2})$. Then:

$$\begin{aligned} \mathsf{Y}(\widehat{t} \cdot (\widehat{t_1})^* \cdot t_1') &= \widetilde{y} \cdot \mathsf{Y}(\widehat{t_1})^* \cdot \mathsf{Y}(\widehat{t_2}) \\ &\sqsubseteq \widetilde{y} \cdot \mathsf{Y}(\widehat{t_1})^* \cdot \mathsf{Y}(\widehat{t_2})^* && \text{(def. of Kleene } ^*) \\ &= \widetilde{y} \cdot \mathsf{Y}(\widehat{t_1})^* + \widetilde{y} \cdot \mathsf{Y}(\widehat{t_2})^* && \text{(Prop. 13 (1))} \end{aligned}$$

The expression $\widetilde{y} \cdot \mathsf{Y}(\widehat{t_1})^*$ equals the pumping yield of a decomposition of an X-tree which is obtained from t by removing the substructure $\widehat{t_2}$. Similarly, the expression $\widetilde{y} \cdot \mathsf{Y}(\widehat{t_2})^*$ is equal to the pumping yield of a decomposition of an X-tree which is obtained from t by removing the substructure $\widehat{t_1}$. By induction on the size, both of those pumping yields are $\sqsubseteq \mathsf{Y}(\mathcal{B}_X)$.

(b) Let t_2 *be* a subtree of $\widehat{t_1}$. Then we can write $\widehat{t_1} = \widehat{\widehat{t_1} \cdot \widehat{t_2} \cdot t_2'}$. We have:

$$\begin{aligned}
\mathsf{Y}(\widehat{t} \cdot (\widehat{t_1})^* \cdot t_1') &= \mathsf{Y}(\widehat{t}) \cdot \mathsf{Y}(\widehat{\widehat{t_1} \cdot \widehat{t_2} \cdot t_2'})^* \cdot \mathsf{Y}(t_1') \\
&= \mathsf{Y}(\widehat{t}) \cdot \Big(\mathsf{Y}(\widehat{\widehat{t_1}}) \cdot \mathsf{Y}(\widehat{t_2}) \cdot \mathsf{Y}(t_2')\Big)^* \cdot \mathsf{Y}(t_1') \\
&\sqsubseteq \mathsf{Y}(\widehat{t}) \cdot \Big(\mathsf{Y}(\widehat{\widehat{t_1}}) \cdot \mathsf{Y}(\widehat{t_2})^* \cdot \mathsf{Y}(t_2')\Big)^* \cdot \mathsf{Y}(t_1') \\
&= \left\{ \begin{array}{l} \mathsf{Y}(\widehat{t}) \cdot \mathsf{Y}(\widehat{\widehat{t_1}}) \cdot \mathsf{Y}(\widehat{t_2})^* \cdot \mathsf{Y}(t_2') \cdot \mathsf{Y}(t_1') + \\ \mathsf{Y}(\widehat{t}) \cdot \Big(\mathsf{Y}(\widehat{\widehat{t_1}}) \cdot \mathsf{Y}(t_2')\Big)^* \cdot \mathsf{Y}(t_1') \end{array} \right\} \quad \text{(Prop. 13 (2))} \\
&= \left\{ \begin{array}{l} \mathsf{Y}(\widehat{t} \cdot (\widehat{\widehat{t_1}} \cdot \widehat{t_2}^* \cdot t_2') \cdot t_1') + \\ \mathsf{Y}(\widehat{t} \cdot (\widehat{\widehat{t_1}} \cdot t_2')^* \cdot t_1') \end{array} \right\}
\end{aligned}$$

The first expression in this sum equals $\mathsf{Y}((\widehat{t} \cdot \widehat{\widehat{t_1}} \cdot t_1') \cdot \widehat{t_2}^* \cdot t_2')$. This is the pumping yield of the decomposition $t = (\widehat{t} \cdot \widehat{\widehat{t_1}} \cdot t_1') \cdot \widehat{t_2} \cdot t_2'$. Since $t_2 = \widehat{t_2} \cdot t_2'$ is a proper subtree of $\widehat{t_1} \cdot t_1'$, it has fewer nodes than $\widehat{t_1} \cdot t_1'$. So this decomposition is smaller (in the second component), i.e., by induction, the first expression in the above sum is $\sqsubseteq \mathsf{Y}(\mathcal{B}_X)$.

The second expression in the above sum equals the pumping yield of the decomposition of an X-tree which is obtained from t by removing the substructure $\widehat{t_2}$. By induction, this pumping yield is $\sqsubseteq \mathsf{Y}(\mathcal{B}_X)$. □

5.1. The $(\min, +)$*-Semiring*

Consider the "tropical" semiring $\mathcal{R} = (\mathbb{R} \cup \{-\infty, \infty\}, \wedge, +_{\mathbb{R}}, \infty, 0)$. By $\wedge$ resp. $+_{\mathbb{R}}$ we mean minimum resp. addition over the reals. Observe that the natural order $\sqsubseteq$ is the order $\geq$ on the reals.[3] As $\mathcal{R}$ is totally ordered, Proposition 12 implies that $\mathcal{R}$ is star-distributive. Assume for the rest of this section that $\boldsymbol{f}$ is a polynomial system over $\mathcal{R}$ of degree at most 2. We can apply Theorem 14, i.e., $\mu\boldsymbol{f} = \mu\boldsymbol{f}_{\mathcal{B}}$ holds. This immediately suggests a polynomial algorithm to compute the least fixed-point: Compute $\boldsymbol{f}^n(\boldsymbol{\infty})$ by performing n Kleene iterations, and solve the linear system $\boldsymbol{X} = D\boldsymbol{f}|_{\boldsymbol{f}^n(\boldsymbol{\infty})}(\boldsymbol{X}) \wedge \boldsymbol{f}(\boldsymbol{\infty})$. The latter can be done by means of the Bellman-Ford algorithm.

Example 15. *Consider the following equation system.*

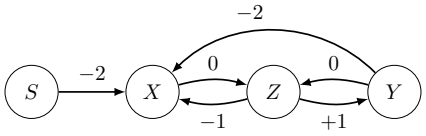
$$\big(X, \quad Y, \quad Z\big) = \big(-2 \wedge (Y +_{\mathbb{R}} Z), \quad Z +_{\mathbb{R}} 1, \quad X \wedge Y\big) =: \boldsymbol{f}(\boldsymbol{X})$$

[3] By symmetry, we could equivalently consider maximum instead of minimum.

We have $\boldsymbol{f}(\boldsymbol{\infty}) = (-2, \infty, \infty), \boldsymbol{f}^2(\boldsymbol{\infty}) = (-2, \infty, -2), \boldsymbol{f}^3(\boldsymbol{\infty}) = (-2, -1, -2)$. *The linear system* $\boldsymbol{X} = D\boldsymbol{f}|_{\boldsymbol{f}^n(\boldsymbol{\infty})}(\boldsymbol{X}) \wedge \boldsymbol{f}(\boldsymbol{\infty}) = \boldsymbol{f}_{\mathcal{B}}(\boldsymbol{X})$ *looks as follows:*

$$\big(X, \quad Y, \quad Z\big) = \big(-2 \wedge (-1 +_{\mathbb{R}} Z) \wedge (Y +_{\mathbb{R}} -2), \quad Z +_{\mathbb{R}} 1, \quad X \wedge Y\big).$$

This equation system corresponds in a straightforward way to the following graph.



We claim that the V*-component of* $\mu\boldsymbol{f}_{\mathcal{B}}$ *equals the least weight of any path from* S *to* V *where* $V \in \{X, Y, Z\}$. *To see this, notice that* $(\boldsymbol{f}^k_{\mathcal{B}}(\boldsymbol{\infty}))_V$ *corresponds to the least weight of any path from* S *to* V *of length at most* k. *The claim follows by Kleene's theorem. So we can compute* $\mu\boldsymbol{f}_{\mathcal{B}}$ *with the Bellman-Ford algorithm. In our example,* X, Y, Z *are all reachable from* S *via a negative cycle, so* $\mu\boldsymbol{f}_{\mathcal{B}} = (-\infty, -\infty, -\infty)$. *By Theorem 14,* $\mu\boldsymbol{f} = \mu\boldsymbol{f}_{\mathcal{B}} = (-\infty, -\infty, -\infty)$. □

The Bellman-Ford algorithm can be used here as it handles negative cycles correctly. The overall runtime of our algorithm to compute $\mu\boldsymbol{f}$ is dominated by the Bellman-Ford algorithm. Its runtime is in $\mathcal{O}(n \cdot m)$, where m is the number of monomials appearing in $\boldsymbol{f}$. We conclude that our algorithm has the same asymptotic complexity as the "generalized Bellman-Ford" algorithm of [15]. It is by a factor of n faster than the algorithm deducible from [10] because our new algorithm uses the Bellman-Ford algorithm only once instead of n times.

5.2. Throughput of Grammars

In [7], a polynomial algorithm for computing the *throughput* of a context-free grammar was given. Now we show that the algorithm can be both simplified and accelerated by computing least fixed-points according to Theorem 14.

Let us define the problem following [7]. Let Σ be a finite alphabet and $\rho : \Sigma \to \mathbb{N}$ a weight function. We extend ρ to words $a_1 \cdots a_k \in \Sigma^*$ by setting $\rho(a_1 \cdots a_k) := \rho(a_1) + \ldots + \rho(a_k)$.[4] The mean weight of a non-empty word w is defined as $\overline{\rho}(w) := \rho(w)/|w|$. The throughput of a non-empty language $L \subseteq \Sigma^+$ is defined as the infimum of the mean weights of the words in L: $tp(L) := \inf\{\overline{\rho}(w) \mid w \in L\}$. Let $G = (\Sigma, \mathcal{X}, P, S)$ be a context-free grammar and $L = L(G)$ its language. The problem is to compute $tp(L)$. As in [7] we assume that G has at most 2 symbols on the right hand side of every production and that L is non-empty and contains only non-empty words.

Note that we cannot simply construct a polynomial system having $tp(L)$ as its least fixed-point, as the throughput of two non-terminals is not additive. In [7] an ingenious algorithm is proposed to avoid this problem. Assume we already know a routine, the *comparing routine*, that decides for a given $t \in \mathbb{Q}$ whether $tp(L) \geq t$ holds. Assume further that this routine has $\mathcal{O}(N^k)$ time complexity for some k. Using

[4] We write $+$ for the addition of reals in this section.

the comparing routine we can approximate $tp(L)$ up to any given accuracy by means of binary search. Let $d = \max_{a\in\Sigma} \rho(a) - \min_{a\in\Sigma} \rho(a)$. A dichotomy result of [7] shows that $\mathcal{O}(N + \log d)$ iterations of binary search suffice to approximate $tp(L)$ up to an ε that allows to compute the exact value of $tp(L)$ in time $\mathcal{O}(N^3)$. This is proved by showing that, once a value t has been determined such that $t - \varepsilon < tp(L) \leq t$, one can:

- transform G in $\mathcal{O}(N^3)$ time into a grammar G' of size $\mathcal{O}(N^3)$ generating a finite language, and having the same throughput as G (this construction does not yet depend on $tp(L)$);
- compute the throughput of G' in linear time in the size of G', i.e., in $\mathcal{O}(N^3)$ time.

The full algorithm for the throughput runs then in $\mathcal{O}(N^k(N + \log d)) + \mathcal{O}(N^3)$ time.

The algorithm of [7] and our new algorithm differ in the comparing routine. In the routine of [7] the transformation of G into the grammar G' is done *before* $tp(L)$ has been determined. Then a linear time algorithm can be applied to G' to decide whether $tp(L) \geq t$ holds. (This algorithm does not work for arbitrary context-free grammars, and that is why one needs to transform G into G'.) Since G' has size $\mathcal{O}(N^3)$, the comparing routine has $k = 3$, and so the full algorithm runs in $\mathcal{O}(N^4 + N^3 \log d)$ time.

We give a more efficient comparing routine with $k = 2$. Given a $t \in \mathbb{Q}$, assign to each word $w \in \Sigma^+$ its *throughput balance* $\sigma_t(w) = \rho(w) - |w| \cdot t$. Notice that $\sigma_t(w) \geq 0$ if and only if $\overline{\rho}(w) \geq t$. Further, for two words w, u we now have $\sigma_t(wu) = \sigma_t(w) + \sigma_t(u)$. So we can set up a polynomial system $\boldsymbol{X} = \boldsymbol{f}(\boldsymbol{X})$ over the tropical semiring $\mathcal{R}$ where $\boldsymbol{f}$ is constructed such that each variable $X \in \mathcal{X}$ in the equation system corresponds to the minimum (infimum) throughput balance of the words derivable from X. More formally, define a map m by setting $m(a) = \rho(a) - t$ for $a \in \Sigma$ and $m(X) = X$ for $X \in \mathcal{X}$. Extend m to words in $(\Sigma \cup \mathcal{X})^*$ by setting $m(\alpha_1 \cdots \alpha_k) = m(\alpha_1) + \cdots + m(\alpha_k)$. Let P_X be the productions of G with X on the left hand side. Then set $\boldsymbol{f}_X(\boldsymbol{X}) := \bigwedge_{(X\to w)\in P_X} m(w)$. For instance, if P_X consists of the rules $X \to aXY$ and $X \to bZ$, we have $\boldsymbol{f}_X(\boldsymbol{X}) = \rho(a) - t + X + Y \wedge \rho(b) - t + Z$.

It is easy to see that the relevant solution of the system $\boldsymbol{X} = \boldsymbol{f}(\boldsymbol{X})$ is the least one w.r.t. $\sqsubseteq$, i.e., $(\mu\boldsymbol{f})_S \geq 0$ if and only if $tp(L) \geq t$. So we can use the algorithm from Section 5.1 as our comparing routine. This takes time $\mathcal{O}(N^2)$ where N is the size of the grammar. With that comparing routine we obtain an algorithm for computing the throughput with $\mathcal{O}(N^3 + N^2 \log d)$ runtime.

6. Lossy Semirings

Definition 16. An io-semiring $\mathcal{S}$ is called *lossy* if $1 \sqsubseteq a$ holds for all $a \neq 0$.

Note that by definition of natural order the requirement $1 \sqsubseteq a$ is equivalent to $a = a+1$. In the free semiring generated by a finite alphabet Σ, and augmented by the equation $a = a + 1$ ($a \in S \setminus \{0\}$), every language $L \subseteq \Sigma^*$ is "downward closed", i.e. for

every word $w = a_1 a_2 \dots a_l \in L$ all possible subwords $\{a'_1 a'_2 \dots a'_l \mid a'_i \in \{\varepsilon, a_i\}\}$ are also included in L. By virtue of Higman's lemma [17] the downward-closure of a context-free language is regular. This has been used in [1] for an efficient analysis of systems with unbounded, lossy FIFO channels. Downward closure was used there to model the loss of messages due to transmission errors.

We say that a system $\boldsymbol{f}$ of polynomials is *clean* if $\mu \boldsymbol{f}_X \neq 0$ for all $X \in \mathcal{X}$. Every system can be *cleaned* in linear time by removing the equations of all variables X such that $\mu \boldsymbol{f}_X = 0$ and setting these variables to 0 in the other equations (the procedure is similar to the one that eliminates non-productive variables in context-free grammars). We consider only clean systems, and introduce a normal form for them.

Definition 17. Let $\boldsymbol{f} \in \mathcal{S}[\mathcal{X}]^{\mathcal{X}}$ be a system of polynomials over a lossy semiring. $\boldsymbol{f}$ is in *quadratic normal form* if every polynomial $\boldsymbol{f}_X$ has the form

$$c + \sum_{Y,Z \in \mathcal{X}} a_{Y,Z} \cdot Y \cdot Z + \sum_{Y \in \mathcal{X}} b_{l,Y} \cdot Y \cdot b_{r,Y}$$

where for all $Y, Z \in \mathcal{X}$: (i) $c \neq 0$, (ii) $a_{Y,Z} \in \{0, 1\}$, and (iii) if $\sum_{Z \in \mathcal{X}} a_{Y,Z} \neq 0$, then $b_{l,Y}, b_{r,Y}, b_{l,Z}, b_{r,Z} \neq 0$.

Lemma 18. *For every clean $\boldsymbol{g} \in \mathcal{S}[\mathcal{X}]^{\mathcal{X}}$ we can construct in linear time a system $\boldsymbol{f} \in \mathcal{S}[\mathcal{X}']^{\mathcal{X}'}$ in quadratic normal form, where $\mathcal{X} \subseteq \mathcal{X}'$ and $\mu \boldsymbol{g}_X = \mu \boldsymbol{f}_X$ for all $X \in \mathcal{X}$.*

PROOF. For every clean $\boldsymbol{g} \in \mathcal{S}[\mathcal{X}]^{\mathcal{X}}$ we can find a $\boldsymbol{f} \in \mathcal{S}[\mathcal{X}']^{\mathcal{X}'}$ in normal form with $\mathcal{X} \subseteq \mathcal{X}'$ such that $\mu \boldsymbol{g}_X = \mu \boldsymbol{f}_X$ for all $X \in \mathcal{X}$ as follows: We first transform $\boldsymbol{g}$ into Chomsky normal-form, which gives us a system $\boldsymbol{g}'$ over the same semiring. As the transformation into Chomsky normal-form introduces new variables, $\boldsymbol{g}'$ is given in a superset $\mathcal{X}'$ of $\mathcal{X}$ with $\mu \boldsymbol{g}'_X = \mu \boldsymbol{g}_X$ for all $X \in \mathcal{X}$. Next, as $\boldsymbol{g}$ is clean, we can ensure that $\boldsymbol{g}'$ is clean, too. We therefore may set $\boldsymbol{g}'' := \boldsymbol{g}' + \mathbf{1}$ without changing the least solution. Hence, every polynomial of $\boldsymbol{g}''_X$ has the form

$$c^{(X)} + \sum_{Y,Z \in \mathcal{X}'} a^{(X)}_{Y,Z} \cdot Y \cdot Z \text{ with } 1 \sqsubseteq c^{(X)} \text{ and } a^{(X)}_{Y,Z} \in \{0, 1\}.$$

Finally, as $\mathbf{1} \sqsubseteq \mu \boldsymbol{g}''$ we have

$$\begin{aligned}
\mu \boldsymbol{g}''_X &= \boldsymbol{g}''_X(\mu \boldsymbol{g}'') \\
&= c^{(X)} + \sum_{Y,Z \in \mathcal{X}'} a^{(X)}_{Y,Z} \cdot \mu \boldsymbol{g}''_Y \cdot \mu \boldsymbol{g}''_Z \\
&= c^{(X)} + \sum_{Y,Z \in \mathcal{X}'} a^{(X)}_{Y,Z} \cdot (1 + \mu \boldsymbol{g}''_Y) \cdot (1 + \mu \boldsymbol{g}''_Z) \\
&= c^{(X)} + \sum_{Y,Z \in \mathcal{X}'} a^{(X)}_{Y,Z} \\
&\quad + \sum_{Y,Z \in \mathcal{X}'} a^{(X)}_{Y,Z} \cdot \mu \boldsymbol{g}''_Y \cdot \mu \boldsymbol{g}''_Z + \sum_{Y \in \mathcal{X}'} \left(\sum_{Z \in \mathcal{X}'} a^{(X)}_{Y,Z} + a^{(X)}_{Z,Y} \right) \cdot \mu \boldsymbol{g}''_Y.
\end{aligned}$$

Note that $\sum_{Y,Z\in\mathcal{X}'} a^{(X)}_{Y,Z} \sqsubseteq 1$ by idempotence. As $c^{(X)} \sqsupseteq 1$, we therefore may write:

$$\begin{aligned}\mu\boldsymbol{g}''_X &= c^{(X)} + \sum_{Y,Z\in\mathcal{X}'} a^{(X)}_{Y,Z}\cdot\mu\boldsymbol{g}''_Y\cdot\mu\boldsymbol{g}''_Z + \sum_{Y\in\mathcal{X}'}\left(\sum_{Z\in\mathcal{X}'} a^{(X)}_{Y,Z} + a^{(X)}_{Z,Y}\right)\cdot\mu\boldsymbol{g}''_Y\\ &= \boldsymbol{g}''_X(\mu\boldsymbol{g}''_X) + \sum_{Y\in\mathcal{X}'}\left(\sum_{Z\in\mathcal{X}'} a^{(X)}_{Y,Z} + a^{(X)}_{Z,Y}\right)\cdot\mu\boldsymbol{g}''_Y.\end{aligned}$$

We now define $\boldsymbol{f}$ by setting for all $X\in\mathcal{X}'$

$$\boldsymbol{f}_X := \boldsymbol{g}''_X + \sum_{Y\in\mathcal{X}'}\left(\sum_{Z\in\mathcal{X}'} a^{(X)}_{Y,Z} + a^{(X)}_{Z,Y}\right)\cdot Y.$$

We then have $\boldsymbol{g}''\sqsubseteq\boldsymbol{f}$, and, thus, $\mu\boldsymbol{g}''\sqsubseteq\mu\boldsymbol{f}$, but also $\boldsymbol{f}(\mu\boldsymbol{g}'')=\mu\boldsymbol{g}''$, i.e. $\mu\boldsymbol{g}''\sqsupseteq\mu\boldsymbol{f}$. □

Our main result in this section is that for *strongly connected* systems $\boldsymbol{f}$ in quadratic normal form we again have that $\mu\boldsymbol{f}=\mu\boldsymbol{f}_{\mathcal{B}}$. We then show how this result leads to an algorithm for arbitrary systems.

Given two variables $X,Y\in\mathcal{X}$, we say that X depends on Y (w.r.t. $\boldsymbol{f}$) if Y occurs in a monomial of $\boldsymbol{f}_X$ or there is a variable Z such that X depends on Z and Z depends on Y. The system $\boldsymbol{f}$ is *strongly connected* if X depends on Y for all variables X,Y.

Theorem 19. *$\mu\boldsymbol{f}=\mu\boldsymbol{f}_{\mathcal{B}}$ holds for strongly connected polynomial systems $\boldsymbol{f}$ in quadratic normal form over lossy semirings.*

We first illustrate the construction underlying the proof of Theorem 19:

Proof Sketch. We consider a concrete example of a tree t that is not a bamboo, and show how to construct a bamboo $\hat{t}$ such that $\mathsf{Y}(t)\sqsubseteq\mathsf{Y}(\hat{t})$. The general procedure for all non-bamboos can be found in the appendix. Let $\mathcal{X}=\{X,Y\}$, and $\boldsymbol{f}$ with $\boldsymbol{f}_X=XY+X+Y+a$, and $\boldsymbol{f}_Y=X+Y+b$. Consider the X-tree t depicted on the left of the picture below, where t_r is some bamboo of height at least 2 (we inductively assume that the original subtree has already been replaced by a bamboo with at least the same yield). Since the left subtree of t has height 2, t itself is not a bamboo. Let s denote the left-most leaf of t, and let r be the parent of s. In our example, we assume that r has s as its only child. Then we proceed as follows:

(i) We remove from t the leaf s, and turn its father r into a leaf. Here, we make use of the assumption that $\boldsymbol{f}$ is in quadratic normal form, and so every polynomial of $\boldsymbol{f}$ contains a constant monomial, in our example b. We change the monomial-label of r to b, and obtain the tree t', which is a derivation tree of $\boldsymbol{f}$. Moreover, t' is a bamboo, because its left subtree has now height 1, and its right subtree t_r is a bamboo.

(ii) We prepend a (partial) derivation tree on top of t_r having two linear chains as subtrees: the left chain leads to the leaf s, and the right chain leads to t'. This gives us the tree $\hat{t}$ depicted on the right of the picture above. The proof of Theorem 19 shows that these chains exist and have at most length $n-1$ (in our example $n-1=1$). It follows that $\hat{t}$ is a bamboo itself with stem $\hat{t}-t''-t'-t_r$, and so $\mathsf{Y}(\hat{t})\sqsubseteq\mathsf{Y}(\mathcal{B}_X)$.

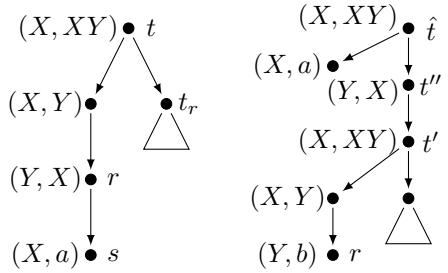


Figure 4: Transforming a derivation tree into a bamboo over a lossy semiring.

We have $\mathsf{Y}(t) = a \cdot \mathsf{Y}(t_r)$ and $\mathsf{Y}(\hat{t}) = a \cdot b \cdot \mathsf{Y}(t_r)$. Since the semiring is lossy, we have $1 \sqsubseteq b$ and so $\mathsf{Y}(t) \sqsubseteq \mathsf{Y}(\hat{t})$. Notice that, since product is not necessarily commutative, it is important that a is the first factor of both yields. □

For the proof of Theorem 19, we first define partial derivation trees: These trees result from derivation trees by removing exactly one subtree, leaving in some sense a "dangling pointer".

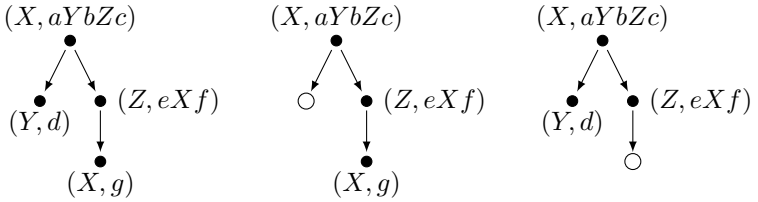
Definition 20. Let $\boldsymbol{f} \in \mathcal{S}[\mathcal{X}]^{\mathcal{X}}$. Let t be some X-tree for $X \in \mathcal{X}$. Further, let $Y \in \mathcal{X}$ be some variable such that t has at least one leaf s with $\lambda_v(s) = Y$. By erasing exactly one such leaf s from t, we obtain an *XY-tree*. We write $\mathcal{T}_{X,Y}$ for the set of all XY-trees.

The set $\mathcal{B}_{X,Y}$ is defined similarly. A tree $t \in \mathcal{B}_{X,Y}$ results from a tree $t \in \mathcal{B}_X$ by removing the exactly one such leaf s of t with $\lambda_v(s) = Y$, where s lies on a longest path from t' to a leaf.

The yield $\mathsf{Y}(t)$ of an XY-tree t is a linear monomial in Y defined analogously to the yield of an X-tree; the single variable occuring in $\mathsf{Y}(t)$ corresponds to the missing Y-subtree in t.

We visualize XY-trees by representing the missing subtree with ○.

Example 21. *Consider the X-tree depicted on the left. By deleting the leaf labeled by (Y, d), we obtain the XY-tree depicted in the middle, where we represent the missing leaf/subtree by ○, with yield $aYbegfc$. Similarily, we obtain the XX-tree shown on the right by deleting the leaf labeled by (X, g) with yield $adbeXfc$.*

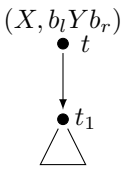


Note that we can replace ○ in the XY-tree by any Y-tree in order to obtain a valid X-tree, again.

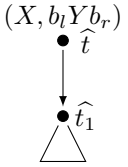
Now we can prove Theorem 19:

PROOF. We again show that we can transform any X-tree t w.r.t. $\boldsymbol{f}$ into a tree $\hat{t}$ contained in $\mathcal{B}_X$ with $\mathsf{Y}(t) \sqsubseteq \mathsf{Y}(\hat{t})$. We proceed by induction on the number N of nodes of t. If $N = 1$, then t has height 0. By definition, we have $t \in \mathcal{B}_X$, so we are done.

Therefore assume $N > 1$. As $\boldsymbol{f}$ is in normal form, we either have $\lambda_m(t) = b_l Y b_r$ or $\lambda_m(t) = YZ$ for some $Y, Z \in \mathcal{X}$, and $b_l, b_r \in S \setminus \{0\}$. If t is labeled by $\lambda_m(t) = b_l Y b_r$,

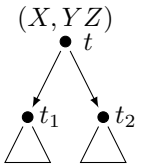


then t has exactly one child t_1, which immediately can be replaced by some tree $\widehat{t_1}$ in $\mathcal{B}_Y$ with $\mathsf{Y}(t_1) = \mathsf{Y}(\widehat{t_1})$ because of induction. This gives us the tree $\widehat{t}$

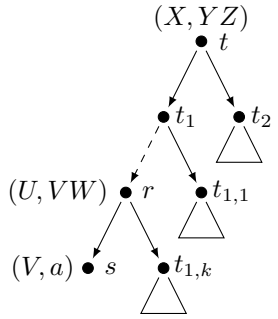


and $\mathsf{Y}(\widehat{t}) = b_l \mathsf{Y}(\widehat{t_1}) b_r = b_l \mathsf{Y}(t_1) b_r = \mathsf{Y}(t)$.

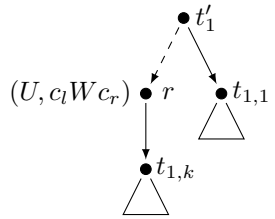
Hence, assume that $\lambda_m(t) = YZ$, i.e. t has two children t_1, t_2.



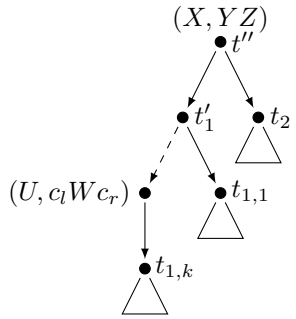
Descending into t_1 by always taking the leftmost child, we end up at the left most leaf s of t. We denote by $t_{1,1}$ to $t_{1,k}$ the "right" children of the nodes located on the path from t_1 to s for some $k \in \mathbb{N}$. Let r then be the father of s with $\lambda_v(s) = V$, and $\lambda_m(s) = a \in S$. We assume that $\lambda_m(r) = VW$ for some $W \in \mathcal{X}$



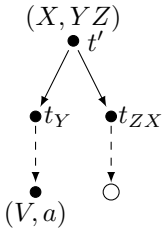
As $\boldsymbol{f}$ is in normal form, and VW is a monomial of $\boldsymbol{f}_U$, there exists also a monomial $c_l W c_r$ appearing in $\boldsymbol{f}_U$ for some $c_l, c_r \in S \setminus \{0\}$. We first remove from t_1 the leaf s, and relabel the node r by setting $\lambda_m(r) := c_l W c_r$. This gives us the tree t_1' with $\mathsf{Y}(t_1) \sqsubseteq a \cdot \mathsf{Y}(t_1')$, as $1 \sqsubseteq c_l, c_r$:



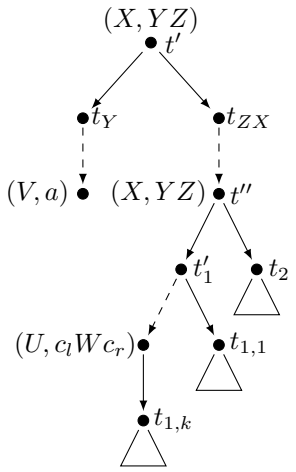
As YZ is a monomial of $\boldsymbol{f}_X$ we can construct from the trees t_1' and t_2 the tree t'':



Now, as $\boldsymbol{f}$ is strongly connected and in normal form, we find an Y-tree t_Y of height at most $n-1$ which has (V, a) as its single leaf, such that $a \sqsubseteq \mathsf{Y}(t_Y)$; similarly, we find a ZX-tree t_{ZX} of height at most $n-1$ having $\bigcirc$ as its single leaf; the "yield" of t_{ZX} is some monomial $d_l X d_r$ for some $d_l, d_r \in S \setminus \{0\}$. Using these, we construct the following tree t' with $\lambda_v(t') = X$, and $\lambda_m(t') = YZ$. As left child of t', we take the Y-tree t_Y, whereas we take t_{ZX} as the right child, giving us:



We complete this partial derivation tree to a derivation tree by replacing $\bigcirc$ with the tree t'':

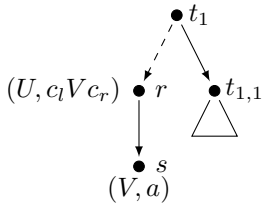


We now have

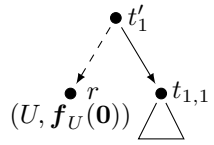
$$\begin{aligned}\mathsf{Y}(t') = \mathsf{Y}(t_Y) \cdot \mathsf{Y}(t_{ZX}) &= \mathsf{Y}(t_Y) \cdot d_l \cdot \mathsf{Y}(t'') \cdot d_r \\ &\sqsupseteq a \cdot d_l \cdot \mathsf{Y}(t'') \cdot d_r \\ &\sqsupseteq a \cdot \mathsf{Y}(t'_1) \cdot \mathsf{Y}(t_2) \\ &\sqsupseteq \mathsf{Y}(t_1) \cdot \mathsf{Y}(t_2) \\ &= \mathsf{Y}(t).\end{aligned}$$

By construction of t', the left child is a Y-tree of height at most $n-1$, while every node from t_{ZX} to t'' has exactly one child. Hence, only the subtree t'' might not have the required form. But as t'' has one node less than t, we find by induction on the number of nodes a tree $\widehat{t''} \in \mathcal{B}_X$ with $\mathsf{Y}(t'') \sqsubseteq \mathsf{Y}(\widehat{t''})$. Replacing in t' the subtree t'' by this tree $\widehat{t''}$, we then obtain the tree $\hat{t}$ with $\hat{t} \in \mathcal{B}_X$ and $\mathsf{Y}(\hat{t}) \sqsupseteq \mathsf{Y}(t') \sqsupseteq \mathsf{Y}(t)$. This ends the case that $\lambda_m(r) = VW$.

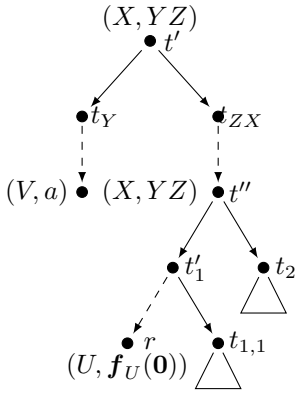
Assume therefore that $\lambda_m(r) = c_l V c_r$ for some $c_l, c_r \in S \setminus \{0\}$, i.e.



We proceed similarly to the previous case, but we define t'_1 as follows: again, we remove the leaf s from t_1, but as r has s as its only child, we now relabel r by $\lambda_m(r) := \boldsymbol{f}_U(\mathbf{0})$. As $\boldsymbol{f}$ is clean, we have $\boldsymbol{f}_U(\mathbf{0}) \sqsupseteq 1$. This gives us:



and



Again, we can find a $\widehat{t''} \in \mathcal{B}_X$ with $\mathsf{Y}(t'') \sqsubseteq \mathsf{Y}(\widehat{t''})$ as t'' has one node less than t, and the induction is complete. □

Because of the preceding theorem, given a strongly connected system $\boldsymbol{f}$, we may use the linear system $\boldsymbol{f}_{\mathcal{B}}(\boldsymbol{X}) = \boldsymbol{f}(\mathbf{0}) + D\boldsymbol{f}|_{\boldsymbol{f}^n(\mathbf{0})}(\boldsymbol{X})$ for calculating $\mu\boldsymbol{f}$. As $\boldsymbol{f}$ is strongly connected, $\boldsymbol{f}_{\mathcal{B}}$ is also strongly connected. The least fixed-point of such a strongly connected linear system $\boldsymbol{f}_{\mathcal{B}}$ is easily calculated: all non-constant monomials appearing in $\boldsymbol{f}_{\mathcal{B}}$ have the form $b_l X b_r$ for some $X \in \mathcal{X}$, and $b_l, b_r \in S \setminus \{0\}$. As $\boldsymbol{f}_{\mathcal{B}}$ is strongly connected, every polynomial $(\boldsymbol{f}_{\mathcal{B}})_Y$ is substituted for Y in $(\boldsymbol{f}_{\mathcal{B}})_X$ again and again when calculating the Kleene sequence $(\boldsymbol{f}^k_{\mathcal{B}}(\mathbf{0}))_{k\in\mathbb{N}}$. So, let l be the sum of all left-handed coefficients b_l (appearing in *any* $\boldsymbol{f}_X$), and similarly define r. We then have $(\mu\boldsymbol{f}_{\mathcal{B}})_X = l^*\big(\sum_{Y\in\mathcal{X}} \boldsymbol{f}_Y(\mathbf{0})\big)r^*$ for all $X \in \mathcal{X}$.

If $\boldsymbol{f}$ is not strongly connected, we first decompose $\boldsymbol{f}$ into strongly connected subsystems, and then we solve these systems bottom-up. Note that substituting the solutions from underlying SCCs into a given SCC leads to a new system in normal form. As there are at most $n = |\mathcal{X}|$ many strongly connected components for a given system $\boldsymbol{f} \in \mathcal{S}[\mathcal{X}]^{\mathcal{X}}$, we obtain the following theorem which was first stated explicitly for context-free grammars in [8].

Theorem 22. *The least fixed-point $\mu\boldsymbol{f}$ of a polynomial system $\boldsymbol{f}$ over a lossy semiring is representable by regular expressions over $\mathcal{S}$. If $\boldsymbol{f}$ is in normal form, $\mu\boldsymbol{f}$ can be calculated solving at most n bamboo systems.*

7. 1-bounded Semirings

Definition 23. An io-semiring $\mathcal{S}$ is called 1-*bounded* if $a \sqsubseteq 1$ holds for all $a \in S$.

Natural examples are the tropical semiring over the natural numbers $(\mathbb{N} \cup \{\infty\}, \wedge, +, \infty, 0)$ and the "maximum-probability" semiring $([0,1], \vee, \cdot, 0, 1)$, where $\wedge$ and $\vee$ denote minimum and maximum, respectively. Notice that any commutative 1-bounded semiring is star-distributive (as $a^* = 1$ for all a), but not all 1-bounded semirings have commutative multiplication. Consider for example the *upward-closed* languages over Σ, i.e., the languages L such that $w \in L$ implies $u \in L$ for all u such that w is a scattered subword of u. If we take union and concatenation of languages as sum and product, and $\emptyset$ and Σ^* as 0- and 1-elements, we obtain a 1-bounded semiring. Upward-closed languages form a natural dual to downward-closed languages from the previous section.
We show that $\mu\boldsymbol{f}$ can be computed very easily in the case of 1-bounded semirings:

Theorem 24. *$\mu\boldsymbol{f} = \boldsymbol{f}^n(\mathbf{0})$ holds for polynomial systems over 1-bounded semirings.*

PROOF. We reuse the notation from the proof of Theorem 14: If t_2 is a subtree of a derivation tree t, we write $t = t_1 \cdot t_2$ where t_1 is the partial derivation tree obtained from t by removing t_2.

Recall that, by Proposition 2, $(\boldsymbol{f}^n(\mathbf{0}))_X = \mathsf{Y}(\mathcal{T}_X^{(n-1)})$, where $\mathcal{T}_X^{(n-1)}$ contains all X-trees of height at most $n-1$. We proceed by derivation tree analysis, i.e., by discharging the precondition of Proposition 4. So it suffices to show that for any X-tree t there is a tree t' of height at most $n-1$ with $\mathsf{Y}(t) \sqsubseteq \mathsf{Y}(t')$. We proceed by induction on the number of nodes in t. For the induction base, t has just one node, so $t \in \mathcal{T}_X^{(0)}$. For the induction step w.l.o.g. let t be an X-tree with a height of at least n. Then there is a pumpable decomposition $t = t_1 \cdot t_2 \cdot t_3$ with $\lambda_1(t_2) = \lambda_1(t_3)$. We have $\mathsf{Y}(t) = y_1y_2y_3y_4y_5$ where $\mathsf{Y}(t_1) = y_1y_5$, $\mathsf{Y}(t_2) = y_2y_4$ and $\mathsf{Y}(t_3) = y_3$. Let $t' = t_1 \cdot t_3$. Notice that t' is a valid X-tree as $\lambda_1(t_2) = \lambda_1(t_3)$. We have $\mathsf{Y}(t') = y_1y_3y_5$ which is, by 1-bounded-ness, at least $y_1y_2y_3y_4y_5 = \mathsf{Y}(t)$. As t' has fewer nodes than t, there is, by induction hypothesis, an X-tree t'' of height at most $n-1$ such that $\mathsf{Y}(t') \sqsubseteq \mathsf{Y}(t'')$. Combined we get $\mathsf{Y}(t) \sqsubseteq \mathsf{Y}(t') \sqsubseteq \mathsf{Y}(t'')$. □

Theorem 24 appears to be rather easy from our point of view, i.e., from the point of view of derivation trees. However, even this simple result has very concrete applications in the domain of interprocedural program analysis [23]. The main algorithms of [23], the so-called *post** and *pre** algorithms, can be seen as solvers of fixed-point equations over *bounded* semirings, which are semirings that do not have infinite ascending chains. Those solvers are based on Kleene's iteration and the complexity result given there depends on the maximal length of ascending chains in the semiring (cf. [23], page 28). Such a bound may not exist, and does not exist for the tropical semiring over the natural numbers $(\mathbb{N} \cup \{\infty\}, \wedge, +, \infty, 0)$ which is considered as an example in [23], pages 13 and 18. However, Theorem 24 can be applied to this semiring, which shows that the program analysis algorithms of [23] applied to 1-bounded semirings are polynomial-time algorithms, independent of the length of chains in the semiring.

8. Conclusion

We have shown that derivation tree analysis, a proof technique first introduced in [10], is an efficient tool for the design of efficient fixed-point algorithms on io-

semirings. We have considered three classes of io-semirings with applications to language theory and verification. We have shown that for star-distributive semirings and lossy semirings the least fixed-point of a polynomial system of equations is equal to the least fixed-point of a *linear system*, the bamboo system. This improves the results of [10]: The generic algorithm given there requires to solve N different systems of linear equations in the star-distributive case (where N is the original number of polynomial equations), and is not applicable to the lossy case.

We have used our results to design an efficient fixed-point algorithm for the $(\min, +)$-semiring. In turn, we have applied this algorithm to provide a cubic algorithm for computing the throughput of a context-free language, improving the $\mathcal{O}(N^4)$ upper bound obtained by Caucal et al. in [7].

For lossy semirings, derivation tree analysis based on bamboos has led to an algebraic generalization of a result of Courcelle stating that the downward-closure of a context-free language is effectively regular. Finally we have used derivation tree analysis to derive a simple proof that $\mu \boldsymbol{f} = \boldsymbol{f}^n(\mathbf{0})$ holds for 1-bounded semirings, with some applications in interprocedural program analysis.

Acknowledgments

We thank two anonymous referees for helpful comments. The first author would like to thank Jan Bergstra for his beautiful course at the Marktoberdorf Summer School in 1990, which left a mark.

References

[1] P. A. Abdulla, A. Bouajjani, and B. Jonsson. On-the-fly analysis of systems with unbounded, lossy FIFO channels. In *CAV'98*, LNCS 1427, pages 305–318. Springer, 1998.

[2] J. Baeten and W.P. Wejland. *Process Algebra*. Cambridge University Press, 1990.

[3] J. Bergstra and J.W. Klop. Fixed point semantics in process algebras. Technical Report IW 206, Math. Centre, Amsterdam, 1982.

[4] Jan A. Bergstra and Jan Willem Klop. Process algebra for synchronous communication. *Information and Control*, 60(1-3):109–137, 1984.

[5] Stefan Blom, Wan Fokkink, Jan Friso Groote, Izak van Langevelde, Bert Lisser, and Jaco van de Pol. μcrl: A toolset for analysing algebraic specifications. In Gérard Berry, Hubert Comon, and Alain Finkel, editors, *CAV*, volume 2102 of *Lecture Notes in Computer Science*, pages 250–254. Springer, 2001.

[6] Ahmed Bouajjani, Javier Esparza, and Tayssir Touili. A generic approach to the static analysis of concurrent programs with procedures. In *POPL*, pages 62–73, 2003.

[7] D. Caucal, J. Czyzowicz, W. Fraczak, and W. Rytter. Efficient computation of throughput values of context-free languages. In *CIAA'07*, LNCS 4783, pages 203–213. Springer, 2007.

[8] B. Courcelle. On constructing obstruction sets of words. *EATCS Bulletin*, 44:178–185, 1991.

[9] J. Esparza, S. Kiefer, and M. Luttenberger. An extension of Newton's method to ω-continuous semirings. In *DLT'07*, LNCS 4588, pages 157–168. Springer, 2007.

[10] J. Esparza, S. Kiefer, and M. Luttenberger. On fixed point equations over commutative semirings. In *STACS'07*, LNCS 4397, pages 296–307. Springer, 2007.

[11] J. Esparza, A. Kučera, and R. Mayr. Model checking probabilistic pushdown automata. *Logical Methods in Computer Science*, 2006.

[12] Javier Esparza, Andreas Gaiser, and Stefan Kiefer. Computing least fixed points of probabilistic systems of polynomials. In Jean-Yves Marion and Thomas Schwentick, editors, *STACS*, volume 5 of *LIPIcs*, pages 359–370. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010.

[13] Kousha Etessami and Mihalis Yannakakis. Recursive markov chains, stochastic grammars, and monotone systems of nonlinear equations. *J. ACM*, 56(1), 2009.

[14] W. Fokkink. *Introduction to Process Algebra*. Springer-Verlag, 2000.

[15] T. Gawlitza and H. Seidl. Precise fixpoint computation through strategy iteration. In *ESOP'07*, LNCS 4421, pages 300–315. Springer, 2007.

[16] T.E. Harris. *The Theory of Branching Processes*. Springer, 1963.

[17] G. Higman. Ordering by divisibility in abstract algebras. *Proc. London Math. Soc.*, 2, 1952.

[18] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[19] M. W. Hopkins and D. Kozen. Parikh's theorem in commutative Kleene algebra. In *LICS'99*, 1999.

[20] 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.

[21] R. Milner. *Communication and concurrency*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

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

[23] 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.

[24] Thomas W. Reps, Susan Horwitz, and Shmuel Sagiv. Precise interprocedural dataflow analysis via graph reachability. In *POPL*, pages 49–61, 1995.

[25] Grzegorz Rozenberg and Arto Salomaa, editors. *Handbook of formal languages, vol. 2: linear modeling: background and application*. Springer-Verlag New York, Inc., New York, NY, USA, 1997.