Weighted parsing for grammar-based language models

Richard Mörbitz
Faculty of Computer Science
Technische Universität Dresden
01062 Dresden
richard.moerbitz@tu-dresden.de

Heiko Vogler
Faculty of Computer Science
Technische Universität Dresden
01062 Dresden
heiko.vogler@tu-dresden.de

Abstract

We develop a general framework for weighted parsing which is built on top of grammar-based language models and employs flexible weight algebras. It generalizes previous work in that area (semiring parsing, weighted deductive parsing) and also covers applications outside the classical scope of parsing, e.g., algebraic dynamic programming. We show an algorithm which terminates and is correct for a large class of weighted grammar-based language models.

1 Introduction

The weighted parsing problem takes as input a weighted language model (LM) and a syntactic object a. For instance, the LM can be given by some grammar G, e.g., a context-free grammar (CFG) or a linear context-free rewriting system (LCFRS), and a can be some string. Each rule r of G has a value (*weight of* r); the weight is an element of some *weight algebra* $\mathbb{K}$. That algebra has a binary commutative and associative operation $\oplus$ on its carrier set, which is used to handle ambiguity of G. As output we expect an element $\Bbbk \in \mathbb{K}$ which is the $\oplus$-accumulation of the weight $\mathrm{wt}(d)_{\mathbb{K}}$ of each abstract syntax tree (AST) d of a in G, i.e.,

$$\Bbbk = \sum_{d \in \mathrm{AST}(G,a)}^{\oplus} \mathrm{wt}(d)_{\mathbb{K}}$$

where $\mathrm{wt}(d)_{\mathbb{K}}$ is computed by other operations of the algebra $\mathbb{K}$ (using the weights of the occurring rules) and $\sum^{\oplus}$ is an extension of $\oplus$ to infinitely many summands (*infinitary sum operation*). For instance, if $\mathbb{K} = [0, 1]$ is the set of probabilities, $\oplus = \max$, $\sum^{\oplus} = \sup$, and $\mathrm{wt}(d)_{\mathbb{K}}$ is the product of all weights of occurrences of rules in d, then $\Bbbk$ is the maximal probability of an AST of a in G.

Goodman (1999) developed a formal framework for weighted parsing, called *semiring parsing*. As weight algebras he used complete semirings $(\mathbb{K}, \oplus, \otimes, \mathbb{0}, \mathbb{1}, \sum^{\oplus})$ (Eilenberg, 1974), i.e., $\sum^{\oplus}$ is the infinitary sum operation extending $\oplus$. The binary operation $\otimes$ is used to compute $\mathrm{wt}(d)_{\mathbb{K}}$. By appropriate choices of the complete semiring, he formalized the following problems as weighted parsing problems for a CFG G and a: the calculation of recognition, string probabilities, number of derivations, derivation forests, probability of best derivation, best derivation, and best n derivations. The algorithm which he proposed for solving the weighted parsing problem is a pipeline with two phases. In the first phase, the CFG G, a deduction system I (Shieber et al., 1995), and the syntactic object a (i.e., a string) are combined into a single CFG G' (using a construction idea of Bar-Hillel et al., 1961). In the second phase, the value $\Bbbk$ (from above) is calculated, if G' is acyclic.[1]

Nederhof (2003) developed a similar framework, called *weighted deductive parsing*. As weight algebras he employed algebras of the form $(\mathbb{K}, \min, \mathbb{0}, \Omega, \sum^{\min})$ where $\mathbb{K}$ is a totally ordered set, $\sum^{\min} = \inf$ (infimum), $\inf(\mathbb{K}) \in \mathbb{K}$, and Ω is a set of superior functions; a superior function f is an operation on $\mathbb{K}$ which is monotone non-decreasing in each argument and $f(\Bbbk_1, \ldots, \Bbbk_m) \geq \max(\Bbbk_1, \ldots, \Bbbk_m)$ holds. The algorithm which he proposed for solving the weighted parsing problem is again a pipeline with two phases, where the first phase is the same as in the framework of Goodman (1999) and the resulting CFG G' is denoted by $c(G, a)$. In the second phase, he employed the algorithm of Knuth (1977), which generalizes the shortest distance algorithm of Dijkstra (1959) from graphs to hypergraphs and also works if G' is cyclic. If the CFG G' is non-branching, i.e., a linear grammar (Khabbaz, 1974, Def. 1), then in the second phase a graph algorithm can

[1] Goodman (1999) actually defines the algorithm so that it attempts to compute an infinite sum. He states that in applications, this computation needs to be replaced by instructions specific to the used semiring.

Proceedings of the 14th International Conference on Finite-State Methods and Natural Language Processing, pages 46–55
Dresden, Germany, September 23-25, 2019.

be used as an alternative to Knuth's algorithm; e.g., the single source shortest distance algorithm of Mohri (2002) if the weight algebra $\mathbb{K}$ is a complete semiring which is *closed for* G'.

In this paper, we generalize the two-phase pipeline approach of Goodman (1999) and Nederhof (2003) as follows. We specify the LM by using the general approach of initial algebra semantics (Goguen et al., 1977). For this, we employ weighted regular tree grammars (wRTG) and evaluate the generated trees (by the unique homomorphism) in some language algebra $\mathcal{L}$, which provides the set of syntactic objects as carrier set. This approach is very flexible and covers LMs for strings (CFG, LCFRS), but also trees and graphs (Drewes et al., 2016). Our second generalization concerns the weight algebra. We consider complete multioperator monoids (Kuich, 1999) which are algebras of the form $(\mathbb{K}, \oplus, \mathbb{0}, \Omega, \sum^{\oplus})$, where Ω is a set of operations on $\mathbb{K}$ and $\sum^{\oplus}$ is the infinitary sum operation which extends $\oplus$. We call the combination of such an LM and weight algebra *weighted RTG-based language model* (wRTG-LM). These combinations are very general and even exceed the scope of parsing; e.g., each algebraic dynamic programming problem (Giegerich et al., 2004), like minimum edit distance or matrix chain multiplication, can be formalized within this framework.

For solving the weighted parsing problem, given a wRTG-LM and a syntactic object a, we formalize the first phase as *canonical weighted deduction system*, which uses a CYK-like deduction system. For the second phase (*value computation algorithm*), we propose a generalization of Mohri's approach to hypergraphs, in the spirit of Knuth's generalization of Dijkstra's algorithm. We prove (in sketches) that our weighted parsing algorithm is terminating and solves the weighted parsing problem for every *closed wRTG-LM* with a *finitely decomposing* language algebra. This covers the approaches of Goodman (1999) and Nederhof (2003); our value computation algorithm subsumes the algorithms of Knuth (1977) and Mohri (2002). Due to space restrictions, we cannot show our detailed proofs of the theorems in this paper.

2 Preliminaries

Mathematical notions. We let $\mathbb{N} = \{0, 1, 2, \ldots\}$ be the set of natural numbers and $[m] = \{1, \ldots, m\}$ for each $m \in \mathbb{N}$. An *alphabet* is a finite, nonempty set. The powerset of a set A is denoted by $\mathcal{P}(A)$. Let $f\colon A \to B$ be a mapping; we extend it to the mapping $f'\colon \mathcal{P}(A) \to \mathcal{P}(B)$ by letting $f(U) = \{f(a) \mid a \in U\}$, and we denote f' also by f. A *family over* A is a mapping $f\colon I \to A$, where I is a countable set (index set). As usual, we represent each family f over A by $(f(i) \mid i \in I)$ and abbreviate $f(i)$ by f_i.

Ranked sets, trees, and regular tree grammars. A *ranked set* is a set Γ such that each $\gamma \in \Gamma$ is associated with a natural number $\mathrm{rk}_\Gamma(\gamma)$, its *rank*. The set of all elements of Γ with rank $m \in \mathbb{N}$ is denoted by Γ_m. A ranked set Σ with $\Sigma \subseteq \Gamma$ is *rank preserving (in* Γ*)* if $\Sigma_m \subseteq \Gamma_m$ for each $m \in \mathbb{N}$. Let H be a set. The set of *trees over* Γ *and* H is defined in the usual way, where elements of H may only occur at leaves. We denote this set by $\mathrm{T}_\Gamma(H)$ and abbreviate $\mathrm{T}_\Gamma(\emptyset)$ by T_Γ. Let $t \in \mathrm{T}_\Gamma(H)$. A *path in* t is a sequence of positions of d from the root to a leaf. Let p be a path in t. The sequence of labels of d along p is denoted by $\mathrm{seq}(d, p)$. A *ranked alphabet* is a ranked set which is an alphabet.

A *regular tree grammar* (RTG) (Brainerd, 1969) is a tuple $G = (N, \Sigma, A_0, R)$ where N is an alphabet (nonterminals), Σ is a ranked alphabet (terminals) with $N \cap \Sigma = \emptyset$, $A_0 \in N$ (initial nonterminal), and R is finite set of rules where each rule has the form $A \to \sigma(A_1, \ldots, A_m)$ with $m \in \mathbb{N}$, $A, A_1, \ldots, A_m \in N$, and $\sigma \in \Sigma_m$. Each RTG G can be considered as a context-free grammar G' (with terminal alphabet $\Sigma \cup \{(,), \mathrm{comma}\}$), which generates well-formed expressions. Thus the derivation relation $\Rightarrow_G$ is the usual derivation relation of G'. The *tree language generated by* G is the set $L(G) = \{t \in \mathrm{T}_\Sigma \mid A_0 \Rightarrow_G^* t\}$.

By viewing each rule $A \to \sigma(A_1, \ldots, A_m)$ of R as symbol with rank m, we can define the set $\mathrm{AST}(G)$ of *abstract syntax trees of* G to be the set of all $d \in \mathrm{T}_R$ such that for each position w of d the following holds: if d has label $A \to \sigma(A_1, \ldots, A_m)$ at w, then the i-th successor of w ($i \in [m]$) is labeled by a rule with left-hand side A_i (cf. Fig. 2). We define the mapping $\pi_\Sigma\colon \mathrm{AST}(G) \to \mathrm{T}_\Sigma$ such that $\pi_\Sigma(d)$ is obtained from d by replacing each label $A \to \sigma(A_1, \ldots, A_m)$ by σ (cf. Fig. 2). Hence $\pi_\Sigma(\mathrm{AST}(G)) = L(G)$.

Γ-algebras. Let Γ be a ranked set. A *Γ-algebra* (or: algebra) is a pair $(\mathcal{A}, \phi)$ where $\mathcal{A}$ is a set (*carrier set*) and ϕ is a mapping (*interpretation map-*

ping) which maps each $\gamma \in \Gamma_m$ ($m \in \mathbb{N}$) to an m-ary operation $\phi(\gamma)$ over $\mathcal{A}$, i.e., $\phi(\gamma)\colon \mathcal{A}^m \to \mathcal{A}$. In the sequel, we will sometimes identify $\phi(\gamma)$ and γ (as it is usual in algebra).

The *Γ-term algebra* is the Γ-algebra $(\mathrm{T}_\Gamma, \phi_\Gamma)$ where $\phi_\Gamma(\gamma)(t_1, \ldots, t_m) = \gamma(t_1, \ldots, t_m)$ for every $m \in \mathbb{N}$, $\gamma \in \Gamma_m$, and $t_1, \ldots, t_m \in \mathrm{T}_\Gamma$. For each Γ-algebra $(\mathcal{A}, \phi)$ there is exactly one homomorphism, denoted by $(.)_{\mathcal{A}}$, from the Γ-term algebra to $(\mathcal{A}, \phi)$ (Wechler, 1992). We write its application to an argument $t \in \mathrm{T}_\Gamma$ as $t_{\mathcal{A}}$. Intuitively, $(.)_{\mathcal{A}}$ evaluates a tree t in $(\mathcal{A}, \phi)$, in the same way as arithmetic expressions (e.g., $3 + 2 \cdot (4 + 5)$) are evaluated in the $\{+, \cdot\}$-algebra $(\mathbb{Z}, +, \cdot)$ to some values (here: 21). Often we abbreviate an algebra $(\mathcal{A}, \phi)$ by its carrier set $\mathcal{A}$. For every $a \in \mathcal{A}$ we let $\mathrm{factors}(a) = \{b \in \mathcal{A} \mid b <_{\mathrm{factor}}{}^* a\}$, where for every $a, b \in \mathcal{A}$, $b <_{\mathrm{factor}} a$ if there is a $\gamma \in \Gamma$ such that b occurs in some tuple $(b_1, \ldots, b_m)$ with $\phi(\gamma)(b_1, \ldots, b_m) = a$. We call $(\mathcal{A}, \phi)$ *finitely decomposable* if $\mathrm{factors}(a)$ is finite for every $a \in \mathcal{A}$.

Monoids. A *monoid* is an algebra $(\mathbb{K}, \oplus, \mathbb{0})$ such that $\oplus$ is a binary, associative operation on $\mathbb{K}$ and $\mathbb{0} \oplus \Bbbk = \Bbbk = \Bbbk \oplus \mathbb{0}$ for each $\Bbbk \in \mathbb{K}$. In the rest of this paper, each occurrence of $\Bbbk, \Bbbk_1, \Bbbk_2, \ldots$ is assumed to be universally quantified over $\mathbb{K}$ if not specified otherwise. The monoid is *commutative* if $\oplus$ is commutative; it is *extremal* (Mahr, 1984) if $\Bbbk_1 \oplus \Bbbk_2 \in \{\Bbbk_1, \Bbbk_2\}$; it is *idempotent* if $\Bbbk \oplus \Bbbk = \Bbbk$. It is *naturally ordered* if the binary relation $\preceq \subseteq \mathbb{K} \times \mathbb{K}$ (defined by $\Bbbk_1 \preceq \Bbbk_2$ if there is a $\Bbbk \in \mathbb{K}$ such that $\Bbbk_1 \oplus \Bbbk = \Bbbk_2$) is anti-symmetric (in which case it is a partial order, since reflexivity and transitivity hold by definition). It is *complete* if for each countable set I, there is an operation $\sum_I^\oplus$ which maps each family $(\Bbbk_i \mid i \in I)$ to an element of $\mathbb{K}$, coincides with $\oplus$ when I is finite, and otherwise satisfies axioms which guarantee commutativity and associativity (Eilenberg, 1974, p. 124). We abbreviate $\sum_I^\oplus (\Bbbk_i \mid i \in I)$ by $\sum_{i \in I}^\oplus \Bbbk_i$. A complete monoid is *d-complete* (Karner, 1992) if for every $\Bbbk \in \mathbb{K}$ and family $(\Bbbk_i \mid i \in \mathbb{N})$ of elements of $\mathbb{K}$ the following holds: if there is an $n_0 \in \mathbb{N}$ such that for every $n \in \mathbb{N}$ with $n \geq n_0$, $\sum_{i \in \mathbb{N}: i \leq n}^\oplus \Bbbk_i = \Bbbk$, then $\sum_{i \in \mathbb{N}}^\oplus \Bbbk_i = \Bbbk$. A complete monoid is *completely idempotent* if for every $\Bbbk \in \mathbb{K}$ and countable set I it holds that $\sum_{i \in I}^\oplus \Bbbk = \Bbbk$.

By easy calculations we obtain the following implications: (1) if $\mathbb{K}$ is extremal, then it is idempotent, (2) if $\mathbb{K}$ is completely idempotent, then it is d-complete, and (3) if $\mathbb{K}$ is d-complete, then it is naturally ordered.

Multioperator monoids. A *multioperator monoid* (M-monoid) (Kuich, 1999) is an algebra $(\mathbb{K}, \oplus, \mathbb{0}, \Omega)$ such that $(\mathbb{K}, \oplus, \mathbb{0})$ is a commutative monoid and Ω is a set of operations on $\mathbb{K}$ which contains at least the unary identity $\mathrm{id}\colon \mathbb{K} \to \mathbb{K}$. We view Ω as a ranked set, and hence $(\mathbb{K}, \phi)$ as an Ω-algebra where $\phi(\omega) = \omega$ for each $\omega \in \Omega$. Thus $t_{\mathbb{K}} \in \mathbb{K}$ is the evaluation of $t \in \mathrm{T}_\Omega$ in the algebra $(\mathbb{K}, \phi)$. An M-monoid inherits the properties of its monoid (e.g., being complete). We denote a complete M-monoid by $(\mathbb{K}, \oplus, \mathbb{0}, \Omega, \sum^\oplus)$.

An M-monoid is *distributive* if for each m-ary $\omega \in \Omega$ and every $i \in [m]$,

$$\begin{aligned}&\omega(\Bbbk_{1,i-1}, \Bbbk_i \oplus \Bbbk, \Bbbk_{i+1,m})\\ &= \omega(\Bbbk_{1,i-1}, \Bbbk_i, \Bbbk_{i+1,m}) \oplus \omega(\Bbbk_{1,i-1}, \Bbbk, \Bbbk_{i+1,m})\end{aligned}$$

where $\Bbbk_{1,i-1}$ and $\Bbbk_{i+1,m}$ abbreviate $\Bbbk_1, \ldots, \Bbbk_{i-1}$ and $\Bbbk_{i+1}, \ldots, \Bbbk_m$, respectively. If $\mathbb{K}$ is complete, then we additionally require that the above equation also holds for each countable set of summands.

Next we show examples of M-monoids.

- Each semiring $(\mathbb{K}, \oplus, \otimes, \mathbb{0}, \mathbb{1})$ can be considered as the M-monoid $(\mathbb{K}, \oplus, \mathbb{0}, \Omega_\otimes)$ (Fülöp et al., 2009) where $\Omega_\otimes = \{\mathrm{mul}_\Bbbk^{(m)} \mid m \in \mathbb{N}, \Bbbk \in \mathbb{K}\}$ and for every $m \in \mathbb{N}$ we define
$$\mathrm{mul}_\Bbbk^{(m)}(\Bbbk_1, \ldots, \Bbbk_m) = \Bbbk \otimes \Bbbk_1 \otimes \cdots \otimes \Bbbk_m \ .$$
Note that $\mathbb{1} = \mathrm{mul}_{\mathbb{1}}^{(0)}()$.
- Knuth (1977) uses complete, distributive M-monoids of the form $(\mathbb{K}, \min, \mathbb{0}, \Omega, \sum^{\min})$ where $\mathbb{K}$ is a totally ordered set, $\inf(\mathbb{K}) \in \mathbb{K}$, and the operations in Ω are superior functions. We will call such M-monoids *superior M-monoids*. We note that each superior M-monoid is d-complete.

3 Weighted RTG-based language models and the weighted parsing problem

As framework for the definition of our language models we use the initial algebra approach (Goguen et al., 1977). An *RTG-based language model* (RTG-LM) is a tuple $(G, (\mathcal{L}, \phi))$ where

- $G = (N, \Sigma, A_0, R)$ is an RTG and
- $(\mathcal{L}, \phi)$ is a Γ-algebra (*language algebra*) such that $\Sigma \subseteq \Gamma$ is rank preserving; the elements of $\mathcal{L}$ are called *syntactic objects*.

The *language generated by* $(G, (\mathcal{L}, \phi))$ is the set

$$L(G)_{\mathcal{L}} = \{t_{\mathcal{L}} \mid t \in L(G)\} \subseteq \mathcal{L} \ ,$$

i.e., the set of all syntactic objects which result

r_1: S $\xrightarrow{1.0}$ $\langle x_1 x_2\rangle$(NP, VP)
r_2: NP $\xrightarrow{0.2}$ $\langle x_1\rangle$(NN)
r_3: NP $\xrightarrow{0.5}$ $\langle x_1 x_2\rangle$(NN, NNS)
r_4: NP $\xrightarrow{0.3}$ $\langle x_1\rangle$(NNS)
r_5: VP $\xrightarrow{0.4}$ $\langle x_1 x_2\rangle$(VBZ, PP)
r_6: VP $\xrightarrow{0.6}$ $\langle x_1 x_2\rangle$(VBP, NP)
r_7: PP $\xrightarrow{1.0}$ $\langle x_1 x_2\rangle$(IN, NP)
r_8: NN $\xrightarrow{1.0}$ $\langle fruit\rangle$
r_9: NNS $\xrightarrow{0.4}$ $\langle flies\rangle$
r_{10}: NNS $\xrightarrow{0.6}$ $\langle bananas\rangle$
r_{11}: VBZ $\xrightarrow{1.0}$ $\langle flies\rangle$
r_{12}: VBP $\xrightarrow{1.0}$ $\langle like\rangle$
r_{13}: IN $\xrightarrow{1.0}$ $\langle like\rangle$

Figure 1: Rules of RTG of Ex. 1.

from evaluating trees of $L(G)$ in the language algebra $\mathcal{L}$. For each $a \in \mathcal{L}$, we let

$$\mathrm{AST}(G, a) = \{d \in \mathrm{AST}(G) \mid \pi_\Sigma(d)_\mathcal{L} = a\} \ .$$

Example 1. We consider the Γ-algebra $\mathcal{CFG}^\Delta = (\Delta^*, \phi)$ as language algebra where $\Delta = \{fruit, flies, like, bananas\}$, $\Gamma = \bigcup_{m\in\mathbb{N}} \Gamma_m$, and $\Gamma_m = \{\langle u_0 x_1 u_1 \cdots x_m u_m\rangle \mid u_i \in \Delta^*\}$. We define

$$\phi(\langle u_0 x_1 u_1 \cdots x_m u_m\rangle)(a_1, \ldots, a_m) = u_0 a_1 u_1 \cdots a_m u_m$$

for every $a_1, \ldots, a_m \in \Delta^*$.

We consider the RTG $G = (N, \Sigma, \mathrm{S}, R)$ with $N = \{\mathrm{S}, \mathrm{NP}, \mathrm{VP}, \mathrm{PP}, \mathrm{NN}, \mathrm{NNS}, \mathrm{VBZ}, \mathrm{VBP}, \mathrm{IN}\}$ and $\Sigma = \{\langle\delta\rangle \mid \delta \in \Delta\} \cup \{\langle x_1\rangle, \langle x_1 x_2\rangle\} \subseteq \Gamma$, and R contains the rules shown in Fig. 1 (ignoring the numbers above the arrows for the time being).

The tree in the middle of the upper row of Fig. 2 is an abstract syntax tree $d \in \mathrm{AST}(G)$. It expresses that certain insects (*fruit flies*) like something (*bananas*). We obtain $\pi_\Sigma(d)$ by dropping the non-highlighted parts of d (left of upper row). The application of the homomorphism $(.)_{\mathcal{CFG}^\Delta}: \mathrm{T}_\Sigma \to \mathcal{CFG}^\Delta$ to $\pi_\Sigma(d)$ yields the string $a = fruit\ flies\ like\ bananas$. We note that there is another abstract syntax tree $d' \in \mathrm{AST}(G)$, viz., $d' = r_1(r_2(r_8), r_5(r_{11}, r_7(r_{13}, r_4(r_{10}))))$ such that $\pi_\Sigma(d')_{\mathcal{CFG}^\Delta} = a$. It expresses how *fruit* performs a certain activity (to fly like bananas). Hence this RTG-LM is ambiguous. □

It should be clear from Ex. 1 that each context-free grammar with terminal alphabet Δ can be represented as an RTG-LM $(G, \mathcal{CFG}^\Delta)$, and vice versa, each RTG-LM $(G, \mathcal{CFG}^\Delta)$ represents a CFG. In the same way, one can characterize LCFRS and tree adjoining grammars by (1) superposing sorts to the set N of nonterminals of the RTG (in order to represent fanout and the characteristic "substitution tree / adjoining tree" of arguments, respectively), and (2) by defining appropriate Γ-algebras $\mathcal{LCFRS}^\Delta$ (Kallmeyer, 2010, Def. 6.2+6.3) and $\mathcal{TAG}^\Delta$ (Büchse et al., 2012; Koller and Kuhlmann, 2012), respectively. The language algebras $\mathcal{CFG}^\Delta$, $\mathcal{LCFRS}^\Delta$, and $\mathcal{TAG}^\Delta$ are finitely decomposable.

A *weighted RTG-based language model* (wRTG-LM) is a tuple

$$(\ (G, (\mathcal{L}, \phi)),\ \ (\mathbb{K}, \oplus, \mathbb{0}, \Omega, \textstyle\sum^\oplus),\ \ \mathrm{wt}\ )\ ,$$

where

- $(G, (\mathcal{L}, \phi))$ is an RTG-LM,
- $(\mathbb{K}, \oplus, \mathbb{0}, \Omega, \sum^\oplus)$ is a complete M-monoid (*weight algebra*), and
- wt maps each rule of G with rank m to an m-ary operation in Ω. We lift wt to the mapping $\mathrm{wt}': \mathrm{T}_R \to \mathrm{T}_\Omega$ and denote wt′ also by wt.

Definition 2. The *weighted parsing problem* is the following problem: given a wRTG-LM $((G, (\mathcal{L}, \phi)), (\mathbb{K}, \oplus, \mathbb{0}, \Omega, \sum^\oplus), \mathrm{wt})$ and an $a \in \mathcal{L}$, compute the value $\mathrm{parse}(a) \in \mathbb{K}$ where

$$\mathrm{parse}(a) = \sum_{d\in\mathrm{AST}(G,a)}^{\oplus} \mathrm{wt}(d)_\mathbb{K} \ . \qquad \square$$

Example 3. (Ex. 1 cont.) The best derivation problem of (Goodman, 1999) consists of computing, given a syntactic object a and a grammar, the abstract syntax trees of a with maximal probability (and this probability). Let R_∞ be a ranked set such that $(R_\infty)_m$ is infinite for each $m \in \mathbb{N}$. In analogy to Goodman, we define the *best derivation M-monoid* to be the d-complete M-monoid

$$\mathbb{BD} = (V,\ \max_{\mathbb{BD}},\ (0, \emptyset),\ \Omega_{\mathbb{BD}},\ \textstyle\sum^{\max_{\mathbb{BD}}}),$$

where $V = [0, 1] \times \mathcal{P}(\mathrm{T}_{R_\infty})$ and $[0, 1]$ is the interval of real numbers from 0 to 1 and

- for every $(p_1, D_1), (p_2, D_2) \in V$, the value $\max_{\mathbb{BD}}((p_1, D_1), (p_2, D_2))$ is (p_i, D_i) if $p_i > p_j$ for $i, j \in \{1, 2\}$, and $(p_1, D_1 \cup D_2)$ if $p_1 = p_2$,
- $\Omega_{\mathbb{BD}} = \{\mathrm{tc}_{p,r} \mid p \in [0, 1]$ and $r \in R_\infty\}$, where for each $p \in [0, 1]$ and $r \in R_\infty$ of rank m, we define $\mathrm{tc}_{p,r}: V^m \to V$ (tc abbreviates top concatenation) such that for every $(p_1, D_1), \ldots, (p_m, D_m) \in V$

  $$\mathrm{tc}_{p,r}((p_1, D_1), \ldots, (p_m, D_m)) = (p', D')$$

  where $p' = p \cdot p_1 \cdot \ldots \cdot p_m$ and $D' = \{r(d_1, \ldots, d_m) \mid d_i \in D_i, 1 \le i \le m\}$, and
- for every family $((p_i, D_i) \mid i \in I)$ over V, we define $\sum^{\max_{\mathbb{BD}}}_{i\in I}(p_i, D_i) = (p, D)$, where $p = \sup\{p_i \mid i \in I\}$ and $D = \bigcup_{i\in I: p_i = p} D_i$.

Since $\mathbb{BD}$ is completely idempotent, it is also d-complete.

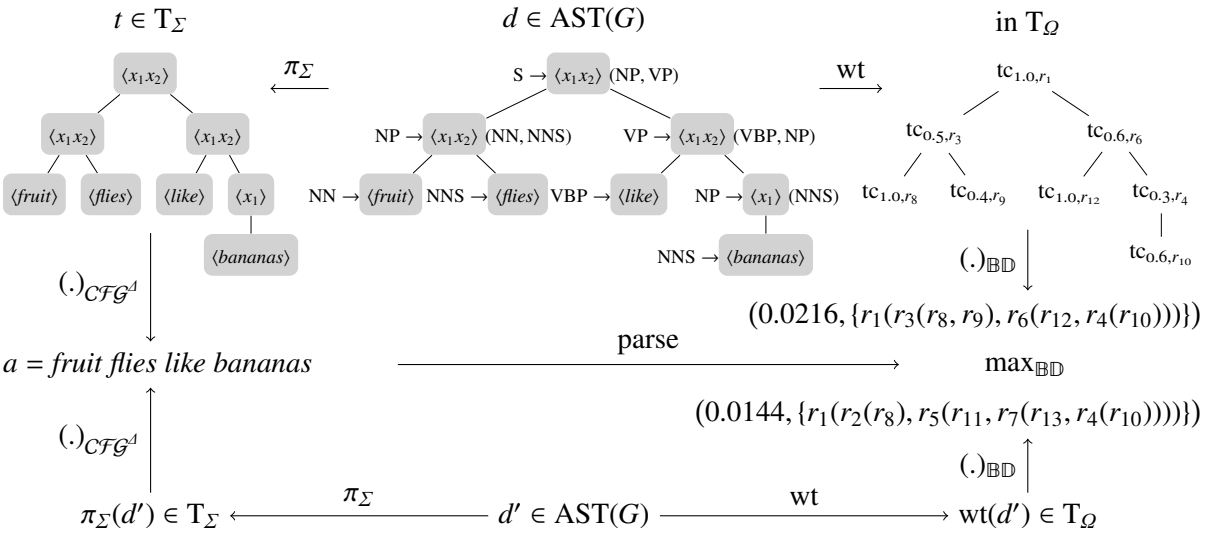

Figure 2: Illustration of the weighted parsing problem for the wRTG-LM $((G, \mathcal{CFG}^\Delta), \mathbb{BD}, \mathrm{wt})$ and the syntactic object a = *fruit flies like bananas* of Δ^*, see Ex. 3.

Now we consider the finite set R of rules of the RTG G given in Ex. 1. We can assume that $R \subseteq R_\infty$ is rank preserving. We define the mapping $\mathrm{wt}\colon R \to \Omega_{\mathbb{BD}}$ by $\mathrm{wt}(r_i) = \mathrm{tc}_{p_i, r_i}$ where p_i is shown in Fig. 1 above the arrow of r_i. For each $d \in \mathrm{AST}(G, a)$, the second component of $\mathrm{wt}(d)_{\mathbb{BD}}$ has exactly one element. Recall d' from Ex. 1, a second AST which is evaluated to a. We obtain $\mathrm{wt}(d')_{\mathbb{BD}} = (0.0144, \{r_1(r_2(r_8), r_5(r_{11}, r_7(r_{13}, r_4(r_{10}))))\})$. Thus

$$\max_{\mathbb{BD}}(\mathrm{wt}(d)_{\mathbb{BD}}, \mathrm{wt}(d')_{\mathbb{BD}}) = \mathrm{wt}(d)_{\mathbb{BD}} \ .$$

As one might expect, it is more likely that a refers to the preferences (to *like bananas*) of certain insects (*fruit flies*). Fig. 2 illustrates the parsing problem for the wRTG-LM $((G, \mathcal{CFG}^\Delta), \mathbb{BD}, \mathrm{wt})$ and a = *fruit flies like bananas*. □

In summary, each wRTG-LM consists of two components: a *syntax component* and a *weight component*. The syntax component (cf. the left of Fig. 2) contains the language algebra $(\mathcal{L}, \phi)$. This is a Γ-algebra whose carrier set is the set of syntactic objects. The mapping π_Σ maps each abstract syntax tree to a tree in the Σ-term algebra T_Σ, which is then evaluated to a syntactic object by the unique homomorphism $(.)_\mathcal{L}$ (recall that $\Sigma \subseteq \Gamma$).

The weight component (cf. the right of Fig. 2) contains a complete M-monoid $(\mathbb{K}, \oplus, \mathbb{0}, \Omega, \sum^\oplus)$ whose carrier set is the set of weights. The mapping wt maps each abstract syntax tree to a tree in the Ω-term algebra T_Ω, which is then evaluated to a weight in $\mathbb{K}$ by the unique homomorphism $(.)_\mathbb{K}$. Weights in $\mathbb{K}$ are accumulated using $\oplus$.

$\mathrm{A} \to \mathrm{del}_a(\mathrm{A})$	$\phi(\mathrm{del}_a)(w) = aw$	$\mathrm{del}_a(n) = n + 1$
$\mathrm{A} \to \mathrm{ins}_a(\mathrm{A})$	$\phi(\mathrm{ins}_a)(w) = wa$	$\mathrm{ins}_a(n) = n + 1$
$\mathrm{A} \to \mathrm{rep}_{a,b}(\mathrm{A})$	$\phi(\mathrm{rep}_{a,b})(w) = awb$	$\mathrm{rep}_{a,b}(n) = n'$
$\mathrm{A} \to \mathrm{nil}$	$\phi(\mathrm{nil})() = \$$	$\mathrm{nil}() = 0$

Figure 3: Rules of G for each $a, b \in \Delta$, the interpretation ϕ, and the operations in Ω where $n' = n + 1$ if $a \neq b$, and n otherwise.

The weighted parsing problem takes as input a wRTG-LM and a syntactic object a, and it computes the $\oplus$-accumulation of the weights of each AST of a.

Example 4. Giegerich et al. (2004) formalized dynamic programming (Bellman, 1952, 1954) in an algebraic setting, called *algebraic dynamic programming* (ADP). We claim that each ADP problem is a weighted parsing problem. To support this statement, we consider the computation of the minimum edit distance (med) between two words over some alphabet Δ by deletion, insertion, and replacement, and we "simulate" its ADP-specification as wRTG-LM $((G, (\mathcal{L}, \phi)), \mathbb{K}, \mathrm{wt})$. The rules of the RTG G and the interpretation ϕ are shown in the first and second columns of Fig. 3, respectively. Thus, for each tree $t \in L(G)$, $t_\mathcal{L} = u\$v$ for some $u, v \in \Delta^*$. We choose the complete, distributive M-monoid $(\mathbb{K}, \oplus, \emptyset, \Omega, \sum^\oplus)$ with $\mathbb{K} = \{h(F) \mid F \in \mathcal{P}(\mathbb{N})\}$ for the single-valued objective function $h\colon \mathcal{P}(\mathbb{N}) \to \mathcal{P}(\mathbb{N})$ with $h(F) = \{\min(F)\}$. We let $F_1 \oplus F_2 = h(F_1 \cup F_2)$ for every $F_1, F_2 \in \mathbb{K}$, and $\sum^\oplus_{i \in \mathbb{N}} F_i = \{\inf(\bigcup_{i \in \mathbb{N}} F_i)\}$. The set Ω is shown in the third column of Fig. 3.

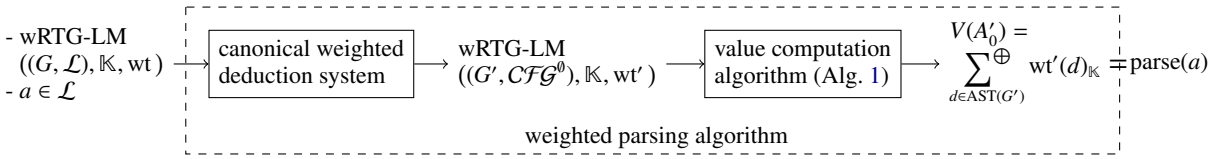

Figure 4: Two-phase pipeline for solving the weighted parsing problem (A'_0 is the initial nonterminal of G').

Note that h satisfies Bellman's principle of optimality: $h(\omega(F)) = h(\omega(h(F)))$ for each unary $\omega \in \Omega$ and $F \in \mathbb{K}$. Then $\mathrm{med}(u, v) = \mathrm{parse}(u\$v^{-1})$ for every $u, v \in \Delta^*$, where v^{-1} is the reversal of v.

This construction can be generalized to a procedure which turns every specification of an ADP problem into a weighted parsing problem. Due to space restrictions, we cannot present this procedure in its entirety. □

4 The weighted parsing algorithm

The weighted parsing algorithm is supposed to solve the weighted parsing problem. As input, it takes a wRTG-LM $\mathcal{G}$ and a syntactic object a. Its output is intended to be $\mathrm{parse}(a)$. The algorithm is a pipeline with two phases (cf. Fig. 4) and follows the modular approach of Nederhof (2003). First, a *canonical weighted deduction system* computes from $\mathcal{G}$ and a a new wRTG-LM $\mathcal{G}'$ with the same weight structure as $\mathcal{G}$, but a different RTG and the language algebra $\mathcal{CFG}^{\emptyset}$. Second, $\mathcal{G}'$ is the input to the *value computation algorithm* (Alg. 1), which computes the value $V(A'_0)$; this is supposed to be $\sum^{\oplus}_{d \in \mathrm{AST}(G')} \mathrm{wt}(d)_{\mathbb{K}} = \mathrm{parse}(a)$.

Weighted deduction systems. Parsing of some string w with some grammar G can be formalized as a deduction system $\mathcal{D}$ (Shieber et al., 1995). $\mathcal{D}$ consists of a set of inference rules

$$\frac{I_1 \ \ldots \ I_m}{I} \ \{c_1, \ldots, c_p\}$$

where $m \in \mathbb{N}$, $I, I_1, \ldots, I_m$ are *items*, and $c_1, \ldots, c_p$ are side conditions. Each item represents a Boolean-valued property (of some combination of nonterminals of G and/or substrings of $a = w$). The meaning of an inference rule is: given that $I_1, \ldots, I_m$ and $c_1, \ldots, c_p$ are true, I is true as well. Nederhof (2003) pointed out that "a deduction system having a grammar G [...] and input string w in the side conditions can be seen as a construction c of a context-free grammar $c(G, w)$ [...]"; also, he extended $\mathcal{D}$ and $c(G, a)$ with weights.

Inspired by this, we define the *canonical weighted deduction system* as a mapping cwds which takes two arguments: (a) a wRTG-LM $\mathcal{G} = ((G, \mathcal{L}), \mathbb{K}, \mathrm{wt})$ such that the language algebra $(\mathcal{L}, \phi)$ is finitely decomposable and (b) a syntactic object $a \in \mathcal{L}$. Let $G = (N, \Sigma, A_0, R)$. Then we define

$$\mathrm{cwds}\,(\mathcal{G}, a) = ((G', \mathcal{CFG}^{\emptyset}), \mathbb{K}, \mathrm{wt}')\ ,$$

where $G' = (N', \Sigma', A'_0, R')$ and

- $N' = \{(A_0, a)\} \cup (N \times \Sigma \times \mathrm{factors}(a))$; N' is finite, because $\mathcal{L}$ is finitely decomposable,
- $\Sigma' = \{\langle x_1 \ldots x_m \rangle \mid \text{a rule with rank } m \text{ is in } R\}$,
- $A'_0 = (A_0, a)$, and
- for each $\sigma \in \Sigma$, the rule $r' = (A_0, a) \to (A_0, \sigma, a)$ is in R' and $\mathrm{wt}'(r') = \mathrm{id}$; for each $r = (A \to \sigma(A_1, \ldots, A_m))$ in R and $a_0, a_1, \ldots, a_m \in \mathrm{factors}(a)$ with $\phi(\sigma)(a_1, \ldots, a_m) = a_0$ and every rule $A_i \to \sigma_i(\ldots)$ $(i \in [m])$ in R, the rule r'

  $$(A, \sigma, a_0) \to \langle x_1 \ldots x_m \rangle((A_1, \sigma_1, a_1), \ldots, (A_m, \sigma_m, a_m))$$

  is in R' and we let $\mathrm{wt}'(r') = \mathrm{wt}(r)$.

Note that cwds implements a CYK-like deduction system. The elements of N' have a very general form. Depending on $\mathcal{L}$, they can be understood as, e.g., spans of strings, occurrences of patterns in trees, or occurrences of subgraphs in graphs. We note that for every $d \in \mathrm{AST}(G')$ it holds that $\pi_{\Sigma}(d)_{\mathcal{CFG}^{\emptyset}} = \varepsilon$, i.e., each abstract syntax tree is evaluated to the empty string. Moreover, cwds is *weight-preserving* in the following sense:

(1) there is a bijective mapping ψ from the set $\mathrm{AST}(G, a)$ to $\mathrm{AST}(G')$ and
(2) for every $d \in \mathrm{AST}(G, a)$ we have that $\mathrm{wt}(d)_{\mathbb{K}} = \mathrm{wt}'(\psi(d))_{\mathbb{K}}$.

Value computation algorithm. This is Alg. 1. Its input is a wRTG-LM $\mathcal{G}'$ with language algebra $\mathcal{CFG}^{\emptyset}$. It maintains a mapping V, which assigns a weight to each nonterminal, and a Boolean variable *changed*. The output is the value $V(A'_0)$. The algorithm starts by assigning the weight $\mathbb{0}$ to each nonterminal (lines 1–2). Then, in a *repeat-until loop* (lines 3–12), the weight of each nonterminal is recomputed in every iteration of that loop as follows (where $\langle x_{1,m} \rangle$ abbreviates $\langle x_1, \ldots, x_m \rangle$):

$$V(A) = \bigoplus_{\substack{r \in R': \\ r = (A \to \langle x_{1,m} \rangle (A_1, \ldots, A_m))}} \mathrm{wt}'(r)(V(A_1), \ldots, V(A_m))\ .$$

Algorithm 1 Value computation algorithm

Input: $((G', \mathcal{CFG}^{\emptyset}),\ (\mathbb{K}, \oplus, \mathbb{0}, \mathbb{\Omega}, \sum^{\oplus}),\ \mathrm{wt}'\ )$ which is a wRTG-LM with $G' = (N', \Sigma', A'_0, R')$
Variables: $V\colon N' \to \mathbb{K}$, $V' \in \mathbb{K}$, *changed* $\in \mathbb{B}$
Output: $V(A'_0)$

1: **for each** $A \in N'$ **do**
2: $\quad V(A) \leftarrow \mathbb{0}$
3: **repeat**
4: $\quad$ *changed* $\leftarrow$ false
5: $\quad$ **for each** $A \in N'$ **do**
6: $\quad\quad V' \leftarrow \mathbb{0}$
7: $\quad\quad$ **for each** $r = (A \to \langle x_{1,m}\rangle(A_1, \ldots, A_m))$ in R' **do**
8: $\quad\quad\quad V' \leftarrow V' \oplus \mathrm{wt}'(r)(V(A_1), \ldots, V(A_m))$
9: $\quad\quad$ **if** $V(A) \neq V'$ **then**
10: $\quad\quad\quad$ *changed* $\leftarrow$ true
11: $\quad\quad V(A) \leftarrow V'$
12: **until** *changed* = false

The algorithm terminates after the first iteration in which no nonterminal has changed its weight.

We note that in practice, a complete computation of $\mathrm{cwds}(\mathcal{G}, a)$ prior to the execution of the value computation algorithm (Alg. 1) is impossible. Similar to Nederhof (2003), we execute the value computation algorithm on an incomplete input which is extended on demand (lazy evaluation). More precisely, $\mathcal{G}'$ is initialized so that it only contains the rules of rank 0 (and the nonterminals in their left-hand sides). Then, each time a value different from $\mathbb{0}$ is first assigned to a nonterminal A in line 11, we compute the following set of rules: each rule whose right-hand side only contains A and other nonterminals for which this computation has already been done is in that set. These new rules (and the nonterminals in their left-hand sides) are added to $\mathcal{G}'$.

5 Termination and correctness

We are interested in two formal properties of the value computation algorithm (Alg. 1) and of the weighted parsing algorithm (Fig. 4): termination and correctness.

The value computation algorithm computes the weights of the ASTs bottom-up and reuses the results of common subtrees (as in dynamic programming); this requires distributivity of the weight algebra. Moreover, solving the weighted parsing problem by a terminating algorithm involves the following difficulty: there may be infinitely many ASTs (due to cycles) which are evaluated to the same syntactic object a. Thus $\mathrm{parse}(a)$ is an infinite sum, which in general cannot be computed in finite time. Hence, a terminating algorithm can only solve the weighted parsing problem if the infinite sum is equal to the sum over some finite subset of the infinite sum's index set.

We have organized this section as follows. In Subsection 5.1 we define the class of *closed* wRTG-LMs (similar to Mohri, 2002) and prove that the value computation algorithm (Alg. 1) is terminating and correct for closed wRTG-LMs as input. We say that the value computation algorithm is *correct* if after termination

$$V(A'_0) = \sum_{d \in \mathrm{AST}(G')}^{\oplus} \mathrm{wt}'(d)_{\mathbb{K}} \ .$$

In Subsection 5.2 we prove that the weighted parsing algorithm (Fig. 4) is terminating and correct for two classes of inputs. We say that the weighted parsing algorithm is *correct* if it computes $\mathrm{parse}(a)$.

5.1 Properties of the value computation algorithm

Since each wRTG-LM has a finite set of rules, an infinite set of ASTs is only possible if the ASTs are cyclic in the following sense. Recall that R' is the set of rules of the input G' to the value computation algorithm (Alg. 1). Let $\rho \in (R')^*$. We call ρ *cyclic* if $|\rho| \geq 2$, $\rho_1 = \rho_{|\rho|}$, and for every $i, j \in \mathbb{N}$, if $1 \leq i < j < |\rho|$, then $\rho_i \neq \rho_j$. From now on, let $\rho \in (R')^*$ be cyclic, $d \in \mathrm{T}_{R'}$, and $c \in \mathbb{N}$. A path p in d is (c, ρ)-*cyclic* if ρ occurs exactly c times in $\mathrm{seq}(d, p)$. We define the set $\mathrm{cutout}(d, \rho)$ which contains every tree obtained from d by cutting out at least one occurrence of ρ. We illustrate cutout by an example in Fig. 5.

Definition 5. Let $c \in \mathbb{N}$. A wRTG-LM $\mathcal{G}' = ((G', \mathcal{CFG}^{\emptyset}), \mathbb{K}, \mathrm{wt}'\ )$ is *c-closed* if $\mathbb{K}$ is distributive and d-complete, and for each $d \in \mathrm{T}_{R'}$ and cyclic string $\rho \in (R')^*$ the following holds: if there is a (c, ρ)-cyclic path in d, then

$$\mathrm{wt}'(d)_{\mathbb{K}} \oplus \bigoplus_{d' \in \mathrm{cutout}(d,\rho)} \mathrm{wt}'(d')_{\mathbb{K}} = \bigoplus_{d' \in \mathrm{cutout}(d,\rho)} \mathrm{wt}'(d')_{\mathbb{K}} \ .$$

$\mathcal{G}'$ is *closed* if it is c-closed for some $c \in \mathbb{N}$. □

For every $c \in \mathbb{N}$ and ranked set R', we let $\mathrm{T}^{(c)}_{R'}$ be the set of all those $d \in \mathrm{T}_{R'}$ such that for every cyclic $\rho \in (R')^*$ and $c' > c$, no path in d is (ρ, c')-cyclic. In other words, $\mathrm{T}^{(c)}_{R'}$ contains all those trees of $\mathrm{T}_{R'}$ which have at most c occurrences of some cycle in some of their paths. Clearly $\mathrm{T}^{(c)}_{R'}$ is finite, $\mathrm{T}^{(c)}_{R'} \subseteq \mathrm{T}^{(c+1)}_{R'}$ for every $c \in \mathbb{N}$,

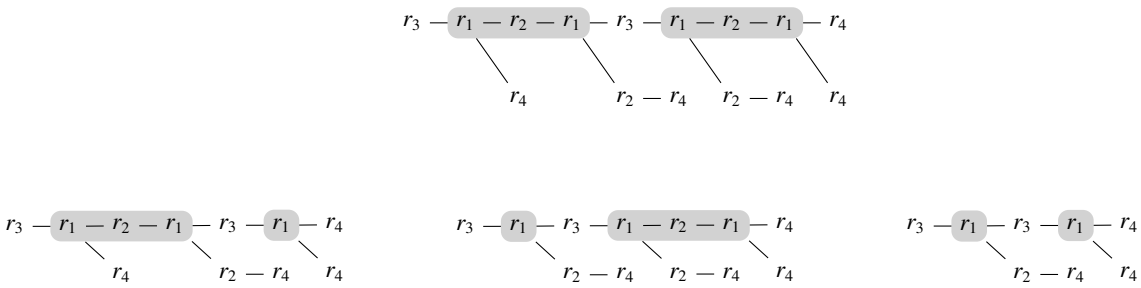

Figure 5: Top: tree d over the ranked set $R' = \{r_1^{(2)}, r_2^{(1)}, r_3^{(1)}, r_4^{(0)}\}$ with a $(2,\rho)$-cyclic path (horizontal line) for $\rho = r_1 r_2 r_1$. Bottom: the set cutout(d,ρ). Please do not confuse the elements of R' with the rules of Ex. 1 and 3.

and $\bigcup_{c\in\mathbb{N}} \mathrm{T}_{R'}^{(c)} = \mathrm{T}_{R'}$. Given a wRTG-LM $\mathcal{G}' = ((G', \mathcal{CFG}^{\emptyset}), \mathbb{K}, \mathrm{wt}')$ with set of rules R', we let $\mathrm{AST}(G')^{(c)} = \mathrm{T}_{R'}^{(c)} \cap \mathrm{AST}(G')$ for every $c \in \mathbb{N}$.

Theorem 6. For every $c \in \mathbb{N}$ and c-closed wRTG-LM $((G', \mathcal{CFG}^{\emptyset}), \mathbb{K}, \mathrm{wt}')$ the following holds:

$$\sum_{d\in\mathrm{AST}(G')}^{\oplus} \mathrm{wt}'(d)_{\mathbb{K}} = \bigoplus_{d\in\mathrm{AST}(G')^{(c)}} \mathrm{wt}'(d)_{\mathbb{K}} \ .$$

Proof (sketch). As $\mathbb{K}$ is distributive, we can show by induction on $n \in \mathbb{N}$ that for every $B \subseteq \mathrm{AST}(G') \setminus \mathrm{AST}(G')^{(c)}$ with $|B| = n$, adding B to the index set of $\oplus$ does not change the sum's value. Then, as $\mathbb{K}$ is d-complete, the equality holds. ∎

This theorem reflects the desired property: given that our wRTG-LM is c-closed (with $c \in \mathbb{N}$), each (possibly infinite) sum over all ASTs can be computed as a sum over the finite set $\mathrm{AST}(G')^{(c)}$.

Theorem 7. The value computation algorithm (Alg. 1) is terminating and correct for every closed wRTG-LM $\mathcal{G}'$ with language algebra $\mathcal{CFG}^{\emptyset}$.

Proof (sketch). Let $\mathcal{G}'$ be c-closed. We note that in line 8, the value in the right-hand side of $\oplus$ always corresponds to the sum over the weights of some trees in $(\mathrm{T}_{R'})_A$; this is due to the fact that $\mathbb{K}$ is distributive. By the form of recomputation in lines 3–12, each $d \in (\mathrm{T}_{R'})_A$ contributes to that sum at most once. Furthermore, V' only differs from $V(A)$ if a tree from the finite set $\mathrm{T}_{R'}^{(c)}$ has been used to compute V', but not $V(A)$ (this is a consequence of $\mathcal{G}'$ being closed). Thus, *changed* is only set to true finitely often and the algorithm eventually terminates. Then, after termination, $V(A'_0) = \bigoplus_{d\in\mathrm{AST}(G')^{(c)}} \mathrm{wt}'(d)_{\mathbb{K}}$ and Theorem 6 implies correctness. ∎

5.2 Properties of the weighted parsing algorithm

We discuss two classes of wRTG-LMs for which the weighted parsing algorithm (Fig. 4) is terminating and correct.

(1) Closed wRTG-LMs with arbitrary language algebras. Each of them is a wRTG-LM $((G, (\mathcal{L}, \phi)), (\mathbb{K}, \oplus, \mathbb{0}, \Omega, \sum^{\oplus}), \mathrm{wt})$ which is c-closed for some $c \in \mathbb{N}$, and *c-closed* is defined as in Def. 5. (We note that this generalization is possible because Def. 5 does not use any property of $\mathcal{CFG}^{\emptyset}$.) The following particular wRTG-LMs are closed:

- wRTG-LMs with acyclic RTG, where an RTG G is acyclic if $\mathrm{AST}(G) = \mathrm{AST}(G)^{(0)}$,
- wRTG-LMs with superior, d-complete M-monoids as weight algebras, and
- wRTG-LMs with weight algebra $\mathbb{BD}$ if no chain rule and ε-rule has probability 1.0 (as in Ex. 3).

(2) Non-looping wRTG-LMs with distributive M-monoids as weight algebras. A wRTG-LM $\mathcal{G}$ is *non-looping* if for every syntactic object a and tree d over the set of rules of $\mathcal{G}$ which is evaluated to a the following holds: no proper subtree of d is evaluated to a. ADP problems can be specified by non-looping wRTG-LMs, because the syntactic objects of ADP represent (sub-)problems which have to be solved. Thus, if $\mathcal{G}$ is looping, then the solution of a subproblem would depend on itself, which contradicts dynamic programming. In general, non-looping is not decidable, but it is for particular language algebras, e.g., $\mathcal{CFG}^{\Delta}$.

Lemma 8. For every closed or nonlooping wRTG-LM $\mathcal{G}$ with finitely decomposable language algebra and syntactic object a, the wRTG-LM $\mathrm{cwds}(\mathcal{G}, a)$ is closed.

Theorem 9. The weighted parsing algorithm (Fig. 4) is terminating and correct for every closed or nonlooping wRTG-LM with finitely decomposable language algebra.

Proof. The weighted parsing algorithm terminates because (a) the computation of cwds is terminating

algorithm	class of valid inputs	class C_1 of RTG	class C_2 of weight algebras
(a) Knuth (1977)	$C_1 \times C_2$	RTG	superior M-monoid
(b) Goodman (1999)	$C_1 \times C_2$	acyclic RTG	complete semiring
(c) Mohri (2002)	C_2 closed for C_1	monadic RTG	commutative, d-complete semiring
(d) Alg. 1	closed wRTG-LM	RTG	distributive, d-complete M-monoid

Table 1: Comparison of four value computation algorithms. The second column represents the class of wRTG-LMs to which the corresponding algorithm is applicable. The expression $C_1 \times C_2$ denotes the class of all wRTG-LMs with RTGs in C_1 and weight algebras in C_2.

for every wRTG-LM with finitely decomposable language algebra and (b) the value computation algorithm (Alg. 1) is terminating by Theorem 7, which we can be applied due to Lemma 8. The weighted parsing algorithm is correct because (a) cwds is weight-preserving and (b) the value computation algorithm is correct by Theorem 7 (which is applicable again due to Lemma 8), hence

$$\mathrm{parse}(a) \overset{\text{(a)}}{=} \sum_{d \in \mathrm{AST}(G')}^{\oplus} \mathrm{wt}'(d)_{\mathbb{K}} \overset{\text{(b)}}{=} V(A'_0) \ . \qquad \blacksquare$$

6 Comparison of value computation algorithms

Here we compare our value computation algorithm (Alg. 1) to the algorithm of Knuth (1977), the second phase of Goodman (1999), and the algorithm of Mohri (2002).

We focus on the question of applicability of the algorithms, i.e., we identify the classes of inputs for which the algorithms are terminating and correct (*class of valid inputs*). In order to have a basis for a fair comparison, we understand the inputs of the algorithms of Knuth (1977), Goodman (1999), and Mohri (2002) as particular wRTG-LMs of the form $((G', \mathcal{CFG}^{\emptyset}), (\mathbb{K}, \oplus, \mathbb{0}, \Omega, \sum^{\oplus}), \mathrm{wt}')$ with $G' = (N', \Sigma', A'_0, R')$. An algorithm is correct for such a wRTG-LM if it returns $\sum^{\oplus}_{d \in \mathrm{AST}(G')} \mathrm{wt}'(d)_{\mathbb{K}}$.

We employ two parameters: C_1 (subset of the class of all RTGs) and C_2 (subset of the class of all weight algebras). Tab. 1 shows the classes of valid inputs parameterized with values for C_1 and C_2. Each valid input in rows (a)–(d) is a closed wRTG-LM. Thus, if one of the value computation algorithms (a)–(c) is applicable, then our value computation algorithm (Alg. 1) is applicable too. In particular, Alg. 1 is applicable to wRTG-LMs with the best derivation M-monoid $\mathbb{BD}$ as weight algebra (cf. Ex. 3), which in general is the case for neither of algorithms (a)–(c). The reason for this is that $\mathbb{BD}$ is not superior (opposing (a)) and RTG-LMs are in general neither acyclic (opposing (b)) nor monadic (opposing (c)). The same holds for ADP problems.

We cannot give a general statement about the complexity of our value computation algorithm (Alg. 1), because the operations in the weight algebra of a wRTG-LM can be undecidable. If we abstract from the costs of particular operations, then we obtain the complexity of Mohri's algorithm. This complexity depends on the number of times the value of a nonterminal changes, which in general is not polynomial in the size of the input wRTG-LM. Mohri circumvents this problem by specifying the order in which nonterminals are processed for well-known classes of inputs, e.g., acyclic graphs or superior weight algebras. We can adapt this idea by imposing such an ordering on the iteration over the nonterminals in line 5. Thus our value computation algorithm achieves the same complexity as Knuth's algorithm (if the input is restricted to superior wRTG-LMs) or the algorithm in Goodman's second phase (if the input is restricted to acyclic wRTG-LMs), respectively.

We note that although our value computation algorithm (Alg. 1) has the same complexity as the other algorithms, in average it performs more computations than those. This is because in each iteration of lines 5–11, the values of all nonterminals are recomputed. This could be avoided by using a direct generalization of Mohri's algorithm to the branching case rather than Alg. 1. However, the intricacies of such a generalization would exceed the scope of this paper.

Acknowledgements

We thank the anonymous reviewers for their helpful comments and our colleagues Kilian Gebhardt and Frederic Dörband for fruitful discussions.

References

Y. Bar-Hillel, M. Perles, and E. Shamir. 1961. On formal properties of simple phrase structure gram-

mars. 14:143–172. Reprinted in Y. Bar-Hillel. (1964). *Language and Information: Selected Essays on their Theory and Application*, Addison-Wesley 1964, 116–150.

R. Bellman. 1952. On the theory of dynamic programming. *Proceedings of the National Academy of Sciences*, 38(8):716–719.

R. Bellman. 1954. The theory of dynamic programming. Technical report, RAND Corp Santa Monica CA.

W. S. Brainerd. 1969. Tree generating regular systems.

M. Büchse, M.-J. Nederhof, and H. Vogler. 2012. Tree parsing for tree-adjoining machine translation. *Journal of Logic and Computation*, 22(6).

E. Dijkstra. 1959. A note on two problems in connexion with graphs. *Numer. Math.*, 1:269–271.

F. Drewes, K. Gebhardt, and H. Vogler. 2016. EM-training for weighted aligned hypergraph bimorphisms. In *Proceedings of the SIGFSM Workshop on Statistical NLP and Weighted Automata*, pages 60–69, Berlin, Germany. Association for Computational Linguistics.

S. Eilenberg. 1974. *Automata, languages, and machines*. Academic press.

Z. Fülöp, A. Maletti, and H. Vogler. 2009. A Kleene theorem for weighted tree automata over distributive multioperator monoids. *Theory of Computing Systems*, 44(3):455–499.

R. Giegerich, C. Meyer, and P. Steffen. 2004. A discipline of dynamic programming over sequence data. *Science of Computer Programming*, 51:215–263.

J. A. Goguen, J. W. Thatcher, E. G. Wagner, and J. B. Wright. 1977. Initial algebra semantics and continuous algebras. *Journal of the ACM (JACM)*, 24(1):68–95.

J. Goodman. 1999. Semiring parsing. *Computational Linguistics*, 25(4):573–605.

L. Kallmeyer. 2010. *Parsing beyond context-free grammars*. Springer.

G. Karner. 1992. On limits in complete semirings. *Semigroup Forum*, 45(1):148–165.

N. A. Khabbaz. 1974. Control sets on linear grammars. *Information and Control*, 25(3):206–221.

D. E. Knuth. 1977. A Generalization of Dijkstra's Algorithm. *Inform. Process. Lett.*, 6(1):1–5.

A. Koller and M. Kuhlmann. 2012. Decomposing TAG algorithms using simple algebraizations. In *Proceedings of the 11th TAG+ Workshop*, Paris.

W. Kuich. 1999. Linear systems of equations and automata on distributive multioperator monoids. *Contributions to general algebra*, 12:247–256.

B. Mahr. 1984. Iteration and summability in semirings. *Annals of Discrete Mathematics*, pages 229–256.

M. Mohri. 2002. Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350.

M.-J. Nederhof. 2003. Squibs and discussions: Weighted deductive parsing and Knuth's algorithm. *Computational Linguistics*, 29(1):135–143.

S. Shieber, Y. Schabes, and F. Pereira. 1995. Principles and implementation of deductive parsing. *The Journal of Logic Programming*, 24(12):3–36.

W. Wechler. 1992. *Universal Algebra for Computer Scientists*, first edition, volume 25 of *Monogr. Theoret. Comput. Sci. EATCS Ser.* Springer-Verlag, Heidelberg/Berlin.