A Chomsky-Schützenberger representation for weighted multiple context-free languages

Tobias Denkinger
Faculty of Computer Science
Technische Universität Dresden
01062 Dresden, Germany
tobias.denkinger@tu-dresden.de

Abstract

We prove a Chomsky-Schützenberger representation theorem for weighted multiple context-free languages.

1 Introduction

Mildly context-sensitive languages receive much attention in the natural language processing community (Kallmeyer, 2010). Many classes of mildly context-sensitive languages are subsumed by the multiple context-free languages, e.g. the languages of head grammars, linear context-free rewriting systems (Seki et al., 1991), combinatory categorial grammars (Vijay-Shanker et al., 1986; Weir and Joshi, 1988), linear indexed grammars (Vijay-Shanker, 1987), minimalist grammars, (Michaelis, 2001a; Michaelis, 2001b), and finite-copying lexical functional grammars (Seki et al., 1993).

The Chomsky-Schützenberger (CS) representation for context-free languages (Chomsky and Schützenberger, 1963, Prop. 2) has recently been generalised to quantitative context-free languages (Droste and Vogler, 2013) and to (unweighted) multiple context-free languages (Yoshinaka et al., 2010). In order to obtain a CS representation for multiple context-free languages, Yoshinaka et al. (2010) introduce multiple Dyck languages.

We give a more algebraic definition of multiple Dyck languages using congruence relations together with a decision algorithm for membership that is strongly related to these congruence relations (Sec. 3). We then provide a CS representation for weighted multiple context-free languages (Sec. 4).

2 Preliminaries

In this section we briefly recall formalisms used in this paper and fix some notation.

We denote by $\mathbb{N}$ the set of natural numbers (including zero). For every $n \in \mathbb{N}$ we abbreviate $\{1, \ldots, n\}$ by $[n]$. Let A be a set. The *power set of A* is denoted by $\mathcal{P}(A)$. Let B be a finite set. A *partition of B* is a set $\mathfrak{P} \subseteq \mathcal{P}(B)$ where the elements of $\mathfrak{P}$ are non-empty, pairwise disjoint, and $\bigcup_{\mathfrak{p} \in \mathfrak{P}} \mathfrak{p} = B$.

Let S be a countable set (of *sorts*) and $s \in S$. An *S-sorted set* is a tuple (B, sort) where B is a set and sort is a function from B to S. We denote the preimage of s under sort by B_s and abbreviate (B, sort) by B; sort will always be clear from the context. An *S-ranked set* is an $(S^* \times S)$-sorted set.

Let A and B be sets. The *set of functions from A to B* is denoted by B^A. Let f and g be functions. The *domain and range of f* are denoted by $\mathrm{dom}(f)$ and $\mathrm{rng}(f)$, respectively. We denote the function obtained by applying g after f by $g \circ f$. Let F be a set of functions and $B \subseteq \bigcap_{f \in F} \mathrm{dom}(f)$. The set $\{f(B) \mid f \in F\} \subseteq \mathcal{P}(\mathrm{rng}(f))$ is denoted by $F(B)$. Let G and H be sets of functions. The set $\{h \circ g \mid h \in H, g \in G\}$ of functions is denoted by $H \circ G$.

We use the notion of nondeterministic finite automata with extended transition function (short: FSA) from Hopcroft and Ullman (1979, Sec. 2.3).

2.1 Weight algebras

A *monoid* is an algebra $(\mathcal{A}, \cdot, 1)$ where $\cdot$ is associative and 1 is neutral with respect to $\cdot$. A *bimonoid* is an algebra $(\mathcal{A}, +, \cdot, 0, 1)$ where $(\mathcal{A}, +, 0)$ and $(\mathcal{A}, \cdot, 1)$ are monoids. We call a bimonoid *strong* if $(\mathcal{A}, +, 0)$ is commutative and for every $a \in \mathcal{A}$ we have $0 \cdot a = 0 = a \cdot 0$. Intuitively, a strong bimonoid is a semiring without distributivity. A strong bimonoid is called *commutative* if $(\mathcal{A}, \cdot, 1)$ is commutative. A commutative strong bimonoid is *complete* if there is an infinitary sum operation $\sum$ that maps every indexed family of elements of $\mathcal{A}$ to $\mathcal{A}$, extends $+$, and satisfies infinitary associativity and commutativity laws; cf. Droste and Vogler (2013, Sec. 2). For the rest of this paper let $(\mathcal{A}, +, \cdot, 0, 1)$, abbreviated by $\mathcal{A}$, be a complete commutative strong bimonoid.

Example 1. We provide a list of complete commutative strong bimonoids (cf. Droste et al. (2010, Ex. 1)) some of which are relevant for natural language processing:

- Any complete commutative semiring, e.g. the *Boolean semiring* $\mathbb{B} = \big(\{0,1\}, \vee, \wedge, 0, 1\big)$, the *probability semiring* $\mathrm{Pr} = \big(\mathbb{R}_{\geq 0}, +, \cdot, 0, 1\big)$, the *Viterbi semiring* $\big([0,1], \max, \cdot, 0, 1\big)$, the *tropical semiring* $\big(\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0\big)$,
- any complete lattice,
- the *tropical bimonoid*
$$\big(\mathbb{R}_{\geq 0} \cup \{\infty\}, +, \min, 0, \infty\big), \text{ and}$$
- the algebra $([0,1], \oplus, \cdot, 0, 1)$ with $\oplus$ being defined for every $a, b \in [0,1]$ as either $a \oplus b = a + b - a \cdot b$ or $a \oplus b = \min\{a+b, 1\}$,

where $\mathbb{R}$ and $\mathbb{R}_{\geq 0}$ denote the set of reals and the set of non-negative reals, respectively, and $+$, $\cdot$, $\max$, $\min$, $\wedge$, $\vee$ denote the usual operations. □

An *$\mathcal{A}$-weighted language (over Δ)* is a function $L\colon \Delta^* \to \mathcal{A}$. The *support of* L, denoted by $\mathrm{supp}(L)$, is $\{w \in \Delta^* \mid L(w) \neq 0\}$. If $|\mathrm{supp}(L)| \leq 1$, we call L a *monomial*. We write $\mu.w$ for L if $L(w) = \mu$ and for every $w' \in \Delta^* \setminus \{w\}$ we have $L(w') = 0$.

2.2 Weighted string homomorphisms

Let Δ and Γ be alphabets and $g\colon \Delta \to \mathcal{A}^{\Gamma^*}$ such that $g(\delta)$ is a monomial for every $\delta \in \Delta$. We define $\widehat{g}\colon \Delta^* \to \mathcal{A}^{\Gamma^*}$ where for every $k \in \mathbb{N}$, $w_1, \ldots, w_k \in \Delta$, and $u \in \Gamma^*$ we have

$$\widehat{g}(w_1 \cdots w_k)(u) = \sum_{\substack{u_1,\ldots,u_k \in \Gamma^* \\ u = u_1 \cdots u_k}} \prod_{i=1}^{k} g(w_i)(u_i)\,.$$

We call $\widehat{g}$ an *$\mathcal{A}$-weighted (string) homomorphism*. An $\mathcal{A}$-weighted homomorphism $h\colon \Delta^* \to \mathcal{A}^{\Gamma^*}$ is *alphabetic* if there is a function $h'\colon \Delta \to \mathcal{A}^{\Gamma \cup \{\varepsilon\}}$ with $h = \widehat{h'}$.

Now assume that $\mathcal{A} = \mathbb{B}$ and for every $\delta \in \Delta$ we have $|\mathrm{supp}(g(\delta))| = 1$. Then g can be construed as a function from Δ to Γ^* and $\widehat{g}$ can be construed as a function from Δ^* to Γ^*. In this case we call $\widehat{g}$ a *(string) homomorphism*. If moreover, g is a function from Δ to $\Gamma \cup \{\varepsilon\}$, we call $\widehat{g}$ *alphabetic*.

The sets of all $\mathcal{A}$-weighted homomorphisms, $\mathcal{A}$-weighted alphabetic homomorphisms, homomorphisms, and alphabetic homomorphisms are denoted by $\mathrm{HOM}(\mathcal{A})$, $\alpha\mathrm{HOM}(\mathcal{A})$, HOM, and $\alpha\mathrm{HOM}$, respectively.

2.3 Weighted multiple context-free languages

We fix a set $X = \{x_i^j \mid i, j \in \mathbb{N}\}$ of *variables*. Variables serve as placeholders for strings. The *set of string functions over Δ* is the $\mathbb{N}$-ranked set F_Δ where for every $\ell, s_1, \ldots, s_\ell, s \in \mathbb{N}$ we have that $(F_\Delta)_{(s_1 \cdots s_\ell, s)}$ is the set of functions $f\colon (\Delta^*)^{s_1} \times \cdots \times (\Delta^*)^{s_\ell} \to (\Delta^*)^s$ that are defined by some equation of the form $f\big(x_1, \ldots, x_\ell\big) = \big(u_1, \ldots, u_s\big)$ where $x_i = (x_i^1, \ldots, x_i^{s_i})$ for every $i \in [\ell]$, $X_f = \{x_i^j \mid i \in [\ell], j \in [s_i]\}$, and $u_1, \ldots, u_s \in (\Delta \cup X_f)^*$.

In this situation, we define the *rank of f*, denoted by $\mathrm{rank}(f)$, and the *fan-out of f*, denoted by $\text{fan-out}(f)$, as ℓ and s, respectively. The string function f is called *linear* if in $u_1 \cdots u_s$ every element of X_f occurs at most once, f is called *non-deleting* if in $u_1 \cdots u_s$ every element of X_f occurs at least once, and f is called *terminal-free* if $u_1, \ldots, u_s \in X_f^*$. If f is non-deleting, it is uniquely determined by the string $[u_1, \ldots, u_s]$. We may therefore write $[u_1, \ldots, u_s]$ for f.

Note that for every $s' \in \mathbb{N}^* \times \mathbb{N}$, the set of linear terminal-free string functions of sort s' is finite.

Definition 2. A *multiple context-free grammar (MCFG)* is a tuple (N, Δ, I, P) where N is a finite $\mathbb{N}$-sorted set (*non-terminals*), $I \subseteq N_1$ (*initial non-terminals*), and $P \subseteq_{\text{fin}} \big\{(A, f, A_1 \cdots A_\ell) \in N \times F_\Delta \times N^\ell \mid \mathrm{sort}(f) = (\mathrm{sort}(A_1) \cdots \mathrm{sort}(A_\ell), \mathrm{sort}(A)), f \text{ is linear}, \ell \in \mathbb{N}\big\}$ (*productions*). We construe P as an $\mathbb{N}$-ranked set where for every $\rho = (A, f, A_1 \cdots A_\ell) \in P$ we have $\mathrm{sort}(\rho) = \mathrm{sort}(f)$. □

Let $G = (N, \Delta, I, P)$ be an MCFG and $w \in \Delta^*$. A production $(A, f, A_1 \cdots A_\ell) \in P$ is usually written as $A \to f(A_1, \ldots, A_\ell)$; it inherits rank and fan-out from f. Also, $\mathrm{rank}(G) = \max_{\rho \in P} \mathrm{rank}(\rho)$ and $\text{fan-out}(G) = \max_{\rho \in P} \text{fan-out}(\rho)$. MCFGs of fan-out at most k are called k-MCFGs. The productions of G form a context-free grammar G' with the elements of F_Δ and '(', ')', and ',' as terminal symbols, N as the set of non-terminals, and I as the set of initial non-terminals. A word in the language of G' is a term over F_Δ and can be evaluated to a word in Δ^*. The *set of derivations of w in G*, denoted by $D_G(w)$, is the set of abstract syntax trees in G' whose corresponding words are evaluated to w. The *language of G* is $L(G) = \{w \in \Delta^* \mid D_G(w) \neq \emptyset\}$. A language L is *multiple context-free* if there is an MCFG G with $L = L(G)$. The *set of multiple context-free languages (for which a k-MCFG* *exists)* is denoted by MCFL (k-MCFL, respectively).

Let $k \in \mathbb{N}$. The class k-MCFL is a substitution-closed full abstract family of languages (Seki et al., 1991, Thm. 3.9). In particular, k-MCFL is closed under intersection with regular languages and under homomorphisms.

Definition 3. An *$\mathcal{A}$-weighted MCFG* is a tuple (N, Δ, I, P, μ) where (N, Δ, I, P) is an MCFG and $\mu\colon P \to \mathcal{A}$ (*weight function*). □

Let $G = (N, \Delta, I, P, \mu)$ be an $\mathcal{A}$-weighted MCFG and $w \in \Delta^*$. The *set of derivations of w in G* is the set of derivations of w in (N, Δ, I, P). G inherits fan-out from (N, Δ, I, P); $\mathcal{A}$-weighted MCFGs of fan-out at most k are called $\mathcal{A}$-weighted k-MCFGs. We apply μ to derivations by applying it at every position (of the derivation) and then multiplying the resulting values (in any order, since $\cdot$ is commutative).

The *$\mathcal{A}$-weighted language induced by G* is the function $[\![G]\!]\colon \Delta^* \to \mathcal{A}$ where for every $w \in \Delta^*$ we have $[\![G]\!](w) = \sum_{d \in D_G(w)} \mu(d)$. Two ($\mathcal{A}$-weighted) MCFGs are *equivalent* if they induce the same ($\mathcal{A}$-weighted) language. An $\mathcal{A}$-weighted language L is *multiple context-free and of fan-out k* if there is an $\mathcal{A}$-weighted k-MCFG G such that $L = [\![G]\!]$; k-MCFL($\mathcal{A}$) denotes the *set of multiple context-free $\mathcal{A}$-weighted languages of fan-out k*.

Example 4. Consider the Pr-weighted MCFG $G = \big(\{S, A, B\}, \Delta, \{S\}, \{\rho_1, \ldots, \rho_5\}, \mu\big)$ where $\Delta = \{a, b, c, d\}$, $\mathrm{sort}(S) = 1$, $\mathrm{sort}(A) = \mathrm{sort}(B) = 2$, and

$$\rho_1\colon S \to [x_1^1 x_2^1 x_1^2 x_2^2](A, B) \quad \mu(\rho_1) = 1$$
$$\rho_2\colon A \to [a x_1^1, c x_1^2](A) \quad \mu(\rho_2) = 1/2$$
$$\rho_3\colon B \to [b x_1^1, d x_1^2](B) \quad \mu(\rho_3) = 1/3$$
$$\rho_4\colon A \to [\varepsilon, \varepsilon]() \quad \mu(\rho_4) = 1/2$$
$$\rho_5\colon B \to [\varepsilon, \varepsilon]() \quad \mu(\rho_5) = 2/3\ .$$

We observe that $\mathrm{supp}([\![G]\!]) = \{a^m b^n c^m d^n \mid m, n \in \mathbb{N}\}$ and for every $m, n \in \mathbb{N}$ we have $[\![G]\!](a^m b^n c^m d^n) = \mu(\rho_1) \cdot \big(\mu(\rho_2)\big)^m \cdot \mu(\rho_4) \cdot \big(\mu(\rho_3)\big)^m \cdot \mu(\rho_5) = 1/(2^m \cdot 3^{n+1})$. The only derivation of a^2bc^2d in G is shown in Fig. 1. □

Non-deleting normal form An ($\mathcal{A}$-weighted) MCFG is called *non-deleting* if the string function in every production is linear and non-deleting. Seki et al. (1991, Lem. 2.2) proved that for every k-MCFG there is an equivalent non-deleting k-MCFG. We generalise this to $\mathcal{A}$-weighted MCFGs.

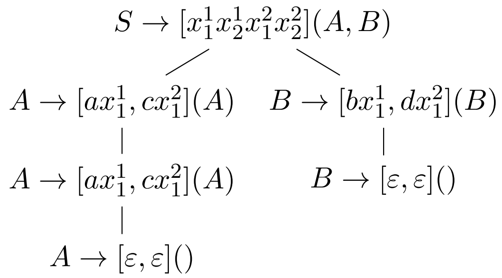

Figure 1: Only derivation of a^2bc^2d in G (Ex. 4).

Lemma 5. For every $\mathcal{A}$-weighted k-MCFG there is an equivalent non-deleting $\mathcal{A}$-weighted k-MCFG.

Proof. Let $G = (N, \Delta, I, P, \mu)$. When examining the proof of Seki et al. (1991, Lem. 2.2), we notice that only step 2 of Procedure 1 deals with non-deletion. We construct N' and P' from (N, Δ, I, P) by step 2 of Procedure 1, but drop the restriction that $\Psi \neq [\mathrm{sort}(A)]$.[1] Let $g\colon P' \to P$ assign to every $\rho' \in P'$ the production in G it has been constructed from. Furthermore, let $I' = \{A[\emptyset] \mid A \in I\}$ and $\mu' = \mu \circ g$. Since the construction preserves the structure of derivations, we have for every $w \in \Delta^*$ that g gives rise to a bijection $\widehat{g}$ between $D_{G'}(w)$ and $D_G(w)$ with $\mu' = \mu \circ \widehat{g}$. Hence $[\![G]\!] = [\![(N', \Delta, I', P', \mu')]\!]$. The fan-out is not increased by this construction. ■

3 Multiple Dyck languages

According to Kanazawa (2014, Sec. 1) there is no definition of multiple Dyck languages using congruence relations. We close this gap by giving such a definition (Def. 7).

3.1 The definition

We recall the definition of multiple Dyck languages (Yoshinaka et al., 2010, Def. 1): Let Δ be a finite $\mathbb{N}$-sorted set,[2] $\overline{(\cdot)}$ be a bijection between Δ and some alphabet $\overline{\Delta}$, $k = \max_{\delta \in \Delta} \mathrm{sort}(\delta)$, and $r \geq k$. The *multiple Dyck grammar with respect to Δ* is the k-MCFG $G_\Delta = \big(\{A_1, \ldots, A_k\}, \widehat{\Delta}, \{A_1\}, P\big)$ where $\widehat{\Delta} = \{\delta^{[i]}, \bar{\delta}^{[i]} \mid \delta \in \Delta, i \in [\mathrm{sort}(\delta)]\}$, $\mathrm{sort}(A_i) = i$ for every $i \in [k]$, and P is the smallest set such that

(i) for every linear non-deleting[3] terminal-free string function $f \in (F_\Delta)_{(s_1 \cdots s_\ell, s)}$ with $\ell \in$

[1] This construction may therefore create productions of fan-out 0.

[2] In Yoshinaka et al. (2010), $\mathbb{N}$-sorted sets are called indexed sets and sort is denoted as dim.

[3] We add the restriction "non-deleting" in comparison to the original definition since in Yoshinaka et al. (2010, Proof of Lem. 1) only non-deleting rules are used.

$[r]$, $s_1, \ldots, s_\ell$, $s \in [k]$ we have $A_s \to f(A_{s_1}, \ldots, A_{s_\ell}) \in P$,

(ii) for every $\delta \in \Delta$ with sort s we have $A_s \to [\delta^{[1]} x_1^1 \bar{\delta}^{[1]}, \ldots, \delta^{[s]} x_1^s \bar{\delta}^{[s]}](A_s) \in P$, and

(iii) for every $s \in [k]$ we have $A_s \to [u_1, \ldots, u_s](A_s) \in P$ where $u_i \in \{x_i,\ x_i \delta^{[1]} \bar{\delta}^{[1]},\ \delta^{[1]} \bar{\delta}^{[1]} x_i \mid \delta \in \Delta_1\}$ for every $i \in [s]$.

The *multiple Dyck language with respect to* Δ, denoted by $mD(\Delta)$, is $L(G_\Delta)$. We call $\max_{\delta \in \Delta} \operatorname{sort}(\delta)$ the *dimension of* $mD(\Delta)$. The *set of multiple Dyck languages of dimension at most k* is denoted by k-mDYCK.

For the rest of this section let Σ be an alphabet. Also let $\overline{\Sigma}$ be a set (disjoint from Σ) and $\overline{(\cdot)}$ be a bijection between Σ and $\overline{\Sigma}$. Intuitively Σ and $\overline{\Sigma}$ are sets of opening and closing parentheses and $\overline{(\cdot)}$ matches an opening to its closing parenthesis.

We define $\equiv_\Sigma$ as the smallest congruence relation on the free monoid $(\Sigma \cup \overline{\Sigma})^*$ where for every $\sigma \in \Sigma$ the cancellation rule $\sigma\overline{\sigma} \equiv_\Sigma \varepsilon$ holds. The *Dyck language with respect to* Σ, denoted by $D(\Sigma)$, is $[\varepsilon]_{\equiv_\Sigma}$. The *set of Dyck languages* is denoted by DYCK.

Example 6. Let $\Sigma = \{(, \langle, [, \llbracket\}$. We abbreviate $\bar{(}$, $\bar{\langle}$, $\bar{[}$, and $\bar{\llbracket}$ by $)$, $\rangle$, $]$, and $\rrbracket$, respectively. Then we have for example $\llbracket()\rrbracket\langle\rangle() \equiv_\Sigma \llbracket\rrbracket\langle\rangle \equiv_\Sigma \llbracket\rrbracket \equiv_\Sigma \varepsilon$ and $(\llbracket)\rrbracket\langle\rangle() \equiv_\Sigma (\llbracket)\rrbracket() \equiv_\Sigma (\llbracket)\rrbracket \not\equiv_\Sigma \varepsilon$. □

Let $\mathfrak{P}$ be a partition of Σ. We define $\equiv_{\Sigma,\mathfrak{P}}$ as the smallest congruence relation on the free monoid $(\Sigma \cup \overline{\Sigma})^*$ such that if $v_1 \cdots v_\ell \equiv_{\Sigma,\mathfrak{P}} \varepsilon$ with $v_1, \ldots, v_\ell \in D(\Sigma)$, then the *cancellation rule*

$$u_0 \sigma_1 v_1 \overline{\sigma_1} u_1 \cdots \sigma_\ell v_\ell \overline{\sigma_\ell} u_\ell \equiv_{\Sigma,\mathfrak{P}} u_0 \cdots u_\ell$$

holds for every $\{\sigma_1, \ldots, \sigma_\ell\} \in \mathfrak{P}$ and $u_0, \ldots, u_\ell \in D(\Sigma)$. Intuitively, every element of $\mathfrak{P}$ denotes a set of *linked* opening parentheses, i.e. parentheses that must be consumed simultaneously by $\equiv_{\Sigma,\mathfrak{P}}$.

Definition 7. The *congruence multiple Dyck language with respect to* Σ *and* $\mathfrak{P}$, denoted by $mD_c(\Sigma, \mathfrak{P})$, is $[\varepsilon]_{\equiv_{\Sigma,\mathfrak{P}}}$. □

Example 8. Let $\Sigma = \{(, \langle, [, \llbracket\}$ and $\mathfrak{P} = \{\mathfrak{p}_1, \mathfrak{p}_2\}$ where $\mathfrak{p}_1 = \{(, \langle\}$ and $\mathfrak{p}_2 = \{[, \llbracket\}$. We abbreviate $\bar{(}$, $\bar{\langle}$, $\bar{[}$, and $\bar{\llbracket}$ by $)$, $\rangle$, $]$, and $\rrbracket$, respectively. Then we have for example $\llbracket()\rrbracket[\langle\rangle] \equiv_{\Sigma,\mathfrak{P}} \varepsilon$ since $\mathfrak{p}_2 = \{[, \llbracket\} \in \mathfrak{P}$, $()\langle\rangle \equiv_{\Sigma,\mathfrak{P}} \varepsilon$, and $u_0 = u_1 = u_2 = \varepsilon$. But $\llbracket()\rrbracket\langle[]\rangle \not\equiv_{\Sigma,\mathfrak{P}} \varepsilon$ since when instantiating the cancellation rule with any of the two elements of $\mathfrak{P}$, we can not reduce $\llbracket()\rrbracket\langle[]\rangle$:

(i) If we choose $\{\sigma_1, \sigma_2\} = \{\llbracket, [\}$ then we would need to set $u_1 = \langle$ and $u_2 = \rangle$, but they are not in $D(\Sigma)$, also $() \not\equiv_{\Sigma,\mathfrak{P}} \varepsilon$;

(ii) If we choose $\{\sigma_1, \sigma_2\} = \{(, \langle\}$ then we would need to set $u_0 = \llbracket$ and $u_1 = \rrbracket$, but they are not in $D(\Sigma)$, also $[] \not\equiv_{\Sigma,\mathfrak{P}} \varepsilon$.

Hence $\llbracket()\rrbracket[\langle\rangle], ()\langle\rangle \in mD_c(\Sigma, \mathfrak{P})$ and $\llbracket()\rrbracket\langle[]\rangle \notin mD_c(\Sigma, \mathfrak{P})$. □

Observation 9. From the definition of $\equiv_{\Sigma,\mathfrak{P}}$ it is easy to see that for every $u_1, \ldots, u_k \in D(\Sigma)$ and $v_1, \ldots, v_\ell \in D(\Sigma)$ we have that $u_1 \cdots u_k, v_1 \cdots v_\ell \in mD_c(\Sigma, \mathfrak{P})$ implies that every permutation of $u_1, \ldots, u_k, v_1, \ldots, v_\ell$ is in $mD_c(\Sigma, \mathfrak{P})$. ■

The *dimension of* $mD_c(\Sigma, \mathfrak{P})$ is $\max_{\mathfrak{p} \in \mathfrak{P}} |\mathfrak{p}|$. The *set of congruence multiple Dyck languages (of at most dimension k)* is denoted by $\mathrm{mDYCK_c}$ ($\mathrm{k\text{-}mDYCK_c}$, respectively).

Note that the dimension of $\mathfrak{P}$ is 1 if and only if $\mathfrak{P} = \{\{\sigma\} \mid \sigma \in \Sigma\}$. In this situation we have $\equiv_\Sigma\ =\ \equiv_{\Sigma,\mathfrak{P}}$ and therefore also $D(\Sigma) = mD_c(\Sigma, \mathfrak{P})$. Hence $\mathrm{DYCK} = \mathrm{1\text{-}mDYCK_c}$.

Proposition 10. $\mathrm{k\text{-}mDYCK} \subseteq \mathrm{k\text{-}mDYCK_c}$

Idea of the proof. We show the property $(*)$ that implies our claim. The "⇒" we prove by induction on the structure of derivations in G_Δ. For "⇐" we construct derivations in G_Δ by induction on the number of applications of the cancellation rule.

Proof. Let $mD \in$ k-mDYCK. Then there is an $\mathbb{N}$-sorted set Δ such that $mD = mD(\Delta)$ and $k \geq \max_{\delta \in \Delta} \operatorname{sort}(\delta)$. We define $\mathfrak{p}_\delta = \{\delta^{[i]} \mid i \in [\operatorname{sort}(\delta)]\}$ for every $\delta \in \Delta$, $\Sigma = \bigcup_{\delta \in \Delta} \mathfrak{p}_\delta$, and $\mathfrak{P} = \{\mathfrak{p}_\delta \mid \delta \in \Delta\}$. Clearly $\max_{\mathfrak{p} \in \mathfrak{P}} |\mathfrak{p}| \leq k$. Thus $mD_c(\Sigma, \mathfrak{P}) \in$ k-mDYCK. Let $\operatorname{Tup}(G_\Delta, A)$ denote the set of tuples generated in G_Δ when starting with non-terminal A where A is not necessarily initial. In the following we show that for every $m \in [\max_{\delta \in \Delta} \operatorname{sort}(\delta)]$ and $w_1, \ldots, w_m \in (\Sigma \cup \bar{\Sigma})^*$:

$$\begin{aligned} &(w_1, \ldots, w_m) \in \operatorname{Tup}(G_\Delta, A_m) \\ \iff\ & w_1 \cdots w_m \in mD_c(\Sigma, \mathfrak{P}) \qquad (*) \\ & \wedge w_1, \ldots, w_m \in D(\Sigma)\ . \end{aligned}$$

We show the "⇒" by induction on the structure of derivations in G_Δ: From the definitions of Tup and G_Δ we have that $(w_1, \ldots, w_m) \in \operatorname{Tup}(G_\Delta, A_m)$ implies that there are a rule $A_m \to f(A_{m_1}, \ldots, A_{m_\ell})$ in G_Δ and a tuple $\vec{u}_i = (u_i^1, \ldots, u_i^{m_i}) \in \operatorname{Tup}(G_\Delta, A_{m_i})$ for every $i \in [\ell]$

such that $f(\vec{u}_1, \ldots, \vec{u}_\ell) = (w_1, \ldots, w_m)$. By applying the induction hypothesis ℓ times, we also have that $u_1^1, \ldots, u_1^{m_1}, \ldots, u_\ell^1, \ldots, u_\ell^{m_\ell} \in D(\Sigma)$ and $u_1^1 \cdots u_1^{m_1}, \ldots, u_\ell^1 \cdots u_\ell^{m_\ell} \in mD(\Sigma, \mathfrak{P})$. We distinguish three cases (each corresponding to one type of rule in G_Δ):

(i) f is linear, non-deleting, and terminal-free. Then we have for every $i \in [m]$ that $w_i \in \{u_1^1, \ldots, u_1^{m_1}, \ldots, u_\ell^1, \ldots, u_\ell^{m_\ell}\}^*$ and therefore also $w_i \in D(\Sigma)$. Furthermore, by applying Obs. 9 $(\ell - 1)$ times, we have that $w_1 \cdots w_m \in mD_c(\Sigma, \mathfrak{P})$.

(ii) $f = [\delta^{[1]} x_1^1 \bar{\delta}^{[1]}, \ldots, \delta^{[m]} x_1^m \bar{\delta}^{[m]}]$; then $\ell = 1$, $m_1 = m$, and for every $i \in [m]$ we have $w_i = \delta^{[i]} u_1^i \bar{\delta}^{[i]}$ and since $u_1^i \in D(\Sigma)$ also $w_i \in D(\Sigma)$. Furthermore, $w_1 \cdots w_m = \delta^{[1]} u_1^1 \bar{\delta}^{[1]} \cdots \delta^{[m]} u_1^m \bar{\delta}^{[m]} \in mD_c(\Sigma, \mathfrak{P})$ due to the cancellation rule.

(iii) $f = [u_1, \ldots, u_m]$ where for every $i \in [m]$: $u_i \in \{x_i,\ x_i \delta^{[1]} \bar{\delta}^{[1]},\ \delta^{[1]} \bar{\delta}^{[1]} x_i \mid \delta \in \Delta_1\}$; then $\ell = 1$, $m_1 = m$, and $w_i \in \{u_i^1,\ x_i^1 \delta^{[1]} \bar{\delta}^{[1]},\ \delta^{[1]} \bar{\delta}^{[1]} x_i^1 \mid \delta \in \Delta_1\}$. Since $\equiv_\Sigma$ is a congruence relation, we have that $w_1, \ldots, w_m \in D(\Sigma)$. By applying Obs. 9 m times, we have that $w_1 \cdots w_m \in mD_c(\Sigma, \mathfrak{P})$.

We show the "$\Leftarrow$" by induction on the number of applications of the cancellation rule (including the number of applications to reduce the word $v_1 \cdots v_\ell$ from the definition on the cancellation rule to ε): If the cancellation rule is applied zero times in order to reduce $w_1 \cdots w_m$ to ε then $w_1 = \ldots = w_m = \varepsilon$. The rule $A_m \to [\varepsilon, \ldots, \varepsilon]()$ clearly derives $(w_1, \ldots, w_m)$. If the cancellation rule is applied $i + 1$ times in order to reduce $w_1 \cdots w_m$ to ε then $w_1 \cdots w_m$ has the form $u_0 \sigma_1 v_1 \overline{\sigma_1} u_1 \cdots \sigma_\ell v_\ell \overline{\sigma_\ell} u_\ell$ for some $u_0, \ldots, u_\ell \in D(\Sigma)$, $v_1, \ldots, v_\ell \in D(\Sigma)$, and $\{\sigma_1, \ldots, \sigma_\ell\} \in \mathfrak{P}$ with $v_1 \cdots v_\ell \equiv_{\Sigma, \mathfrak{P}} \varepsilon$. Then we need to apply the cancellation rule at most i times to reduce $v_1 \cdots v_\ell$ to ε, hence, by induction hypothesis, there is some $d \in D_{G_\Delta}$ that derives $(v_1, \ldots, v_\ell)$. We use an appropriate rule ρ of type (ii) such that $\rho(d)$ derives $(\sigma_1 v_1 \overline{\sigma_1}, \ldots, \sigma_\ell v_\ell \overline{\sigma_\ell})$. Also, we need to apply the cancellation rule at most i times in order to reduce $u_0 \cdots u_\ell$ to ε, hence, by induction hypothesis, there are derivations $d_1, \ldots, d_n$ that derive tuples containing exactly $u_0, \ldots, u_\ell$ as components. Then there is a rule ρ' such that $\rho'(\rho(d), d_1, \ldots, d_n) \in D_{G_\Delta}$ derives the tuple $(w_1, \ldots, w_m)$.

From $(*)$ with $m = 1$ and the fact "$w_1 \in mD_c(\Sigma, \mathfrak{P})$ implies $w_1 \in D(\Sigma)$" we get that $mD_c(\Sigma, \mathfrak{P}) = mD$. ■

Lemma 11. k-mDYCK$_c$ $\subseteq$ k-MCFL

Proof idea. For any congruence multiple Dyck language we construct a multiple Dyck grammar that is equivalent up to a homomorphism. We then use the closure of k-MCFL under homomorphisms.

Proof. Let $L \in$ k-mDYCK$_c$. Then there are an alphabet Σ and a partition $\mathfrak{P}$ of Σ such that $mD_c(\Sigma, \mathfrak{P}) = L$. Consider $\mathfrak{P}$ as an $\mathbb{N}$-sorted set where the sort of an element is its cardinality. Then $\widehat{\Delta} = \{\mathfrak{p}^{[i]}, \bar{\mathfrak{p}}^{[i]} \mid \mathfrak{p} \in \mathfrak{P}, i \in [|\mathfrak{p}|]\}$. For every $\mathfrak{p} \in \mathfrak{P}$ assume some fixed enumeration of the elements of $\mathfrak{p}$. We define a bijection $g\colon \widehat{\Delta} \to \Sigma \cup \overline{\Sigma}$ such that every $\mathfrak{p}^{[i]}$ (for some $\mathfrak{p}$ and i) is assigned the i-th element of $\mathfrak{p}$ and $g(\bar{\mathfrak{p}}^{[i]}) = \overline{g(\mathfrak{p}^{[i]})}$. Then $g(L(G_\mathfrak{P})) = L$, where $G_\mathfrak{P}$ is the multiple Dyck grammar with respect to $\mathfrak{P}$. Since k-MCFLs are closed under homomorphisms (Seki et al., 1991, Thm. 3.9), $L \in$ k-MCFL.[4] ■

Observation 12. Examining the definition of multiple Dyck grammars, we observe that some production in item (ii) has fan-out k for at least one $\delta \in \Delta$. Then, using Seki et al. (1991, Thm. 3.4), we have for every $k \geq 1$ that (k + 1)-mDYCK$_c$ $\setminus$ k-MCFL $\neq \emptyset$. ■

Proposition 13.

$$\text{1-mDYCK}_c \subsetneq \text{2-mDYCK}_c \subsetneq \ldots$$

Proof. We get '$\subseteq$' from the definition of k-mDYCK$_c$ and '$\neq$' from Obs. 12. ■

3.2 Membership in a congruence multiple Dyck language

We provide a recursive algorithm (Alg. 1) to decide whether a word w is in a given congruence multiple Dyck language $mD_c(\Sigma, \mathfrak{P})$. This amounts to checking whether $w \equiv_{\Sigma, \mathfrak{P}} \varepsilon$, and it suffices to only apply the cancellation rule from left to right.

Outline of Alg. 1 If w is the empty word, we return 1 on line 2 since the empty word is in $mD_c(\Sigma, \mathfrak{P})$. Then we check if w is in $D(\Sigma)$, e.g. with the context-free grammar in (7.6) in Salomaa (1973). If w is not in $D(\Sigma)$, it is also not in $mD_c(\Sigma, \mathfrak{P})$ and we return 0. Otherwise, we split w into shortest non-empty Dyck words (on

[4] This construction shows that Def. 7 is equivalent to Def. 1 in Yoshinaka et al. (2010) modulo the application of g.

Algorithm 1 Membership in $mD_{\mathrm{c}}(\Sigma, \mathfrak{P})$

Input: Σ, $\mathfrak{P}$, and $w \in (\Sigma \cup \overline{\Sigma})^*$
Output: 1 if $w \in mD_{\mathrm{c}}(\Sigma, \mathfrak{P})$, 0 otherwise

1: **function** MAIN($\Sigma, \mathfrak{P}, w$)
2:   **if** $w = \varepsilon$ **then** **return** 1 **end if**
3:   **if** $w \notin D(\Sigma)$ **then** **return** 0 **end if**
4:   $(\sigma_1 u_1 \overline{\sigma}_1, \ldots, \sigma_\ell u_\ell \overline{\sigma}_\ell) \leftarrow$ SPLIT(Σ, w) ▷ such that $\sigma_1, \ldots, \sigma_\ell \in \Sigma$
5:   $\mathcal{I} \leftarrow \{I \subseteq \mathcal{P}([\ell]) \mid I$ partition of $[\ell], \forall \{i_1, \ldots, i_k\} \in I \colon \{\sigma_{i_1}, \ldots, \sigma_{i_k}\} \in \mathfrak{P}\}$
6:   **for** $I \in \mathcal{I}$ **do**
7:     $b \leftarrow 1$
8:     **for** $\{i_1, \ldots, i_k\} \in I$ **do** ▷ such that $i_1 < \ldots < i_k$
9:       $b \leftarrow b \cdot$ MAIN($\Sigma, \mathfrak{P}, u_{i_1} \cdots u_{i_k}$)
10:     **end for**
11:     **if** $b = 1$ **then** **return** 1 **end if**
12:   **end for**
13:   **return** 0
14: **end function**

15: **function** SPLIT(Σ, w)
16:   $(u_1, \ldots, u_\ell) \leftarrow$ sequence of shortest words $u_1, \ldots, u_\ell \in D(\Sigma) \setminus \{\varepsilon\}$ with $w = u_1 \cdots u_\ell$
17:   **return** $(u_1, \ldots, u_\ell)$
18: **end function**

Table 1: Run of Alg. 1 on the word $\llbracket()\rrbracket[\langle\rangle]$, cf. Exs. 8 and 14.

MAIN($\Sigma, \mathfrak{P}, \llbracket()\rrbracket[\langle\rangle]$)
l. 4: $\sigma_1 = \llbracket, \sigma_2 = [, u_1 = (), u_2 = \langle\rangle$
l. 5: $\mathcal{I} = \big\{\{\{1,2\}\}\big\}$
l. 6: $I = \big\{\{1,2\}\big\}$
l. 8: $k = 2, i_1 = 1, i_2 = 2$
l. 9: $b = 1 \cdot$ MAIN($\Sigma, \mathfrak{P}, ()\langle\rangle$)
    l. 4: $\sigma_1 = (, \sigma_2 = \langle, u_1 = \varepsilon = u_2$
    l. 5: $\mathcal{I} = \big\{\{\{1,2\}\}\big\}$
    l. 6: $I = \big\{\{1,2\}\big\}$
    l. 8: $k = 2, i_1 = 1, i_2 = 2$
    l. 9: $b = 1 \cdot$ MAIN($\Sigma, \mathfrak{P}, \varepsilon$)
        l. 2: **return** 1
    l. 9: $b = 1 \cdot 1 = 1$
    l. 11: **return** 1
l. 9: $b = 1 \cdot 1$
l. 11: **return** 1

line 4) using the function SPLIT. Since each of those shortest non-empty Dyck words has the form $\sigma u \overline{\sigma}$ for some $\sigma \in \Sigma$ and $u \in (\Sigma \cup \overline{\Sigma})^*$, we write $(\sigma_1 u_1 \overline{\sigma}_1, \ldots, \sigma_\ell u_\ell \overline{\sigma}_\ell)$ for the left-hand side of the assignment on line 4. On line 5 we calculate the set $\mathcal{I}$ of all partitions I such that each element of I specifies a set of components of the tuple $(\sigma_1 u_1 \overline{\sigma}_1, \ldots, \sigma_\ell u_\ell \overline{\sigma}_\ell)$ whose outer parentheses can be removed with one application of the cancellation rule. Since I is a partition, we know that the outer parentheses of every component can be removed via the cancellation rule. Then it remains to be shown that there is a partition I such that for each element $\{i_1, \ldots, i_k\}$ of I the word $u_{i_1} \cdots u_{i_k}$ is an element of $mD_{\mathrm{c}}(\Sigma, \mathfrak{P})$; this is done on lines 6 to 12. If there is no such partition, then we return 0 on line 13.

Example 14 (Ex. 8 continued)**.** Tab. 1 shows a run of Alg. 1 on the word $\llbracket()\rrbracket[\langle\rangle]$ where we report return values and a subset of the variable assignment whenever we reach the end of lines 4, 5, 6, 8, 9. The recursive calls to MAIN are indented. □

In light of the close link between Alg. 1 and the relation $\equiv_{\Sigma,\mathfrak{P}}$ we omit the proof of correctness.

Proof of termination for Alg. 1. If $w = \varepsilon$, the algorithm terminates on line 2. If $w \notin D(\Sigma)$, the algorithm terminates on line 3. Since $\mathcal{I}$ is finite and each element $I \in \mathcal{I}$ is also finite, there are only finitely many calls to MAIN on line 9 for each recursion. In each of those calls, the length of the third argument is strictly smaller then the length of w. Therefore, after a finite number of recursions, the third argument passed to MAIN is either the empty word, then the algorithm terminates on line 2, or not an element of $D(\Sigma)$, then the algorithm terminates on line 2. ■

4 CS theorem for weighted MCFLs

In this section we generalise the CS representation of (unweighted) MCFLs (Yoshinaka et al., 2010, Thm. 3) to the weighted case. We prove that an $\mathcal{A}$-weighted MCFL L can be decomposed into an $\mathcal{A}$-weighted alphabetic homomorphism h, a regular language R and a congruence multiple Dyck language mD_c such that $L = h(R \cap mD_\mathrm{c})$.

To show this, we use the proof idea from Droste and Vogler (2013). The outline of our proof is as follows:

(i) We separate the weights from L (Lem. 15), obtaining an MCFL L' and a weighted alphabetic homomorphism.

(ii) We use a corollary of the CS representation of (unweighted) MCFLs (Cor. 16) to obtain a CS representation of L'.

(iii) Using the two previous points and an observation for the composition of weighted and unweighted alphabetic homomorphisms (Lem. 18), we obtain a CS representation of L (Thm. 19).

Lemma 15.

$$\text{k-MCFL}(\mathcal{A}) = \alpha\mathrm{HOM}(\mathcal{A})\big(\text{k-MCFL}\big)$$

Proof. ($\subseteq$) Let $L \in \text{k-MCFL}(\mathcal{A})$. By Lem. 5 there is a non-deleting $\mathcal{A}$-weighted k-MCFG $G = (N, \Delta, I, P, \mu)$ such that $[\![G]\!] = L$. We define a non-deleting k-MCFG $G' = (N, \Delta', I, P')$ where $\Delta' = \Delta \cup \{\rho^i \mid \rho \in P, i \in [\text{fan-out}(\rho)]\}$ and P' is the smallest set such that for every production $\rho = A \to [u_1, \ldots, u_s](A_1, \ldots, A_m) \in P$ there is a production $A \to [\rho^1 u_1, \ldots, \rho^s u_s](A_1, \ldots, A_m) \in P'$. We define an $\mathcal{A}$-weighted alphabetic homomorphism $h\colon (\Delta')^* \to \mathcal{A}^{\Delta^*}$ where $h(\delta) = 1.\delta$ for every $\delta \in \Delta$, $h(\rho^1) = \mu(\rho).\varepsilon$ for every $\rho \in P$, and $h(\rho^i) = 1.\varepsilon$ for every $\rho \in P$ and $i \in \{2, \ldots, \text{fan-out}(\rho)\}$. Since 1 is neutral in multiplication, $\cdot$ is commutative, and G' is non-deleting, we have $L = h(L(G'))$.

($\supseteq$) Let $L \in \text{k-MCFL}$ and $h\colon \Gamma^* \to \mathcal{A}^{\Delta^*}$ an $\mathcal{A}$-weighted alphabetic homomorphism. By Seki et al. (1991, Lem. 2.2) there is a non-deleting k-MCFG $G = (N, \Gamma, I, P)$ such that $L(G) = L$. We construct the $\mathcal{A}$-weighted k-MCFG $G' = (N, \Delta, I, P', \mu)$ as follows: We extend h to $h'\colon (\Gamma \cup X)^* \to \mathcal{A}^{(\Delta \cup X)^*}$ where $h'(x) = 1.x$ for every $x \in X$ and $h'(\gamma) = h(\gamma)$ for every $\gamma \in \Gamma$. We define P' as the smallest set such that for every $\rho = A \to [u_1, \ldots, u_s](A_1, \ldots, A_m) \in P_{(s_1 \cdots s_m, s)}$ and $(u'_1, \ldots, u'_s) \in \mathrm{supp}(h'(u_1)) \times \ldots \times \mathrm{supp}(h'(u_s))$ we have that P' contains the production $\rho' = A \to [u'_1, \ldots, u'_s](A_1, \ldots, A_m)$ and $\mu(\rho') = h'(u_1)(u'_1) \cdot \ldots \cdot h'(u_s)(u'_s)$. Since $\cdot$ is commutative and G non-deleting, we have that $[\![G']\!] = h([\![G]\!])$.[5] ■

By setting $k = 1$ in the above lemma we reobtain the equivalence of 1 and 3 in Thm. 2 of Droste and Vogler (2013) for complete commutative strong bimonoids.

The following is a corollary to Yoshinaka et al. (2010, Thm. 3) where the homomorphism is replaced by an alphabetic homomorphism and the multiple Dyck language is replaced by a congruence multiple Dyck language.

Corollary 16. Let L be a language and $k \in \mathbb{N}$. Then the following are equivalent:

(i) $L \in \text{k-MCFL}$

(ii) there are an alphabetic homomorphism h, a regular language R, and a congruence multiple Dyck language mD_c of at most dimension k with $L = h(R \cap mD_\mathrm{c})$.

Proof. The construction of h in Yoshinaka et al. (2010, Sec. 3.2) already satisfies the definition of an alphabetic homomorphism. We may use a congruence multiple Dyck language instead of a multiple Dyck language since, for (i) $\Rightarrow$ (ii), $\text{k-mDYCK} \subseteq \text{k-mDYCK}_\mathrm{c}$ and, for (ii) $\Rightarrow$ (i), $\text{k-mDYCK}_\mathrm{c} \subseteq \text{k-MCFL}$ and k-MCFL is closed under intersection with regular languages and under homomorphisms. ■

We give an example to show how Lem. 15 and Yoshinaka et al. (2010, Sec. 3.2) can be employed to construct a regular language for the CS representation of weighted MCFLs. The regular language is represented by an FSA.

Example 17 (Ex. 4 continued)**.** We construct an MCFG G' from G as described in the proof of ($\subseteq$) in Lem. 15. Fig. 2 shows the FSA R obtained from G' by the construction in Yoshinaka et al. (2010, Sec. 3.2). An edge labelled with a set L of words denotes a set of transitions each reading a word in L. Note that the language of R is not finite. □

[5] The same two constructions also work to show that $\text{k-MCFL}(\mathcal{A}) = \mathrm{HOM}(\mathcal{A})\big(\text{k-MCFL}\big)$.

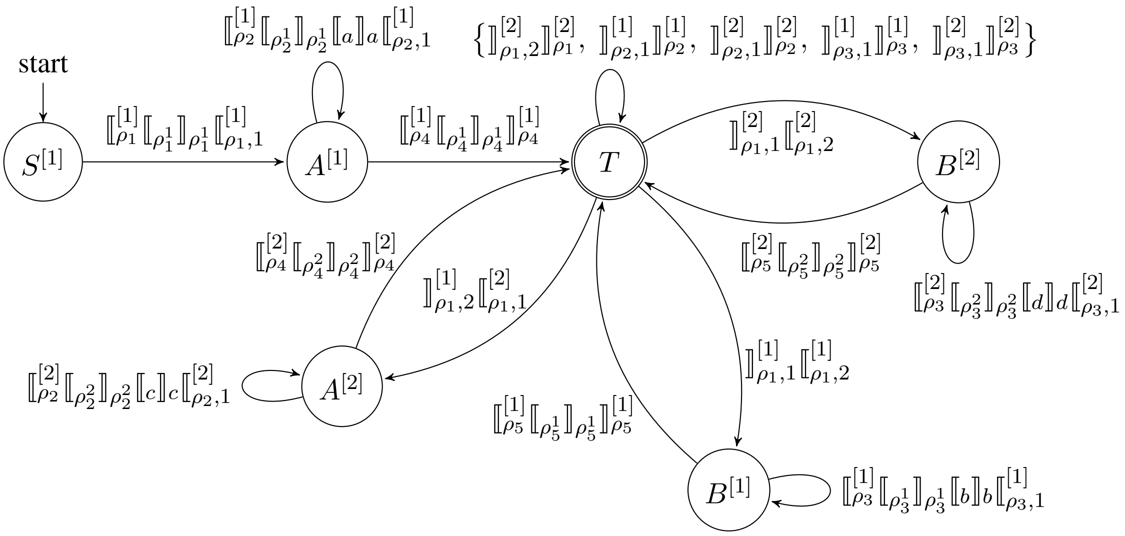

Figure 2: Automaton R obtained from G' (cf. Exs. 4 and 17) by Lem. 15 and Cor. 16.

Lemma 18.

$$\alpha\mathrm{HOM}(\mathcal{A}) \circ \alpha\mathrm{HOM} = \alpha\mathrm{HOM}(\mathcal{A})$$

Proof. ($\subseteq$) Let $h_1 \colon \Delta^* \to \mathcal{A}^{\Gamma^*} \in \alpha\mathrm{HOM}(\mathcal{A})$ and $h_2 \colon \Sigma^* \to \Delta^* \in \alpha\mathrm{HOM}$. By the definitions of $\alpha\mathrm{HOM}(\mathcal{A})$ and $\alpha\mathrm{HOM}$ there must exist $h_1' \colon \Delta \to \mathcal{A}^{\Gamma \cup \{\varepsilon\}}$ and $h_2' \colon \Sigma \to \Delta \cup \{\varepsilon\}$ such that $\widehat{h_1'} = h_1$ and $\widehat{h_2'} = h_2$. Since $h_1(\mathrm{rng}(h_2')) \subseteq \mathcal{A}^{\Gamma \cup \{\varepsilon\}}$ there is some $h \in \alpha\mathrm{HOM}(\mathcal{A})$ such that $h = h_1 \circ h_2$; hence $h_1 \circ h_2 \in \alpha\mathrm{HOM}(\mathcal{A})$.

($\supseteq$) Follows from the fact that $\alpha\mathrm{HOM}$ contains the identity. ■

Theorem 19. Let L be an $\mathcal{A}$-weighted language and $k \in \mathbb{N}$. The following are equivalent:

(i) $L \in$ k-MCFL($\mathcal{A}$)

(ii) there are an $\mathcal{A}$-weighted alphabetic homomorphism h, a regular language R, and a congruence multiple Dyck language mD_c of dimension at most k with $L = h(R \cap mD_\mathrm{c})$.

Proof. For (i) $\Rightarrow$ (ii): There are some $L' \in$ k-MCFL, $h, h_1 \in \alpha\mathrm{HOM}(\mathcal{A})$, $h_2 \in \alpha\mathrm{HOM}$, $mD_\mathrm{c} \in$ k-mDYCK$_\mathrm{c}$, and $R \in \mathrm{REG}$ such that

$$\begin{aligned} L &= h_1(L') && \text{(by Lem. 15)} \\ &= h_1(h_2(R \cap mD_\mathrm{c})) && \text{(by Cor. 16)} \\ &= h(R \cap mD_\mathrm{c}) && \text{(by Lem. 18)} \end{aligned}$$

For (ii) $\Rightarrow$ (i): We use Lems. 11 and 15, and the closure of k-MCFG under intersection with regular languages and application of homomorphisms. ■

5 Conclusion and outlook

We defined multiple Dyck languages using congruence relations (Def. 7), gave an algorithm to decide whether a word is in a given multiple Dyck language (Alg. 1), and established that multiple Dyck languages with increasing maximal dimension form a hierarchy (Prop. 13).

We obtained a weighted version of the CS representation of MCFLs for complete commutative strong bimonoids (Thm. 19) by separating the weights from the weighted MCFG and using Yoshinaka et al. (2010, Thm. 3) for the unweighted part.

Thm. 19 may be used to develop a parsing algorithm for weighted multiple context-free grammars in the spirit of Hulden (2011).

References

Noam Chomsky and Marcel-Paul Schützenberger. 1963. The algebraic theory of context-free languages. *Computer Programming and Formal Systems, Studies in Logic*, pages 118–161.

Manfred Droste and Heiko Vogler. 2013. The Chomsky-Schützenberger theorem for quantitative context-free languages. In Marie-Pierre Béal and Olivier Carton, editors, *Developments in Language Theory*, volume 7907 of *Lecture Notes in Computer Science*, pages 203–214. Springer.

Manfred Droste, Torsten Stüber, and Heiko Vogler. 2010. Weighted finite automata over strong bimonoids. *Information Sciences*, 180(1):156–166.

John E. Hopcroft and Jeffrey D. Ullman. 1979. *Introduction to automata theory, languages and computation*. Addison-Wesley, 1st edition.

Mans Hulden. 2011. Parsing CFGs and PCFGs with a Chomsky-Schützenberger representation. In Zygmunt Vetulani, editor, *Human Language Technology. Challenges for Computer Science and Linguistics*, volume 6562 of *Lecture Notes in Computer Science*, pages 151–160.

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

Makoto Kanazawa. 2014. Multidimensional trees and a Chomsky-Schützenberger-Weir representation theorem for simple context-free tree grammars. *Journal of Logic and Computation*.

Jens Michaelis. 2001a. Derivational minimalism is mildly context–sensitive. In Michael Moortgat, editor, *Logical Aspects of Computational Linguistics*, volume 2014 of *Lecture Notes in Computer Science*, pages 179–198. Springer.

Jens Michaelis. 2001b. Transforming linear context-free rewriting systems into minimalist grammars. In Philippe Groote, Glyn Morrill, and Christian Retoré, editors, *Logical Aspects of Computational Linguistics*, volume 2099 of *Lecture Notes in Computer Science*, pages 228–244. Springer.

Arto Salomaa. 1973. *Formal languages*. Academic Press.

Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88(2):191–229.

Hiroyuki Seki, Ryuichi Nakanishi, Yuichi Kaji, Sachiko Ando, and Tadao Kasami. 1993. Parallel multiple context-free grammars, finite-state translation systems, and polynomial-time recognizable subclasses of lexical-functional grammars. In *Proceedings of the 31st Annual Meeting on Association for Computational Linguistics*, pages 130–139.

Krishnamurti Vijay-Shanker, David Jeremy Weir, and Aravind K Joshi. 1986. Tree adjoining and head wrapping. In *Proceedings of the 11th International Conference on Computational Linguistics*, pages 202–207.

Krishnamurti Vijay-Shanker. 1987. *A study of tree adjoining grammars*. Ph.D. thesis.

David Jeremy Weir and Arvind K. Joshi. 1988. Combinatory categorial grammars: Generative power and relationship to linear context-free rewriting systems. In *Proceedings of the 26th annual meeting on Association for Computational Linguistics*, pages 278–285.

Ryo Yoshinaka, Yuichi Kaji, and Hiroyuki Seki. 2010. Chomsky-Schützenberger-type characterization of multiple context-free languages. In Adrian-Horia Dediu, Henning Fernau, and Carlos Martín-Vide, editors, *Proceedings of the 4th International Conference on Language and Automata Theory and Applications*, volume 6031 of *Lecture Notes in Computer Science*, pages 596–607. Springer.

A Supplemental definitions

Let G be an MCFG and A a non-terminal in G.

A *subderivation in G* is a derivation in the underlying context-free grammar of G that does not necessarily start with an initial non-terminal.

The *set of subderivations in G from A*, denoted by $D_G(A)$, is the set of subderivations in G that start with the non-terminal A.

B Supplementals to the proof of Lem. 5

Observation 20. $\widehat{g}$, obtained by position-wise application of g, is a tree homomorphism. ■

Claim 21. $\widehat{g}$ is a bijection.

Proof. We show that $\widehat{g}$ is bijective by induction on the structure of subderivations:

Induction hypothesis: For every $A \in N$ and $\Psi \in M(A)$: $\widehat{g}$ is a bijection between $D_{G'}(A[\Psi])$ and $D_G(A)$.

Induction step: Let $\Psi \in M(A)$ and $d = \rho(d_1, \ldots, d_k) \in D_G(A)$ with $d_1 \in D_G(A_1), \ldots, d_k \in D_G(A_k)$. The construction defines $\Psi_1 \in M(A_1), \ldots, \Psi_k \in M(A_k)$ and a production ρ' which is unique for every ρ and Ψ. By the induction hypothesis, we know that there are derivations $d'_1, \ldots, d'_k$ which are unique for $(d_1, \Psi_1), \ldots, (d_k, \Psi_k)$, respectively. Therefore, $d' = \rho(d'_1, \ldots, d'_k)$ is unique for d and Ψ. Hence for every Ψ: $\widehat{g}$ induces a bijection on $D_{G'}(A[\Psi])$ and $D_G(A)$. By construction, all elements of I' have the form $A[\emptyset]$ for some $A \in I$; hence for every element of D'_G we set $\Psi = \emptyset$ and by the induction hypothesis we obtain Cl. 21. ■

Claim 22. $\mu' = \mu \circ \widehat{g}$

Proof. Since $\widehat{g}$ is a tree homomorphism (cf. Obs. 20), it preserves the tree structure. By the definition of μ' we obtain Cl. 22. ■

C Supplementals to the proof of Lem. 11

Claim 23. $g(L(G_{\mathfrak{P}})) = L$.

Proof. Let $\mathrm{Tup}_g(G_{\mathfrak{P}})$ be the set of tuples that are obtained by interpreting the terms corresponding to every subderivation in $G_{\mathfrak{P}}$ and then applying g to every component. We show the following equivalence by induction:

$$\begin{aligned} & w \in mD(\mathfrak{P}) \\ \iff & \forall \ell \in \mathbb{N}, u_0, \ldots, u_\ell, w_1, \ldots, w_\ell \in D(\Sigma) \\ & \text{with } w = u_0 w_1 u_1 \cdots w_\ell u_\ell \colon \\ & (u_0, w_1, u_1, \cdots, w_\ell, u_\ell) \in \mathrm{Tup}_g(G_{\mathfrak{P}}) \end{aligned} \tag{IH}$$

Note that in the following the indices in $\mathfrak{p} = \{\sigma_1, \ldots, \sigma_\ell\}$ are chosen such that for every $i \in [\ell]$: $g(\mathfrak{p}^{[i]}) = \sigma_i$. We derive

$$\begin{aligned} & w \in mD(\mathfrak{P}) \\ \iff & \forall \ell \in \mathbb{N}, u_0, \ldots, u_\ell, v_1, \ldots, v_\ell \in D(\Sigma), \\ & \mathfrak{p} = \{\sigma_1, \ldots, \sigma_\ell\} \in \mathfrak{P} \text{ with} \\ & w = u_0 \sigma_1 v_1 \overline{\sigma_1} u_1 \cdots \sigma_\ell v_\ell \overline{\sigma_\ell} u_\ell \colon \\ & u_0 v_1 u_1 \cdots v_\ell u_\ell \in mD(\mathfrak{P}) \\ & \text{(by def. of } mD(\mathfrak{P})) \\ \iff & \forall \ell \in \mathbb{N}, u_0, \ldots, u_\ell, v_1, \ldots, v_\ell \in D(\Sigma), \\ & \mathfrak{p} = \{\sigma_1, \ldots, \sigma_\ell\} \in \mathfrak{P} \text{ with} \\ & w = u_0 \sigma_1 v_1 \overline{\sigma_1} u_1 \cdots \sigma_\ell v_\ell \overline{\sigma_\ell} u_\ell \colon \\ & (u_0, v_1, u_1, \ldots, v_\ell, u_\ell) \in \mathrm{Tup}_g(G_{\mathfrak{P}}) \\ & \text{(by (IH))} \\ \iff & \forall \ell \in \mathbb{N}, u_0, \ldots, u_\ell, v_1, \ldots, v_\ell \in D(\Sigma), \\ & \mathfrak{p} = \{\sigma_1, \ldots, \sigma_\ell\} \in \mathfrak{P} \text{ with} \\ & w = u_0 \sigma_1 v_1 \overline{\sigma_1} u_1 \cdots \sigma_\ell v_\ell \overline{\sigma_\ell} u_\ell \colon \\ & (u_0, \sigma_1 v_1 \overline{\sigma_1}, u_1, \ldots, \sigma_\ell v_\ell \overline{\sigma_\ell}, u_\ell) \in \mathrm{Tup}_g(G_{\mathfrak{P}}) \\ & \text{(by def. of } G_{\mathfrak{P}}) \\ \iff & \forall \ell \in \mathbb{N}, u_0, \ldots, u_\ell, w_1, \ldots, w_\ell \in D(\Sigma) \\ & \text{with } w = u_0 w_1 u_1 \cdots w_\ell u_\ell \colon \\ & (u_0, w_1, u_0, \ldots, w_\ell, u_\ell) \in \mathrm{Tup}_g(G_{\mathfrak{P}}) \\ & \text{(using permuting productions in } G_{\mathfrak{P}}) \end{aligned}$$

Cl. 23 follows by instantiating (IH) for $\ell = 0$ and discovering that $\{t \mid (t) \in \mathrm{Tup}_g(G_{\mathfrak{P}})\} = g(L(G_{\mathfrak{P}}))$. ■

D Supplementals to Alg. 1

Alg. 2 implements SPLIT from Alg. 1 (lines 30–33) in an explicit manner.

For this purpose we define a data structure *pushdown* as a string over some alphabet and two functions with side-effects on pushdowns. Let Γ be an alphabet, $\gamma \in \Gamma$, and $pd \subseteq \Gamma^*$ be a pushdown.

- $\mathrm{pop}(pd)$ returns the left-most symbol of pd and removes it from pd.
- $\mathrm{push}(pd, \gamma)$ prepends γ to pd.

Algorithm 2 Algorithm to split a word in $D(\Sigma)$ into shortest non-empty strings from $D(\Sigma)$

Input: alphabet Σ, $w \in D(\Sigma)$
Output: sequence $(u_1, \dots, u_\ell)$ of shortest words $u_1, \dots, u_\ell \in D(\Sigma) \setminus \{\varepsilon\}$ with $w = u_1 \cdots u_\ell$

1: **function** SPLIT'(Σ, w)
2: $\quad pd \leftarrow \varepsilon$
3: $\quad j \leftarrow 1$
4: $\quad u_j \leftarrow \varepsilon$
5: $\quad$ **for** $0 \leq i \leq |w|$ **do**
6: $\quad\quad u_j \leftarrow u_j w_i$
7: $\quad\quad$ **if** $w_i \in \overline{\Sigma}$ **then**
8: $\quad\quad\quad \mathrm{pop}(pd)$
9: $\quad\quad\quad$ **if** $pd = \varepsilon$ **then**
10: $\quad\quad\quad\quad j \leftarrow j + 1$
11: $\quad\quad\quad\quad u_j \leftarrow \varepsilon$
12: $\quad\quad\quad$ **end if**
13: $\quad\quad$ **else**
14: $\quad\quad\quad \mathrm{push}(pd, \overline{w_i})$
15: $\quad\quad$ **end if**
16: $\quad$ **end for**
17: $\quad$ **return** $(u_1, \dots, u_{j-1})$
18: **end function**

Note that pop() is only a partial function, it is undefined for $pd = \varepsilon$. But since the input word w is in $D(\Sigma)$, the expression on line 8 is always defined.

Tab. 2 shows the run of Alg. 1 on the word $⟦()⟧[]⟦⟧[\langle\rangle]$.

E Supplementals to the proof of Lem. 15

Claim 24. There are bijections $f\colon D_G \to D_{G'}$ and $g\colon D_{G'} \to L(G')$.

Proof. Let f be the function that is obtained by applying the construction position-wise to a derivation in D_G. The function f only inserts symbols into the functions in the productions; by removing these elements, we get the original function, hence f is bijective.

Let $g\colon D_{G'} \to L(G')$ be the function that assigns for every derivation $d \in D_{G'}$ the word in $L(G')$ obtained by interpreting the term corresponding to d. For every $w \in L(G')$ we can calculate the corresponding derivation (as a tree with domain $\mathrm{dom}(t)$ and labelling function t) using Alg. 3, hence g is bijective. ■

Table 2: Run of Alg. 1 on the word $⟦()⟧[]⟦⟧[\langle\rangle]$.

MAIN$(\Sigma, \mathfrak{P}, ⟦()⟧[]⟦⟧[\langle\rangle])$
l. 4: $\sigma_1 = ⟦ = \sigma_3, \sigma_2 = [= \sigma_4,$
$\quad u_1 = (), u_2 = \varepsilon = u_3, u_4 = \langle\rangle$
l. 5: $\mathcal{I} = \big\{\{\{1,2\},\{3,4\}\}, \{\{1,4\},\{2,3\}\}\big\}$

l. 6: $I = \big\{\{1,2\},\{3,4\}\big\}$
l. 8: $k = 2, i_1 = 1, i_2 = 2$
l. 9: $b = 1 \cdot$ MAIN$(\Sigma, \mathfrak{P}, ())$
$\quad\quad$ l. 4: $\sigma_1 = (, u_1 = \varepsilon$
$\quad\quad$ l. 5: $\mathcal{I} = \emptyset$
$\quad\quad$ l. 13: **return** 0
l. 9: $b = 1 \cdot 0 = 0$
l. 8: $k = 2, i_1 = 3, i_2 = 4$
l. 9: $b = 0 \cdot$ MAIN$(\Sigma, \mathfrak{P}, \langle\rangle)$
$\quad\quad$ l. 4: $\sigma_1 = \langle, u_1 = \varepsilon$
$\quad\quad$ l. 5: $\mathcal{I} = \emptyset$
$\quad\quad$ l. 13: **return** 0
l. 9: $b = 0 \cdot 0 = 0$

l. 6: $I = \big\{\{1,4\},\{2,3\}\big\}$
l. 8: $k = 2, i_1 = 1, i_2 = 4$
l. 9: $b = 1 \cdot$ MAIN$(\Sigma, \mathfrak{P}, ()\langle\rangle)$
$\quad\quad$ l. 4: $\sigma_1 = (, \sigma_2 = \langle, u_1 = \varepsilon = u_2$
$\quad\quad$ l. 5: $\mathcal{I} = \big\{\{\{1,2\}\}\big\}$
$\quad\quad$ l. 6: $I = \big\{\{1,2\}\big\}$
$\quad\quad$ l. 8: $b = 1 \cdot$ MAIN$(\Sigma, \mathfrak{P}, \varepsilon)$
$\quad\quad\quad\quad$ l. 2: **return** 1
$\quad\quad$ l. 8: $b = 1 \cdot 1$
$\quad\quad$ l. 11: **return** 1
l. 9: $b = 1 \cdot 1$
l. 8: $k = 2, i_1 = 2, i_2 = 3$
l. 9: $b = 1 \cdot$ MAIN$(\Sigma, \mathfrak{P}, \varepsilon)$
$\quad\quad$ l. 2: **return** 1
l. 9: $b = 1 \cdot 1$

l. 11: **return** 1

Claim 25. For every $d \in D_G$ and $w \in \Delta^*$:

$$(h \circ g \circ f)(d)(w) = \begin{cases} \mu(d) & \text{if } d \in D_G(w), \\ 0 & \text{otherwise.} \end{cases}$$

Proof. Follows directly from the definitions of f, g, and h. ■

Claim 26. $⟦G⟧ = h(L(G'))$.

Proof. For every $w \in \Delta^*$:

$$\begin{aligned} L(w) &= ⟦G⟧(w) \\ &= \textstyle\sum_{d \in D_G(w)} \mu(d) \\ &= \textstyle\sum_{d \in D_G} (h \circ g \circ f)(d)(w) \quad \text{(by Cl. 25)} \end{aligned}$$

Algorithm 3 Algorithm to calculate for every word in $L(G')$ the corresponding derivation in $D_{G'}$ (cf. Cl. 24)

Input: $w \in (\Delta')^*$
Output: $t \colon \mathbb{N}^* \to P$

procedure MAIN($w \in (\Delta')^*$)
  let t be the empty function
  DESCEND($\varepsilon, 1$)
  return t
end procedure

procedure DESCEND($\pi \in \mathbb{N}^*, j \in \mathbb{N}$)
  $\rho^j u \leftarrow w$ where $\rho \in P$ and $u \in (\Delta')^*$
  $t(\pi) \leftarrow \rho$
  $w \leftarrow u$
  $A \to [u_1, \ldots, u_s](A_1, \ldots, A_k) \leftarrow \rho$
  for every symbol δ' in u_j **do**
    if $\delta' \in \Delta$ **then**
      remove δ' from the beginning of w
    else
      $x_i^{j'} \leftarrow \delta'$ for some i and j'
      DESCEND($\pi i, j'$)
    end if
  end for
end procedure

$$
\begin{aligned}
&= \textstyle\sum_{\substack{d \in D_G, u \in L(G') \\ u = (g \circ f)(d)}} h(u)(w) \\
&= \textstyle\sum_{u \in L(G')} h(u)(w) && \text{(by Cl. 24)} \\
&= h(L(G'))(w) && \blacksquare
\end{aligned}
$$