Regulation by Valences

Henning Fernau
Wilhelm-Schickhard-Institut für Informatik
Universität Tübingen
Sand 13, D-72076 Tübingen, Germany
email: fernau@informatik.uni-tuebingen.de

Ralf Stiebe
Institut für Informatik
Martin-Luther-Universität Halle-Wittenberg
Kurt-Mothes-Str. 1, D-06120 Halle, Germany
email: stiebe@informatik.uni-halle.de

Abstract

Valences are a very simple and yet powerful method of regulated rewriting. In this paper we give an overview on different aspects of this subject. We discuss closure properties of valence languages. It is shown that matrix grammars can be simulated by valence grammars over finite monoids. A Chomsky normal form theorem is proved for multiplicative valence grammars, thereby solving the open question of the existence of normal forms for unordered vector grammars. This also gives an alternative proof of the inclusion of context-free unordered vector languages in LOG(CFL). Moreover, we investigate valences in parallel systems, thereby solving part of open problems posted in [5, p. 267].

Remark: A short version of this paper appeared in the proceedings of the 22nd MFCS conference 1997, see [6].

1 Introduction

Valences were introduced in 1980 by Păun [15] as a mechanism of regulated rewriting. The original idea was to assign to a context-free core rule an integer, the valence, and to compute for a derivation a value by adding the valences of the applied rule. A derivation is valid iff this sum evaluates to 0, reflecting the balance of positive and negative valences in chemical molecules or in directed graphs. This mechanism can be easily extended to monoids different from $(\mathbb{Z}, +, 0)$.

Valence languages have been in the focus of several papers [7, 12, 13, 16, 17, 24, 25]. We think for several reasons that valence grammars are worth a deeper investigation. First of all, valences are a very natural and simple mechanism. The context-free derivation process is not changed at all; the validity of a derivation is only checked at the end. Thus, many attractive properties of context-free grammars can immediately be transferred. Moreover, it is possible to describe several language families by valence grammars over different monoids, and one can hope to simplify investigations concerning these families by studying the respective valence grammars. For example, unordered vector languages can be characterized via valence languages over the monoid of positive rational numbers with multiplication, and Greibach's family *BLIND* of languages accepted by blind multi-counter automata can be generated by regular valence grammars over $(\mathbb{Q}_+, \cdot, 1)$. Finally, valences are very flexible and can be incorporated into parallel systems, grammars with other means of regulation and machine models (in fact, "finite valence automata" were discussed by several authors even before the introduction of valence grammars, e.g., in [8, 11, 18]).

This paper discusses several of the mentioned aspects. After giving the necessary notations we discuss valence languages over arbitrary monoids. The known results regarding closure properties for specific monoids are extended and generalized, while the proofs are simplified. It is shown that valence grammars over finite monoids are equivalent to matrix grammars. We show how to construct normal forms for valence grammars over $(\mathbb{Q}_+, \cdot, 1)$, a result which also applies to the equivalent class of unordered vector grammars. Finally, we investigate several variants of valences in parallel systems. For reasons of space, we can only list further interesting connections: we showed that valence languages over the monoid of integers correspond to homomorphic images of vertex languages of edge grammars as introduced in [3]. Taking Abelian monoids as valence regulators, we found algebraic characterizations of the corresponding language families in the spirit of [1, 2].

2 Preliminaries

Throughout the paper we assume the reader to be familiar with the theory of context-free languages, see e.g., [21].

Let $V = \{a_1, \ldots, a_n\}$, $n \geq 1$, be an alphabet. The set of all words over V is denoted by V^*, the empty word by λ, and $V^+ = V^* \backslash \{\lambda\}$. For $w \in V^*$, the length of w is denoted by $|w|$, the number of appearances of the letter $a \in V$ in w is denoted by $|w|_a$. The *Parikh mapping* associated with V is a map $\Psi : V^* \to \mathbb{N}^n$ such that $\Psi(w) = (|w|_{a_1}, \ldots, |w|_{a_n})$. For a language $L \subseteq V^*$, we define the *Parikh set of* L by $\Psi(L) = \{\Psi(w) : w \in L\}$. Two languages $L_1, L_2 \subseteq V^*$ are called *letter equivalent* iff their Parikh sets are equal. For a word w, let $\mathrm{Perm}(w)$ denote the set of all words obtained by permuting the symbols of w. For a language L, we define $\mathrm{Perm}(L) = \bigcup_{w \in L} \mathrm{Perm}(w)$.

Let G be a grammar. For a derivation Δ applying the rules $p_1, p_2, \ldots, p_n$ (in this sequence), the control word of Δ is defined as $c(\Delta) = p_1 p_2 \ldots p_n$.

A *valence grammar* (over the monoid $\mathbf{M} = (M, \circ, e)$) is a construct $G = (N, T, P, S, \varphi, \mathbf{M})$, where $G' = (N, T, P, S)$ is a context-free grammar, and $\varphi : P \to M$ is the valence mapping (which can be extended to a monoid morphism from P^* to M). To avoid explicit reference to the monoid, we write $\mathrm{Lab}(G)$ instead of $\varphi(P)$. P is called the set of core rules; a pair (p, m), $p = (A, \alpha)$, $m = \varphi(p)$ is called a valence rule and usually denoted by $(m : A \to \alpha)$. The yield relation $\Rightarrow$ over $(N \cup T)^* \times M$ is defined as: $(w, m) \Rightarrow (w', m')$ iff there is a rule $(n : A \to \alpha)$ such that $w = w_1 A w_2$ and $w' = w_1 \alpha w_2$ and $m' = m \circ n$. The language generated by G is $L(G) = \{w \in T^* : (S, e) \stackrel{*}{\Rightarrow} (w, e)\}$. A valence grammar is called regular if all its core rules are regular; a valence grammar is λ-free if it has no core rule of the form $A \to \lambda$. The language families generated by context-free, context-free λ-free, and regular, resp., valence grammars over $\mathbf{M}$, are denoted by $\mathcal{L}(\mathrm{Val}, \mathrm{CF}, \mathbf{M}), \mathcal{L}(\mathrm{Val}, \mathrm{CF} - \lambda, \mathbf{M}), \mathcal{L}(\mathrm{Val}, \mathrm{REG}, \mathbf{M})$, respectively. For brevity, let $\mathbb{Z}^0$ denote the trivial monoid. Then, $\mathcal{L}(\mathrm{Val}, X, \mathbb{Z}^0) = \mathcal{L}(X)$ for $X \in \{\mathrm{REG}, \mathrm{CF}[-\lambda]\}$.

In terms of control words, a derivation in the underlying context-free grammar is valid in a valence grammar iff its control word is mapped by the valence morphism to the neutral element. Next, we define some other regulation mechanisms depending on control words, and hence related to valence grammars. A *matrix grammar* is a quintuple $G = (N, T, P, S, M)$ where $G' = (N, T, P, S)$ is a context-free grammar, and $M \subset P^*$ is a finite set of matrices. A terminal derivation Δ in G' is valid in G iff $c(\Delta) \in M^*$. $L(G)$ consists of all words obtained by valid derivations. An *unordered vector grammar* is defined like a matrix grammar, with the difference that a terminal derivation Δ in G' is valid in G iff $c(\Delta) \in \mathrm{Perm}(M^*)$. The families of unordered vector languages of type

$X \in \{\mathrm{REG}, \mathrm{CF}[-\lambda]\}$ are denoted by $\mathcal{L}(\mathrm{UV}, X)$. A *grammar with regular control* is a quintuple $G = (N, T, P, S, R)$, where (N, T, P, S) is a context-free grammar, and R is a regular subset of P^*. A terminal derivation Δ is valid iff $c(\Delta) \in R$. Grammars with regular control are equivalent to matrix grammars, [5]. The corresponding language class is denoted by $\mathcal{L}(\mathrm{rC}, \mathrm{CF}[-\lambda])$.

An important tool for proving closure properties is the notion of a finite transducer which is defined next. A *finite transducer* is a sixtuple

$$\mathcal{A} = (Z, I, O, z_0, \delta, z_f),$$

consisting of the finite set of states Z, the input and output alphabets I and O, the initial state $z_0 \in Z$, the finite transition relation $\delta \subset I^* \times Z \times Z \times O^*$, and the final state $z_f \in Z$. $\mathcal{A}$ is called λ-free iff $\delta \subset I^* \times Z \times Z \times O^+$.

The yield relation $\models$ over $I^* \times Z \times O^*$ is defined as: $(w_{in}, z, w_{out}) \models (w'_{in}, z, w'_{out})$ iff $w_{in} = xw'_{in}$ with $x \in I^*$, $w'_{out} = w'_{out}u$ with $u \in O^*$, and $(x, z, z', u) \in \delta$.

For $w \in I^*$, the transduced image of w under $\mathcal{A}$ is defined as $T_{\mathcal{A}}(w) = \{w' \in O^* : (w, z_0, \lambda) \models^* (\lambda, z_f, w')\}$. For a language $L \subseteq I^*$, the transduced image is $T_{\mathcal{A}}(L) = \bigcup_{w \in L} T(w)$. The operator $T_{\mathcal{A}}$ on languages is also called *rational transduction*.

We briefly mention two well-known facts on finite transducers:

1. A language family is a full trio (or trio, resp.) iff it is closed under rational transductions (or λ-free rational transductions, resp.).

2. Every rational transduction can be defined by a finite transducer in *normal form*, i.e., with transition relation $\delta \subset (I \cup \{\lambda\}) \times Z \times Z \times (O \cup \{\lambda\})$.

Let us further mention without proof that every λ-free rational transduction is representable as the composition $T = T_2 T_1$ of a λ-free "normal form" transducer $\mathcal{A}_1$ followed by a restricted erasing T_2, where a *k-restricted erasing* is a finite-state transduction τ which realizes the morphism $g_X : (T \cup \{\$\})^* \to X^*$ (where $\$ \notin X$), given by $a \mapsto a$ for $a \in X$ and $\$ \mapsto \lambda$, on the domain

$$\mathrm{dom}(\tau) = (\bigcup_{i=0}^{k} \{\$^i\} X)^+ .$$

In analogy to valence grammars, one can define valence transducers, where a valence is assigned to each element of the valence relation, and a transduction is valid iff the valence product evaluates to the neutral element.

We close this section with some conventions used in this paper. The inclusion relation is denoted by $\subseteq$, proper inclusion by $\subset$. If $\mathbf{M} = (M, \circ, e)$ is a monoid and $M' \subseteq M$, the monoid generated by M' is denoted by $\langle M' \rangle$. The monoids $(\mathbb{Z}^k, +, \vec{0})$ and $(\mathbb{Q}_+, \cdot, 1)$ are sometimes simply denoted by $\mathbb{Z}^k$ and $\mathbb{Q}_+$.

For a vector $\vec{r} = (r_1, r_2 \ldots, r_k) \in \mathbb{Z}^k$, we define the max-norm by $||\vec{r}||_{\max} = \max\{|r_i| : 1 \leq i \leq k\}$ and the 1-norm by $||\vec{r}||_1 = \sum_{i=1}^{k} |r_i|$. We define the modulo and integer division operations for vectors, $\mathrm{mod}, \mathrm{div} : \mathbb{Z}^k \times \mathbb{Z} \to \mathbb{Z}^k$, as the component-wise application of the integer operations $\mathrm{mod}, \mathrm{div} : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$, and denote them by $\vec{r} \bmod m$ and $\vec{r} \operatorname{div} m$, for $\vec{r} \in \mathbb{Z}^k, m \in \mathbb{Z}$.

3 Valence grammars over monoids

Up to now, valence grammars have been investigated only for a few specific groups. In this section, we are first going to prove some general closure properties of classes of valence languages over arbitrary monoids. Later we discuss valence grammars over particular monoids: the monoid $(\mathbb{Q}_+, \cdot, 1)$, Abelian monoids and finite monoids.

In short, valence language classes form semi-AFLs, i.e., they are closed under union and finite transducer mappings, and this fact can be shown quite generally, not requiring a separate proof for each monoid, as done in [13, 25].

Theorem 3.1 *For each monoid* $\mathbf{M}$ *and each* $X \in \{\mathrm{REG}, \mathrm{CF}[-\lambda]\}$*, the class* $\mathcal{L}(\mathrm{Val}, X, \mathbf{M})$ *is a semi-AFL. Moreover,* $\mathcal{L}(\mathrm{Val}, X, \mathbf{M})$ *is closed under mirror image and substitution by* $\mathcal{L}(X)$*-languages.*

Proof. The triple constructions known for the basic Chomsky families showing closure under rational transductions given by finite transducers in normal form can be adapted for the families of valence languages. Furthermore, the "block coding technique" used for showing closure under restricted erasing can be adapted for our purposes, too.

Thus $\mathcal{L}(\mathrm{Val}, X, \mathbf{M})$ is closed under λ-free transductions for $X = \mathrm{CF} - \lambda$, and closed under arbitrary transducer mappings for $X \in \{\mathrm{REG}, \mathrm{CF}\}$.

As regards the other properties, namely closure under union, mirror and $\mathcal{L}(X)$-substitutions, the standard constructions known from the theory of context-free languages can easily be carried over. Analogous results for finite automata with valences can be found in [14]. □

The classes of valence languages over $\mathbb{Q}_+$ are, additionally, closed under catenation. Another interesting closure property is:

Theorem 3.2 *The families* $\mathcal{L}(\mathrm{Val}, X, \mathbb{Q}_+)$*,* $X \in \{\mathrm{REG}, \mathrm{CF}\}$*, are closed under valence transductions over* $\mathbb{Q}_+$*.*

Proof. First note that, for a valence grammar (transducer) over $\mathbb{Q}_+$, exists an equivalent valence grammar (transducer) over some $\mathbb{Z}^k$ for some $k \geq 0$, and vice versa. Let G be a valence grammar over $(\mathbb{Z}^k, +, \vec{0})$ and $\mathcal{A}$ be a valence transducer over $(\mathbb{Z}^l, +, \vec{0})$. Again, the triple construction for the classical language families can be modified to obtain a valence grammar H over $(\mathbb{Z}^{k+l}, +, \vec{0})$ such that $L(H) = T_{\mathcal{A}}(L(G))$. Note that the commutativity of addition is essential for the construction. □

Corollary 3.3 *Each of the families* $\mathcal{L}(\mathrm{Val}, X, \mathbb{Q}_+)$, $X \in \{\mathrm{CF}, \mathrm{REG}\}$, *is closed under permutation and under intersection by languages from* $\mathcal{L}(\mathrm{Val}, \mathrm{REG}, \mathbb{Q}_+)$.

This generalizes the older result that *BLIND* is closed under intersection [8].
Proof. It can easily be shown that the mentioned operations are λ-free valence transductions. □

Next we are going to show that the generative power of valence grammars is not improved when admitting arbitrary Abelian valence monoids instead of $(\mathbb{Q}_+, \cdot, 1)$.

Theorem 3.4 *Let* $\mathbf{M}$ *be an Abelian monoid and* $X \in \{\mathrm{REG}, \mathrm{CF}[-\lambda]\}$. *Then the class* $\mathcal{L}(\mathrm{Val}, X, \mathbf{M})$ *equals either* $\mathcal{L}(\mathrm{Val}, X, \mathbb{Q}_+)$ *or* $\mathcal{L}(\mathrm{Val}, X, \mathbb{Z}^k)$, *for some* $k \geq 0$.

Proof. First consider an arbitrary monoid $\mathbf{M} = (M, \circ, e)$. An element m of a sub-monoid $\mathbf{M}'$ is called completable in $\mathbf{M}'$ iff there are m_1, m_2 in $\mathbf{M}'$ such that $m_1 \cdot m \cdot m_2 = e$. Obviously, any valence grammar G over $\mathbf{M}$ is a valence grammar over the finitely generated monoid $\langle \mathrm{Lab}(G) \rangle$. Note that for a successful derivation only valence rules labeled by elements completable in $\langle \mathrm{Lab}(G) \rangle$ are of interest. We can thus remove the valence rules labeled by non-completable elements without changing the generated language. By iterating this procedure, we finally obtain a valence grammar G', equivalent to G, such that all elements of $\mathrm{Lab}(G')$ are completable in $\langle \mathrm{Lab}(G') \rangle$. If $\mathbf{M}$ is Abelian, then $\langle \mathrm{Lab}(G') \rangle$ is an Abelian group. By the well-known fundamental theorem for finitely generated Abelian groups, see e.g., [19], $\langle \mathrm{Lab}(G') \rangle$ is isomorphic to some $\mathbf{M} \times \mathbb{Z}^k$, where $\mathbf{M}$ is a finite Abelian group, and $k \geq 0$. For any valence grammar G over $\mathbf{M} \times \mathbb{Z}^k$, one can find by a triple construction an equivalent valence grammar over $\mathbb{Z}^k$. □

As mentioned in the definitions, regulation by valences is very similar to regulation by matrices and unordered vectors. This is stressed by proving that valence grammars over finite monoids are as powerful as matrix grammars.

Theorem 3.5 *For the family of all finite monoids* $\mathbf{FM}$ *and for* $X \in \{\mathrm{CF}, \mathrm{CF}-\lambda\}$, $\bigcup_{\mathbf{M} \in \mathbf{FM}} \mathcal{L}(\mathrm{Val}, X, \mathbf{M}) = \mathcal{L}(\mathrm{rC}, X)$.

Proof. The proof is based on the equivalence of matrix grammars and grammars with regular control. Let $\mathbf{M} = (M, \circ, e)$ be some finite monoid, and let $G = (N, T, P, S, \varphi, \mathbf{M})$ be a valence grammar over $\mathbf{M}$. One can easily construct a finite automaton $\mathcal{A}_{\mathbf{M}}$ accepting $m_1 m_2 \ldots m_k \in M$ iff $m_1 \circ m_2 \circ \ldots \circ m_k = e$, and starting with $\mathcal{A}_{\mathbf{M}}$, a finite automaton $\mathcal{A}_G$ accepting $\pi \in P^*$ iff $\varphi(\pi) = e$. The equivalent grammar with regular control is $G' = (N, T, P, S, L(\mathcal{A}_G))$.

On the other hand, let $G = (N, T, M, S)$ be a matrix grammar in 2-normal form, cf. [5]. This means, especially, that terminal words can only be derived after a complete application of a matrix. So, there is an equivalent grammar $G' = (N, T, P, S, R)$ with regular control such that $\pi \in P^*$ is in R iff it is syntactically equivalent to λ (i.e., iff R is accepted by a deterministic finite automaton whose only accepting state is the initial state). Hence, there is a straightforward construction of an equivalent valence grammar over the syntactic monoid of R. $\square$

4 Normal forms for valence grammars

Since the introduction of unordered vector grammars it has been an open question whether there are normal form representations. Sudborough [23] showed that, in analogy to the context-free case, the membership problem can be efficiently solved for unordered vector grammars without erasing and unit productions, more exactly, it was shown to be in LOG(CFL). Satta [22] could prove that complexity result for arbitrary unordered vector grammars, while leaving open the problem of the existence of normal forms. Our result yields an alternative proof for the relation $\mathcal{L}(\mathrm{Val}, \mathrm{CF}, \mathbb{Q}_+) \subseteq \mathrm{LOG(CFL)}$.

The general idea of the construction of Chomsky normal forms for valence grammars over $\mathbb{Z}^k$, $k \geq 1$ is the same as in the classical case. First λ-rules, then unit core rules are eliminated. Due to valences, these steps are harder to perform than in the plain context-free case, but many observations can be transferred. Additionally, the valence vectors are normalized.

Theorem 4.1 *For any valence grammar over $\mathbb{Z}^k$, there is an equivalent valence grammar over $\mathbb{Z}^k$ with core rules of the form $A \to BC$ or $A \to a$, and valence vectors with 1-norms not greater than 1.*

Proof. See Appendix A. $\square$

Note that the proof mainly depends on the commutativity of the monoid $(\mathbb{Z}^k, +, \vec{0})$. Thus it cannot be modified for valence grammars over, e.g., finite monoids (and hence for matrix grammars). This remains a challenging open question. In [5] it is shown that, for any valence grammar with unit or zero valence vectors, there is an equivalent unordered vector grammar with the same set of core rules. This implies

Corollary 4.2 *For any unordered vector grammar, there is an equivalent unordered vector grammar with core rules of the form $A \to BC$ or $A \to a$.*

As regards language families, we obtain via [23, 24]:

Corollary 4.3 *Let $k \geq 0$. Then, we have:*

$$\begin{array}{rcl cl}
\mathcal{L}(\mathrm{Val}, \mathrm{CF}-\lambda, \mathbb{Z}^k) & = & \mathcal{L}(\mathrm{Val}, \mathrm{CF}, \mathbb{Z}^k) & \subset & \mathcal{L}(\mathrm{Val}, \mathrm{CF}-\lambda, \mathbb{Z}^{k+1}) \\
 & = & \mathcal{L}(\mathrm{Val}, \mathrm{CF}, \mathbb{Z}^{k+1}) & \subset & \mathcal{L}(\mathrm{Val}, \mathrm{CF}-\lambda, \mathbb{Q}_+) \\
 & = & \mathcal{L}(\mathrm{Val}, \mathrm{CF}, \mathbb{Q}_+) & = & \mathcal{L}(\mathrm{UV}, \mathrm{CF}-\lambda) \\
 & = & \mathcal{L}(\mathrm{UV}, \mathrm{CF}) & \subset & \mathrm{LOG}(\mathrm{CFL}) .
\end{array}$$

5 Valences in parallel systems

Valences can also be attached to systems with parallel derivations. There are several modes to be discussed. In the plain mode, the validity of a derivation is checked in the end, while in the test mode the valence product has to be the neutral element after each parallel step. Finally, in table systems, the valences are attached to the tables (instead to individual rules).

We assume that the reader is familiar with the basics of Lindenmayer systems, see e.g., [20]. We use the common notation for variants of Lindenmayer systems, such as 0L, E0L, D0L, T0L, P0L, ET0L, and so on. For a derivation in an ET0L system $G = (\Sigma, \Delta, \mathbf{P}, \omega)$, the control word over $\mathbf{P}$ is defined as the sequence of applied tables; the Szilard language is the set of all control words of terminal derivations.

A valence ET0L system (over $\mathbb{Q}_+$), $G = (\Sigma, \Delta, \mathbf{P}, \omega)$ is defined similar to an ET0L system with the difference that the tables $P \in \mathbf{P}$ are finite sets of valence rules, i.e., $P \subset \Sigma \times \Sigma^* \times \mathbb{Q}_+$. As usual a valence rule (a, w, r) is denoted by $(r : a \to w)$. We say that the pair $(w, r) \in \Sigma^* \times \mathbb{Q}_+$ directly derives (w', r') in *plain mode* and write $(w, r) \Rightarrow_P (w', r')$ iff there is some $P \in \mathbf{P}$ such that $w = a_1 a_2 \ldots a_n$, $w' = (w_1 w_2 \ldots w_n)$, $r' = r_1 \cdot r_2 \cdot \ldots \cdot r_n \cdot r$, and $(r_i : a_i \to w_i) \in P$, for $1 \leq i \leq n$. The language generated by G in plain mode is $L(G) = \{w \in \Delta^* : (\omega, 1) \stackrel{*}{\Rightarrow}_\pi (w, 1), \pi \in P^*\}$. We say that the word $w \in \Sigma^*$ directly derives $w' \in \Sigma^*$ in *test mode* and write $w \Rightarrow_t w'$ iff there is some $P \in \mathbf{P}$ such that $w = a_1 a_2 \ldots a_n$, $w' = w_1 w_2 \ldots w_n$, $r_1 \cdot r_2 \cdot \ldots \cdot r_n = 1$, and $(r_i : a_i \to w_i) \in P$, for $1 \leq i \leq n$. The language generated by G in test mode is $L_t(G) = \{w \in \Delta^* : \omega \stackrel{*}{\Rightarrow}_t w\}$.

Finally, an *ET0L system with table valences* is a quintuple $G = (\Sigma, \Delta, \mathbf{P}, \omega, \varphi)$, where $G' = (\Sigma, \Delta, \mathbf{P}, \omega)$ is an ET0L system and φ is a mapping from $\mathbf{P}$ into $\mathbb{Q}_+$. We say that the pair $(w, r) \in \Sigma^* \times \mathbb{Q}_+$ directly derives (w', r') and write $(w, r) \Rightarrow_P (w', r')$ iff there is some $P \in \mathbf{P}$ such that $w = a_1 a_2 \ldots a_n$, $w' = (w_1 w_2 \ldots w_n)$, $r' = \varphi(P) \cdot r$, and $a_i \to w_i \in P$, for $1 \leq i \leq n$. The language generated by G is $L(G) = \{w \in \Delta^* : (\omega, 1) \stackrel{*}{\Rightarrow} (w, 1)\}$.
The families of languages generated by ET0L systems, by valence ET0L systems in plain mode, in test mode, and by ET0L systems with table valences, resp., are denoted by $\mathcal{L}(\mathrm{ET0L})$, $\mathcal{L}(\mathrm{Val}, \mathrm{ET0L})$, $\mathcal{L}(\mathrm{Val}_t, \mathrm{ET0L})$, and $\mathcal{L}(\text{Tab-Val}, \mathrm{ET0L})$, respectively.

Theorem 5.1

$$\mathcal{L}(\mathrm{ET0L}) \subset \mathcal{L}(\text{Tab-Val}, \mathrm{ET0L}) \subseteq \mathcal{L}(\mathrm{Val}, \mathrm{ET0L}) \subseteq \mathcal{L}(\mathrm{Val}_t, \mathrm{ET0L}) = \mathcal{L}(\mathrm{RE}).$$

We conjecture that the second inclusion is also proper, but have no proof for it.
Proof. The chain of inclusions $\mathcal{L}(\mathrm{ET0L}) \subset \mathcal{L}(\text{Tab-Val}, \mathrm{ET0L}) \subseteq \mathcal{L}(\mathrm{Val}, \mathrm{ET0L})$ is easily obtained. The language $\{(a^m b)^n : m \geq n \geq 1\}$ is a witness for the properness of the first inclusion. Namely, consider the ET0L system with table valences $G = (\Sigma, \Delta, P, \omega, \varphi)$ where $\Sigma = \{a, b, A, B, C_1, C_2, X\}$, $\Delta = \{a, b\}$, $P = \{P_1, P_1', P_2, P_3\}$ with
$P_1 = \{B \to B, C_1 \to BC_1\} \cup \{Y \to X : Y \in \{a, b, A, C_2, X\}\}$, $P_1' = P_1$,
$P_2 = \{B \to AB, A \to A, C_1 \to C_2, C_2 \to C_2\} \cup \{Y \to X : Y \in \{a, b, X\}\}$,
$P_3 = \{A \to a, B \to b, C_1 \to \lambda, C_2 \to \lambda\} \cup \{Y \to X : Y \in \{a, b, X\}\}$,
$\omega = C_1$, $\varphi(P_1) = 2$, $\varphi(P_1') = 1$, $\varphi(P_2) = 1/2$, $\varphi(P_3) = 1$. The inclusion $\mathcal{L}(\mathrm{RE}) \subseteq \mathcal{L}(\mathrm{Val}_t, \mathrm{ET0L})$ is shown by simulating a programmed grammar with appearance checking by a valence ET0L system in test mode. □

As regards closure properties, $\mathcal{L}(\mathrm{Val}_t, \mathrm{ET0L})$ is by the above theorem a full AFL. Similar to the case of context-free valence grammars, the other variants of valence ET0L languages are semi-AFL's. (All proofs known from the theory of ET0L systems, see [20], can be adapted to the valence variants.)

Theorem 5.2 $\mathcal{L}(\text{Tab-Val}, \mathrm{ET0L})$ *and* $\mathcal{L}(\mathrm{Val}, \mathrm{ET0L})$ *are catenation-closed semi-AFLs which are, in addition, closed under substitution by ET0L languages and mirror image. Moreover,* $\mathcal{L}(\mathrm{Val}, \mathrm{ET0L})$ *is closed under valence transductions, in particular under permutations and intersection by regular valence languages.*

Note that the construction for closure under valence transductions works only if valences are attached to the productions, and therefore not for table valence systems.

(Valence) ET0L systems are the most general case of (valence) Lindenmayer systems. It is therefore natural to discuss the existence of normal forms (i.e., the comparison with propagating, non-extended, deterministic, ..., variants). We shall give here a few results.

Theorem 5.3 *The family* $\mathcal{L}(\mathrm{Val}_t, \mathrm{T0L})$ *is strictly included in* $\mathcal{L}(\mathrm{Val}_t, \mathrm{ET0L})$, *while for* $L \in \mathcal{L}(\mathrm{Val}, \mathrm{ET0L}) \subseteq \Delta, S \notin \Delta$, $L \cup \{S\} \in \mathcal{L}(\mathrm{Val}, \mathrm{T0L})$.

Proof. Let $G = (\Sigma, \Delta, \mathbf{P}, \omega)$ be a valence ET0L system. By introducing a new symbol S as axiom and a prime number q not appearing in the factorization of some number in $\mathrm{Lab}(G)$, it is possible to keep track of the number of nonterminals in the derived sentential form by the power of q in the factorization of the valence product. Hence, the product 1 in a derivation in the T0L system (with alphabet Σ) can only be computed if the sentential form contains only symbols from Δ. The language of the T0L system is $L(G) \cup \{S\}$. We could not prove or disprove whether it is possible to construct a T0L system generating exactly $L(G)$. Note, however, that any finite language is in $\mathcal{L}(\mathrm{Val}, \mathrm{T0L})$.

The strict inclusion for valence test languages can be shown by the example $\{a^n b^n a^{2n+1} : n \geq 1\} \in \mathcal{L}(\mathrm{Val}_t, \mathrm{ET0L}) \setminus \mathcal{L}(\mathrm{Val}_t, \mathrm{T0L})$. □

Theorem 5.4 *We have* $\mathcal{L}(\mathrm{Val}_t, \mathrm{EPT0L}) \subset \mathcal{L}(\mathrm{Val}_t, \mathrm{ET0L})$, *while on the other hand,* $\mathcal{L}(\text{Tab-Val}, \mathrm{ET0L}) = \mathcal{L}(\text{Tab-Val}, \mathrm{EPT0L})$.

Proof. The strictness of the inclusion in the valence test case follows from the equivalence of (propagating) valence test languages and (λ-free) matrix grammars with appearance checking. To show equivalence in the case of table valence families, one can simply adapt the proof for classical L systems. □

Theorem 5.5 *The membership, the emptiness, and the finiteness problems are decidable for table valence ET0L systems.*

Proof. By the previous theorem, we can restrict to EPT0L table valence systems. For such systems, the membership problem is immediately reduced to the membership problem for monotone valence grammars over $\mathbb{Q}_+$, which is decidable.

To show decidability of the emptiness and finiteness problems we can, similar to the case without valences [20], assign to an EPT0L table valence system G an EDPT0L table valence system H, the so called combinatorially complete version. It can be shown that $L(G)$ is empty (finite) iff $L(H)$ is empty (finite). The problems with respect to H can be reduced to the corresponding problems for the Szilard language of H, which is a regular valence language over $\mathbb{Q}_+$. □

6 Conclusions

We have given an overview of the potentials of valence grammars. Many problems remain open. It would, for instance, be very interesting to investigate valence grammars over other specific monoids than discussed here in order to describe different language classes. Regarding parallel systems, decision questions for valence systems in plain mode are completely open, as well as questions with respect to determinism. (Of course, one has to be aware of the close equivalences to k-*BLIND* and reversal-bounded multi-counter machines and their deterministic variants, see [8, 9].) The latter question could be very interesting keeping in mind the correspondence of Chomsky normal form grammars and machine models, since we can hope for efficient LR-style parsing algorithms for valence grammars. Valences can such be used as attributes which can be easily handled.

It should be mentioned that similar results as given in this paper for ET0L systems with valences hold also for systems with limited parallelism. Moreover, there are close connections to regulated valence grammars, which have been also investigated by us, hence answering questions raised in [5, p. 267]. An interesting special case of valence grammars are grammars with valuations, where values are assigned to terminals. We have identified such languages as homomorphic pre-images of special groupoids.

Finally, we would like our colleagues, especially H. Bordihn, M. Holzer, and K. Reinhardt, for stimulating discussions.

A Proof of Theorem 4.1

The proof of the existence of normal forms for valence grammars over $\mathbb{Z}^k$ is done in several steps. First, the concept of a derivation tree is introduced. Derivation trees are very useful for the proof. Moreover, by the similarity of valence grammars and plain context-free grammars, the generalization of derivation trees is a very natural idea. In the first phase of the construction of the normal form, the erasing rules are replaced; then unit productions are eliminated; in the end the valences are normalized.

A.1 Derivation trees for valence grammars

Let $G = (N, T, P, S)$ be a valence grammar over $\mathbb{Z}^k$. A tree is a *derivation tree* for G if:

1. Every node has a *label*, which is a symbol of $N \cup T \cup \{\lambda\}$.
2. Every node with a label in N has a *value*, which is a vector in $\mathbb{Z}^k$.
3. The label of the root is S.
4. If a node is interior and has label A then A must be in N.
5. If node v has label A, value $\vec{r}$, and the nodes $v_1, v_2, \ldots, v_k$ are the sons of node v, in order from left to right, with labels $X_1, X_2, \ldots, X_k$, respectively, then $(A \to X_1 X_2 \ldots X_k, \vec{r})$ must be in P.
6. If node v has label λ then it is the only son of its father and a leaf.

As in [10], the leaves of the tree can be ordered from left to right. The *yield* of a derivation tree is the word obtained by reading the labels of the leaves from left. The *value* of the tree is obtained by adding the values of all interior nodes of the tree.

Let D be a derivation tree and v be a node in D. The subtree of D consisting of root v, all its descendants, the edges connecting them, their labels, and their values, is denoted by $D(v)$ and referred to as the subtree of D with root v. Let v_1 and v_2 be two different nodes in D where v_2 is a descendent of v_1. The (labeled and valuated) subtree induced by v_1 and all its descendants that are not descendants of v_2 is denoted by $D(v_1 - v_2)$ and called the subtree of D between v_1 and v_2.

The yield and value of a subtree are defined as for a derivation tree. For a derivation tree D in G, we denote by N_D the set of all nonterminals appearing as a label in D. Analogously, for a derivation Δ, let N_Δ be the set of nonterminals appearing in some word during the derivation process.

By the commutativity of addition in $\mathbb{Z}^k$ we obtain, as in the case of context-free grammars, a correspondence between derivations and derivation trees.

Lemma A.1 *Let $G = (N, T, P, S)$ be a valence grammar over $\mathbb{Z}^k$. A pair $(\alpha, \vec{r}) \in (N \cup T)^* \times \mathbb{Z}^k$ can be derived in G iff there is a derivation tree whose yield is α, and whose value is $\vec{r}$.*

A.2 Elimination of erasing rules

In a first step we simplify the structure of the rules with right-hand sides of length two or greater. Moreover, the proposition below shows that, given a valence grammar over $\mathbb{Z}^k$, an equivalent valence grammar over $\mathbb{Z}^k$ can be found where the set of nonterminals is partitioned into a set generating only non-empty terminal words, and a second set generating λ as the only terminal word.

Proposition A.2 *Let G be a context-free valence grammar over $\mathbb{Z}^k$. There is a valence grammar $H = (N, T, P, S)$ over $\mathbb{Z}^k$ with $L(H) = L(G) \setminus \{\lambda\}$ such that*

- $N = N_1 \cup N_2$, $N_1 \cap N_2 = \emptyset$
- *The core rules of P have one of the following forms:*
  - $A \to BC$, $A, B, C \in N, A \in N_1 \wedge (B \in N_1 \vee C \in N_1)$
  - $A \to BC$, $A \in N_2 \wedge B \in N_2 \wedge C \in N_2$
  - $A \to B$, $A \in N_1 \wedge B \in N_1$
  - $A \to B$, $A \in N_2 \wedge B \in N_2$
  - $A \to a$, $A \in N_1 \wedge a \in T$
  - $A \to \lambda$, $A \in N_2$

Proof. Let $G = (N', T, P', S')$. Completely analogous to the case of context-free grammars, we can construct $G_1 = (N_1, T, P_1, S_1)$, $L(G_1) = L(G)$, where the core rules of P_1 have one of the forms $A \to BC$ or $A \to B$ or $A \to a$ or $A \to \lambda$, $A, B, C \in N_1$, $a \in T$. Now we choose $H = (N, T, P, S)$ as follows. $N = N_1 \cup N_2$ where N_2 is a disjoint copy of N_1, the copy of $A \in N_1$ is denoted by A', $S = S_1$, and P is defined as:

$$
\begin{aligned}
(A \to BC, \vec{r}) \in P_1 \quad &\iff \quad (A \to BC, \vec{r}), (A \to B'C, \vec{r}), \\
&\qquad\;\; (A \to BC', \vec{r}), (A' \to B'C', \vec{r}) \in P, \\
(A \to B, \vec{r}) \in P_1 \quad &\iff \quad (A \to B, \vec{r}), (A' \to B', \vec{r}) \in P,
\end{aligned}
$$

$$(A \to a, \vec{r}) \in P_1 \iff (A \to a, \vec{r}) \in P,$$
$$(A \to \lambda, \vec{r}) \in P_1 \iff (A' \to \lambda, \vec{r}) \in P.$$

Let σ be the finite substitution from N_1^* into N^* with $\sigma(A) = \{A, A'\}$, for all $A \in N_1$. It can easily be shown by induction on the number of derivation steps that a derivation $(A, 0) \overset{*}{\Rightarrow}_H (\alpha, \vec{r}),\ A \in N_1, \alpha \in N^*$ is possible iff there is a derivation $(A, 0) \overset{*}{\Rightarrow}_{G_1} (\alpha_1, r)$ such that $\alpha \in \sigma(\alpha_1)$ and α contains a letter from N_1. The construction of the terminating rules guarantees that $(A, 0) \overset{*}{\Rightarrow}_H (w, \vec{r}),\ A \in N_1, w \in T^*$ iff $(A, 0) \overset{*}{\Rightarrow}_{G_1} (w, \vec{r})$ and $|w| > 0$. We have thus shown that $L(H) = L(G) \setminus \{\lambda\}$. □

Let us consider the tree of some derivation in the grammar H of the previous proposition. Let v be a node labeled by $A \in N_1$, having two sons, v_1 and v_2, respectively, labeled by $B' \in N_2$ and $C \in N_1$, respectively. The idea of the construction of the λ-free grammar is to delete in the new derivation tree the subtree with the root v_1 and to insert a path from v to v_2 whose valuation is that of the deleted subtree.

For this purpose, the following lemma is of crucial importance.

Lemma A.3 *Let $K = (N, T, P, S)$ be a context-free valence grammar over $\mathbb{Z}^k$ with core rules of the forms $A \to BC$, $A \to B$, and $A \to \lambda$. There is a regular valence grammar $K' = (N', T', P', S')$ over $\mathbb{Z}^k$ with core rules of the forms $A \to B$ and $A \to \lambda$ such that $(S, 0) \overset{*}{\Rightarrow}_K (\lambda, \vec{r})$ iff $(S', 0) \overset{*}{\Rightarrow}_{K'} (\lambda, \vec{r})$. Moreover, if $(\lambda, \vec{r})$ can be derived in K, it can be obtained in K' in* $\max\{1, ||\vec{r}||_1\}$ *steps.*

Proof. Let $U = \{a_1, \ldots, a_k\} \cup \{b_1, \ldots, b_k\}$. We define the mappings word : $\mathbb{Z}^k \to U^*$ and vec : $U^* \to \mathbb{Z}^k$ by

$$\begin{aligned} \mathrm{word}(\vec{r}) &= c_1^{|r_1|} \ldots c_k^{|r_k|}, c_i = a_i \text{ if } r_i \geq 0, c_i = b_i \text{ if } r_i < 0, \vec{r} = (r_1, \ldots, r_k), \\ \mathrm{vec}(w) &= (|w|_{a_1} - |w|_{b_1}, \ldots, |w|_{a_k} - |w|_{b_k}). \end{aligned}$$

Note that the mapping $\tau : U^* \to U^*$ with $\tau(w) = \mathrm{word}(\mathrm{vec}(w))$ is a rational valence transduction over $\mathbb{Z}^k$.

In the first step we construct the context-free grammar $K_1 = (N, U, P_1, S)$ with $P_1 = \{A \to \mathrm{word}(\vec{r})\alpha : (A \to \alpha, \vec{r}) \in P\}$. After that construct the valence grammar K_2 over $\mathbb{Z}^k$ generating $\tau(L(K_1))$. Hence, $L(K_2) = \{\mathrm{word}(\vec{r}) : (S, 0) \overset{*}{\Rightarrow}_K (\lambda, \vec{r})\}$. Since the Parikh set of $L(K_2)$ is semilinear[5], we can find a regular grammar $K_3 = (N', U, P_3, S')$ generating a symbol equivalent language, see [21]. We can assume that the rules of K_3 have the forms $A \to aB$ or $A \to a$, $A, B \in N', a \in U$.

Finally, we define the regular valence grammar $K' = (N', T, P', S')$ with $P' = \{(A \to \alpha, \mathrm{vec}(a)) : A \to a\alpha \in P_3\}$. A pair $(\lambda, \vec{r})$ can be derived in K' iff there is some $w \in L(K_3)$ with $\mathrm{vec}(w) = \vec{r}$, i.e., by the symbol equivalence of $L(K_2)$ and $L(K_3)$, iff $\mathrm{word}(\vec{r}) \in L(K_2)$. Moreover, note that $|\mathrm{word}(\vec{r})| = ||\vec{r}||_1$ and that any word $w \in L(K_3)$ is generated in $\max\{|w|, 1\}$ steps. We can conclude that $(S, 0) \stackrel{*}{\Rightarrow} (\lambda, \vec{r})$ in K iff $(S', 0) \stackrel{*}{\Rightarrow} (\lambda, \vec{r})$ in K' (in $\max\{||\vec{r}||_1, 1\}$ steps). □

Theorem A.4 *For any context-free valence grammar H over $\mathbb{Z}^k$, there is an equivalent valence grammar H' over $\mathbb{Z}^k$ with core rules of the forms $A \to BC$ or $A \to B$ or $A \to a$.*

Proof. Without loss of generality, we can assume that $H = (N, T, P, S)$ has the form as in proposition A.2. For any $B' \in N_2$, we define the context-free valence grammar $K_{B'} = (N_2, T, P_2, B')$ where P_2 is the set of all rules whose left-hand side is an element of N_2. By Lemma A.3 there is a regular valence grammar $K'_{B'} = (N_{B'}, T, P_{B'}, S_{B'})$ such that $(B', 0) \stackrel{*}{\Rightarrow} (\lambda, \vec{r})$ holds in $K_{B'}$ iff $(S_{B'}, 0) \stackrel{*}{\Rightarrow} (\lambda, \vec{r})$ holds in $K'_{B'}$. Without loss of generality we assume that $N_{B'}$ and $N_{C'}$ are disjoint for different $B', C' \in N_2$, and set $M = \bigcup_{B' \in N_2} N_{B'}$, $Q = \bigcup_{B' \in N_2} P_{B'}$.

Now we are ready to construct the λ-free valence grammar $H' = (N', T, P', S')$. We choose the set of nonterminals as $N' = N_1 \cup M \times N_1$. P' consists of P_1, the set of all rules in P with a left-hand side from N_1, and of the following rules:

$$
\begin{aligned}
(A \to (S_{B'}, C), \vec{r}) \in P' &\iff (A \to B'C, \vec{r}) \in P \text{ or } (A \to CB', \vec{r}) \in P,\\
((X, C) \to (Y, C), \vec{r}) \in P' &\iff (X \to Y, \vec{r}) \in Q,\\
((X, C) \to C, \vec{r}) \in P' &\iff (X \to \lambda, \vec{r}) \in Q.
\end{aligned}
$$

H and H' are equivalent since any leftmost derivation $(A, 0) \Rightarrow (B'C, \vec{r}) \Rightarrow^* (C, \vec{r} + \vec{s})$ in H can be replaced by the derivation $(A, 0) \Rightarrow ((S_{B'}, C), \vec{r}) \Rightarrow^* (C, \vec{r} + \vec{s})$ in H', and vice versa. □

A.3 Elimination of unit productions

Next we are going to substitute the unit productions of the form $(A \to B, \vec{r})$, where A and B are nonterminals. The basic idea is to construct, for a given valence grammar $G = (N, T, P, S)$ over $\mathbb{Z}^k$, a context-free valence grammar $G_{M,2}$ in Chomsky normal form and a regular valence grammar $G_{M,1}$, for all $M \subseteq N$. It is shown that $(w, 0)$ can be derived in G iff there are $M \subseteq N$ and $\vec{r} \in \mathbb{Z}^k$ such that $(w, \vec{r})$ is derivable in $G_{M,2}$ and $(\lambda, -\vec{r})$ is derivable in $G_{M,1}$.

From $G_{M,1}$ and $G_{M,2}$ we construct a valence grammar G_M in Chomsky normal form generating $(w, 0)$ iff $(\lambda, -\vec{r})$ and $(w, \vec{r})$, respectively, are derivable in $G_{M,1}$, $G_{M,2}$, respectively, for some $\vec{r}$. Finally, a valence grammar G' in normal form is constructed generating $\bigcup_{M \subseteq N} L(G_M) = L(G)$.

We start with some useful definitions. Let $G = (N, T, P, S)$ be a valence grammar over $\mathbb{Z}^k$ with rules of the form as in proposition A.4. Let D be a derivation tree in G. If a subtree $D(v_1 - v_2)$ is a nontrivial path and v_1 and v_2 have the same label, it is called a *loop-path*. A derivation is called loop-free if the corresponding derivation tree does not contain any loop-path. For a nonterminal $A \in N$ and for a subset $M \subseteq N$, we define

$$\begin{aligned} \mathit{LOOP}(A) &= \{\vec{r} \in \mathbb{Z}^k \mid (A, \vec{0}) \overset{*}{\Rightarrow} (A, \vec{r})\}, \\ \mathit{LOOP}(M) &= \{\vec{r} \in \mathbb{Z}^k \mid \vec{r} = \sum_{i=1}^{n} \vec{r_i},\, \vec{r_i} \in \mathit{LOOP}(A) \text{ for some } A \in M, 1 \leq i \leq n\}. \end{aligned}$$

Proposition A.5 *Let $G = (N, T, P, S)$ be a λ-free valence grammar over $\mathbb{Z}^k$. Then $(S, 0) \overset{*}{\Rightarrow} (w, 0)$, $w \in T^+$, iff there is a loop-free derivation Δ yielding $(w, \vec{r})$ where $-\vec{r} \in \mathit{LOOP}(N_\Delta)$. (Here, N_Δ is the set of nonterminals occurring in Δ.)*

Proof. Let $\vec{r} \in \mathit{LOOP}(M)$, with $\vec{r} = \sum_{i=1}^{n} \vec{r_i}$, $\vec{r_i} \in \mathit{LOOP}(A_i)$, $A_i \in M, i \leq n$, for some $M \subseteq N$. Let D be a derivation tree with $M \subseteq N_D$ (where N_D is the set of nonterminals in D) yielding $(w, \vec{s})$. By induction over n it is easily shown that there exists a derivation tree D' yielding $(w, \vec{s} + \vec{r})$.

On the other hand, let D_0 be a derivation tree yielding $(w, 0)$. We shall construct a (finite) sequence $D_0, D_1, \ldots, D_s$ of derivation trees such that

1. $\mathit{yield}(D_i) = w$, for all $1 \leq i \leq s$,
2. D_s has no loop-paths,
3. if the value of D_i is $\vec{r}$ then $-\vec{r}$ is in $\mathit{LOOP}(N_{D_s})$, for $0 \leq i \leq s$.

If the tree D_i contains a loop-path we define D_{i+1} as follows. Let V_i be the set of all nodes v in D_i that are starting points of some loop-path.

We choose $v_i \in V_i$ such that no other node of V_i is an ancestor of v_i, and u_i such that $D_i(v_i - u_i)$ is the loop-path of maximal length starting in v_i. Let $\vec{r_i}$ be the value of this path, and let A_i be the label of v_i and u_i. D_{i+1} is obtained by cutting the path $D_i(v_i - u_i)$ from D_i, i.e., the subtree with root v_i is removed and replaced by the subtree with root u_i.

The newly obtained tree D_{i+1} is obviously a derivation tree in G yielding w with $\text{value}(D_{i+1}) = \text{value}(D_i) - \vec{r}_i$. Since D_{i+1} has less nodes than D_i there must be some s such that D_s is the tree of a loop-free derivation.

Note that the choice of v_i and u_i guarantees that the node u_i is not deleted in the further process. Hence, the node set of D_s contains u_i, for all $0 \leq i \leq s-1$. Therefore, $A_i \in N_{D_s}$, for all $0 \leq i \leq s-1$. It follows that $\text{value}(D_s) = -\sum_{i=0}^{s-1} \vec{r}_i$. Since the vectors $\vec{r}_i$ belong to $LOOP(A_i)$ and $A_i \in N_{D_s}$, $1 \leq i \leq s$, we get $-\text{value}(D_s) \in LOOP(N_{D_s})$. □

Proposition A.6 *Let $G = (N, T, P, S)$ be a valence grammar over $\mathbb{Z}^k$ without λ-rules. For any $M \subseteq N$, there is a regular valence grammar over $\mathbb{Z}^k$ $G_{M,1} = (N_{M,1}, T, P_{M,1}, S_{M,1})$ such that a pair $(\lambda, \vec{r})$ is derived in $G_{M,1}$ iff $\vec{r} \in LOOP(M)$; the number of derivation steps is $\max\{1, ||\vec{r}||_1\}$.*

Proof. For $A \in N$, set $G_A = (N, T, P_A, A)$ with $P_A = \{(B \to C, \vec{r}) \in P\} \cup \{(A \to \lambda, 0)\}$. A pair $(\lambda, \vec{r})$ is generated in G_A iff $\vec{r} \in LOOP(A)$. By renaming the symbols in G_A we obtain valence grammars $K_A = (N_A, T, P'_A, S_A)$, such that $N_A \cap N_B = \emptyset$, for $A \neq B$. For $M \subseteq N$, let K_M be the valence grammar with the set of nonterminals $\bigcup_{A \in M} N_A \cup \{S\}$, the set of valence rules $\bigcup_{A \in M} P_A \cup \{(S \to S_A S, 0), (S \to S_A, 0) : A \in M\}$, and the start symbol S. Obviously, $(\lambda, \vec{r})$ is generated in K_M iff $\vec{r} \in LOOP(M)$.

The regular valence grammar $G_{M,1}$ with the properties mentioned above can be constructed following Lemma A.3. □

Proposition A.7 *Let $G = (N, T, P, S)$ be a valence grammar over $\mathbb{Z}^k$ with core rules of the forms $A \to BC$, $A \to B$, $A \to a$. For any $M \subseteq N$, there is a valence grammar over $\mathbb{Z}^k$ $G_{M,2} = (N_{M,2}, T, P_{M,2}, S_{M,2})$ with:*

1. *For any pair $(w, \vec{r})$, there is a derivation in $G_{M,2}$ iff there is a loop-free derivation Δ in G for this pair with $M \subseteq N_\Delta$.*
2. *The core rules of $P_{M,2}$ have the forms $A \to BC$ or $A \to a$.*

Proof. Consider the following set Q. For any loop-free derivation $\Delta : (A, 0) \stackrel{*}{\Rightarrow} (A', \vec{r'}) \Rightarrow (\alpha, \vec{r})$ with $\alpha \in T \cup N^2$, Q contains the triple $(A \to \alpha, \vec{r}, N_\Delta)$.

Set $G_{M,2} = (N \times \mathcal{P}(N), T, P', (S, M))$ where P' contains all valence rules

- $((A, M_0) \to (B, M_1)(C, M_2), \vec{r})$ with $(A \to BC, \vec{r}, M_3) \in Q$ and $M_0 \subseteq M_1 \cup M_2 \cup M_3$ for some $M_3 \subseteq N$, and
- $((A, M_0) \to a, \vec{r})$ with $(A \to a, \vec{r}, M_1) \in Q$ and $M_0 \subseteq M_1$ for some $M_1 \subseteq N$.

By induction on the number of derivation steps it is shown that $((A, M_0), 0) \stackrel{*}{\Rightarrow} (w, \vec{r})$ in $G_{M,2}$ iff there is a loop-free derivation $\Delta : (A, 0) \stackrel{*}{\Rightarrow} (w, \vec{r})$ in G with $M_0 \subseteq N_\Delta$. □

Proposition A.8 *Let c be a positive integer, $G_1 = (N_1, T, P_1, S_1)$ be a regular valence grammar over $\mathbb{Z}^k$, and let $G_2 = (N_2, T, P_2, S_2)$ be a context-free valence grammar over $\mathbb{Z}^k$ with core rules of the forms $A \to BC$, $A \to a$.*

There is a context-free valence grammar over $\mathbb{Z}^k$ $G = (N, T, P, S)$ with core rules of the forms $A \to BC$, $A \to a$ where a pair $(w, \vec{r})$, $w \in T^*$, $\vec{r} \in \mathbb{Z}^k$ can be derived in G iff there are $\vec{r}_1, \vec{r}_2$ such that $\vec{r} = \vec{r}_1 + \vec{r}_2$, $(\lambda, \vec{r}_1)$ is derivable in G_1 in at most $c|w|$ steps, and $(w, \vec{r}_2)$ is derivable in G_2.

Proof. Given G_1 and G_2, the desired grammar G is obtained by the following triple construction:

$$\begin{aligned}
N &= (N_1 \cup \{\lambda\}) \times N_2 \times (N_1 \cup \{\lambda\}),\\
S &= (S_1, S_2, \lambda),\\
P &= \{((A_1, A, A_2) \to (A_1, B, B_2)(B_2, C, A_2), \vec{r}) : A_1, A_2, B_2 \in N_1 \cup \{\lambda\},\\
&\quad A, B, C \in N_2, (A \to BC, \vec{r}) \in P_2\}\\
&\quad \cup \{((A_1, A, A_2) \to a, \vec{r}) : A_1, A_2 \in N_1 \cup \{\lambda\}, A \in N_2, a \in T,\\
&\quad (A_1, 0) \stackrel{*}{\Rightarrow}_{G_1} (A_2, \vec{r}_1) \text{ in maximal } c \text{ steps},\\
&\quad (A \to a, \vec{r}_2) \in P_2, \vec{r} = \vec{r}_1 + \vec{r}_2\}
\end{aligned}$$

The correctness of the construction can be shown by induction on the length of $|w|$. □

Theorem A.9 *For any context-free valence grammar $G = (N, T, P, S)$ over $\mathbb{Z}^k$, there is an equivalent context-free valence grammar $G' = (N', T, P', S')$ over $\mathbb{Z}^k$ such that the core rules of P' have the form $A \to BC$, $A \to a$.*

Proof. For any subset M of N, let $G_{M,1}$ and $G_{M,2}$ be constructed as in Propositions A.6 and A.7. There is a derivation Δ of $(w, 0)$, $w \in T^*$, in G with $M \subseteq N_\Delta$ iff there are $\vec{r} \in \mathbb{Z}^k$ such that $(\lambda, \vec{r})$ is generated in $G_{M,1}$, and a derivation of $(w, -\vec{r})$ in $G_{M,2}$. Since $(w, -\vec{r})$ is generated in $(2|w| - 1)$ steps in $G_{M,2}$ the inequation $||\vec{r}||_1 \leq c'(2|w| - 1)$, where c' is the greatest 1-norm appearing in the rules of $G_{M,2}$, holds. Hence, $(\lambda, \vec{r})$ is generated in $G_{M,1}$ in less than $2c'|w|$ steps.

Following Proposition A.8 with $c = 2c'$ and Propositions A.6,A.7, we can construct a valence grammar over $\mathbb{Z}^k$ $G_M = (N_M, T, P_M, S_M)$ such that $(w, 0)$ is generated by G_M iff there are a vector $\vec{r} \in LOOP(M)$ and a loop-free derivation Δ of $(w, -\vec{r})$ in G with $M \subseteq N_\Delta$.

From the valence grammars G_M one constructs a valence grammar G' with core rules of the forms $A \to BC$, $A \to a$, and $L(G') = \bigcup_{M \subseteq N} L(G_M)$. By Proposition A.5, $L(G') = L(G)$. □

A.4 Normalization of the valences

Theorem A.10 *Let G be a valence grammar over $\mathbb{Z}^k$. There is an equivalent valence grammar over $\mathbb{Z}^k$ G' with valence rules of the forms $(A \to BC, \vec{r})$, $||r||_1 \leq 1$, and $(A \to a, 0)$.*

Proof. Let $G = (N, T, P, S)$ be a valence grammar over $\mathbb{Z}^k$, with core rules in Chomsky normal form. We shall first construct a valence grammars over $\mathbb{Z}^k$ in Chomsky normal form whose valence vectors are multiples of unit vectors. After that, the valences are normalized by a triple construction.

Let $G_1 = (N, T, P_1, S)$ with the set of valence rules

$$\begin{aligned} P_1 &= \{(S \to w, 0) : w \in L(G), 1 \leq w \leq k\} \cup \\ &\quad \{(A \to \alpha, \vec{r}) : (A, 0) \overset{*}{\Rightarrow}_G (\alpha, \vec{r}), \alpha \in (N \cup T)^*, k+1 \leq |\alpha| \leq k(k+1)\} \end{aligned}$$

It is clear that, for any derivation of $(\alpha, \vec{r})$ in G_1, there is a derivation of the same pair in G. On the other hand, it can be shown by induction on the length of α, that $(\alpha, \vec{r})$ is derivable in G_1 if $(\alpha, \vec{r})$ is derivable in G. Consequently, $L(G_1) = L(G)$.

Next, let T' be a copy of T and let $h : (N \cup T)^* \to (N \cup T')^*$ be the coding mapping $A \in N$ to A and $a \in T$ to its copy $a' \in T'$. A valence rule $(A_1 \to B_1 \cdots B_m, \vec{r}) \in P_1$ with $m \geq k+1$, $\vec{r} = r_1\vec{e_1} + \cdots + r_k\vec{e_k}$ is split into $(m-1)$ rules $(A_1 \to h(B_1)A_2, \vec{s_1}), (A_2 \to h(B_2)A_3, \vec{s_2}), \ldots, (A_{m-1} \to h(B_{m-1})h(B_m), \vec{s}_{k-1})$, where $A_2, \ldots, A_{k-1}$ are new nonterminals and $\vec{s_j} = r_j\vec{e_j}$, for $j \leq k$, $\vec{s_j} = 0$, for $j > k$. Analogously, all rules $(S \to a_1 \cdots a_m, 0)$, $2 \leq m \leq k$, are replaced by rules $(S \to a'_1 A_2, 0), (A_2 \to a'_2 A_3, 0), \ldots, (A_{m-1} \to a'_{m-1}a'_m, 0)$. For all $a \in T$, the rule $(a' \to a, 0)$ is introduced. The grammar $G_2 = (N_2, T, P_2, S_2)$ obtained this way is equivalent to G_1.

Finally, we construct $G_3 = (N_3, T, P_3, (0, S_2, 0))$ with

$$\begin{aligned} N_3 &= \{0, \ldots, c-1\} \times N_2 \times \{0, \ldots, c-1\} \\ P_3 &= \{((\vec{r}, A, \vec{r_2}) \to (\vec{r_0}, B, \vec{r_1})(\vec{r_1}, C, \vec{r_2}), \vec{s'}) : (A \to BC, \vec{s}) \in P_2, \\ &\quad \vec{r_0} = (\vec{r} + \vec{s}) \bmod c, \vec{s'} = (\vec{r} + \vec{s}) \operatorname{div} c \text{ for some } \vec{s}\} \\ &\quad \cup \{((\vec{r}, A, \vec{r}) \to a, 0) : (A \to a, 0) \in P_2\} \end{aligned}$$

where $c = \max\{||\vec{r}||_1 : \exists(A \to \alpha, \vec{r}) \in P_2\}$. Again it can be shown by induction that G_3 is equivalent to G_2 and hence to G. Moreover, the rules of G_3 have the announced form. □

A.5 Greibach normal form

Finally, let us mention that it is also possible to find, valence grammars and unordered vector grammars, admit the construction of Greibach normal forms.

Theorem A.11 *Let $G = (N, T, P, S)$ be a valence grammar over $\mathbb{Z}^k$. There is an equivalent k-valence grammar $G' = (N', T, P', S')$ such that the rules of P' have the form $(A \to a\alpha, \vec{r})$, $A \in N', a \in T, \alpha \in N'^*, r \in \mathbb{Z}^k, ||r||_1 \le 1$.*

Proof. Let G be a valence grammar over $\mathbb{Z}^k$ with core rules in Chomsky normal form. Nearly literally the same construction as in the context-free case [10] can be applied to obtain an equivalent k-valence grammar G_1 with core rules in Greibach normal form. A technique similar to theorem A.10 leads to a grammar in Greibach normal form with valence vectors of 1-norm not greater than 1. □

Theorem A.12 *For any context-free unordered vector grammar H, there is an equivalent unordered vector grammar H' whose corresponding context-free grammar is in Greibach normal form.*

Remark A.13 *The last theorem immediately yields a simple machine characterization of valence languages over $\mathbb{Z}^k$ via pushdown machines (without λ-steps) endowed with k blind counters. The proofs of [10, Theorems 5.3,5.4] can easily be extended to valence grammars and pushdown automata.*

References

[1] D. A. M. Barrington and D. Thérien. Finite monoids and the fine structure of NC^1. *Journal of the Association for Computing Machinery*, 35(4):941–952, October 1988.

[2] F. Bédard, F. Lémieux, and P. McKenzie. Extensions to Barrington's M-program model. *Theoretical Computer Science*, 107:31–61, 1993.

[3] F. Berman. Edge Grammars and parallel computation. In: *Proceedings of the 1983 Allerton Conference*, pages 214–223, 1983.

[4] J. Berstel. *Transductions and Context-Free Languages.* Stuttgart: Teubner, 1979.

[5] J. Dassow and Gh. Păun. *Regulated Rewriting in Formal Language Theory.* Berlin: Springer, 1989.

[6] H. Fernau and R. Stiebe. Regulation by valences. In: *Mathematical Foundations of Computer Science 1997 (LNCS 1295)*, pages 239–248, 1997. Berlin: Springer, 1997.

[7] M. Gheorge. Linear valence grammars. In: *International Meeting of Young Computer Scientists*, pages 281–285, 1986.

[8] S. Greibach. Remarks on blind and partially blind one-way multicounter machines. *Theoretical Computer Science*, 7:311–324, 1978.

[9] E. M. Gurari and O. H. Ibarra. The complexity of decision problems for finite-turn multicounter machines. *Journal of Computer and System Sciences*, 22:220–229, 1981.

[10] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Reading (MA): Addison-Wesley, 1979.

[11] O. H. Ibarra, S. K. Sahni, and C. E. Kim. Finite automata with multiplication. *Theoretical Computer Science*, 2:271–296, 1976.

[12] M. Marcus and Gh. Păun. Valence gsm-mappings. *Bull. Math. Soc. Sci. Math. Roumanie*, 31(3):219–229, 1987.

[13] V. Mitrana. Valence grammars on a free generated group. *EATCS Bulletin*, 47:174–179, 1992.

[14] V. Mitrana and R. Stiebe. The accepting power of finite automata over groups. In Gh. Păun and A. Salomaa, editors, *New Trends in Formal Languages (LNCS 1218)*, pages 39–48. Berlin: Springer, 1997.

[15] Gh. Păun. A new generative device: valence grammars. *Rev. Roumaine Math. Pures Appl.*, XXV(6):911–924, 1980.

[16] Gh. Păun. On a class of valence grammars. *Stud. cerc. mat.*, 42:255–268, 1990.

[17] Gh. Păun. Valences: increasing the power of grammars, transducers, grammar systems. *EATCS Bulletin*, 48:143–156, 1992.

[18] V. Red'ko and L. Lisovik. Regular events in semigroup. *Problems of Cybernetics*, 37:155–184, 1980. In Russian.

[19] J. J. Rotman. *An Introduction to the Theory of Groups.* New York: Springer, 5th edition, 1995.

[20] G. Rozenberg and A. K. Salomaa. *The Mathematical Theory of L Systems.* Academic Press, 1980.

[21] A. K. Salomaa. *Formal Languages.* Academic Press, 1973.

[22] G. Satta. The membership problem for unordered vector grammars. In J. Dassow, G. Rozenberg, and A. Salomaa, editors, *Developments in Language Theory II*, pages 267–275. Singapore: World Scientific, 1996.

[23] I. H. Sudborough. The complexity of the membership problem for some extensions of context-free languages. *International Journal of Computer Mathematics*, 6:191–215, 1977.

[24] S. Vicolov. Hierarchies of valence languages. In J. Dassow and A. Kelemenova, editors, *Developments in Theoretical Computer Science*, pages 191–196. Basel: Gordon and Breach, 1994.

[25] S. Vicolov-Dumitrescu. Grammars, grammar systems, and gsm mappings with valences. In Gh. Păun, editor, *Mathematical Aspects of Natural and Formal Languages*, pages 473–491. Singapore: World Scientific, 1994.