International Journal of Foundations of Computer Science

LINKS*

FRANK DREWES
Department of Computing Science, Umeå University, S-90187 Umeå, Sweden
drewes@cs.umu.se
http://www.cs.umu.se/~drewes

Received (Day Month Year)
Accepted (Day Month Year)
Communicated by (xxxxxxxxxx)

We define the notion of links, thus generalizing the chain-code, turtle geometry, and collage formalisms. As links can be concatenated, link grammars can be defined in a similar way as chain-code grammars. Context-free link grammars are shown to be strictly more powerful than both context-free chain-code grammars and context-free collage grammars. Moreover, we study the generalization obtained by extending the formalism with the brackets known from the turtle geometry.

Keywords: chain-code grammar; collage grammar; picture concatenation.

1991 Mathematics Subject Classification: 68Q50, 68Q45

1. Introduction

Chain-code picture grammars and collage grammars are two well-studied formalisms for generating pictures. The former were introduced in [10] and have intensively been studied by Dassow et al., e.g., in [1, 2, 3]. A chain-code picture is a line drawing with distinguished start and end points. The latter make it possible to concatenate pictures. Thus, every string language over an alphabet of primitive chain-code pictures yields a picture language. In contrast, collage grammars [9] are based on the idea of transforming collages (i.e., finite sets of geometric objects) and taking unions.

These mechanisms are known to be incomparable with respect to generative power. While this statement is trivial in one direction (as collages need not consist of lines), the other direction is less trivial. Nevertheless, not even all linear context-free chain-code picture languages are context-free collage languages (see Section 3.2.2 of [5]). Intuitively, the reason is that concatenation moves the second argument to a place depending on the first – an ability which the collage formalism lacks.

A third formalism, quite similar to the chain code, is the well-known turtle geometry. In this paper, the notion of links is proposed as a generalization of all

*dedicated to Jürgen Dassow on the occasion of this 60th birthday

three. Moreover, we study the addition of brackets to the formalism, a generalization well known from the turtle geometry.

Our main results are that (a) link collage languages generalize chain-code, turtle, and collage languages, (b) there are linear context-free link collage languages which are not context-free collage languages, even if collages that can be transformed into each other by some affine transformation are considered to be equivalent, and (c) if branching grammars [6] are used to generate the underlying string languages, then brackets can be avoided by increasing the nesting depth of tables by one. Moreover, we give an example that supports the conjecture that, in the context-free case, the requirement of well-formed right-hand sides (disallowing, e.g., $A]B[A[$ as a right-hand side) strictly reduces the generative power.

An implementation of the link formalism has been added to the system TREEBAG, which can be downloaded from http://www.cs.umu.se/~drewes/treebag.

2. Context-free link collage languages

We denote the sets of natural numbers (including 0) and real numbers by $\mathbb{N}$ and $\mathbb{R}$, resp. Given a function $f\colon A \to B$, we denote the canonical extension of f to subsets of A by f as well, i.e., $f\colon 2^A \to 2^B$ is given by $f(S) = \{f(a) \mid a \in S\}$ for all $S \subseteq A$. For a set A, A^* denotes the set of all strings over A. The empty string is denoted by ϵ. If $\mathcal{C}$ is a class of string languages, then A-$\mathcal{C}$ denotes the class of all languages in $\mathcal{C}$ which are subsets of A^*. We assume that the reader is familiar with some of the most fundamental definitions and properties of affine geometry.

Let us fix some dimension $d > 0$. A *collage* is a finite set of nonempty bounded subsets of the d-dimensional Euclidean space $\mathbb{R}^d$. A *link* is a pair $\lambda = \langle C, \tau\rangle$ consisting of a collage C and a transformation $\tau \in \mathbb{AFF}$, where $\mathbb{AFF}$ denotes the set of all injective affine transformations of $\mathbb{R}^d$. These components may also be referred to as C_λ and τ_λ, resp. The *empty link* is given by $\lambda_{\mathrm{e}} = \langle \emptyset, \mathrm{id}\rangle$, where id is the identity. The set of all links (in $\mathbb{R}^d$) is denoted by Λ.

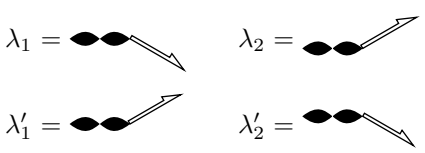
Two links $\lambda = \langle C, \tau\rangle$ and $\lambda' = \langle C', \tau'\rangle$ are concatenated by transforming C' by means of τ and composing τ and τ'. More precisely, $\lambda \cdot \lambda' = \langle C \cup \tau(C'), \tau \circ \tau'\rangle$, where $(\tau \circ \tau')(x) = \tau(\tau'(x))$ for all $x \in \mathbb{R}^d$. Clearly, concatenation of links is an associative operation, and λ_{e} is its neutral element.

For a string $w \in \Lambda^*$, we let $\mathrm{draw}(w)$ be defined inductively, as follows: $\mathrm{draw}(\epsilon) = \lambda_{\mathrm{e}}$ and, for $w = u\lambda$ with $u \in \Lambda^*$ and $\lambda \in \Lambda$, $\mathrm{draw}(w) = \mathrm{draw}(u) \cdot \lambda$. Thus, draw is the unique monoid homomorphism from the free monoid over Λ to $(\Lambda; \cdot, \lambda_{\mathrm{e}})$. For $w \in \Lambda^*$, the collage $C_{\mathrm{draw}(w)}$ is called the *link collage drawn by* w, denoted by $\mathrm{lc}(w)$.

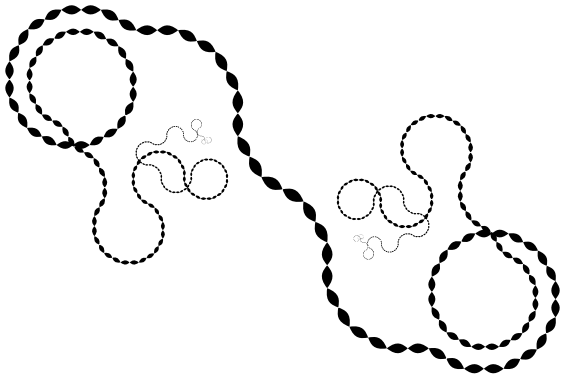
Given any class $\mathcal{C}$ of string languages, such as the classes CF and ET0L of context-free languages and ET0L languages, resp., we let $\mathrm{LC}_{\mathcal{C}}$ denote the class of *link collage languages* obtained from languages in $\mathcal{C}$; more precisely, $\mathrm{LC}_{\mathcal{C}} = \mathrm{lc}(\Lambda\text{-}\mathcal{C})$. Thus, e.g., $\mathrm{LC}_{\mathrm{CF}}$ is the class of *context-free link collage languages*, and $\mathrm{LC}_{\mathrm{ET0L}}$ is the class of *ET0L link collage languages*.

Example 1. *Let us have a look at an example in* $\mathrm{LC}_{\mathrm{CF}}$*, where* $d = 2$*.*

Consider the links to the right, where the solid black objects are the parts, each having its left and right ends at $(0,0)$ *and* $(1,0)$*, resp. The hollow arrows indicate the second components of the links. Each arrow is the image of the arrow* ⟹ *(whose base line extends from* $(0,0)$ *to* $(1,0)$*) under the respective transformation. Whereas* τ_{λ_1} *and* τ_{λ_2} *include a scaling by a factor a slightly larger than* 1*,* $\tau_{\lambda_1'}$ *and* $\tau_{\lambda_2'}$ *include a scaling by* $1/a$*. (In particular,* $\tau_{\lambda_1} \neq \tau_{\lambda_2'}$ *and* $\tau_{\lambda_2} \neq \tau_{\lambda_1'}$*, even though the difference is difficult to see.)*



Thus, τ_{λ_1} *and* τ_{λ_2} *are equal except for their rotation directions, and* $\tau_{\lambda_i'}$ *undoes the rotation and scaling of* τ_{λ_i} *(while the translation vector is* $(1,0)$ *in each). As a consequence, the context-free grammar with the initial nonterminal* S *and the rules* $S \to A\lambda_1$ *and* $A \to \lambda_1 A\lambda_1' \mid \lambda_2 A\lambda_2' \mid \lambda_{\mathrm{e}}$ *generates symmetric pictures such as the one shown to the right (which has been scaled down as it is actually about 30,000 units wide).*



3. Generating line drawings

Two well-known mechanisms for generating line drawings are the chain-code formalism and the turtle geometry.[a] Their definitions can easily be restated in terms of links. More precisely, let $d = 2$ and define, for every point $x \in \mathbb{R}^2$, $\overrightarrow{x}$ to be the link whose first component is (the singleton consisting of) the closed line segment whose end points are the origin and x, and whose second component is the translation that maps the origin to x.

Now, to obtain the chain-code formalism, let $\Lambda_{\mathrm{CC}} = \{\mathtt{u}, \mathtt{l}, \mathtt{d}, \mathtt{r}\}$, where

$$\mathtt{u} = \overrightarrow{(0,1)},\ \mathtt{d} = \overrightarrow{(0,-1)},\ \mathtt{l} = \overrightarrow{(-1,0)}, \text{ and } \mathtt{r} = \overrightarrow{(1,0)}.$$

A set of collages of the form $\mathrm{lc}(L)$, where $L \subseteq \Lambda_{\mathrm{CC}}^*$, is commonly called a *chain-code picture language*. Given any class $\mathcal{C}$ of string languages, we let $\mathrm{CC}_{\mathcal{C}}$ denote the class of chain-code picture languages obtained from languages in $L \in \mathcal{C}$, i.e., $\mathrm{CC}_{\mathcal{C}} = \mathrm{lc}(\Lambda_{\mathrm{CC}}\text{-}\mathcal{C})$. The class $\mathrm{CC}_{\mathcal{C}}$ is usually named after $\mathcal{C}$ in the obvious way. For instance, $\mathrm{CC}_{\mathrm{CF}}$ is the class of *context-free chain-code picture languages*, and

[a] see, e.g., [10, 1, 2, 3] and [11]; additional references can be found in [5]

$\mathrm{CC}_{\mathrm{ET0L}}$ is the class of *ET0L chain-code picture languages*. As indicated above, all these definitions are obviously equivalent to the traditional ones. In particular, the classes $\mathrm{CC}_{\mathrm{CF}}$ and $\mathrm{CC}_{\mathrm{ET0L}}$ are those known from the literature.

Similarly, the turtle geometry (without brackets; cf. Section 5) can be defined in terms of links. For this, choose some angles α_0 and α, and let ρ_α denote the rotation by α around the origin. Now, define $\Lambda_{\mathrm{TG}} = \{\mathtt{F}, \mathtt{f}, +, -\}$, where

$$\mathtt{F} = \overrightarrow{(\cos(\alpha_0), \sin(\alpha_0))},\ \mathtt{f} = \langle \emptyset, \tau_{\mathtt{F}} \rangle,\ + = \langle \emptyset, \rho_\alpha \rangle, \text{ and } - = \langle \emptyset, \rho_{-\alpha} \rangle.$$

Again, these definitions are obviously equivalent to those used traditionally used in connection with the turtle geometry. For example, the languages in $\mathrm{lc}(\Lambda_{\mathrm{TG}}\text{-ET0L})$ are those generated by ET0L systems under the turtle geometry interpretation.

4. Collage languages are link collage languages

Let us, very briefly and informally, recall some of the definitions in connection with collage grammars. For a more detailed and formal exposition of the subject, see [5]. For a ranked alphabet Σ, we let T_Σ denote the set of all terms (or trees) over Σ. Thus, T_Σ is the smallest set of strings such that (a) every symbol of rank 0 in Σ is in T_Σ, and (b) $f[t_1, \ldots, t_k] \in T_\Sigma$ for all $f \in \Sigma$ of rank $n > 0$ and all $t_1, \ldots, t_n \in T_\Sigma$. For a class $\mathcal{C}$ of tree languages (i.e., of sets of trees) and a ranked alphabet Σ, we let $\Sigma\text{-}\mathcal{C} = \{L \in \mathcal{C} \mid L \subseteq T_\Sigma\}$, similar to the string case.

In order to simplify and shorten the presentation, we restrict ourselves to the so-called basic collage operations in order to generate collages (see [5], and, in particular, Theorem 3.1.7). Consider the ranked alphabet Σ_{COL} containing the symbol $\cup$ of rank 2, all elements of $\mathbb{AFF}$, viewed as symbols of rank 1, and all collages viewed as symbols of rank 0. Given a tree $t \in T_{\Sigma_{\mathrm{COL}}}$, the *collage drawn by* t, $\mathrm{col}(t)$, is obtained by interpreting the symbols in the obvious way:

$$\mathrm{col}(t) = \begin{cases} C & \text{if } t = C \text{ for a collage } C \\ \tau(\mathrm{col}(t')) & \text{if } t = \tau[t'] \text{ for some } \tau \in \mathbb{AFF} \text{ and } t' \in T_{\Sigma_{\mathrm{COL}}} \\ \mathrm{col}(t') \cup \mathrm{col}(t'') & \text{if } t = t' \cup t'' \text{ for trees } t', t'' \in T_{\Sigma_{\mathrm{COL}}}. \end{cases}$$

Let REGT, ET0LT, and BST denote the classes of all regular, ET0L, and branching synchronization tree languages, resp. Then $\mathrm{COL}_{\mathrm{CF}} = \mathrm{col}(\Sigma_{\mathrm{COL}}\text{-REGT})$, $\mathrm{COL}_{\mathrm{ET0L}} = \mathrm{col}(\Sigma_{\mathrm{COL}}\text{-ET0LT})$, and $\mathrm{COL}_{\mathrm{BS}} = \mathrm{col}(\Sigma_{\mathrm{COL}}\text{-BST})$ are the classes known in the literature as the context-free, ET0L, and BS collage languages, resp. (Owing to lack of space, the reader is referred to [5] for definitions.)

The *yield* of a tree t, denoted by $\mathrm{yield}(t)$ is the string obtained by reading its leaves (i.e., the symbols of rank 0) from left to right. Given a class $\mathcal{C}$ of tree languages, we let $\mathrm{y}\mathcal{C}$ denote the class of string languages obtained by taking yields, i.e., $\mathrm{y}\mathcal{C} = \mathrm{yield}(\mathcal{C})$. We remark that, with these definitions, CF = yREGT, ET0L = yET0LT, and BS = yBST, resp., where BS denotes the class of branching synchronization (string) languages introduced and studied in [6]; see also Section 5.

For the next theorem, recall that a tree homomorphism $h\colon T_\Sigma \to T_{\Sigma'}$ (defined in the usual way) is called linear if it does not duplicate subtrees of its argument

tree (e.g., $h(f[t_1, t_2]) = f'[h(t_1), f'[h(t_2), h(t_1)]]$ is not allowed because t_1 is duplicated), and nondeleting if it does not delete any subtree of the argument tree (e.g., $h(g[t_1, t_2, t_3]) = g'[h(t_2), h(t_1)]$ is not allowed because t_3 is deleted). For exact definitions see, e.g., [8].

Theorem 2. *For every class $\mathcal{C}$ of tree languages which is closed under linear nondeleting tree homomorphisms, it holds that $\mathrm{COL}_{\mathcal{C}} \subseteq \mathrm{LC}_{\mathrm{y}\mathcal{C}}$. In particular, $\mathrm{COL}_{\mathrm{CF}} \subseteq \mathrm{LC}_{\mathrm{CF}}$, $\mathrm{COL}_{\mathrm{ET0L}} \subseteq \mathrm{LC}_{\mathrm{ET0L}}$, and $\mathrm{COL}_{\mathrm{BS}} \subseteq \mathrm{LC}_{\mathrm{BS}}$.*

Proof. Choose any linear nondeleting tree homomorphism h from $T_{\Sigma_{\mathrm{COL}}}$ to T_Σ, for some suitable ranked alphabet Σ, such that $yh = \mathrm{yield} \circ h$ is as follows:

- $yh(C) = \langle C, \mathrm{id}\rangle$ for every collage C,
- $yh(\tau[t]) = \langle \emptyset, \tau\rangle yh(t) \langle \emptyset, \tau^{-1}\rangle$ for all $\tau \in \mathbb{AFF}$ and $t \in T_{\Sigma_{\mathrm{COL}}}$, and
- $yh(\cup[t, t']) = yh(t)yh(t')$ for all trees $t, t' \in T_{\Sigma_{\mathrm{COL}}}$.

By an obvious induction, $\mathrm{draw}(yh(t)) = \langle \mathrm{col}(t), \mathrm{id}\rangle$ for all trees $t \in T_{\Sigma_{\mathrm{COL}}}$. In particular, $\mathrm{lc}(yh(t)) = \mathrm{col}(t)$, and thus, as $\mathcal{C}$ is closed under h, $\mathrm{col}(L) = \mathrm{lc}(yh(L)) \in \mathrm{LC}_{\mathrm{y}\mathcal{C}}$ for every tree language $L \in \mathcal{C}(\Sigma_{\mathrm{COL}})$. This completes the proof. □

Since it is known that the classes $\mathrm{CC}_{\mathrm{CF}}$ and $\mathrm{COL}_{\mathrm{CF}}$ are incomparable (see Section 3.2.2 of [5]), Section 3 and Theorem 2 show that the inclusion $\mathrm{COL}_{\mathrm{CF}} \subseteq \mathrm{LC}_{\mathrm{CF}}$ is strict.[b] In fact, according to Theorem 3.2.3 of [5], there is a linear chain-code picture language L that is not in $\mathrm{COL}_{\mathrm{CF}}$ – and this holds even if we identify collages that are equal up to translation.

We shall now strengthen this result. First, let us recall a few things. A *collage replacement rule* is a pair of collages (l, r). For a set $\mathcal{R}$ of collage replacement rules and collages C, C', we write $C \Rightarrow_{\mathcal{R}} C'$ if $C' = (C \setminus \tau(l)) \cup \tau(r)$ for some $\tau \in \mathbb{AFF}$ such that $\tau(l) \subseteq C$. Now, by inspecting the proof of Lemma 3.2.1 of [5] (originally from [7]), one can easily see that a minor modification proves the following.

Lemma 3. *For every collage language $L \in \mathrm{COL}_{\mathrm{CF}}$, there exist a finite set $\mathcal{R}$ of collage replacement rules and a finite subset L_0 of L such that the following holds:*

> *For every collage $C_0 \in L$, there are collages $C_1, \ldots, C_n \in L$ (for some $n \in \mathbb{N}$) such that $C_0 \Rightarrow_{\mathcal{R}} \cdots \Rightarrow_{\mathcal{R}} C_n$ and $C_n \in L_0$.*

Hence, $\mathcal{R}$ can be used to transform any collage in L in a stepwise manner into one of a finite number of collages without ever leaving L.

Let us now return to the comparison of language classes. If the aim is to generate a certain collage language, one may not necessarily be interested in, e.g., the exact placement, size, or rotation of the generated collages. To take this into account, we say that collages C and C' are equivalent, $C \sim C'$, if $C' = \tau(C)$ for some $\tau \in \mathbb{AFF}$. Clearly, $\sim$ is an equivalence relation. For a class $\mathcal{C}$ of collage languages, let $\mathcal{C}^{\sim}$

[b] Of course, the strictness of the inclusion $\mathrm{CC}_{\mathrm{CF}} \subseteq \mathrm{LC}_{\mathrm{CF}}$ is trivial.

denote the class of all its quotient languages under $\sim$, i.e., $\mathcal{C}^{\sim} = \{L/{\sim} \mid L \in \mathcal{C}\}$. We prove the following, where LIN denotes the set of all linear context-free languages.

Theorem 4. $\mathrm{LC}^{\sim}_{\mathrm{LIN}} \not\subseteq \mathrm{COL}^{\sim}_{\mathrm{CF}}$.

Proof. Consider the language $L = \{\lambda\lambda_{\mathrm{sc}}^n\lambda_{\mathrm{tr}}(\lambda'_{\mathrm{sc}})^n\lambda \mid n \in \mathbb{N}\} \in \mathrm{LIN}$, where $\lambda, \lambda_{\mathrm{sc}}, \lambda'_{\mathrm{sc}}, \lambda_{\mathrm{tr}}$ are given as follows:

(1) $\lambda = \langle\{p\}, \mathrm{id}\rangle$, where p is any part such that, for any other part p', there is at most one $\tau \in \mathbb{AFF}$ such that $p = \tau(p')$. (For example, the arrow shaped part on p. 3 is of this kind.)
(2) $\lambda_{\mathrm{sc}} = \langle\emptyset, \sigma\rangle$ and $\lambda'_{\mathrm{sc}} = \langle\emptyset, \sigma^{-1}\rangle$, where σ is the scaling by the factor 2.
(3) λ_{tr} is the translation by $(1, 0)$.

Thus, $\mathrm{lc}(L)$ is the set of all collages $C_i = \{p, p_i\}$, $i \in \mathbb{N}$, where p_i is p translated by $(2^i, 0)$. Note that the C_i are pairwise inequivalent.

Now, suppose that $L'/{\sim} = \mathrm{lc}(L)/{\sim}$ for some context-free collage language L', and let $\mathcal{R}$ be the finite set of collage replacement rules obtained by applying Lemma 3 to L'. By the definition of $\Rightarrow_{\mathcal{R}}$, if $C_1 \Rightarrow_{\mathcal{R}} C'_1$ and $C_1 \sim C_2$, then $C_2 \Rightarrow_{\mathcal{R}} C'_2$ for some C'_2 such that $C'_2 \sim C'_1$. In particular, we have the following:

$$\text{For all but finitely } i \in \mathbb{N}, \text{ there exists } j \in \mathbb{N} \setminus \{i\} \text{ such that } C_i \Rightarrow_{\mathcal{R}} C \text{ for some collage } C \sim C_j. \tag{4}$$

Without loss of generality, we may assume that the left-hand side l of every rule $(l, r) \in \mathcal{R}$ contains p. This is because, if l does not contain any part of the form $\tau(p)$, then the rule can be discarded; otherwise, it can be replaced with $(\tau^{-1}(l), \tau^{-1}(r))$ without affecting $\Rightarrow_{\mathcal{R}}$.

Another simplification results from the observation that each rule $(l, r) \in \mathcal{R}$ with $|l| > 1$ applies to at most one C_i. This is a direct consequence of the choice of p. Therefore, (4) remains valid if we require that $l = \{p\}$ for all $(l, r) \in \mathcal{R}$, and thus $r = \{\tau(p)\}$ for some translation τ. Looking at all pairs $i, j \in \mathbb{N}$ as in (4), we thus find that the distance between p_i and p_j is bounded by some constant k (namely the maximum of the lengths of the translation vectors underlying the translations τ). Hence, no such j can exist for any i with $2^{i-1} > k$ – a contradiction. □

5. Bracketed link collage languages

A widely used extension of the turtle geometry discussed in Section 3 adds square brackets to the alphabet. Given a string $u[v]w$ with a matching pair of parentheses surrounding the substring v, the interpretation is that w is drawn using the transformation that was in effect when u had been drawn. From an operational point of view, upon encountering the '[' the transformation component of draw(u) is pushed onto a stack, and is restored later upon reading the ']'. In the following, we extend the link formalism in a similar manner (but without explicitly introducing a stack).

Let $\Lambda_{[]} = \Lambda \cup \{[,]\}$. In the following, we say that $w \in \Lambda_{[]}^*$ is well formed if each of its opening brackets has a matching closing bracket, and vice versa. In other words, every $w \in \Lambda^*$ is well formed, and if w, w' are well formed, then so are ww' and $[w]$.

In order to extend draw from Λ^* to $\Lambda_{[]}^*$, let $u \in \Lambda_{[]}^*$.

- If $u \in \Lambda^*$, then $\mathrm{draw}(u)$ is defined as in Section 2.
- If $u = ww'$ for well-formed $w, w' \in \Lambda_{[]}^*$, then $\mathrm{draw}(u) = \mathrm{draw}(w)\mathrm{draw}(w')$.
- If $u = [w]$, where $w \in \Lambda_{[]}^*$ is well formed, then $\mathrm{draw}(u) = \langle C_{\mathrm{draw}(w)}, \mathrm{id}\rangle$.
- If u is not well-formed, then it can be written in the form waw', where either w is well formed and $a =]$, or $a = [$ and w' is well formed. We define $\mathrm{draw}(u) = \mathrm{draw}([u)$ in the former case, and $\mathrm{draw}(u) = \mathrm{draw}(u])$ in the latter.

Note that $\mathrm{draw}(u)$ is uniquely defined even if u is not well formed. However, draw is of course not a homomorphism any more. As before, we let $\mathrm{lc}(u) = C_{\mathrm{draw}(u)}$.

When generating link collage languages in this generalized formalism, we speak of *bracketed link collage languages*. Similar to the notation introduced in Section 2, for every class $\mathcal{C}$ of formal languages, we let $\mathrm{bLC}_{\mathcal{C}} = \mathrm{lc}(\Lambda_{[]}\text{-}\mathcal{C})$ denote the class of bracketed link collage languages it defines.

It may be interesting to note that brackets can be used to simplify the simulation of context-free collage grammars given in Section 4: yh may, instead of turning $\tau[t]$ into $\langle \emptyset, \tau\rangle yh(t)\langle \emptyset, \tau^{-1}\rangle$, simply turn it into $[\langle \emptyset, \tau\rangle yh(t)]$. In addition to being more elegant, this construction remains valid in settings where a restricted class of affine transformations (not closed under inverse) is considered, such as the set of all contractions.

The issue of well-formedness seems to deserve more attention than it has received. The definitions of the turtle geometry that can be found in the literature are typically quite unclear in this respect – neither do they require well-formedness explicitly, nor do they define how ill-formed strings are interpreted. In fact, in almost all publications, there seems to be the implicit assumption that all right-hand sides of the grammars (or L systems) considered are themselves well formed.[c] The author is not aware of any example presented in the literature which is not of this kind. Under the assumption that only well-formed right-hand sides are allowed, brackets are equivalent to the unary *encapsulation operation* used in [5], where $\mathrm{enc}(\lambda) = \langle C_\lambda, \mathrm{id}\rangle$ (see [4], where this equivalence is proved, for a more formal statement).

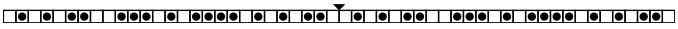
For the case of the turtle geometry, it seems to be an open question whether, say, context-free grammars with well-formed right-hand sides have the same generative power as those without. Naturally, the question is open for bracketed link collage languages. However, consider the linear context-free grammar whose rules are

$$S \to [A,\ A \to \lambda_a A \lambda_a,\ A \to \lambda_b A \lambda_b,\ A \to]\lambda.$$

It generates the language L_0 consisting of all (well-formed!) strings of the form $[w]\lambda w^R$, where $w \in \{\lambda_a, \lambda_b\}^*$, and w^R denotes the reverse of w. Now, let λ_a be

[c]We extend the notion of well-formedness to strings containing nonterminals in the obvious way.

the link consisting of the hollow unit square whose lower left corner is $(0, -1/2)$, and the translation by one unit to the right. Let λ_b be defined as λ_a, but with an additional circle inside the square. Finally, let $\lambda = \langle C, \tau \rangle$, where C is a triangle placed half a unit over the origin, and τ is the rotation by 180 degrees. A typical member of the language $\mathrm{lc}(L_0)$ looks like this:



Seen from the left to the right, each picture consists of a left half which is an arbitrary sequence of squares with and without circles, and a right half, being an exact copy of the left half. Less importantly, a triangle is placed in the middle, i.e., on top of the edge between the last square of the left half and the first square of the right half. Thus, $\mathrm{lc}(L_0)$ is a pictorial variant of $\{w\$w \mid w \in \{a, b\}^*\}$, which is well-known not to be context free. We conjecture that $\mathrm{lc}(L_0)$ cannot be generated by a context-free grammar with well-formed right-hand sides. To make the example work for the turtle geometry as well, we modify it slightly. Let $\alpha_0 = 0°$, $\alpha = 90°$, and use a linear context-free grammar similar to the one above to generate the language $L_1 = \{[w]+[F]+w^R \mid w \in \{F, f\}^*\}$. Apart from using different pictorial primitives, $\mathrm{lc}(L_1)$ has structural properties similar to those of $\mathrm{lc}(L_0)$.

Conjecture 5. *Let* WF_0 *denote the set of all languages in* $\Lambda_{\square}$-CF *that can be generated by context-free grammars whose right-hand sides are well formed. Then* $\mathrm{lc}(L_1)/{\sim} \notin \mathrm{bLC}^{\sim}_{\mathrm{WF}_0}$.

In the rest of this section, we restrict our attention to the case where right-hand sides are required to be well formed. The type of grammars we consider is given by the branching synchronization (string) grammars with tables of nesting depth n introduced in [6]. We show that every bracketed link collage language that can be generated using tables of nesting depth n is an (unbracketed) link collage languages that can be generated using nesting depth $n + 1$.

Let us briefly (and somewhat informally) recall the definition of branching grammars. These grammars generalize ET0L systems by turning the set of tables into a tree structure of depth n whose leaves are the tables. Nonterminals can be synchronized with each other at each of the levels $n, \ldots, 0$, which restricts the choice of tables allowed more or less strictly.

As in a context-free grammar, we have disjoint sets T and N of terminals and nonterminals, including an initial nonterminal S. In a branching grammar of nesting depth n, the rules are organized in tables $R(tab)$, where $tab \in J^n$, J being the finite set of so-called table symbols. This gives rise to a hierarchical structure of so-called supertables: for $tab' \in J^m$ with $m \in \{0, \ldots, n\}$, the set $R(tab') = \bigcup\{R(tab) \mid tab \in J^n,\ tab' \text{ is a prefix of } tab\}$ is a *supertable at depth* m.

Another ingredient of a branching grammar is the finite set I of *synchronization symbols*. A *synchronized nonterminal* is a pair $(A, \varphi) \in N \times (I^n)^*$. Its so-called synchronization string φ may alternatively be viewed as an n-tuple of strings

$\varphi_1, \ldots, \varphi_n \in I^*$ of equal length: if $\varphi = (a_{1,1}, \ldots, a_{n,1}), \ldots, (a_{1,m}, \ldots, a_{n,m})$, then $\varphi_l = a_{l,1} \cdots a_{l,m}$ for every $l \in \{1, \ldots, n\}$.

Thus, a branching grammar is a system $G = (N, T, I, J, R, S)$ consisting of the components mentioned above. As in L systems, branching grammars work by replacing every nonterminal in each step. The idea behind branching synchronization is to let the nonterminals in the sentential forms accumulate synchronization strings, i.e., after m derivation steps the sentential form consists of terminals and synchronized nonterminals of length m. Now, if there are two synchronized nonterminals (A, φ) and (A', φ') and $l \in \{0, \ldots, n\}$ is the maximum index such that $(\varphi_1, \ldots, \varphi_l) = (\varphi'_1, \ldots, \varphi'_l)$, then rules from the same supertable at depth l must be applied to them in the next step. In particular, the same table has to be used if $l = n$ (i.e., if the synchronization strings are equal), and arbitrary tables can be used if $l = 0$ (i.e., if $n = 0$ or $\varphi_1 \neq \varphi'_1$).

Every rule $A \to w$ has a nonterminal A as its left-hand side and a string $w \in (T \cup (N \times I^n))^*$ as its right-hand side. In a derivation step $u \to u'$, if the rule is applied to an occurrence of a synchronized nonterminal (A, φ) in u, and w contains an occurrence of (B, ψ), then the resulting synchronized nonterminal in u' is $(B, \varphi\psi)$. The derivation starts with (S, ϵ). Naturally, the generated language consists of all terminal strings that can be derived from (S, ϵ).

A branching grammar is *deterministic* if none of its tables contains two distinct rules with the same left-hand side. The set of all languages over $\Lambda_{[]}$ that can be generated by (deterministic) branching grammars of depth n with well-formed right-hand sides is denoted by WF_n (dWF_n, resp.). This is consistent with the notation used above because the branching grammars of depth 0 can be identified with the context-free grammars in an obvious way. Note also that the (deterministic) branching grammars of depth 1 generalize the (deterministic) ET0L systems. The latter correspond to the case where $J = 1$.

We can now prove the main theorem of this section, whose proof also ends the paper.

Theorem 6. *For every* $n \in \mathbb{N}$, $\mathrm{bLC}_{\mathrm{dWF}_n} = \mathrm{LC}_{\mathrm{dWF}_n}$ *and* $\mathrm{bLC}_{\mathrm{WF}_n} \subseteq \mathrm{LC}_{\mathrm{dWF}_{n+1}}$.

Proof. By Lemma 4.5 of [6] (and the fact that the construction used in its proof preserves well-formedness of right-hand sides), we have $\mathrm{bLC}_{\mathrm{WF}_n} \subseteq \mathrm{bLC}_{\mathrm{dWF}_{n+1}}$. Thus, the second part of the statement is a consequence of the first. Moreover, the inclusion $\mathrm{LC}_{\mathrm{dWF}_n} \subseteq \mathrm{bLC}_{\mathrm{dWF}_n}$ is trivial. Thus, it remains to be proved that $\mathrm{bLC}_{\mathrm{dWF}_n} \subseteq \mathrm{LC}_{\mathrm{dWF}_n}$. For this, assume that we are given a deterministic branching grammar $G = (N, \Lambda_{[]}, I, J, R, S)$ of depth n with well-formed right-hand sides. It is easy to see that we may assume that the right-hand sides of the rules in R do not contain nested brackets. This assumption simplifies the construction of a branching grammar $G' = (N', \Lambda, I, J, R', S)$ of depth n such that $\mathrm{lc}(L(G')) = \mathrm{lc}(L(G))$. We let $N' = N \cup \{A' \mid A \in N\}$, where $A' \notin N$ for each $A \in N$. For every rule in a table $R(tab)$ of G, we let $R'(tab)$ contain two rules, namely one whose left-hand side is A

and one whose left-hand side is A'. Intuitively, if A generates a (string yielding a) link λ, then A' generates $\langle \emptyset, \tau_\lambda^{-1} \rangle$. This is used to undo the transformation component of λ, thus simulating a closing bracket.

To describe this formally, we need the following auxiliary construction: for a string $w \in (N \cup \Lambda)^*$, let w' be obtained from the reverse w^R of w by replacing every $A \in N$ with A', and every $\lambda \in \Lambda$ with $\langle \emptyset, \tau_\lambda^{-1} \rangle$. Now, suppose a table $R(\tau)$ contains the rule $A \to u$. Then $R'(\tau)$ contains

(1) the rule $A \to v$, where v is obtained from u by replacing every substring of the form $[w]$ ($w \in (N \cup \Lambda)^*$) with ww', and
(2) the rule $A' \to u'_0$, where u_0 is the string obtained from u by deleting all substrings of the form $[w]$ ($w \in (N \cup \Lambda)^*$).

Since G' is deterministic, it is not difficult to show that this construction serves its purpose, i.e., $\mathrm{lc}(L(G')) = \mathrm{lc}(L(G))$. Due to limitations of space, this part of the proof is omitted. □

Acknowledgments

I am very grateful to J. Engelfriet, who suggested the notion of links as a common extension of chain-code pictures and collages during personal communication.

References

[1] Jürgen Dassow. Graph-theoretic properties and chain code picture languages. *Journ. Inform. Process. Cybern. EIK.*, 25:423–433, 1989.
[2] Jürgen Dassow. On the connectedness of pictures in chain code picture languages. *Theoretical Computer Science*, 81:289–294, 1991.
[3] Jürgen Dassow and Friedhelm Hinz. Decision problems and regular chain code picture languages. *Discrete Applied Mathematics*, 45:29–49, 1993.
[4] Frank Drewes. Tree-based picture generation. *Theoretical Computer Science*, 246:1–51, 2000.
[5] Frank Drewes. *Grammatical Picture Generation – A Tree-Based Approach*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006.
[6] Frank Drewes and Joost Engelfriet. Branching synchronization grammars with nested tables. *Journal of Computer and System Sciences*, 68:611–656, 2004.
[7] Frank Drewes, Hans-Jörg Kreowski, and Denis Lapoire. Criteria to disprove context-freeness of collage languages. *Theoretical Computer Science*, 290:1445–1458, 2003.
[8] Joost Engelfriet. Bottom-up and top-down tree transformations – a comparison. *Mathematical Systems Theory*, 9:198–231, 1975.
[9] Annegret Habel and Hans-Jörg Kreowski. Collage grammars. In H. Ehrig, H.-J. Kreowski, and G. Rozenberg, editors, *Proceedings of the Fourth Intl. Workshop on Graph Grammars and Their Application to Computer Science*, volume 532 of Lecture Notes in Computer Science, pages 411–429, Springer, 1991.
[10] Hermann A. Maurer, Grzegorz Rozenberg, and Emo Welzl. Using string languages to describe picture languages. *Information and Control*, 54:155–185, 1982.
[11] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The Algorithmic Beauty of Plants*. Springer, 1990.