Chapter 11

PICTURE GENERATION BY COLLAGE GRAMMARS

F. DREWES H.-J. KREOWSKI

University of Bremen, Department of Computer Science,
P.O. Box 33 04 40, D-28334 Bremen, Germany
E-mail: {drewes,kreo}@informatik.uni-bremen.de

The purpose of this chapter is to give an introduction to collage grammars—picture generating devices whose generation mechanism is based on hyperedge replacement as known from the area of graph transformation. We summarize the main results that have, so far, been obtained for collage grammars and the languages they generate. In particular, we address structural properties, decision problems, comparisons with other picture-generating devices, and aspects relating collage grammars to concepts from fractal geometry.

Contents

11.1 Introduction

As picture generation is one of the central issues of computer graphics and grammars are well-known and successful devices for the generation of objects of various kinds, one encounters quite a variety of syntactic approaches to the generation of artificial pictures and classes of pictures in the literature (see, e.g., the books by Barnsley [1], Edgar [17], Falconer [18], Gips [19], Grünbaum and Shephard [20], Mandelbrot [31], Peitgen, Jürgens, and Saupe [34], Prusinkiewicz and Lindenmayer [36], Stiny [38], Toffoli and Margolus [41], and Wolfram [42]). Collage grammars belong to this variety and are used in this survey in order to explain some of the principles and possibilities of picture generation. They show some specific characteristics that seem to be worth mentioning beforehand.

(1) Collage grammars allow to derive collages from collages, that is, they work directly on pictorial objects rather than on representations of some kind that need an extra pictorial interpretation.

(2) The derivation process of collage grammars employs the concept of hyperedge replacement which stems from the field of graph grammars. Hence, the study of collage grammars can be seen as an attempt to carry over some experience in graph grammar theory to computer graphics.

(3) As hyperedge replacement is a context-free mode of rewriting, the theory of collage grammars presents various results with counterparts in the theory of context-free Chomsky grammars and of Lindenmayer systems.

A collage consists of a set of parts and a sequence of pin points. A part may be an arbitrary set of points (in a Euclidean space)—usually taken from some standard set of geometric objects like line segments, circles, triangles, polygons, polyhedra, etc. that have simple finite descriptions and are easy to deal with on graphical user interfaces. The pin points are used to paste collages into collages. A collage represents a picture by the overlay of its parts.

To generate collages from collages by the application of productions, they are decorated with hyperedges in intermediate steps. A hyperedge is a labelled item with an ordered number of tentacles each of which is attached to a point. A hyperedge serves as a place holder. It may eventually be replaced by a decorated collage, provided that there is a transformation of the pin points into the attachment points of the hyperedge. This kind of hyperedge replacement establishes the rewrite steps of a collage grammar if the label of the replaced hyperedge and the replacing decorated collage form a production. So, a collage grammar generates a set of collages in the usual way of language generation. As collages represent pictures, a set of pictures is generated at the same time. The overlay of all generated pictures and their intersection yields two particular pictures that are often of a fractal nature.

Most of the results reported in this survey can be found in [22,23,40,13,7,14, 15,8]. The structure of the chapter is as follows. The concept of collage grammars is recalled in Section 11.2. In particular, a useful normal form of collage grammars is given. In Section 11.3 variations and extensions are discussed, namely regulation by context and the table-driven parallel mode of rewriting. Section 11.4 provides a context-freeness lemma that justifies calling collage languages context-free if they are generated by unconstrained collage grammars. The context-freeness lemma yields a fixed-point theorem and allows to introduce a notion of derivation trees. In Section 11.5, further properties that all context-free collage languages have in common are studied. These properties can be used as criteria to show that certain collage languages are not context-free. In Section 11.6 we summarize some results about the decidability or undecidability of questions concerning properties of context-free collage languages. Section 11.7 is devoted to the pictures obtained by taking the overlay or intersection of all pictures given by the collages in a context-free collage language. The presented results show that these pictures have indeed a fractal nature because, for instance, all self-affine fractals can be specified by a particular kind of collage grammar. In Section 11.8, we relate collage grammars with other picture generating and modelling devices, namely iterated function systems (as known from fractal geometry) and chain-code picture languages. Finally, Section 11.9 contains some information about existing implementations of collage grammars. In Section 11.10 a few concluding remarks are given.

11.2 Collage Grammars

In this section, the basic notions and notations concerning collages and collage grammars (cf. [22,23]) are recalled and illustrated by an example.

For a set X, $|X|$ denotes its cardinality, X^* denotes the set of all finite sequences over X, including the empty sequence λ, $\wp(X)$ denotes its powerset, and $X \setminus Y$ denotes the complement of a set Y in X. For a function $f\colon A \to B$, the canonical extensions of f to $\wp(A)$ and to A^* are denoted by f as well, i.e., $f(S) = \{f(a) \mid a \in S\}$ for $S \subseteq A$ and $f(a_1 \cdots a_n) = f(a_1) \cdots f(a_n)$ for $a_1, \ldots, a_n \in A$. The composition of functions $f\colon A \to B$ and $g\colon B \to C$ is denoted by $f \circ g$, where $f \circ g(a) = g(f(a))$ for all $a \in A$.

As usual, for a binary relation $\Longrightarrow \subseteq S \times S$ one may write $s \Longrightarrow s'$ instead of $(s, s') \in \Longrightarrow$ and $s_0 \Longrightarrow^n s_n$ or $s_0 \Longrightarrow^* s_n$ to abbreviate $s_0 \Longrightarrow s_1 \Longrightarrow \cdots \Longrightarrow s_n$.

The sets of natural numbers, integers, rational numbers, and real numbers are denoted by $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, and $\mathbb{R}$, respectively. By $\mathbb{N}_+$ we denote the set $\mathbb{N} \setminus \{0\}$.

Familiarity with the basic notions of Euclidean geometry is assumed (see, e.g., Coxeter [3]). $\mathbb{R}^d$ denotes the Euclidean space of dimension d for some $d \geq 1$. $\mathbb{R}^d$ is equipped with the usual Euclidean distance function $dist\colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. By δP we denote the boundary of a compact set P of points, and $[P]$ denotes the convex hull of a set or sequence P of points.

11.2.1 Collages and Collage Grammars

A collage consists of a set of parts being geometric objects, and a sequence of so-called pin points. To generate sets of collages, they are decorated with hyperedges. A hyperedge has a label and an ordered finite set of tentacles, each of which is attached to a point. A handle—a collage consisting of a single hyperedge—is particularly useful as initial collage. Each collage specifies a picture, called induced pattern, by the overlay of all its parts. More precisely, all these notions are defined as follows.

A *collage* (in $\mathbb{R}^d$) is a pair $(PART, pin)$, where $PART \subseteq \wp(\mathbb{R}^d)$ is a finite set of so-called *parts* and $pin \in (\mathbb{R}^d)^*$ is the sequence of *pin points*. $\mathcal{C}$ denotes the class of all collages.

Let N be a set of *labels*. A *(hyperedge-)decorated collage* (over N) is a construct $C = (PART, E, att, lab, pin)$ such that $(PART, pin)$ is a collage, E is a finite set of *hyperedges*, $att\colon E \to (\mathbb{R}^d)^*$ is a mapping, called the *attachment*, and $lab\colon E \to N$ is a mapping, called the *labelling*. The class of all decorated collages over N is denoted by $\mathcal{C}(N)$, and $PART_C$, E_C, att_C, lab_C, and pin_C denote the components of $C \in \mathcal{C}(N)$.

A collage can be seen as a decorated collage C where $E_C = \emptyset$. In this sense, $\mathcal{C} \subseteq \mathcal{C}(N)$. Thus, we may drop the components E_C, att_C, and lab_C if $E_C = \emptyset$. Furthermore, we will often use the term *collage* also in the case of decorated collages as long as this is not a source of misunderstandings.

Example 11.1 (collage)

Collages will be depicted as shown in Figure 11.1. Pin points and attached points of hyperedges are drawn as small bullets. The pin points are numbered according to their order in the sequence of pin points. A hyperedge is represented as a square carrying its label inside, together with numbered lines indicating the attachment. If the sequence of attached nodes is obtained from the sequence of pin points by scaling and translation only, then the numbers are dropped (see the B-labelled hyperedge). □

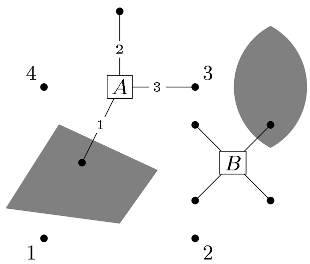


Figure 11.1: A collage

Let $A \in N$ and $pin \in (\mathbb{R}^d)^*$. The *handle induced by* A and pin is the decorated collage $(A, pin)^\bullet = (\emptyset, \{e\}, att, lab, pin)$ with $lab(e) = A$ and $att(e) = pin$. The overlay of the set of all parts of a collage C yields the *induced pattern* $pattern(C) = \bigcup_{part \in PART_C} part$. Removing a set $B \subseteq E_C$ of hyperedges from a collage yields the collage $C - B = (PART_C, E_C \setminus B, att, lab, pin_C)$, where att and lab are the restrictions of att_C and lab_C to $E_C \setminus B$, respectively.

The addition of a collage C' to a collage C is defined by $C + C' = (PART_C \cup PART_{C'}, E_C \uplus E_{C'}, att, lab, pin_C)$, where $\uplus$ denotes the disjoint union of sets,

$$att(e) = \begin{cases} att_C(e) & \text{for } e \in E_C \\ att_{C'}(e) & \text{for } e \in E_{C'}, \end{cases}$$

and

$$lab(e) = \begin{cases} lab_C(e) & \text{for } e \in E_C \\ lab_{C'}(e) & \text{for } e \in E_{C'}. \end{cases}$$

As an abbreviation, one may write $C + \sum_{i=1}^{n} C_i$ for $C + C_1 + \cdots + C_n$. The transformation of a collage C by a transformation $a\colon \mathbb{R}^d \to \mathbb{R}^d$ is given by $a(C) = (a(PART_C), E_C, att_C \circ a, lab_C, a(pin_C))$.

Hyperedges in decorated collages serve as place holders for (decorated) collages. Hence, the key construction is the replacement of a hyperedge in a decorated collage with a (decorated) collage. While a hyperedge is attached to some points, a (decorated) collage is equipped with a number of pin points. If there is an affine transformation that maps the pin points to the attached points of the hyperedge, the transformed (decorated) collage may replace the hyperedge.

Let $C \in \mathcal{C}(N)$ be a collage and let $B \subseteq E_C$. Furthermore, let $repl\colon B \to \mathcal{C}(N)$ be a mapping such that, for all $e \in B$, there is a unique affine transformation $a(e)$ that satisfies $att_C(e) = a(e)(pin_{repl(e)})$. Then the *replacement of* B *in* C *through repl* yields the decorated collage $C[repl]$ constructed by

1. removing the hyperedges in B from C,
2. transforming $repl(e)$ by $a(e)$ for all $e \in B$, and
3. adding the transformed collages $a(e)(repl(e))$ to $C - B$.

Thus, to be more precise, $C[repl] = (C - B) + \sum_{e \in B} a(e)(repl(e))$.

Whenever a notion of replacement like the one above is given, one may easily define productions, grammars, and derivations. In our particular case this leads to the notions of collage grammars and languages.

Let N be a set of labels. A *production* (over N) is a pair $p = (A, R)$ with $A \in N$ and $R \in \mathcal{C}(N)$. A is called the *left-hand side* of p and R is its *right-hand side*, denoted by $lhs(p)$ and $rhs(p)$, respectively. A production $p = (A, R)$ is also denoted by $A \underset{p}{::=} R$ or simply $A ::= R$. Furthermore, $A ::= R_1 \mid \cdots \mid R_n$ abbreviates $A ::= R_1, \ldots, A ::= R_n$.

Let $C \in \mathcal{C}(N)$, $B \subseteq E_C$, and let P be a set of productions over N. We call $b \colon B \to P$ a *base* on B in C if, for all $e \in B$,

(1) $lab_C(e) = lhs(b(e))$ and

(2) there is a unique affine transformation $a(e)$ that satisfies the equation $att_C(e) = a(e)(pin_{rhs(b(e))})$.

C directly derives $C' \in \mathcal{C}(N)$ through b if $C' = C[repl]$, where $repl(e) = rhs(b(e))$ for all $e \in B$. A direct derivation is denoted by $C \Longrightarrow_P C'$ or $C \Longrightarrow C'$ if P is clear from the context. A sequence $C \Longrightarrow^* C'$ of direct derivations is called a *derivation* from C to C'.

Definition 11.2.1 (*collage grammar*)

A *collage grammar* is a system $G = (N, P, Z)$, where N is a finite set of *nonterminal labels* (or *nonterminals*, for short), P is a finite set of productions (over N), and $Z \in \mathcal{C}(N)$ is a decorated collage, called the *axiom*.

The *collage language generated by G* consists of all collages which can be derived from Z by applying productions of P:

$$L(G) = \{C \in \mathcal{C} \mid Z \overset{*}{\underset{P}{\Longrightarrow}} C\}.$$

Example 11.2

Consider the collage grammar $G = (\{S\}, \{p_1, p_2\}, Z)$, where Z, p_1, and p_2 are as shown in Figure 11.2. It generates collages that can be seen as approximations of the well-known Sierpiński gasket. Some of the collages in this language are depicted in Figure 11.3. (Like in this figure, we will often scale down pictures of generated collages in order to save space.) □

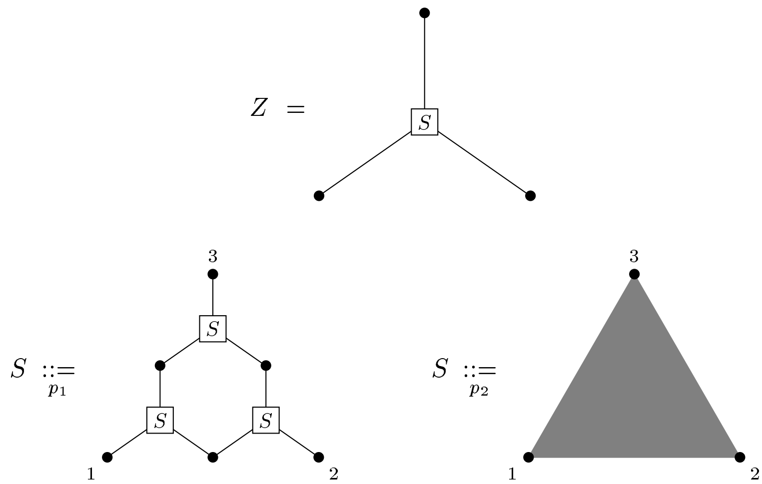


Figure 11.2: The axiom and the productions used in Example 11.2

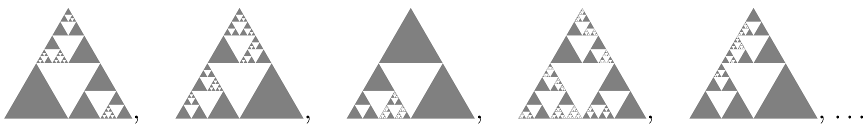


Figure 11.3: Approximations of the Sierpiński gasket generated by the collage grammar considered in Example 11.2

11.2.2 Proper Collage Grammars

By the definition of direct derivations the application of a production to a hyperedge e requires the existence of an appropriate base on $\{e\}$. Thus, a replacement may take place only if there is a unique affine transformation that maps the pin points of the right-hand side to the attached points of e. This fact is sometimes inconvenient because it means that the applicability of a production to a hyperedge cannot be decided by comparing labels only. Fortunately, every collage grammar can be turned into an equivalent one in which productions with appropriate left-hand sides can always be applied. Two further, also quite useful normalizations affect the axiom and the pin points of right-hand sides: The axiom may be turned into a handle, and the pin points of right-hand sides of productions having the same left-hand side can be made equal. Thus, every nonterminal A has its associated sequence pin_A of pin points.

Definition 11.2.2 (*proper collage grammar*)
A collage grammar $G = (N, P, Z)$ is said to be *proper* if

1. $Z = (S, pin_S)^\bullet$ for some $S \in N$ and $pin_S \in (\mathbb{R}^d)^*$,
2. for all $C \in \{Z\} \cup \{rhs(p) \mid p \in P\}$, every $b\colon E_C \to P$ with $lhs(b(e)) = lab_C(e)$ for all $e \in E_C$ is a base on B in C, and
3. for all $p, p' \in P$, $lhs(p) = lhs(p')$ implies $pin_{rhs(p)} = pin_{rhs(p')}$.

In the following, we denote by pin_A the sequence of pin points associated with a nonterminal A in a given proper collage grammar (N, P, Z), as in Definition 11.2.2 above.

Clearly, if $G = (N, P, Z)$ is a collage grammar then all collages in $L(G)$ have the same pin points, as they all inherit them from the axiom Z of G. Furthermore, as the pin points of the axiom never play any role in the derivation process, for every other sequence pin of pin points the language $\{(PART_C, pin) \mid C \in L(G)\}$ is generated by (N, P, Z'), where Z' equals Z except that $pin_{Z'} = pin$. As a consequence, in the context discussed here it is meaningless to distinguish between collage languages whose elements differ only in their sequences of pin points. Therefore, let us say that two collage languages L and L' are equivalent, $L \cong L'$, if $\{PART_C \mid C \in L\} = \{PART_C \mid C \in L'\}$. Now, the following lemma, which is due to Habel and Kreowski [22, Theorem 4.3], can be shown.

Lemma 11.2.3
For every collage grammar G one can effectively construct a proper collage grammar G' such that $L(G') \cong L(G)$. □

The construction of G' merely consists in a refinement of the set of nonterminal labels in order to satisfy the third requirement given in the definition. In the following we assume that collage grammars are proper whenever this is convenient.

Notice that, now, the emptiness of the language generated by a collage grammar can be decided by converting the grammar into a proper one, and then using the well-known algorithm from the theory of context-free string languages in order to decide whether there is at least one terminating derivation. Hence, Lemma 11.2.3 yields the following corollary.

Corollary 11.2.4
There is an algorithm that takes as input a collage grammar G and decides whether $L(G)$ is empty. □

11.3 Variations and Extensions

We shall now briefly discuss two generalized types of collage grammars, namely the table-driven and the context-sensitive ones. The reader may skip this section at first reading and return to it later, because the rest of this chapter mainly deals with the ordinary type of collage grammars.

The definition of table-driven collage grammars is inspired by the well-known notion of TOL systems, i.e., table-driven context-free Lindenmayer systems (cf. [37,36,28]). This is why we shall call these grammars TOL collage grammars.

Definition 11.3.1 (*TOL collage grammar*)
A *TOL collage grammar* is a system $G = (N, T, Z)$, where N is the finite set of nonterminals, $Z \in \mathcal{C}(N)$ is the axiom, and T is a finite set of *tables*, every table $P \in T$ being a finite set of productions over N such that

(1) $\{lhs(p) \mid p \in P\} = N$ and
(2) the collage grammar $(N, \bigcup T, Z)$ is proper.

For collages $C, C' \in \mathcal{C}(N)$ we write $C \Longrightarrow_T C'$ if C derives C' through a base $b\colon E_C \to P$ for some $P \in T$. The collage language generated by G is

$$L(G) = \{C \in \mathcal{C} \mid Z \overset{*}{\underset{T}{\Longrightarrow}} C\}.$$

As in the case of ordinary collage grammars, one may write $C \Longrightarrow C'$ for $C \Longrightarrow_T C'$ if T is understood. Notice that every direct derivation is required to replace *all* hyperedges of a collage in parallel. This is essential because otherwise the division of the set of productions into tables would be meaningless. Notice also that every table contains at least one production for each left-hand side $A \in N$. Together with the fact that $(N, \bigcup T, Z)$ is proper this implies that every table can be applied at any step of a derivation.

It should be obvious that TOL collage grammars can be considered as a generalization of collage grammars: For every proper collage grammar (N, P, Z) the TOL collage grammar $(N, \{P \cup P_{id}\}, Z)$ with $P_{id} = \{A ::= (A, pin_A)^\bullet \mid A \in N\}$ generates the same language.

Example 11.3
As an example, consider the TOL collage grammar $G = (\{L\}, \{\{p_1, p_1'\}, \{p_2\}, \{p_1'\}\}, (L, pin)^\bullet)$, where the productions are as shown in Figure 11.4 and $pin = (0,0)(1,0)(0,1)$. One of the generated collages is shown in Figure 11.5. □

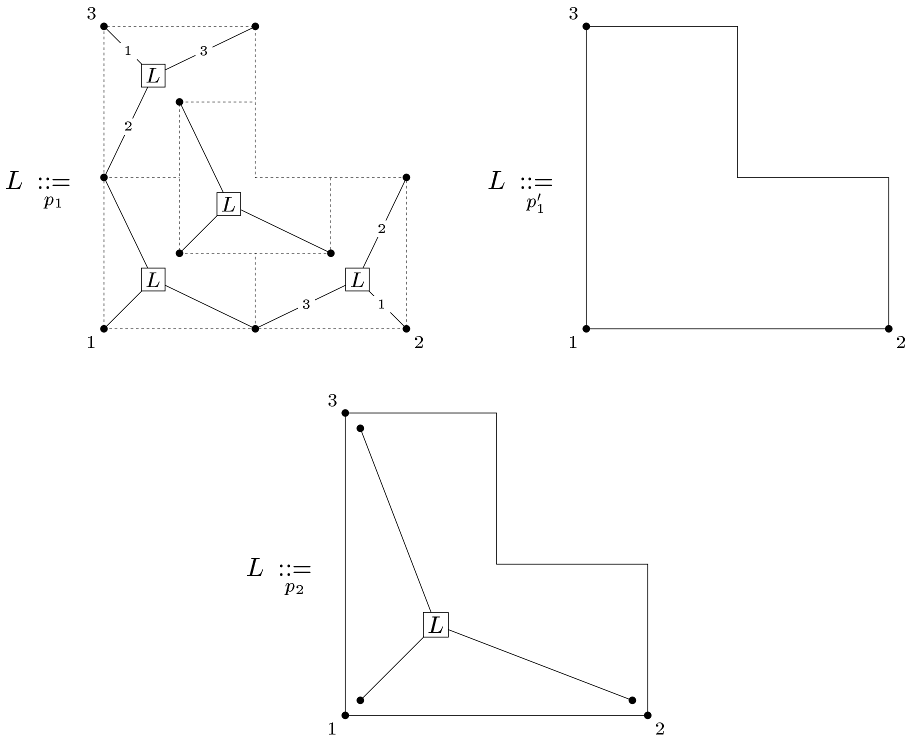


Figure 11.4: The productions used in Example 11.3 (the dashed lines in $rhs(p_1)$ do not belong to the collage; they are added for clearness' sake)

Let us now turn to the discussion of context-sensitive collage grammars. Intuitively, collage grammars as defined in Section 11.2.1 are context-free devices since productions replace atomic items (i.e., hyperedges) in a way that neither depends on nor affects the rest of the collage. A context-sensitive version of collage grammars should also replace hyperedges without affecting the rest of the collage, but one should be able to specify that a production can only be applied if certain items (specified in the production) are present in the neighbourhood of the replaced hyperedge.

In the case of Chomsky grammars the applicability of productions may depend on the presence of both terminal and nonterminal context. In fact, for Chomsky grammars it is well known that context-sensitive grammars with terminal context only can be simulated by context-free grammars. For collage grammars the situation turns out to be different because, intuitively, hyperedges can move

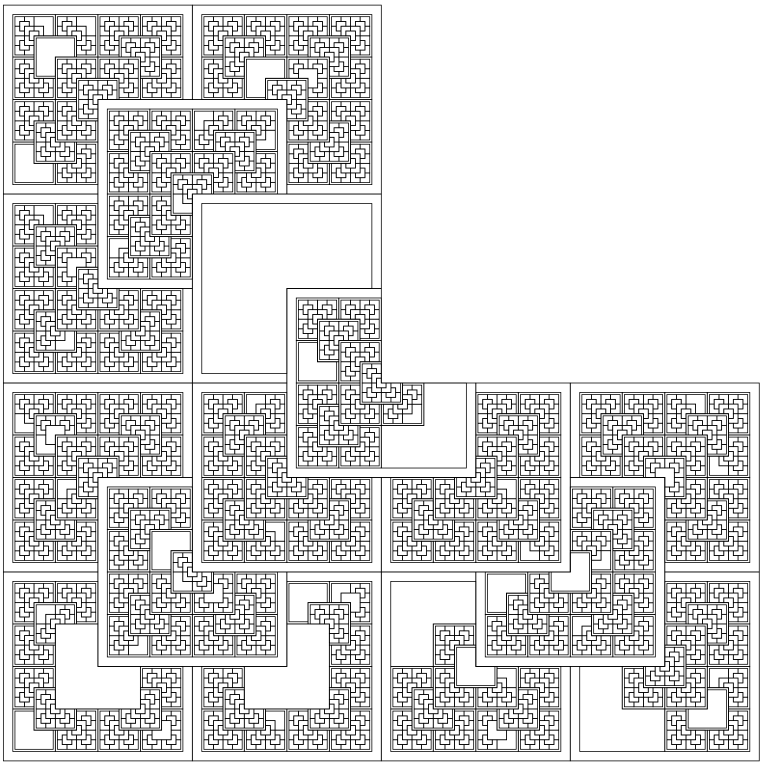


Figure 11.5: One of the collages generated by the TOL collage grammar in Example 11.3

around. Even if the context is required to consist entirely of parts (i.e., one cannot ask for the presence of other hyperedges) context-sensitivity leads to a proper increase in power. Moreover, the more restricted version has nicer properties than the one in which the required context is allowed to contain hyperedges, too. This is why we shall only discuss this type of context-sensitivity, which we call part-sensitivity.

A *part-sensitive production* $p = (L ::= R)$ over N is a pair of collages $L, R \in \mathcal{C}(N)$ such that $E_L = \{e\}$ for some hyperedge e and $att_L(e) = pin_R$. The collage $K(p) = L - \{e\}$ is then called the *context collage* and $U(p) = (lab_L(e), R)$ is its *underlying production*. For a set P of part-sensitive productions we define $U(P) = \{U(p) \mid p \in P\}$.

Let P be a set of part-sensitive productions and consider a mapping $b\colon B \to$

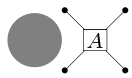


Figure 11.6: The axiom Z of the part-sensitive collage grammar considered in Example 11.4

P, where $B \subseteq E_C$ for a given collage C. Assume that the mapping $b' = b \circ U$ is a base on B in C, C' is the collage such that $C \Longrightarrow C'$ through b' and, for every $e \in B$, $a(e)$ is the unique affine transformation that satisfies $att_C(e) = a(e)(pin_{rhs(b'(e))})$. Then C *directly derives* C' *through* b, $C \Longrightarrow_P C'$, if $a(e)(K(b(e))) \subseteq C$ for all $e \in B$ (where $C_0 \subseteq C$ means $PART_{C_0} \subseteq PART_C$ for $C_0 \in \mathcal{C}$).

Thus, intuitively, a part-sensitive production p applies to a hyperedge $e \in E_C$ if the underlying production $U(p)$ applies to e and the transformed context collage $K(p)$ is contained in C.

Definition 11.3.2 (*part-sensitive collage grammar*)

A *part-sensitive collage grammar* is a system $G = (N, P, Z)$, where N is the finite set of nonterminals, $Z \in \mathcal{C}(N)$ is the axiom, and P is a finite set of part-sensitive productions over N. The collage language generated by G is

$$L(G) = \{C \in \mathcal{C} \mid Z \overset{*}{\underset{P}{\Longrightarrow}} C\}.$$

Clearly, if $PART_{K(p)} = \emptyset$ for a part-sensitive production p then this production can be identified with the ordinary production $U(p)$. Thus, a collage grammar may be seen as a part-sensitive collage grammar whose productions have trivial context collages. In this sense, part-sensitive collage grammars generalize collage grammars.

Example 11.4

Consider the part-sensitive collage grammar $G = (\{A, B, C\}, P, Z)$, where Z is as shown in Figure 11.6 and P consists of the productions shown in Figure 11.7. Notice that it makes no difference whether the context parts in the left-hand side of a production are repeated in the right-hand side (as in this case) or not. Since these parts must already be present in order to apply the production, including them in the right-hand side is redundant but harmless (and maybe more intuitive because the effect of a production is then just what you see in the picture).

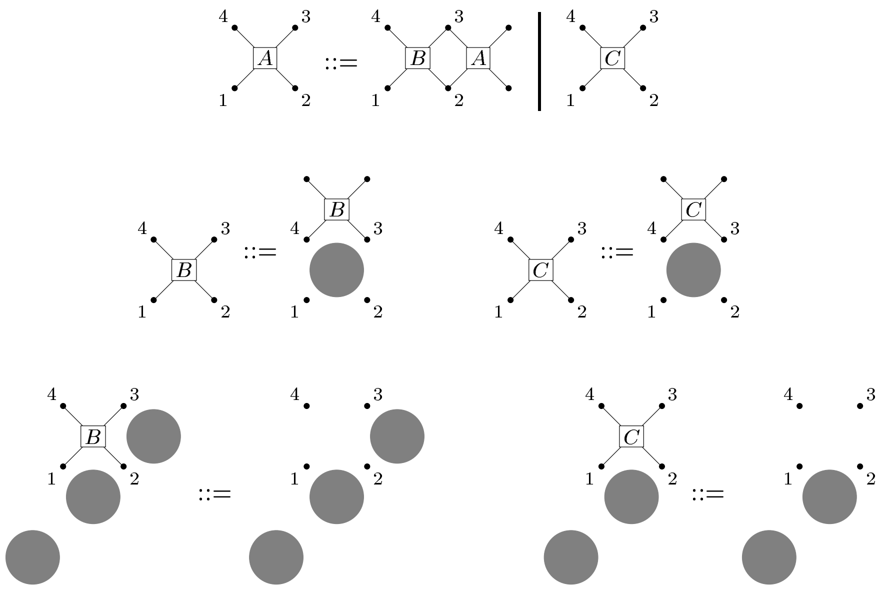


Figure 11.7: The productions of the part-sensitive collage grammar of Example 11.4

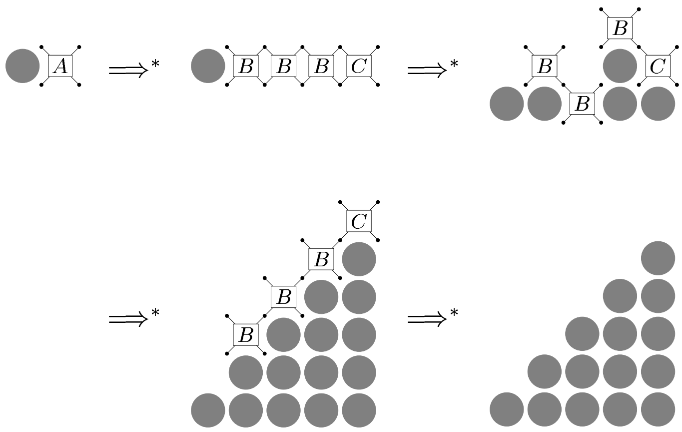


Figure 11.8: A derivation using the part-sensitive productions shown in Figure 11.7

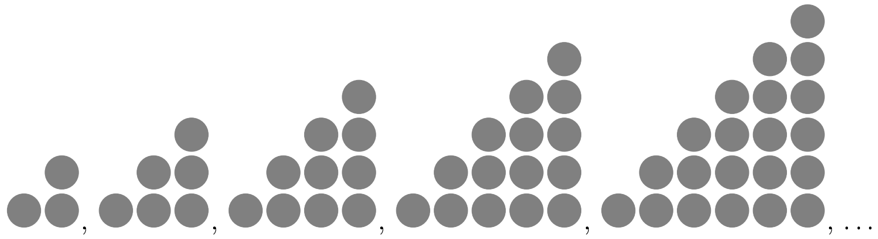


Figure 11.9: The collages generated by the part-sensitive collage grammar in Example 11.4

A derivation is shown in Figure 11.8. It is not hard to verify that the language generated by this grammar consists of all isosceles triangles of the kind depicted in Figure 11.9. It will be shown in Section 11.5 that this language cannot be generated by an ordinary collage grammar. □

It should be observed that for every direct derivation $C \Longrightarrow C'$ through a base $b\colon B \to P$ (where P is a set of part-sensitive productions) and every partition of B into disjoint subsets B_1, B_2, it holds that $C \Longrightarrow C'' \Longrightarrow C'$ through the restrictions of b to B_1 and B_2, respectively. This is because $PART_C \subseteq PART_{C''}$ (i.e., parts which have already been generated never get lost), which implies that every production $b(e)$, $e \in B_2$, can be applied to e in C'' because it can be applied to e in C. Thus, we have the following sequentialization theorem.

Theorem 11.3.3
Let $C \Longrightarrow_P^* C'$, where C and C' are collages and P is a set of part-sensitive productions. Then there is a derivation $C = C_0 \Longrightarrow_P C_1 \Longrightarrow_P \cdots \Longrightarrow_P C_n = C'$ such that, for all $i \in \{1, \ldots, n\}$, $C_{i-1} \Longrightarrow_P C_i$ through a base $b_i\colon B_i \to P$ for which $|B_i| = 1$. □

11.4 Context-Freeness

Let us now return to the ordinary type of collage grammars. As mentioned in Section 11.3 these grammars are, in an intuitive sense, context-free devices. In this section we shall show how this can be captured formally. We discuss three notions which are quite closely related: the context-freeness lemma, derivation trees, and a characterization of generated languages as least fixed points of systems of formal equations.

11.4.1 The Context-Freeness Lemma

In order to obtain a precise statement that captures the essence of context-freeness, consider a proper collage grammar. If a derivation starts in the handle $(A, pin_A)^\bullet$ for some $A \in N$, the first non-trivial step (i.e., the first one with a non-empty base) applies a production to the unique hyperedge and derives the right-hand side R of the production. The remaining derivation can be restricted to the induced handles $(lab_R(e), att_R(e))^\bullet$ $(e \in E_R)$ of R and can be reconstructed as the embedding of all the restrictions into R. Moreover, each restriction can be transformed into a derivation starting in the handle $(lab_R(e), pin_{lab_R(e)})^\bullet$. This reasoning results in the context-freeness lemma.

Lemma 11.4.1 (context-freeness lemma, cf. [22])
Let $G = (N, P, Z)$ be a proper collage grammar and let $A \in N$ and $C \in \mathcal{C}$. Then there is a derivation $(A, pin_A)^\bullet \Longrightarrow^n C$ if and only if there is a production $(A, R) \in P$ and, for each $e \in E_R$, there is derivation $(lab_R(e), pin_{lab_R(e)})^\bullet \Longrightarrow^{n(e)} C_e$ such that $n(e) < n$ and $C = R[repl]$ with $repl(e) = C_e$ for $e \in E_R$. □

Justified by the context-freeness lemma, collage languages that can be generated by a collage grammar are called *context-free collage languages* in the following. If a collage C is generated from a handle as in the lemma (and the derivation is clear from the context), then the notation C_e is used to denote the component of C generated from a hyperedge e.

11.4.2 Derivation Trees

The context-freeness implies the possibility to use derivation trees as a convenient representation of derivations. For this, let us consider some derivation $(A, pin_A)^\bullet \Longrightarrow^* C$. If $k = 0$ then $C = (A, pin_A)^\bullet$ and we may represent the derivation by a one-node tree whose only node is the label A. Otherwise, the context-freeness lemma states that the derivation has the form $(A, pin_A)^\bullet \Longrightarrow R \Longrightarrow^* R[repl]$, where (A, R) is a production and $(lab_R(e), pin_{lab_R(e)})^\bullet \Longrightarrow^* repl(e)$ for all $e \in E_R$. Thus, the derivation gives rise to a tree whose root is the production (A, R) and whose subtrees are the trees obtained recursively from the shorter derivations. Thus, derivation trees represent derivations up to some reordering of direct derivation steps (which does not affect the result of a derivation, as we know). The resulting collage of a derivation tree is a handle if the derivation tree a single node labelled with a nonterminal. Otherwise, the result is $R[repl]$, where R is the right-hand side of the production in its root and $repl(e)$ is recursively obtained as the result of the subtree corresponding to $e \in E_R$. Formally, this amounts to the following definition.

Definition 11.4.2 (*derivation tree*)

Let $G = (N, P, Z)$ be a proper collage grammar. The set of *derivation trees* of G is the smallest set S satisfying the following requirements.

- $N \subseteq S$ with $root(A) = A$ and $result(A) = (A, pin_A)^\bullet$ for all $A \in N$.
- For every production $(A, R) \in P$ and every mapping $branch\colon E_R \to S$ such that $lab_R(e) = root(branch(e))$ for all $e \in E_R$, the triple $t = (A, R, branch)$ is in S. In this case, $root(t) = A$ and $result(t) = R[repl]$ where, for all $e \in E_R$, $repl(e) = result(branch(e))$.

The reader should notice that a derivation tree contains nonterminal labels as leaves if and only if its result is a collage containing hyperedges (i.e., a decorated collage). The theorem below states the expected correspondence between derivations and derivation trees. The proof is by a straightforward induction using the context-freeness lemma for both directions.

Theorem 11.4.3

Let $G = (N, P, Z)$ be a proper collage grammar, let $A \in N$ and $C \in \mathcal{C}(N)$. Then there is a derivation $(A, pin_A)^\bullet \Longrightarrow^* C$ if and only if there is a derivation tree t of G such that $root(t) = A$ and $result(t) = C$. □

Theorem 11.4.3 can, in fact, be seen as a reformulation of the context-freeness lemma because the two statements are obviously equivalent. Another such reformulation is discussed next.

11.4.3 Context-Free Collage Languages as Fixed Points

In [22] it is shown that context-free collage languages are fixed points of systems of equations induced by the generating grammars. Like Theorem 11.4.3 this result is a mere reformulation of the context-freeness lemma, which can be seen as follows.

Consider a proper collage grammar $G = (N, P, Z)$. With each $A \in N$, one may associate the collage grammar $G_A = (N, P, (A, pin_A)^\bullet)$ by using the handle induced by A and pin_A as axiom instead of Z. This family of grammars $(G_A)_{A \in N}$ yields the family of generated collage languages $L^G = (L(G_A))_{A \in N}$.

On the other hand, consider the set $rhs_G(A)$ of all right-hand sides of productions in P with the left-hand side $A \in N$. Then the family $EQ(G)$ of formal equations $(A = rhs_G(A))_{A\in N}$ can be seen as a system of collage language equations (induced by G) in the following way. Given a family of collage languages $L = (L(A))_{A\in N}$ with $pin_C = pin_A$ for all $C \in L(A)$ and $A \in N$, $rhs_G(A)$ yields a collage language by replacing hyperedges of collages in $rhs_G(A)$ with corresponding collages in L in all possible ways, i.e.,

$$L(rhs_G(A)) = \bigcup_{R\in rhs_G(A)} \{R[repl] \mid repl(e) \in L(lab_R(e)) \text{ for all } e \in E_R\}.$$

The language family L is called a *solution* or a *fixed point* of $EQ(G)$ if $L(A) = L(rhs_G(A))$ for all $A \in N$. As a corollary of the context-freeness lemma, we get that the family L^G is a fixed point of $EQ(G)$. To see this, notice the following. By definition, $C \in L(G_A)$ means that there is a derivation $(A, pin_A)^\bullet \Longrightarrow^* C$. This is the case if and only if $C = R[repl]$ for some $(A, R) \in P$ and $repl\colon E_R \to \mathcal{C}$ with $(lab_R(e), pin_{lab_R(e)})^\bullet \Longrightarrow^* repl(e)$ for all $e \in E_R$. This can be reformulated as $C = R[repl]$ for some $R \in rhs_G(A)$ and $repl\colon E_R \to \mathcal{C}$ with $repl(e) \in L^G(lab_R(e))$ for all $e \in E_R$, which means $C \in L^G(rhs_G(A))$.

Moreover, using again the context-freeness lemma, one can easily show by induction on the length of derivations that $L(G_A) \subseteq L(A)$ for every $A \in N$ provided that $(L(A))_{A\in N}$ is a fixed point of $EQ(G)$. Altogether, this proves the following result.

Theorem 11.4.4 (fixed point theorem)
Let $G = (N, P, Z)$ be a proper collage grammar and let $EQ(G)$ be the associated system of equations. Then the family L^G of generated languages is the smallest fixed point of $EQ(G)$. □

As a consequence, one gets an alternative way to generate context-free collage languages because the smallest fixed point of $EQ(G)$ can also be obtained iteratively. For this, let L_i ($i \in \mathbb{N}$) be defined as follows:

(i) $L_0(A) = \emptyset$ and
(ii) $L_{i+1}(A) = L_i(rhs_G(A))$ for $i \in \mathbb{N}$.

Then the family $L^* = (L^*(A))_{A\in N}$ with $L^*(A) = \bigcup_{i\in\mathbb{N}} L_i(A)$ equals L^G.

11.5 Necessary Conditions for Context-Freeness

The results reported in this section can be found in [15,8]. We discuss two criteria that can be applied to disprove context-freeness of collage languages.

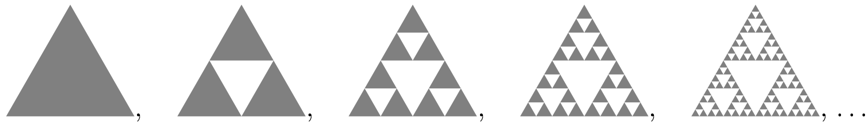


Figure 11.10: The iteration sequence of the Sierpiński gasket

The first criterion concerns the numbers of parts in a language generated by a collage grammar. As one would expect, the number of parts grows at most linearly. The second criterion concerns the volumes of such parts, which, roughly speaking, can only grow or shrink exponentially.

Consider, for example, the typical iteration sequence of the Sierpiński gasket shown in Figure 11.10, forming the collage language $L_{Sierpiński}$. As replacements of different nonterminals by the productions of a collage grammar are completely independent of each other (cf. the context-freeness lemma), one should expect that such a language cannot be generated. Intuitively, the refinement is too uniform to be the result of a context-free derivation process. In order to prove that, indeed, such a language cannot be generated by a collage grammar, we shall concentrate on the fact that it involves a non-linear growth concerning the number of parts of collages: the elements of this language have 3^i parts ($i \in \mathbb{N}$).

Instead of the number of parts one may also consider the volumes of parts which occur in the collages of a language. If an affine transformation f is applied to a part, then the volume of this part is multiplied by a factor α depending on f (namely, the determinant of the matrix associated with f). Thus, by iteration one obtains factors like $\alpha, \alpha^2, \alpha^3, \ldots$. This makes it easy to generate a language whose collages consist of, e.g., squares of side length 2^n. However, it seems to be impossible to achieve a modest growth of parts. Consider, for example, the collage language L_{square} indicated in Figure 11.11 each member of which is a square of integer side length and where all positive integers occur as side lengths. Such a language cannot be generated since finitely many affine transformations in the productions of a collage grammar do not suffice to describe all these parts. In fact, the argument shows that it is not even possible to generate any language in whose collages all these squares appear. Similarly, the set of all squares of side length $1/n$ ($n \in \mathbb{N}_+$) cannot be generated.

In analogous situations concerning string or graph languages, pumping lemmas are often helpful. To recall the basic argument, a sufficiently long string in a context-free language induces three derivations of the form $S \to^* uAy$, $A \to^* vAx$, $A \to^* w$ where u, v, w, x, and y are terminal strings and vx is not

Figure 11.11: Collages that consist of squares with integer side lengths

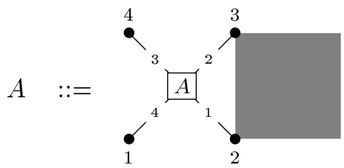


Figure 11.12: A production of a collage grammar

empty. Then one gets by induction that the strings uv^iwx^iy for $i \in \mathbb{N}$ belong to the given language, which in particular shows that the length of strings in the language grows at most linearly (and never quadratically or exponentially). In the case of collage grammars, due to the context-freeness lemma one still gets three derivations of a form corresponding to the three above. But iteration may fail to produce an infinite number of collages of the language as can be seen in the following example. Consider the production in Figure 11.12. By iteration, one gets just three further derived collages, as shown in Figure 11.13, because the same parts are reproduced again and again after four steps. In other words, one obtains a pumping lemma for collage grammars as in the case of context-free Chomsky grammars (and in the case of hyperedge replacement graph grammars), but it seems useless if one wants to show that a certain collage language cannot be generated by a collage grammar because the essential property—that pumping yields infinitely many different results—is missing.

11.5.1 The Number of Parts of Collages

As mentioned above, a direct consequence of the pumping lemma for context-free string languages is that, if the strings in such a language are ordered by length, then there is a constant $c \in \mathbb{N}$ such that consecutive strings differ in length by at most c symbols. In other words, the length of strings increases

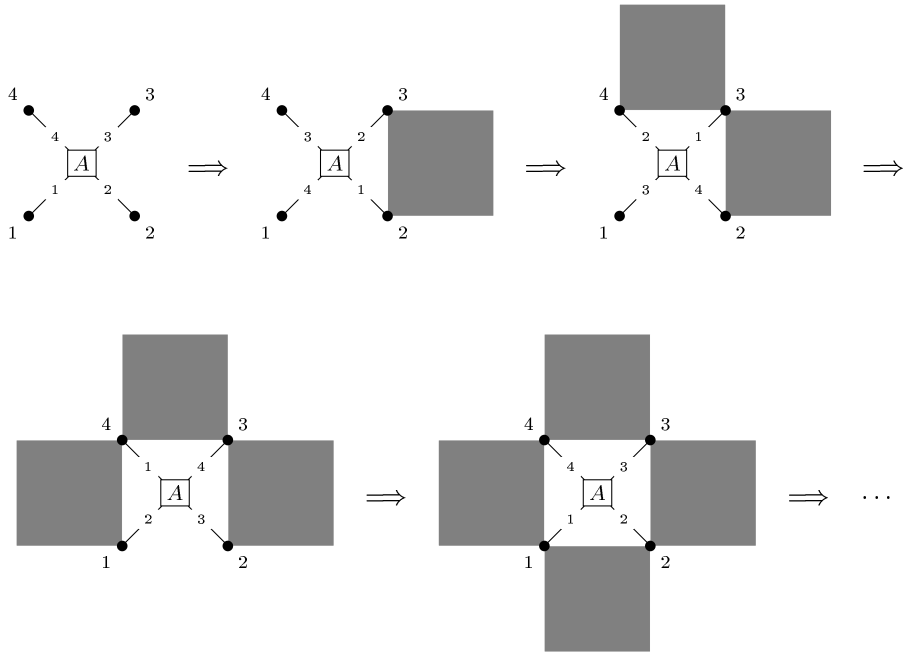


Figure 11.13: A derivation using the production in Figure 11.12

at most linearly. Consequently, languages like $\{a^{n^2} \mid n \in \mathbb{N}\}$ are not context-free. The purpose of this section is to discuss a similar result for context-free collage languages. The result states that the number of parts increases at most linearly.

As explained above, until now no pumping lemma for context-free collage languages is known which is powerful enough to establish such a result as a consequence. Therefore, a direct proof is needed in order to show the desired property.

Theorem 11.5.1

Let L be a context-free collage language such that $S = \{|PART_C| \mid C \in L\}$ is infinite. Then there is some $c \in \mathbb{N}$ such that $S \cap \{m, \ldots, m + c\} \neq \emptyset$ for all $m \in \mathbb{N}$. □

Proof

Let L be generated by a collage grammar G. Call a derivation tree t of G *small* if (1) it contains no proper subtree t' such that $root(t') = root(t)$ and (2) all its

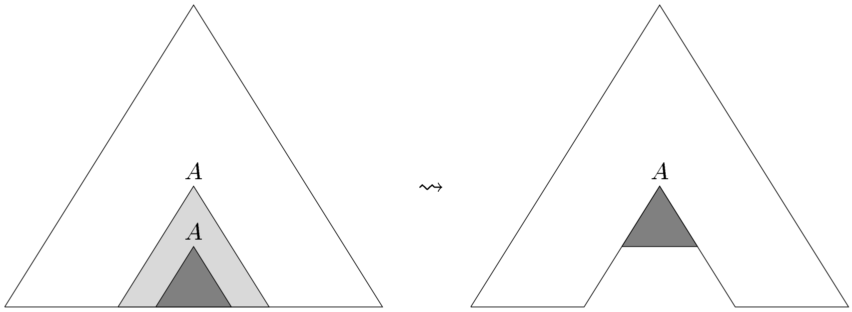


Figure 11.14: Reducing the size of a derivation tree in the proof of Theorem 11.5.1

proper subtrees are small. Let T be the set of all derivation trees t of G such that all proper subtrees of t are small, and choose $c = \max\{|PART_{result(t)}| \mid t \in T\}$. Clearly, this is well defined because T is finite.

If $C \in L$ is a collage consisting of at least $m + c$ parts, then it has a derivation tree as shown in Figure 11.14 on the left, containing a subtree s which is in T but is not small (the grey part). Replacing s with one of its small subtrees s' such that $root(s') = root(s)$ yields a strictly smaller derivation tree whose result C' satisfies $-c \leq |PART_C| - |PART_{C'}| \leq c$. (Notice that the number of parts may possibly increase!)

Repeating this construction as long as possible[1], one obtains a sequence $C = C_0, C_1, \ldots, C_k$ of collages such that the number of parts in C_i and C_{i+1} differs by at most c, and C_k contains at most c parts. Consequently, there must be at least one index i such that $m \leq |PART_{C_i}| \leq m + c$. □

By Theorem 11.5.1 collage languages for which the number of parts of collages grows faster than linearly cannot be context-free. Examples of such languages are the one given in Figure 11.10 and the one in Figure 11.15. The number of parts in the approximations of the Sierpiński gasket shown in Figure 11.10 grows exponentially, while the number of parts in the square patterns of Figure 11.15 grows quadratically. Thus, both these languages cannot be generated. There are, of course, many other examples of this kind. In fact, almost all the well-known fractals generated by iterated function systems (cf. [1,31,34]), like the Sierpiński sponge, the Cantor dust, the Barnsley fern, etc. are obtained by taking the limit of a sequence which, considered as a set of collages, shows the exponential behaviour of the example in Figure 11.10. If one tries to gen-

[1] Termination is of course guaranteed as the trees get smaller and smaller.

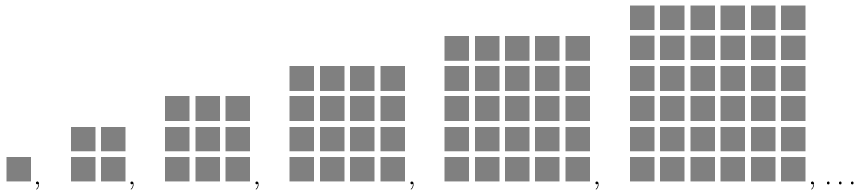


Figure 11.15: A non-context-free collage language (the number of parts grows quadratically)

erate such sets using collage grammars, Theorem 11.5.1 says that there is no way to make sure that all regions of a given collage show the same degree of refinement.

Notice that, as shown by Example 11.2, one can easily construct a collage grammar that generates the collages in Figure 11.10 together with its "irregular" approximations (like those shown in Figure 11.3). Clearly, the intersection of all the pictures given by collages of this sort also yields the Sierpiński gasket. Thus, if the aim is to investigate grammatical descriptions of fractals one may say that nothing essential is lost. This argument will be formalized in Section 11.7. Furthermore, it is rather obvious that exactly the collages in Figure 11.10 are generated if the grammar of Example 11.2 is turned into a TOL collage grammar by taking two different tables containing the productions p_1 and p_2, respectively. This will be discussed in more detail in Section 11.8.1.

The number of parts of the collages generated by the part-sensitive collage grammar in Example 11.4 (see Figure 11.9) grows quadratically. Thus, this language is not context-free either. On the other hand, it is easy to generate the same language using a TOL collage grammar, which yields the following result.

Theorem 11.5.2

The set of context-free collage languages is a proper subset of the intersection of the sets of collage languages generated by TOL collage grammars and part-sensitive collage grammars. ☐

It should be mentioned that, as observed in [8], the proof of Theorem 11.5.1 does in fact yield a more general statement. For this, let us call a pair $r = (L, R)$ of finite sets of parts a *collage-replacement rule* and define $C \Longrightarrow_r C'$ for undecorated collages C, C' with $pin_C = pin_{C'}$ if there is an affine transformation a such that $a(L) \subseteq PART_C$ and $PART_{C'} = (PART_C \setminus a(L)) \cup a(R)$. Intuitively, r just replaces an affine image of L in C by the correspondingly transformed right-hand side R. Clearly, in the proof of Theorem 11.5.1 the replacement of

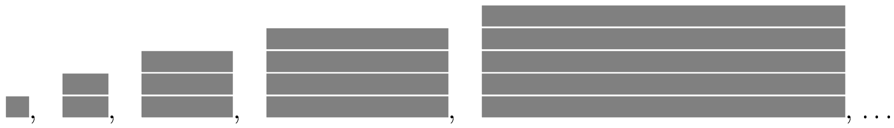


Figure 11.16: Another collage language which is not context-free

a subtree s with s' has exactly the same effect (where a is given by the composition of the transformations determined by the path from the root of the derivation tree to the root of s). Therefore, the finitely many pairs (s, s') to be considered correspond to a finite set of collage-replacement rules, which yields the following theorem.

Theorem 11.5.3

For every context-free collage language L there is a finite set S of collage-replacement rules and a constant $c \in \mathbb{N}$ such that, for every $C_0 \in L$, there exist $C_1, \ldots, C_n \in L$ such that $|PART_{C_n}| \leq c$ and $C_0 \Longrightarrow_{r_1} C_1 \Longrightarrow_{r_2} \cdots \Longrightarrow_{r_n} C_n$ for some $r_1, \ldots, r_n \in S$. $\square$

In particular, since S is finite it holds that two collages C_{i-1}, C_i in the sequence $C_0, C_1, \ldots, C_n$ differ by at most a constant number of parts. From this it follows, for instance, that the collage language indicated in Figure 11.16 is not context-free (which could not be shown by the use of Theorem 11.5.1).

11.5.2 The Volume of Parts of Collages

If one is interested in properties of context-free collage languages one may also consider the set $parts(L) = \bigcup_{C \in L} PART_C$ of all parts that occur in the collages of a given language L instead of looking at the collages themselves. It is immediately clear that $parts(L)$ must be of a rather special nature if L is context-free because this means that the parts are obtained by repeatedly applying affine transformations (taken from a finite set!) to finitely many basic parts. In this section the aim is to study the distribution of volumes of parts in $parts(L)$.

Let us denote by $vol(p)$ the volume of a part p. In order to avoid unnecessary technical difficulties, in the following it is assumed that only bounded parts with a non-zero volume appear in the considered collage languages, i.e., for $p \in parts(L)$ it always holds that $vol(p) \in \mathbb{R} \setminus \{0\}$. We shall now consider the set $vol(parts(L))$ of all volumes of parts in $parts(L)$, and, for technical reasons, also the set $log\,vol(parts(L)) = \{log\, r \mid r \in vol(parts(L))\}$, i.e., the logarithms of volumes of parts in *parts*(L). It can be shown that this set is a semi-linear set of real numbers if L is context-free. Using this, two variants of a criterion can be proved which are useful in order to show that certain collage languages are not context-free.

The notion of semi-linearity of subsets of $\mathbb{R}$ mentioned above is defined as follows.

Definition 11.5.4

A *linear subset of* $\mathbb{R}$ is a set S such that there exist real numbers $t_0, \ldots, t_m \in \mathbb{R}$ (for some $m \in \mathbb{N}$) with $S = \{t_0 + \sum_{i=1}^{m} n_i t_i \mid n_1, \ldots, n_m \in \mathbb{N}\}$. In this case we say that S *is generated by* $(t_0, \ldots, t_m)$ and $\{t_1, \ldots, t_m\}$ is a *period* of S.

A subset of $\mathbb{R}$ is *semi-linear* if it is a finite union of linear subsets of $\mathbb{R}$.

We are now able to formulate the result mentioned above.

Theorem 11.5.5

Let L be a context-free collage language. Then the set $\mathit{log}\, \mathit{vol}(\mathit{parts}(L))$ is semi-linear. □

Proof sketch

It is well known that applying an affine transformation a to a part p means to multiply its volume by the factor $\mathit{det}(a)$, where $\mathit{det}(a)$ is the determinant of the matrix that defines the linear part of a. Taking logarithms, this results in the addition of the constant $\mathit{log}\, \mathit{det}(a)$. Furthermore, the sequences of affine transformations encountered on the paths in the derivation trees of a collage grammar form a regular language L_0 (whose "symbols" are affine transformations). Thus, $\mathit{log}\, \mathit{vol}(\mathit{parts}(L)) = \{\sum_{i=1}^{k} \mathit{det}(a_i) \mid a_1 \cdots a_k \in L_0\}$. From this, the assertion follows quite directly using the well-known Parikh theorem for regular string languages. □

Although Theorem 11.5.5 can of course be used as a criterion to disprove context-freeness of collage languages, it is a rather cumbersome tool for such an aim. However, the theorem can be used to derive simpler properties which are easy to use. This is based on the following lemma.

Lemma 11.5.6

Let S be a finite union of linear subsets of $\mathbb{R}$, each of which admits a period in which only positive or only negative numbers occur. Let $\mu_S(x) = |\{y \in S \mid -x \leq y \leq x\}|$ for all positive $x \in \mathbb{R}$. Then there is some $k \in \mathbb{N}$ such that $\mu_S(x) \in O(x^k)$. □

The proof of the lemma is fairly easy. Of course, it is sufficient to prove it for linear subsets of $\mathbb{R}$. If the period of such a linear set is a k-tuple consisting only of positive or only of negative numbers then $\mu_S(x)$ is, up to a constant factor, not larger than the number of k-tuples of natural numbers $\leq x$, that is, $O(x^k)$. Now, consider the semi-linear subset $S = \mathit{log}\,\mathit{vol}(\mathit{parts}(L))$ of $\mathbb{R}$ associated with a context-free language according to Theorem 11.5.5. If there is an upper or a non-zero lower bound on the volumes of parts then L can be generated using only transformations a such that $\mathit{log}\,\mathit{det}(a) < 0$, or only transformations such that $\mathit{log}\,\mathit{det}(a) > 0$. But then the periods of S contain only negative or only positive numbers, so that the following theorem is obtained, using Lemma 11.5.6 and Theorem 11.5.5.

Theorem 11.5.7

Let L be a context-free collage language with $\mathit{parts}(L) \neq \emptyset$ such that $\mathit{inf}\,\mathit{vol}(\mathit{parts}(L)) > 0$ or $\mathit{sup}\,\mathit{vol}(\mathit{parts}(L)) < \infty$, and let, for every positive $x \in \mathbb{R}$, $\mu(x) = |\{y \in \mathit{vol}(\mathit{parts}(L)) \mid 1/x \leq y \leq x\}|$. There is some $k \in \mathbb{N}$ such that $\mu(x) \in O((\mathit{log}\,x)^k)$. □

To consider an example in $\mathbb{R}^d$, let $\mathit{cube}(r) = \{x \in \mathbb{R}^d \mid 0 \leq x_i \leq r \text{ for } i = 1, \ldots, d\}$ and $C_{\mathit{cube}(r)} = (\{\mathit{cube}(r)\}, \lambda)$, for $r \in \mathbb{R}$, $r \geq 0$. Thus, $\mathit{cube}(r)$ is a d-dimensional cube of edge length r. Theorem 11.5.7 says that, for instance, $L = \{C_{\mathit{cube}(n)} \mid n \in \mathbb{N}_+\}$ and $L' = \{C_{\mathit{cube}(1/n)} \mid n \in \mathbb{N}_+\}$ are not context-free because in these cases $\mu(x) = \lfloor \sqrt[d]{x} \rfloor$, which is not polynomial in $\mathit{log}\,x$. For $d = 2$ this shows that the language L_{square} discussed in the introduction is not context-free. Note that, in fact, Theorem 11.5.7 says that there cannot even be any context-free collage language L_0 without arbitrarily small parts such that $\{\mathit{cube}(n) \mid n \in \mathbb{N}_+\} \subseteq \mathit{parts}(L_0)$. Similarly, there cannot be any context-free collage language L_0' without arbitrarily large parts such that $\{\mathit{cube}(1/n) \mid n \in \mathbb{N}_+\} \subseteq \mathit{parts}(L_0')$.

Since Theorem 11.5.7 is restricted to languages that do not contain arbitrarily small as well as arbitrarily large parts at the same time, it cannot be used to show that there is in fact *no* context-free collage language $\widehat{L}$ which satisfies $\{\mathit{cube}(n) \mid n \in \mathbb{N}_+\} \subseteq \mathit{parts}(\widehat{L})$. For example, Theorem 11.5.7 cannot be used to show that $L \cup L'$ (where L and L' are as above) is not context-free. It is easy to see that the theorem does not hold if we allow for languages L such that $\mathit{inf}\,\mathit{vol}(\mathit{parts}(L)) = 0$ and $\mathit{sup}\,\mathit{vol}(\mathit{parts}(L)) = \infty$ because then $\mu(x) = \infty$ is possible. However, as the following lemma shows, one can avoid the restriction by considering parts with integer volumes and volumes of the form $1/n$ ($n \in \mathbb{N}_+$) only. For the (rather technical) proof see [15].

Theorem 11.5.8

Let L be a context-free collage language and let $\mu(n) = |\, vol(parts(L)) \cap \{1/n, \ldots, 1/2, 1, \ldots, n\}\,|$ for every $n \in \mathbb{R}$. There is some $k \in \mathbb{N}$ such that $\mu(n) \in O((log\, n)^k)$. □

As an example, it follows from Theorem 11.5.8 that there cannot be a context-free collage language L such that $\{cube(n) \mid n \in \mathbb{N}_+\} \subseteq parts(L)$. Thus, for instance, the language $\{C_{cube(n)} \mid n \in \mathbb{N}_+\} \cup \{C_{cube(1/n)} \mid n \in \mathbb{N}_+\}$ discussed above is not context-free. Note that this example is not covered by Theorem 11.5.7, while Theorem 11.5.8 is not able to handle examples like $\{C_{cube(n+1/2^n)} \mid n \in \mathbb{N}_+\}$, which Theorem 11.5.7 is able to deal with.

It should be noticed that, in fact, Theorem 11.5.7 and Theorem 11.5.8 hold also for collage languages generated by TOL or part-sensitive collage grammars. This is because, obviously, every collage language generated by one of these devices is contained in a context-free collage language. (Just take the union of all tables in the TOL case, or forget about the context in the part-sensitive case.)

The criteria discussed in this section allow to disprove context-freeness of collage languages in a reasonable way. On the other hand, they fail in a number of cases where one would intuitively also expect to be able to prove non-context-freeness. Future investigations should therefore try to find further criteria that cover other (and perhaps more sophisticated) quantitative or geometric properties of context-free collage languages. A more ambitious aim is to find further criteria for TOL and part-sensitive collage grammars that can be used to separate the corresponding language classes. Results in this respect may eventually lead to a thorough comparison of classes of picture languages.

11.6 Decidability and Undecidability

One of the most intriguing questions about collage grammars is which geometric properties of the generated pictures can be decided by inspecting the generating grammar. This question has been investigated in [14] in the following respects.

(1) The decidability results for context-free hypergraph languages based on so-called compatible properties (cf. [2,30,24,25]) can be carried over to collage grammars. In particular, one gets decidability of the following problems provided that the considered geometric property is compatible with respect to a class of collage grammars: Is there a picture in the generated language having the property? Do all pictures in the generated language have the property?

Unfortunately, only a few geometric properties turn out to be compatible, but nevertheless there exist some. In the following we discuss three concrete properties with respect to certain classes of collage grammars. Furthermore, we discuss a special case of collage grammars given by the so-called grid restriction [7], for which the notion of compatibility is particularly useful.

(2) Replacing the classes of collage grammars mentioned above by more general ones we obtain undecidability results for the same problems. Therefore, one cannot hope to prove compatibility of reasonable problems without restricting the class of collage grammars considered. This contrasts the situation for hyperedge replacement grammars, where most of the compatibility results known so far hold for the class of *all* hyperedge replacement grammars.

We shall first present the concept of compatibility and the mentioned meta results for the decidability of compatible properties. Afterwards, some concrete properties are discussed, which are compatible with respect to certain classes of collage grammars. The first property requires that the generated pictures contain full balls. The second requires the generated pictures to be space-filling in some sense, and the third concerns connectedness.

The mentioned undecidability results are obtained by a simulation of two-counter machines and by a reduction of Post's correspondence problem. These results indicate that the range of compatible properties and of decidable properties in general is rather small for collage grammars.

11.6.1 The Decidability of Compatible Properties

The main characteristics of compatible properties is that they can be checked using the context-freeness lemma. More precisely, a property is compatible if it holds for a collage generated from a handle if and only if some of the component collages have the property (where an auxiliary predicate determines the meaning of the term "some"). For the definition, recall that C_e denotes the component of C derived from $e \in E_R$, as it was formalized in the context-freeness lemma.

Definition 11.6.1 (*compatible property*)
Let $\mathcal{G}$ be a class of proper collage grammars and π a predicate on collages. Then π is called $\mathcal{G}$-*compatible* if for all $G = (N, P, Z) \in \mathcal{G}$ there is a decidable auxiliary predicate π' on pairs (C, B), with $C \in \mathcal{C}(N)$ and $B \subseteq E_C$, such that the following holds: For all derivations of the form $(A, pin_A)^\bullet \Longrightarrow R \Longrightarrow^* C$ with $C \in \mathcal{C}$ and $(A, R) \in P$,

$\pi(C)$ holds if and only if there is a subset $B \subseteq E_R$ such that

(1) $\pi'(R, B)$ holds and

(2) $\pi(C_e)$ holds for all $e \in B$.

Remark:
Compatibility can be defined in a slightly more general way (as it is done for hyperedge replacement graph grammars). Then, π is a predicate out of a finite family indexed by some set I. In this case, $\pi(C) = \pi_{i_0}(C)$ need not be obtained from the $\pi(C_e)$ alone, but may depend on predicates $\pi_1, \ldots, \pi_k$ that test related properties instead. However, the simple form introduced above is sufficient for the applications presented below.

Theorem 11.6.2 (decidability of compatible properties)
Let π be a $\mathcal{G}$-compatible predicate for some class $\mathcal{G}$ of collage grammars.

1. *It is decidable for $G \in \mathcal{G}$ whether π holds for some $C \in L(G)$.*
2. *It is decidable for $G \in \mathcal{G}$ whether π holds for all $C \in L(G)$.* □

For completeness' sake we mention another interesting result which does not directly say something about decidability questions: the so-called filter theorem. This result gives further evidence for the usefulness of compatibility.

Theorem 11.6.3 (filter theorem)
Let π be a $\mathcal{G}$-compatible predicate for some class $\mathcal{G}$ of collage grammars. Then, for every collage grammar $G \in \mathcal{G}$, the collage language $\{C \in L(G) \mid \pi(C)$ holds$\}$ is context-free. □

The construction in the proof of Theorem 11.6.3 is effective. In fact, in order to transform the collage grammar G into a collage grammar that generates all $C \in L(G)$ for which π holds, one just has to change the labels of hyperedges in order to keep track of certain additional information. Thus, the collages used as right-hand sides of the resulting grammar are the same as in G, up to the relabelling of hyperedges. Notice that, since the emptiness problem is decidable for the languages generated by collage grammars (see Corollary 11.2.4), the first assertion of Theorem 11.6.2 follows from (the effectivenes of) Theorem 11.6.3.

11.6.2 Some Compatible Properties

Let us now discuss some geometric properties with respect to compatibility. Unfortunately, it seems to be difficult to find compatible properties in the geometric setting (as opposed to the graph case), but three of them are provided in the following. At first, we consider the question whether a collage contains a full ball. Then we consider so-called pin-filling collages (which means that the underlying patterns fill the convex hulls spanned by the pin points) and show compatibility in case of compact pin-bounded and non-overlapping collage grammars. Finally, connected collages are looked at and compatibility is shown for this case if the considered collage grammars are non-overlapping and pin-framed.

Let us first define the mentioned properties of collages and collage grammars.

Definition 11.6.4
A collage $C \in \mathcal{C}(N)$ is

1. *compact* if each of its parts is compact (that is, bounded and closed),
2. *pin-bounded* if $X \subseteq [pin_C]$ for all $X \in PART_C \cup \{[att_C(e)] \mid e \in E_C\}$,
3. (essentially) *non-overlapping* if for all $X, Y \in \{[att_C(e)] \mid e \in E_C\} \cup PART_C$ with $X \neq Y$ we have $X \cap Y = \delta X \cap \delta Y$, and
4. *pin-framed* if it is pin-bounded and satisfies $\delta[pin_C] \subseteq pattern(C) \cup \bigcup_{e \in E_C}[att_C(e)]$.

A proper collage grammar $G = (N, P, Z)$ is *compact* (*pin-bounded*, *non-overlapping*, *pin-framed*) if all right-hand sides of productions in P are compact (respectively pin-bounded, non-overlapping, pin-framed).

We first consider the question whether the generated collages contain balls. A collage C in $\mathbb{R}^d$ is said to *contain a ball* if there is a point $x_0 \in \mathbb{R}^d$ and an $\epsilon > 0$ such that $x \in pattern(C)$ for all $x \in \mathbb{R}^d$ with $dist(x, x_0) \leq \epsilon$.

Theorem 11.6.5
Let $\mathcal{G}_c$ denote the class of all compact collage grammars. Let π be the predicate that holds for (and only for) collages that contain a ball. Then π is $\mathcal{G}_c$-compatible.

Proof sketch
If the collage grammar in question is compact, looking at $R[repl]$ we know that it consists entirely of compact parts since affine transformations preserve compactness. However, if a ball is covered by a finite number of compact parts, one of these parts must itself contain a (possibly smaller) ball. Therefore, if E is the set of all hyperedges $e \in E_R$ such that the associated transformation $a(e)$ is injective, $R[repl]$ contains a ball if and only if $pattern(R - E_R)$ contains a ball or there is a hyperedge $e \in E$ such that $repl(e)$ contains a ball. $\square$

As a direct consequence we obtain the following corollary, using Theorem 11.6.2.

Corollary 11.6.6

1. *For $G \in \mathcal{G}_{\mathrm{c}}$, it is decidable whether some collages in $L(G)$ contain balls.*
2. *For $G \in \mathcal{G}_{\mathrm{c}}$, it is decidable whether all collages in $L(G)$ contain balls.* □

Let us now turn to the question whether a collage grammar generates pin-filling collages, where a collage C is called *pin-filling* if $[pin_C] \subseteq pattern(C) \cup \bigcup_{e \in E_C}[att_C(e)]$, that is, if all points in the convex hull spanned by the pin points of C belong to the induced pattern. We have the following theorem.

Theorem 11.6.7

Let $\mathcal{G}_{\mathrm{cbn}}$ be the class of compact, pin-bounded, and non-overlapping collage grammars, and let π be the predicate which holds for (and only for) pin-filling collages. Then π is $\mathcal{G}_{\mathrm{cbn}}$-compatible.

Proof sketch
Consider a collage grammar $G \in \mathcal{G}_{\mathrm{cbn}}$ and look at a collage $C = R[repl]$ generated from a handle. It is not hard to show that, besides compactness, also pin-boundedness is preserved by hyperedge replacement. Therefore, $repl(e)$ is compact and pin-bounded for all $e \in E_R$. Together with the fact that R is non-overlapping this means that C is pin-filling if and only if (1) R is pin-filling and (2) for every hyperedge $e \in E_R$, $repl(e)$ is pin-filling. □

Corollary 11.6.8

1. *For $G \in \mathcal{G}_{\mathrm{cbn}}$, it is decidable whether some collages in $L(G)$ are pin-filling.*
2. *For $G \in \mathcal{G}_{\mathrm{cbn}}$, it is decidable whether all collages in $L(G)$ are pin-filling.* □

Finally, let us consider the connectedness of generated collages, where a collage C is said to be *connected* if $pattern(C) \cup \bigcup_{e \in E_C}[att_C(e)]$ is connected[2].

Theorem 11.6.9

Let $\mathcal{G}_{\mathrm{nf}}$ be the class of non-overlapping pin-framed collage grammars, and let π be the predicate that holds for (and only for) connected collages. Then π is $\mathcal{G}_{\mathrm{nf}}$-compatible.

Proof sketch
Clearly, pin-framedness is another property of collages which is preserved by

[2] A set $X \in \mathbb{R}^d$ is disconnected if $X \subseteq Y \cup Z$ for disjoint open sets Y, Z such that both $X \cap Y$ and $X \cap Z$ are nonempty. Otherwise, X is connected.

hyperedge replacement. Together with the fact that only non-overlapping collage grammars are considered, this means that $R[repl]$ is connected if and only if $repl(e)$ is connected for every hyperedge $e \in E_R$ and, furthermore, the graph g, obtained by taking as nodes the parts and hyperedges of R and drawing an edge between two of these nodes whenever there is an overlapping in R, is connected. □

Corollary 11.6.10

1. *For $G \in \mathcal{G}_{\mathrm{nf}}$, it is decidable whether some collages in $L(G)$ are connected.*
2. *For $G \in \mathcal{G}_{\mathrm{nf}}$, it is decidable whether all collages in $L(G)$ are connected.* □

Of course, there are many other questions one would like to be able to answer algorithmically: Is the volume of the picture larger (smaller) than some bound? Is the picture covered by a ball of a given radius? Is the number of holes bounded by a constant? For all these properties it is an open problem to find natural classes of collage grammars such that the properties turn out to be compatible. One promising candidate is the grid restriction discussed next.

11.6.3 Grid Collage Grammars

Grid collage grammars are based on a division of the d-dimensional unit cube into m^d smaller cubes, all having the same size, using an evenly spaced grid. The attached points of a hyperedge in the right-hand side of a production are then required to span such a small cube, while the pins are given by the corners of the unit cube.

Grid collage grammars are studied in detail in [7]. It turns out that they have many nice properties. In particular, the class of languages generated by m-grid collage grammars comes with powerful closure properties and interesting questions can be decided. As some of the proofs are rather involved we only give a rough overview here, omitting most of the details, and in particular the proofs.

For the following discussion let *pin* be a sequence of pin points consisting of all points of the form $(x_1, \dots, x_d) \in \{0,1\}^d$ (in an arbitrary but fixed order) and denote by *cube* the unit cube $[pin]$. Now, consider some $m \geq 2$, the size of the grid, and let, for every $\alpha = (\alpha_1, \dots, \alpha_d) \in \{1, \dots, m\}^d$,

$$cube_\alpha = \{(x_1, \dots, x_d) \in \mathbb{R}^d \mid (\alpha_i - 1)/m \leq x_i \leq \alpha_i/m \text{ for } i = 1, \dots, d\}.$$

Then, an *m-grid collage grammar* in $\mathbb{R}^d$ is a proper collage grammar $G = (N, P, Z)$ such that

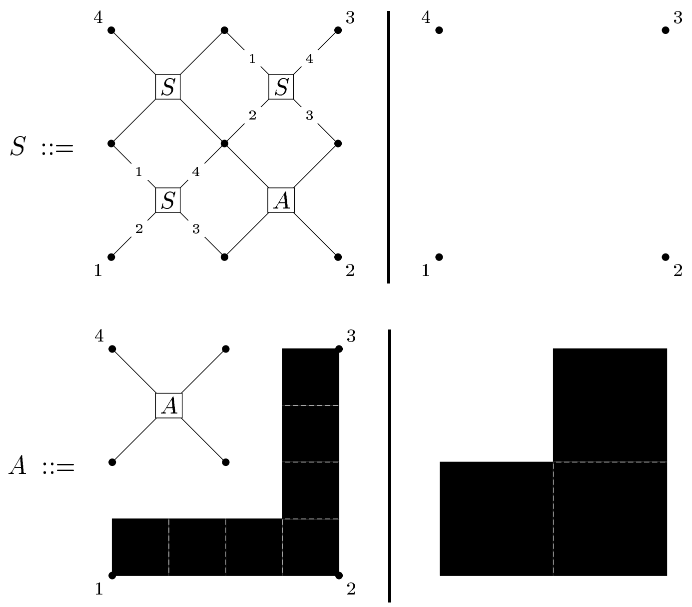


Figure 11.17: The productions of the 2-grid collage grammar in Example 11.5

- $pin_A = pin$ (where pin is as above) for all $A \in N$ and
- for all $(A ::= R) \in P$ either $PART_R = \{cube\}$ and $E_R = \emptyset$ or $PART_R = \emptyset$ and, for all $e \in E_R$,
  - (i) $[att_R(e)] = cube_\alpha$ for some $\alpha \in \{1, \ldots, m\}^d$ and
  - (ii) $[att_R(e)] \neq [att_R(e')]$ for all $e' \in E_R \setminus \{e\}$.

  Note that this includes the case $PART_R = \emptyset = E_R$, i.e., the empty right-hand side.

The language generated by an m-grid collage grammar is called a *context-free m-grid collage language.*

Example 11.5 (grid collage grammar)
As an example in $\mathbb{R}^2$, consider the collage grammar $G = (\{S, A\}, P, (pin, S)^\bullet)$, where P consists of the productions shown in Figure 11.17. Strictly speaking, G is not a 2-grid collage grammar, but it can easily be turned into the form required, using three additional labels and decomposing the two productions with left-hand side A in a suitable way. Thus, the generated language, of which

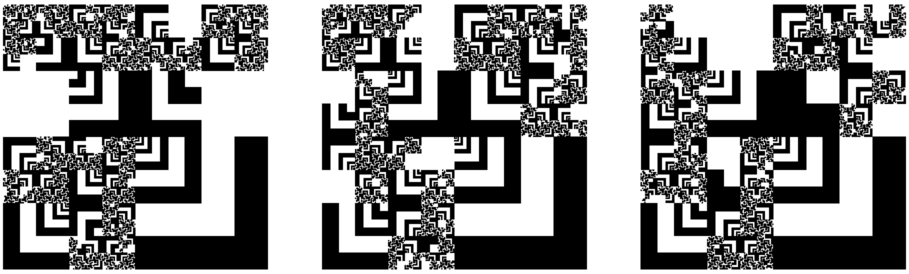


Figure 11.18: Elements of the 2-grid collage language discussed in Example 11.5

Figure 11.18 shows some elements, is a context-free 2-grid collage language.

Clearly, one can apply the grid restriction to TOL collage grammars as well, yielding TOL grid collage grammars. For instance, the grammar in Example 11.5 can be turned into a TOL collage grammar with two tables, using one table for the terminating and one for the other two productions (again neglecting the fact that the productions must be changed slightly in order to obey the restrictions). Applying the terminating table after a couple of derivation steps yields the collage shown in Figure 11.19.

Using the results of Section 11.5 it is easy to show that the class of context-free collage languages is neither closed under intersection nor under set difference. For example, the language indicated in Figure 11.10 can easily be obtained as the intersection of two suitably chosen context-free collage languages. In the grid case the situation turns out to be different. Intuitively, if two m-grid collage grammars are given, one can build a third whose nonterminal labels are pairs of labels in order to ensure that a derived collage can be derived by both of the original grammars. The construction also has to take into account the rotations and reflections that take place, but this is possible since there are only finitely many in this case. In a similar way set difference can be dealt with. Furthermore, the construction can be modified in such a way that it works for the generated patterns rather than for the generated collages. Thus, the following closure result is obtained.

Theorem 11.6.11
Let L, L' be context-free m-grid collage languages for some $m \in \mathbb{N}_+$. Then, for every $\oplus \in \{\cup, \cap, \setminus\}$, one can effectively construct m-grid collage grammars G and G' such that $L(G) = L \oplus L'$ and $pattern(L(G')) = pattern(L) \oplus pattern(L')$. $\square$

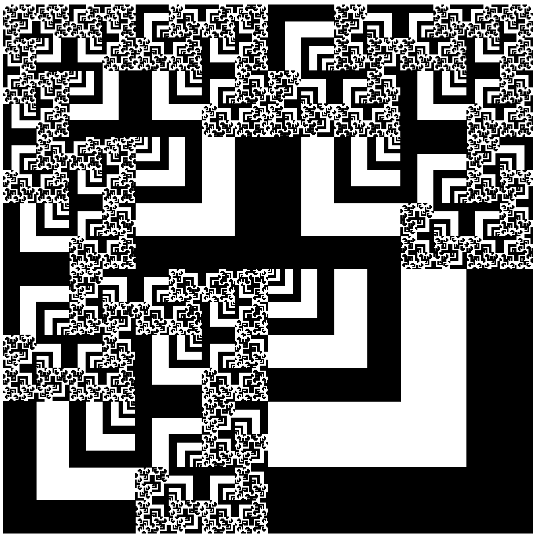


Figure 11.19: A collage derived by a TOL grid collage grammar obtained from the grammar in Example 11.5

In the following, denote by $\emptyset$ the collage $(\emptyset, pin)$. Using a standard construction (cf. the removal of λ-productions from context-free string languages) every m-grid collage grammar G can be turned into an m-grid collage grammar $G' = (N, P, Z)$ such that $L(G') = L(G) \setminus \{\emptyset\}$ and P contains no production of the form $A ::= \emptyset$. Now, it is clear that $L(G')$ is finite if and only if its set of derivation trees is finite. By a very similar construction one can achieve in addition that $(A, pin)^\bullet \Longrightarrow^* C$ with $pattern(C) = [pin]$ implies $C = (\{cube\}, pin)$. While the latter construction cannot preserve $L(G)$, it does in fact preserve patterns, i.e., $pattern(L(G')) = pattern(L(G)) \setminus \{\emptyset\}$. Now, for this second construction, $pattern(L(G'))$ is finite if and only if G' has only finitely many derivation trees.

For sets of derivation trees finiteness can be decided like in the string case. Consequently, we have the following theorem.

Theorem 11.6.12

There is an algorithm that takes as input a grid collage grammar G and decides whether $L(G)$ is finite, and whether $pattern(L(G))$ is finite. □

Notice that, taking into account Theorem 11.6.11, this means that we can decide whether, for example, $pattern(L(G)) \cap pattern(L(G'))$ is finite (where G and G' are two m-grid collage grammars).

Let us now turn to the question of compatible predicates. In [7] mainly two of them are discussed. It was mentioned above that the notion of compatibility can actually be generalized in such a way that a finite family of predicates (which may mutually depend on each other) is used instead of just a single predicate. In this case, each of the predicates in the finite family is called compatible[3]). This notion of compatibility, of which the one defined above is a special case, is referred to in the following.

First, let us consider the question whether the pattern of a collage C contains a rational point r. If we are considering a grid of size m, then the coordinates of r can be written down with respect to the base m. Now, looking at the m^d cubes of the grid, we can easily tell in which cube the point lies—just by inspecting the first digits of its d coordinates. If the cube is again divided into its m^d subcubes, we can proceed recursively in the same way, now looking at r', which is obtained from r by cancelation of the first digit of each coordinate. The fact that r is rational means that our repeated cancelation of digits yields only finitely many different possibilities, which correspond to the finite number of predicates needed to establish compatibility. This leads to the following theorem.

Theorem 11.6.13
Let $r \in \mathbb{Q}^d$. Then the predicate π_r which holds for $C \in \mathcal{C}$ if (and only if) $r \in pattern(C)$ is compatible[4] with respect to the class of all m-grid collage grammars, for all $m \in \mathbb{N}_+$. □

We note the customary corollary.

Corollary 11.6.14
Let $r \in \mathbb{Q}^d$.

(1) For m-grid collage grammars G ($m \in \mathbb{N}_+$) it is decidable whether $r \in pattern(C)$ for all $C \in L(G)$ and whether $r \in pattern(C)$ for at least one $C \in L(G)$.

(2) For every context-free m-grid collage language L the language $\{C \in L \mid r \in pattern(C)\}$ is a context-free m-grid collage language, too. □

[3] For the exact definition see [21, Definition VI.2.1] or [12].

[4] Recall that this refers to the generalized notion of compatibility which was briefly discussed above.

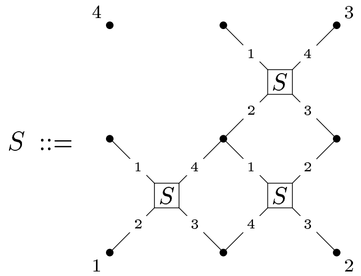


Figure 11.20: Production used to generate the collage in Figure 11.21

Here, (2) holds because the construction in the proof of Theorem 11.6.3 yields an m-grid collage grammar if the original grammar is of this type (which is true because, as mentioned after Theorem 11.6.3, mainly labels need to be changed).

The proof of the second compatibility result is considerably more difficult. It concerns the question whether the generated collages are connected. While it is still unknown whether connectedness is compatible with respect to all m-grid collage grammars, there is a positive result for a special case of TOL grid collage grammars.

In the following, consider some $m \in \mathbb{N}_+$ and let $\mathcal{G}$ denote the class of all TOL grid collage grammars of the form $(\{S\}, \{\{S ::= R_1\}, \{S ::= R_2\}\}, (S, pin)^\bullet)$ where $R_2 = (\{cube\}, pin)$. This class, although quite special, allows to generate an astonishing variety of languages approximating some famous fractals like the Cantor set in $\mathbb{R}$ and the Cantor dust in $\mathbb{R}^2$, the Sierpiński gasket and the Sierpiński carpet in $\mathbb{R}^2$, and the Sierpiński sponge in $\mathbb{R}^3$. In fact, as will be discussed in Section 11.8.1, this is not very surprising because the considered type of TOL collage grammars is as powerful as iterated function systems, a well-known picture-generating device from fractal geometry. The patterns in $\mathbb{R}^2$ that can be defined in this way, using the smallest sensible grid size $m = 2$, are discussed by Peitgen, Jürgens, and Saupe in [34], pp. 244, where they are called the "relatives of the Sierpiński gasket". This naming refers to the fact that the Sierpiński gasket is certainly the most famous member of this class.

Figure 11.20 shows the rule from the first table of a grammar in $\mathcal{G}$; in Figure 11.21 one of the resulting collages is depicted.

For $G \in \mathcal{G}$ and $C \in L(G)$ with $(S, pin)^\bullet \Longrightarrow R_1 \Longrightarrow^* C$, all the component derivations $(S, pin)^\bullet \Longrightarrow^* C_e$ ($e \in E_{R_1}$) are equal because each table consists of a single production. In particular, all component collages C_e are equal and,

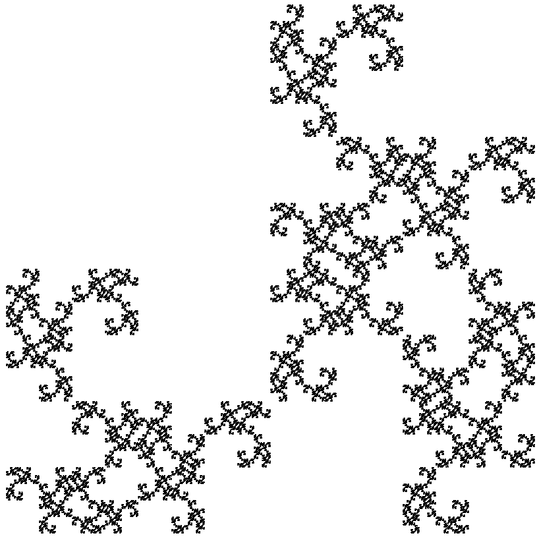


Figure 11.21: One of the collages generated using the production in Figure 11.20

therefore, C consists of rotated and reflected copies of this collage C', each copy occupying one of the cubes $cube_\alpha$. Now, consider the graph g whose nodes correspond to the copies of C' in C and whose edges indicate whether two of these copies intersect in their boundaries. Then one can show that g is a connected graph if and only if C is connected. Since g has only $|E_{R_1}|$ nodes there are only finitely many possible graphs of this kind, which can be used to prove compatibility[5]. The difficult part is to express the requirement that two copies of the component collage C' intersect in their boundaries.

Theorem 11.6.15

Let $\mathcal{G}$ be as above and let π be the predicate which holds for a collage C if (and only if) $pattern(C)$ is connected. Then π is $\mathcal{G}$-compatible. □

Corollary 11.6.16

Let $\mathcal{G}$ be as above. For $G \in \mathcal{G}$ it is decidable whether $pattern(C)$ is connected for all collages $C \in L(G)$. □

Notice that we always have $(\{cube\}, pin) \in L(G)$, so the respective existentially

[5]The definition of compatibility for TOL collage grammars is the same as for ordinary collage grammars.

quantified question is trivial. Furthermore, it is not very interesting that $L = \{C \in L(G) \mid \mathit{pattern}(C) \text{ is connected}\}$ can be generated by a TOL collage grammar. This is because $L(G)$ always has the form $\{C_0, C_1, C_2, \dots\}$ where $\mathit{pattern}(C_i)$ is connected if $\mathit{pattern}(C_{i+1})$ is connected. Thus, the language L in question is either finite or equals $L(G)$.

11.6.4 Undecidable Problems

In this section we first show that for Corollary 11.6.8 (hence for Theorem 11.6.7) the restriction to compact parts is vital. More precisely speaking, the problem to determine whether a non-overlapping, pin-bounded proper collage grammar generates a pin-filling collage is undecidable. Moreover, this holds even for linear grammars, that is, grammars with at most one hyperedge in every right-hand side and in the axiom. It is easy to modify the proof in such a way that it proves the same for the question of connectedness of the generated pictures: For linear, non-overlapping, pin-bounded collage grammars it is undecidable whether a connected collage is generated. In other words, Theorem 11.6.9 and Corollary 11.6.10 depend on the pin-framedness of the considered grammars. Here is the theorem.

Theorem 11.6.17
Let $\mathcal{G}_{\mathrm{lbn}}$ be the class of linear, pin-bounded, non-overlapping collage grammars in $\mathbb{R}^2$.

1. *For $G \in \mathcal{G}_{\mathrm{lbn}}$ as input it is undecidable whether some collage $C \in L(G)$ is pin-filling.*
2. *For $G \in \mathcal{G}_{\mathrm{lbn}}$ as input it is undecidable whether some collage $C \in L(G)$ is connected.* □

The theorem can be proved by simulating a two-counter machine (cf. [26]) by a collage grammar in such a way that a well-formed accepting computation corresponds to the generation of a pin-filling collage. In the following, the construction is explained in an informal way.

The constructed collage grammars will simulate a nondeterministic version of two-counter machines (TCMs) especially suitable for this proof. These TCMs are restricted Turing machines having a read-only input tape with end markers (i.e., it is not possible to move past the left or right end marker delimiting the input), and two counters that can be incremented and decremented (instead of the usual work tape). Depending on the currently scanned input symbol and the current state the machine can switch to another state and either

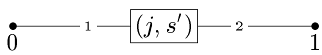
- move the input head one cell to the left or right,
- increment or decrement one of the counters by one,
- verify that a counter has the value zero, or
- accept the input and stop.

The machine rejects whenever a decrementation step is applied to a counter whose value is zero, and whenever a verification step takes place while the respective counter has a non-zero value. As usual, the machine is said to accept its input if there is at least one accepting computation.

It is easy to see that this type of TCMs is equivalent to the usual deterministic version with respect to computational power. Since TCMs are known to be universal computing devices there is no general procedure to decide whether a TCM M accepts an input w. Therefore, the aim now is to construct for a given pair (M, w) a collage grammar $G \in \mathcal{G}_{\mathrm{lbn}}$ such that G generates a pin-filling collage (or, for the second part of the theorem, a connected collage) if and only if M accepts w.

The main problem is, of course, how to simulate the counters. In order to explain the basic idea, let us first discuss the simulation of an "OCM"—a machine with only one counter—working on an input string w. For this, it is possible to use a collage grammar in $\mathbb{R}$ rather than $\mathbb{R}^2$. The axiom of the grammar is a collage that contains a single part, namely the set $\{x \in \mathbb{R} \mid 0 \leq x \leq 1\} \setminus \{1/6^n \mid n \in \mathbb{N}\}$, and a single hyperedge attached to the points 0 and 1. Every derivation step corresponds to a single step of the simulated OCM, and at each moment the label of the hyperedge is of the form (i, s), where i is an index indicating the current position of the input head and s is the current state of the OCM. The different types of steps that the OCM can perform are turned into productions as follows. All right-hand sides have the pin points 0 and 1. Consider a hyperedge label (i, s), and let a be the input symbol scanned at position i.[6] If, in this situation, the transition relation allows to switch to state s' and to

- move to the right (left), the grammar contains the production[7]

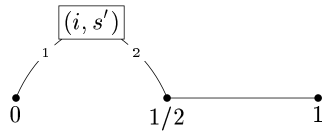
  $(i, s) ::=$ 

  where $j = i + 1$ ($j = i - 1$, respectively);

[6] Note that we are constructing a grammar for an arbitrary but fixed input w, so i really determines a concrete symbol read at that position.

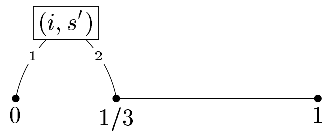
[7] In the following pictures all right-hand sides of productions have the pin points 0 and 1.

- increment the counter, the grammar contains the production

$(i, s) ::=$ 

  where the line between 1/2 and 1 is the part $\{x \in \mathbb{R} \mid 1/2 < x \leq 1\}$ (i.e., the point 1/2 is excluded);

- decrement the counter, the grammar contains the production

$(i, s) ::=$ 

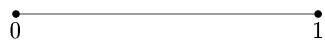
  where the line between 1/3 and 1 is the part $\{x \in \mathbb{R} \mid 1/3 < x < 1\}$ (i.e., both 1/3 and 1 are excluded);

- verify that the counter is zero, the grammar contains the production

$(i, s) ::=$ 

  where the right-hand side now contains a part consisting of the point 1 only;

- accept the input, the grammar contains the production

$(i, s) ::=$ 

  where the part in the right-hand side is $\{x \in \mathbb{R} \mid 0 \leq x \leq 1\}$.

It should be clear that, due to these productions, the hyperedge will always be attached to 0 and a point of the form $1/2^m 3^n$ for some $m, n \in \mathbb{N}$, where $m - n$ is the difference of incrementation and decrementation steps that have taken place. In particular, the second attached point is one of the points $1/6^k$ (which are missing in the part in the axiom) if and only if the value of the counter is zero. Using this fact it is easy to verify that only derivations corresponding to well-formed accepting computations will lead to a collage whose pattern contains all the points between 0 and 1. For example, if the second attached point is $p = 1/2^m 3^n$ and a production of the third kind is applied, then p is

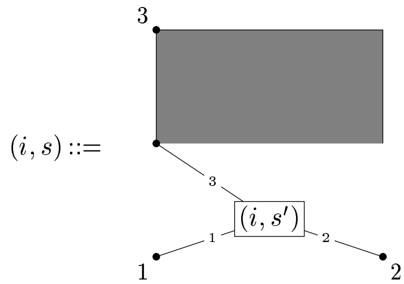


Figure 11.22: A production to simulate the incrementation of the first counter of a TCM; the bottom edge of the rectangle is not included in the part

missing in the resulting collage if and only if $m = n$ (because in the other case p is a point contained in the part of the axiom). Altogether, the grammar generates a pin-filling collage (or, equivalently, a connected collage) if and only if the OCM accepts its input word.

Finally, the extension to TCMs is easily accomplished by looking at $\mathbb{R}^2$ instead of $\mathbb{R}$ and using the two dimensions in order to deal with two counters simultaneously. The hyperedge is then initially attached to $(0,0)$, $(1,0)$, and $(0,1)$; the part in the axiom consists of all points $(x,0)$ and $(0,x)$ such that $0 \leq x \leq 1$ and $x \notin \{1/6^n \mid n \in \mathbb{N}\}$. All right-hand sides of productions have the pin points $(0,0)$, $(1,0)$, and $(0,1)$. In the two-dimensional case there are two slightly different production sets one has to use for the pin-filling property and for connectedness, respectively, but both are easy to obtain from the one-dimensional case. Just to quote one of the productions for the pin-filling property, the one to increment the second counter (associated to the vertical axis) is shown in Figure 11.22, where the part is the set $\{(x,y) \in \mathbb{R}^2 \mid 0 \leq x \leq 1, 1/2 < y \leq 1\}$, i.e., the boundary is included except for the bottom line. In order to deal with connectedness, the right-hand side is the same except that the right edge of the part is also missing, i.e., the part is $\{(x,y) \in \mathbb{R}^2 \mid 0 \leq x < 1, 1/2 < y \leq 1\}$.

There is also a reduction of Post's correspondence problem (PCP) to the question whether *all* generated collages are connected. This will not be discussed in detail here, however. The interested reader is referred to [14]. The construction yields a non-overlapping, pin-bounded, and compact collage grammar for each instance of PCP, such that the given instance has a solution if and only if the grammar generates a disconnected collage. Intuitively, in the collage grammar a derivation tries to "guess" a solution to the given instance of PCP. A deriva-

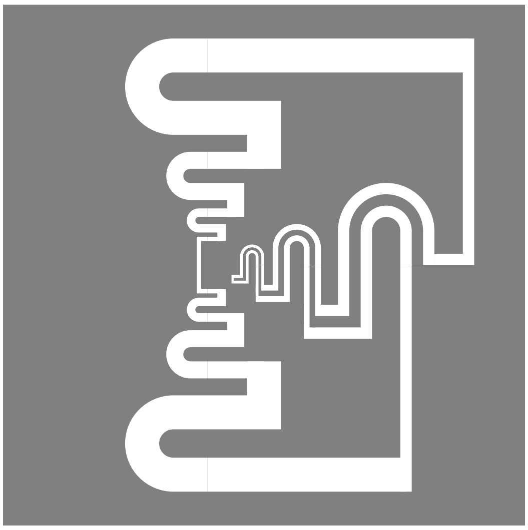


Figure 11.23: Typical example of the sort of pattern generated to prove Theorem 11.6.18

tion that corresponds to a well-formed solution yields a collage whose pattern looks like the one indicated in Figure 11.23. Roughly speaking, the vertical and horizontal curves code the chosen sequence of indices and the resulting words, respectively. (Remember that an instance of PCP consists of two sequences $u_1, \ldots, u_n$ and $v_1, \ldots, v_n$ of words, and a solution is a sequence $i_1, \ldots, i_m$ of indices $i_j \in \{1, \ldots, n\}$ such that the two words $u_{i_1} \ldots u_{i_m}$ and $v_{i_1} \ldots v_{i_m}$ are equal.) If everything fits well, a disconnected pattern like the one shown in the figure is obtained. In all other cases the vertical or the horizontal part will contain substructures as shown in Figure 11.24, resulting in a connected pattern. In other words, we have the following theorem.

Theorem 11.6.18
Let $\mathcal{G}_{\mathrm{cbn}}$ be the class of compact, pin-bounded, non-overlapping collage grammars in $\mathbb{R}^2$. For a collage grammar $G \in \mathcal{G}_{\mathrm{cbn}}$ as input it is undecidable whether all collages $C \in L(G)$ are connected. □

11.7 Generation of Self-Affine Fractals

In this section some links between collage grammars and the concept of self-affinity known from fractal geometry are discussed, summarizing the results

Figure 11.24: Substructures of a generated pattern indicating a malformed solution

of [13]. Fractal geometry is a flourishing field of modern mathematics with significant applications to picture processing and computer graphics (see, e.g., Mandelbrot [31], Barnsley [1], Edgar [17], Falconer [18], and Peitgen, Jürgens, and Saupe [34]). As collages represent patterns, a collage grammar can be viewed as a device that generates a set of patterns. The overlay and the intersection of all generated patterns yield two particular patterns that are called upper and lower generated fractal.

Definition 11.7.1 (*upper* and *lower fractal*)
Let G be a collage grammar. The *upper fractal* generated by G is the overlay of the induced patterns of all generated collages:

$$\mathit{fractal}_{\mathrm{u}}(G) = \bigcup_{C \in L(G)} \mathit{pattern}(C).$$

The *lower fractal* generated by G is the intersection of the induced patterns of all generated collages:

$$\mathit{fractal}_{\mathrm{l}}(G) = \bigcap_{C \in L(G)} \mathit{pattern}(C).$$

Is it justified to call the patterns $\mathit{fractal}_{\mathrm{u}}(G)$ and $\mathit{fractal}_{\mathrm{l}}(G)$ fractals? Clearly, the answer depends on the meaning of the term *fractal*, and there seems to be no standard definition. One motivation to use the term fractal is that the considered overlay and intersection may have a fine structure with details on arbitrarily small scale and a shape too bizarre and irregular to be described in traditional geometry. Moreover, several well-known fractals can be generated by means of collage grammars (cf. [23]).

There is, however, a formal justification as well. In this section two types of collage grammars will be discussed, called increasing and decreasing generalized

Sierpiński grammars, that generate self-affine fractals. More exactly speaking, the upper fractal of an increasing generalized Sierpiński grammar and the lower fractal of a decreasing one are self-affine. This shows that collage grammars provide syntactic descriptions of fractals and, therefore, may be useful for the analysis of fractals. Moreover, every subset of the generated languages and every set of derivations provide approximations of the generated fractals. In this way collage grammars become means to visualize fractals.

Another result is that every self-affine fractal of a certain type is the lower fractal of a suitably chosen decreasing generalized Sierpiński grammar. Together with the result mentioned above this gives a characterization of self-affine patterns as those which can be generated using a decreasing generalized Sierpiński grammar.

The following definition of self-affinity imposes a strong requirement on the way in which transformed patterns are allowed to intersect. To meet the more general notions of self-affinity one usually encounters in the literature (see, e.g., Mandelbrot [31] and Edgar [17]) this requirement must be relaxed or dropped.

Definition 11.7.2 (self-affinity and self-similarity)

A bounded pattern $pat \subseteq \mathbb{R}^d$ is *self-affine* if there are pairwise distinct affine contractions[8] $a_i \colon \mathbb{R}^d \to \mathbb{R}^d$ for $i = 1, \ldots, k$ with $k \geq 2$ such that

$$pat = \bigcup_{i=1}^{k} a_i(pat) \quad \text{and}$$

$$[a_i(pat)] \cap [a_j(pat)] \subseteq \delta[a_i(pat)] \cap \delta[a_j(pat)] \text{ for all } i \neq j.$$

A self-affine pattern is *self-similar* if the involved contractions are similarities.

Now, we are going to define increasing and decreasing generalized Sierpiński grammars. A generalized Sierpiński grammar is a very special proper collage grammar with only one nonterminal S and two productions. The right-hand side of the first production is a decorated collage with at least two hyperedges (and possibly some parts). The second right-hand side is an undecorated collage. Moreover, the pin points, the attachment points, and the parts of the right-hand sides must be related in a certain way. The requirements ensure that the relative positions of pin and attachment points reflect certain aspects of self-affinity. The conditions are easy to check because there are only a finite

[8]A transformation $a \colon \mathbb{R}^d \to \mathbb{R}^d$ is a *contraction* if there is some $c < 1$ such that $dist(a(x), a(y)) \leq c \cdot dist(x, y)$ for all $x, y \in \mathbb{R}^d$.

number of pin and attachment points. A generalized Sierpiński grammar is called increasing if the parts lie inside the convex hull of the pin points and the overlay of the transformed parts (using the associated affine transformations) covers the parts. It is called decreasing if the parts of the second right-hand side cover the overlay of its transformed parts (again using the associated affine transformations) as well as the parts of the first right-hand side.

Definition 11.7.3 (generalized Sierpiński grammar)

1. A proper collage grammar $G = (\{S\}, \{(S, R_1), (S, R_2)\}, (S, pin)^\bullet)$ is a *generalized Sierpiński grammar* if the following conditions are fulfilled:

   (1) E_{R_1} contains at least two hyperedges,

   (2) $E_{R_2} = \emptyset$,

   (3) $[att_{R_1}(e)] \subseteq [pin]$ for all $e \in E_{R_1}$,

   (4) for every hyperedge $e \in E_{R_1}$ the unique affine transformation $a(e)$ that maps pin to $att_{R_1}(e)$ is a contraction,

   (5) $[att_{R_1}(e)] \cap [att_{R_1}(e')] \subseteq \delta[att_{R_1}(e)] \cap \delta[att_{R_1}(e')]$ for all $e, e' \in E_{R_1}$, $e \neq e'$,

   (6) $pattern(R_1) \cup pattern(R_2)$ is non-empty and bounded.

2. G is called *increasing* if the following conditions hold in addition:

   (7) $pattern(R_1) \cup pattern(R_2)$
   $\subseteq \bigcup_{e \in E_{R_1}} a(e)(pattern(R_1) \cup pattern(R_2))$,

   (8) $pattern(R_1) \cup pattern(R_2) \subseteq [pin]$.

3. G is called *decreasing* if, in addition to (1)–(6),

   (7′) $pattern(R_1) \cup \bigcup_{e \in E_{R_1}} a(e)(pattern(R_2)) \subseteq pattern(R_2)$,

   (8′) $pattern(R_1) \subseteq \bigcup_{e \in E_{R_1}} a(e)(pattern(R_1))$.

The reference to Sierpiński in the name of this type of grammars is made because the form of productions we require is inspired by famous fractals like the Sierpiński gasket and the Sierpiński carpet.

We are now ready to state the first result mentioned above.

Theorem 11.7.4

Let G be an increasing generalized Sierpiński grammar. Then $fractal_{\mathrm{u}}(G)$, the upper fractal generated by G, is self-affine.

Moreover, if all affine transformations involved in the definition of G are similarities, $fractal_{\mathrm{u}}(G)$ is self-similar. □

The proof is not very difficult. It consists merely of a rather straightforward verification of the fact that the transformations associated with the hyperedges in E_{R_1} satisfy the requirements of Definition 11.7.2. The proof of the corresponding result for the decreasing case and the lower fractal is more complicated. In particular, it makes use of the following two lemmas stating that (for decreasing generalized Sierpiński grammars) it may be assumed that only the "terminal" right-hand side R_2 contains parts and, secondly, the points of patterns induced by generated collages are exactly those which can be obtained from the right-hand sides by a sequence of the affine transformations associated with the given grammar.

Lemma 11.7.5
Let $G = (\{S\}, \{(S, R_1), (S, R_2)\}, (S, pin)^\bullet)$ be a decreasing generalized Sierpiński grammar and let $G' = (\{S\}, \{(S, R_1'), (S, R_2)\}, (S, pin)^\bullet)$ be given by $R_1' = (\emptyset, E_{R_1}, att_{R_1}, lab_{R_1}, pin_{R_1})$. Then

$$\{pattern(C) \mid C \in L(G)\} = \{pattern(C) \mid C \in L(G')\}. \qquad \square$$

Intuitively, the essence of the second lemma can be explained as follows. Suppose we are working in $\mathbb{R}^2$ and consider the polygon spanned by pin. Then, if one of the hyperedges in a derived collage is replaced by R_1, the new hyperedges span polygons that are nested within the old one. The lemma now says that the points of $fractal_1(G)$ are exactly those x for which we can find an infinite sequence of these nested polygons, of which x is the limit. In other words, a sequence $(e_{x,i})_{i\geq 1}$ of hyperedges must exist such that

$$x \in [pin],$$
$$x \in a(e_{x,1})([pin]) \subseteq [pin],$$
$$x \in a(e_{x,1})(a(e_{x,2})([pin])) \subseteq a(e_{x,1})([pin]) \subseteq [pin],$$

and so on. Formally, we have the following lemma.

Lemma 11.7.6
Let $G = (\{S\}, \{(S, R_1), (S, R_2)\}, (S, pin)^\bullet)$ be a generalized Sierpiński grammar with $PART_{R_1} = \emptyset$ and let $x \in \mathbb{R}^d$. Then x is a point of $fractal_1(G)$ if and only if there is a sequence $(e_{x,i})_{i\geq 1}$ of hyperedges $e_{x,i} \in E_{R_1}$ such that $x \in a(e_{x,1}) \circ \cdots \circ a(e_{x,i})(pattern(R_2))$ for all $i \in \mathbb{N}$. $\square$

Lemmas 11.7.5 and 11.7.6 can now be used to establish the self-affinity of $fractal_1(G)$ for decreasing generalized Sierpiński grammars.

Theorem 11.7.7

Let G be a decreasing generalized Sierpiński grammar. Then $\mathit{fractal}_{\mathrm{l}}(G)$, the lower fractal generated by G, is self-affine.

Moreover, if all affine transformations involved are similarities, $\mathit{fractal}_{\mathrm{l}}(G)$ is self-similar. □

Again, the proof is omitted. As mentioned above, it is also possible to prove the converse of the theorem. In order to state this result, the following definition is useful.

Definition 11.7.8 (canonical grammar)

Let $pin \in (\mathbb{R}^d)^*$, let $E = \{e_1, \ldots, e_k\}$ be a finite set, and let $a(e)$ be an affine transformation for every $e \in E$. Then the *canonical grammar* $G(pin, a)$ given by a and pin is $G(pin, a) = (\{S\}, \{(S, R_1), (S, R_2)\}, (S, pin)^\bullet)$, where $R_1 = (\emptyset, E, att, lab, pin)$ with $att(e_i) = a(e_i)(pin)$ and $lab(e_i) = S$, and $R_2 = (\{[pin]\}, pin)$.

In general, the canonical grammar is not a generalized Sierpiński grammar. Notice, however, that it is a decreasing generalized Sierpiński grammar if it is a generalized Sierpiński grammar at all. Now, suppose *pat* is self-similar with respect to affine contractions $a_1, \ldots, a_k$ (see Definition 11.7.2). The fact that, for distinct i, j, the convex hulls of $a_i(pat)$ and $a_j(pat)$ overlap only in their boundaries, can be used to construct a finite sequence pin of pin points such that $a_i([pin]) \subseteq [pin]$ and $a_i([pin]) \cap a_j([pin]) \subseteq \delta[a_i(pin)] \cap \delta[a_j(pin)]$ for $1 \leq i < j \leq k$. Now, using pin and the a_i to build the canonical grammar, a decreasing generalized Sierpiński grammar is obtained which generates *pat* as its lower fractal. This is stated in the following theorem.

Theorem 11.7.9

Let pat be a self-affine pattern. Then there exist a finite sequence pin of pin points, a finite set E of hyperedges, and a mapping a which assigns an affine transformation $a(e)$ to every hyperedge $e \in E$, such that $G(pin, a)$ is a decreasing generalized Sierpiński grammar with

$$\mathit{fractal}_{\mathrm{l}}(G(pin, a)) = pat. \qquad \square$$

The proof of the theorem relies on the construction mentioned above, as well as on the following lemma concerning self-affine patterns, which is similar to Lemma 11.7.6.

Lemma 11.7.10
Let pat $\subseteq \mathbb{R}^d$ be self-affine with respect to affine mappings $a_1, \ldots, a_k$ and let $x \in \mathbb{R}^d$. Then $x \in pat$ if and only if there is a sequence $(i_j)_{j \geq 1}$, $1 \leq i_j \leq k$, such that $x \in a_{i_1} \circ \cdots \circ a_{i_j}(pat)$ for all $j \in \mathbb{N}$. □

As a consequence of Theorem 11.7.9 one obtains a corollary stating that, no matter what the induced patterns of R_1 and R_2 are, if two decreasing generalized Sierpiński grammars use the same affine transformations in R_1 then their lower fractals are the same (and are both equal to the self-affine pattern described by the given transformations). This statement is closely related to the well-known result from fractal geometry saying that the fractal generated by an iterated function system is independent of the picture the iteration starts with (cf. the *contraction mapping principle* [35]).

Corollary 11.7.11
Let pat be a self-affine pattern and let G, G' be decreasing generalized Sierpiński grammars, all three using the same affine transformations. Then

$$\mathit{fractal}_l(G) = \mathit{pat} = \mathit{fractal}_l(G'). \qquad \square$$

Further research in this area may investigate more general types of productions, that perhaps still yield self-affine upper and lower fractals. Theorem 11.7.9 would then yield a kind of normal form for such collage grammars because it says that every collage grammar that generates a self-affine upper or lower fractal can be transformed (although perhaps not effectively) into a decreasing generalized Sierpiński grammar that generates the same fractal (as its lower fractal).

Of course, self-affine pictures are not the only type of fractal one may consider. There are other sorts of fractals, and, from an intuitive point of view, it seems to be clear that some of them can and others cannot be generated by collage grammars. Also, taking the union or intersection of potentially infinite sets of patterns is a somewhat rough way to obtain fractals from collage languages. More sophisticated limit constructions may yield different results. Moreover, to prove the results summarized in this section elementary notions and standard techniques from geometry suffice. The application of more evolved concepts like, for instance, Hausdorff spaces, geometric measure theory, and fractal dimensions may lead to further insight or simpler proofs.

11.8 Comparison with Other Methods of Picture Generation

In this section collage grammars are compared with iterated function systems and chain-code picture grammars.

11.8.1 Iterated Function Systems

In the previous section, the discussion of the relationship between collage grammars and fractal geometry concentrated on the semantic aspects of self-affinity. In this section, we include syntactic aspects. Iterated function systems are well-known devices to describe fractals (see, e.g., Barnsley [1], Culik and Dube [4], Hutchinson [27], and Peitgen, Jürgens, and Saupe [35]). An iterated function system consists of a finite set of contractions on a Hilbert space. It induces the so-called Hutchinson operator which can be applied to any set of points and yields the union of all images of this set under the given functions. Therefore, it can be iterated yielding an infinite sequence of sets of points that converges if the initial set of points is compact. The limit—often called the attractor—is the unique fixed point of the Hutchinson operator. We show that iterated function systems can be translated into collage grammars in a way that preserves semantics. To keep technicalities simple, we shall continue to concentrate on the space $\mathbb{R}^d$, instead of considering an arbitrary Hilbert space (as is usually done in fractal geometry).

Definition 11.8.1

1. An *iterated function system* (IFS) is a finite set $I = \{a_1, \ldots, a_k\}$ of affine transformations on $\mathbb{R}^d$ (for some $d \in \mathbb{N}$).
2. The *Hutchinson operator* H_I of I is defined on patterns $pat \subseteq \mathbb{R}^d$ by
$$H_I(pat) = \bigcup_{i=1}^{n} a_i(pat).$$
3. Choosing some initial pattern pat_0, the application of H_I can be iterated ad infinitum yielding the *Hutchinson sequence* $(pat_i)_{i \in \mathbb{N}}$ given by
$$pat_{i+1} = H_I(pat_i).$$

We will now construct, for a given IFS, a corresponding TOL collage grammar. For this, the notion of significant sequences of points is needed. A sequence $x_0 \cdots x_d$ of $d+1$ points is called *significant* if $x_1 - x_0, \ldots, x_d - x_0$ form a basis of $\mathbb{R}^d$ (and thus every affine transformation a is uniquely determined by the images of these points under a).

Construction 11.8.2

1. *Let* $I = \{a_1, \ldots, a_k\}$ *be an IFS,* $pat_0 \subseteq \mathbb{R}^d$*, and* pin *be a significant sequence of points. Then the corresponding TOL collage grammar is given by*

$$CG(I, pat_0, pin) = (\{S\}, \{P_1, P_2\}, (S, pin)^\bullet)$$

with the single nonterminal label S*, the axiom being the handle induced by* S *and* pin*, and two tables* $P_i = \{(S, R_i)\}$*,* $i = 1, 2$*, each containing a single production.*

R_1 *is a decorated collage without parts. It has the sequence* pin *of pin points and* k *hyperedges* $e_1, \ldots, e_k$ *each of which is labelled with* S*. Each hyperedge* e_i *is attached to* $a_i(pin)$*, thus representing the transformation* a_i*. Hence,*

$$R_1 = (\emptyset, \{e_1, \ldots, e_k\}, att_{R_1}, lab_{R_1}, pin)$$

with $att_{R_1}(e_i) = a_i(pin)$ *and* $lab_{R_1}(e_i) = S$ *for* $i = 1, \ldots, k$*.*

R_2 *consists of the single part* pat_0 *and the pin-point sequence* pin*, i.e.,* $R_2 = (\{pat_0\}, pin)$*.*

2. *Given a decorated collage the hyperedges of which are labelled with* S*, the application of one of the two tables yields a unique result. It is terminal if* P_2 *is applied, because all hyperedges are replaced by the (transformed) part* pat_0*. The application of* P_1 *can be repeated. In other words, there is a unique derivation of the form*

$$(S, pin)^\bullet \overset{n}{\underset{P_1}{\Longrightarrow}} \overline{C}_n \underset{P_2}{\Longrightarrow} C_n$$

for every $n \in \mathbb{N}$*. In this way, the grammar* $CG(I, pat_0, pin)$ *induces a sequence* $(C_i)_{i \in \mathbb{N}}$ *of derived collages.* □

Example 11.6

Consider the IFS which is given by the three transformations $zoom(1/2)$, $translate((1,0)) \circ zoom(1/2)$, and $translate((1/2, \sqrt{3}/2)) \circ zoom(1/2)$, where $zoom(1/2)$ is the contraction by the factor 1/2 and $translate(x)$ is the translation by the vector x. Let the initial set of points be the triangle given by the corners $x_1 = (0,0)$, $x_2 = (1,0)$, and $x_3 = (1/2, \sqrt{3}/2)$, and choose $x_1x_2x_3$ as the sequence of pin points. Then the first members of the Hutchinson sequence are those depicted in Figure 11.10 and the corresponding TOL collage grammar is established by the two productions shown in Figure 11.2. □

It can be shown that the pattern induced by the i-th derived collage equals the i-th element of the Hutchinson sequence. In this sense, the translation of IFS's into TOL collage grammars preserves the semantics and can be considered as correct.

Theorem 11.8.3

Let $I = \{a_1, \ldots, a_k\}$ be an IFS, $pat_0 \subseteq \mathbb{R}^d$, and pin be a significant sequence of points. Let $(pat_i)_{i\in\mathbb{N}}$ be the resulting Hutchinson sequence.

1. *If $(C_i)_{i\in\mathbb{N}}$ is the sequence of collages derived by $CG(I, pat_0, pin)$ then $pat_i = pattern(C_i)$ for all $i \in \mathbb{N}$.*
2. *Let $a_1, \ldots, a_n$ be contractions and pat_0 be compact, so that the Hutchinson sequence converges to a limit (or attractor) frac. Let, finally, $pattern(repl(e)) = frac$ for all hyperedges $e \in E_{R_1}$, where R_1 is as in the construction of $CG(I, pat_0, pin)$. Then it holds that*

$$frac = pattern(R_1[repl]).$$

Proof sketch

For the first part of the theorem, let $A_i = \{a_{j_1} \circ \cdots \circ a_{j_i} \mid j_1, \ldots, j_i \in \{1, \ldots, k\}\}$ for all $i \in \mathbb{N}$, and thus $pat_i = \bigcup_{a\in A_i} a(pat_0)$. Then the asserted equality $pat_i = pattern(C_i)$ follows from the more general fact that $PART_{C_i} = \{a(pat_0) \mid a \in A_i\}$, which holds because A_i is the set of all affine transformations a such that $a(pin) = att_{\overline{C}_i}(e)$ for some $e \in E_{\overline{C}_i}$. The latter can be shown by induction, as follows. For $i = 0$ we have $A_0 = \{id\}$ and $pin = att_{\overline{C}_0}(e)$, where e is the unique hyperedge of $\overline{C}_0 = (S, pin)^\bullet$. Now, assume that the claim holds for $\overline{C}_i$. Then, by the definition of hyperedge replacement, replacing a hyperedge $e \in E_{\overline{C}_i}$ by R_1, where $a(pin) = att_{\overline{C}_i}(e)$, yields hyperedges $e'_1, \ldots, e'_k \in E_{\overline{C}_{i+1}}$ such that $att_{\overline{C}_{i+1}}(e'_j) = a(att_{R_1}(e_j)) = a(a_j(pin))$, which means $A_{i+1} = \bigcup_{a\in A_i}\{a_1 \circ a, \ldots, a_k \circ a\}$.

For the second part, notice that, by the definition of hyperedge replacement, $R_1[repl] = (\{a_i(frac) \mid 1 \le i \le k\}, pin)$. Consequently, $pattern(R_1[repl]) = H_I(frac) = frac$ because the attractor of I is known to be the unique fixed point of H_I. □

In similar ways, more general types of iterated function systems can be translated into TOL collage grammars like iterated function systems with condensation (see, e.g., Barnsley [1]) or mutually recursive iterated function systems (see, e.g., Culik and Dube [4]), which are also called hierarchical iterated function systems (see, e.g., Peitgen, Jürgens, and Saupe [35]). Such translations (including the one presented in this section) are studied in [40] in the more general setting of arbitrary Hilbert spaces.

11.8.2 Chain-Code Picture Languages

In this section the capabilities of collage grammars on the one hand and chain-code grammars on the other are compared, summarizing results from [5,8].

The notion of chain-code grammars has been introduced in [32] (see also [39, 29,6]). These grammars are in fact Chomsky grammars (here, context-free and regular ones are considered) that generate languages over a particular alphabet, namely $A_{cc} = \{u, d, l, r, \uparrow, \downarrow\}$. A string over these symbols is then interpreted as a sequence of instructions to a plotter-like device: $\uparrow$ and $\downarrow$ cause it to lift the pen and to set it down again, respectively, whereas u, d, l, and r make the pen move one unit up, down, to the left, and to the right, respectively.

Formally, for every point $p = (x, y) \in \mathbb{Z}^2$ let $u(p)$, $d(p)$, $l(p)$, and $r(p)$ denote its neighbours $(x, y+1)$, $(x, y-1)$, $(x-1, y)$, and $(x+1, y)$, respectively. Furthermore, for $a \in \{u, d, l, r\}$ let $aline(p)$ be the straight line segment between p and $a(p)$. Now, a *line drawing* is a finite set D of parts in $\mathbb{R}^2$ such that every $part \in D$ has the form $uline(p)$ or $rline(p)$ for some $p \in \mathbb{Z}^2$. In the following, we shall identify a line drawing D with the collage (D, λ), i.e., we consider a line drawing as a collage whose sequence of pin points is empty. The set of all line drawings is denoted by $\mathcal{D}$.

A *picture description* is a word over the alphabet $A_{cc} = \{u, d, l, r, \uparrow, \downarrow\}$. Every word $w \in A_{cc}^*$ determines a *drawn picture* $dpic(w) \in \mathcal{D} \times \mathbb{Z}^2 \times \{\uparrow, \downarrow\}$:

(i) $dpic(\lambda) = (\emptyset, (0,0), \downarrow)$ and

(ii) for every picture description $v \in A_{cc}^*$ with $dpic(v) = (D, p, s)$ and every $a \in A_{cc}$, if $a \in \{\uparrow, \downarrow\}$ then $dpic(va) = (D, p, a)$. Otherwise,

$$dpic(va) = \begin{cases} (D \cup \{aline(p)\}, a(p), s) & \text{if } s = \downarrow \\ (D, a(p), s) & \text{if } s = \uparrow. \end{cases}$$

The line drawing $drawing(w)$ described by $w \in A_{cc}^*$ is the first component of $dpic(w)$, i.e., $drawing(w) = D$ if $dpic(w) = (D, p, s)$.

A *chain-code grammar* is a Chomsky grammar g whose alphabet of terminal symbols is A_{cc}. The *chain-code picture language* generated by g is the set $\mathcal{L}(g) = \{drawing(w) \mid w \in L(g)\}$.[9]

[9] For a Chomsky grammar g, $L(g)$ denotes the language generated by g.

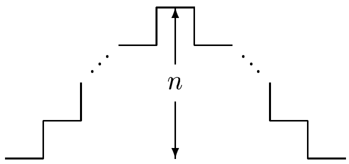


Figure 11.25: The "twin stairs" TS_n in the language of line drawings generated by the productions $A ::= ruAdr$ and $A ::= r$

Are context-free chain-code grammars more powerful than collage grammars? Or vice versa? Of course, this question makes sense only if we restrict our considerations to collage grammars that generate line drawings. Furthermore, a collage grammar simulating a chain-code grammar g (or vice versa) should probably not that strictly be required to generate $\mathcal{L}(g)$. Instead, two languages should be considered equivalent if they consist of the same line drawings *up to translation*. This is because of the fact that one is usually interested in the figures being generated, but not so much in their exact position. Therefore, let $D \equiv D'$ for line drawings D, D' if D' is obtained from D via translation, and define $L \equiv L'$ for languages L, L' of line drawings if their quotient sets with respect to $\equiv$ are equal.

It turns out that the classes of line-drawing languages generated by the two devices are incomparable even if only equivalence with respect to $\equiv$ is required. For the first direction, consider the chain-code grammar $g = (\{A\}, A_{cc}, \{A ::= ruAdr, A ::= r\}, A)$ (which, as the reader should notice, does not even make use of $\uparrow$ and $\downarrow$). It generates the language of all "twin stairs" $TS_0, TS_1, \ldots$ as indicated in Figure 11.25.

Suppose there is a collage grammar G such that $L(G) \equiv \mathcal{L}(g)$ and let S be a finite set of collage-replacement rules as in Theorem 11.5.3. Due to Theorem 11.5.3 there are infinitely many $m \in \mathbb{N}$ such that $C \Longrightarrow_S C'$ for some $C \equiv TS_m$ and $C' = TS_n$, where $n < m$. Consider one such situation, i.e., $C' = (C \setminus a(L)) \cup a(R)$ for an affine transformation a, where $(L ::= R) \in S$. As $n < m$, the two bottom lines in C cannot be in C', which means that they must be in $a(L)$. Now, if the pre-images of these lines in L are of length c then their distance must be $(2m-1)/c$ because this is the respective ratio in C, which is preserved by affine transformations. In other words, m is uniquely determined by L, contradicting the finiteness of S. Hence we have proved that $\mathcal{L}(g)$ cannot be generated by a collage grammar.[10]

[10] As an exercise the reader may try to find a part-sensitive collage grammar that generates $\mathcal{L}(g)$ (yes, there is one!).

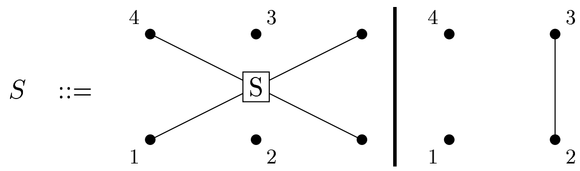


Figure 11.26: Productions of a collage grammar generating a language of line drawings that cannot be generated by a context-free chain-code grammar

As for the converse, consider the collage grammar $G = (\{S\}, P, Z)$, where $Z = (\{\mathit{rline}((0,0))\}, \{e\}, \mathit{att}, \mathit{lab}, \lambda)$ with $\mathit{att}(e) = (0,0)(1,0)(1,1)(0,1)$ and $\mathit{lab}(e) = S$, and P consists of the two productions shown in Figure 11.26. Then $L_1 = L(G)$ is the set of all collages $\{\mathit{rline}((0,0)), \mathit{uline}((2^n, 0))\}$ $(n \in \mathbb{N})$. The set of points in the grid $\mathbb{Z}^2$ that occur as end-points of lines in the collages of this language is thus given by $\mathit{grid\text{-}points}(L(G)) = \{(0,0),(1,0)\} \cup \bigcup_{n \in \mathbb{N}} \{(2^n, 0), (2^n, 1)\}$. Now, assume that $L(G)$ can be generated by a context-free chain-code grammar. Using Parikh's theorem for context-free string languages [33] it can be shown quite easily that this implies semi-linearity of $\mathit{grid\text{-}points}(L(G))$. (Recall that a subset of $\mathbb{Z}^m$ is semi-linear if it is a finite union of sets of the form $\{z_0 + \sum_{i=1}^{n} a_i z_i \mid a_1, \ldots, a_n \in \mathbb{N}\}$, where $z_0, \ldots, z_n \in \mathbb{Z}^m$.) However, $\mathit{grid\text{-}points}(L(G))$ is obviously not semi-linear. Thus, we cannot have $\mathcal{L}(g) = L(G)$ for any chain-code grammar g. In fact, we cannot even have $\mathcal{L}(g) \equiv L(G)$: It is an easy exercise in modifying context-free grammars to show that such a grammar g could actually be turned into a grammar g' such that $\mathcal{L}(g') = L(G)$.

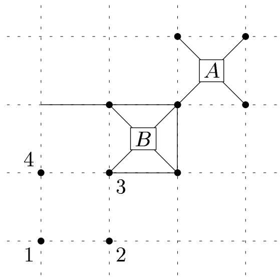
Notice finally that in both directions the counterexample is given by a linear grammar, i.e., a grammar with at most one nonterminal in every right-hand side and in the axiom. Altogether, this amounts to the following theorem.

Theorem 11.8.4

(1) *There is a linear context-free chain-code grammar generating a language L_0 of line drawings, such that there does not exist any context-free collage language L'_0 with $L'_0 \equiv L_0$.*

(2) *There is a linear context-free collage language L_1 consisting of line drawings, such that there does not exist any context-free chain-code grammar that generates a language L'_1 with $L'_1 \equiv L_1$.* □

The natural question now is whether one can find restrictions under which the two devices yield comparable classes of line-drawing languages. For this, let us say that a collage grammar $G = (N, P, Z)$ *uses* a transformation a if there are

$A ::=$ 

$$\leadsto A ::= \underbrace{\uparrow u\, u \downarrow r\, r\, d\, l}_{\substack{\text{generate}\\ PART_R}} \underbrace{\uparrow\downarrow B \uparrow u\, r \downarrow A}_{\substack{\text{generate}\\ \text{nonterminals}}} \underbrace{\uparrow d\, d\, l\, l \downarrow}_{\substack{\text{return to}\\ \text{origin}}}$$

Figure 11.27: How to convert a production of a collage grammar into a chain-code production

$(A ::= R) \in P$, $R' \in \{Z\} \cup \{rhs(p) \mid p \in P\}$, and $e \in E_{R'}$ with $lab_{R'}(e) = A$ such that $a(pin_R) = att_{R'}(e)$.

Notice that the collage grammar in the proof of Theorem 11.8.4(2) uses a non-uniform scaling in its first production: while there is no vertical scaling, horizontally a scaling by the factor 2 is imposed. In effect, this yields an exponential translation of the vertical line being generated. The fact that there is no vertical scaling ensures that this line remains a unit line, as required. A uniform scaling could not be used in the same way because it would necessarily affect the length of lines. For similar reasons, the generation of line drawings cannot significantly make use of rotations. This is why it is possible to show that collage grammars which generate line drawings by using only similarity transformations can be simulated by context-free chain-code grammars.

Theorem 11.8.5

Every language of line drawings that can be generated by a collage grammar using only similarity transformations can be generated by a context-free chain-code grammar. □

The main part of the proof is to show that, based on the observations made above, a collage grammar of the type considered in the theorem can be modified in such a way that it uses only translations in the grid $\mathbb{Z}^2$. More precisely, if R is a right-hand side of a production in G, then (a) $PART_R$ is a line drawing, (b) $pin_R = (0,0)(1,0)(1,1)(0,1)$, and (c) for all $e \in E_R$ there is a point $(x,y) \in \mathbb{Z}^2$ such that $att_R(e) = (x,y)(x+1,y)(x+1,y+1)(x,y+1)$. Based on a collage grammar of this kind a corresponding context-free chain-code grammar can be constructed in a straightforward way. For a sample production this is indicated in Figure 11.27. In the case of linear collage grammars the last part ("return to origin") is not required. Thus, a regular grammar is obtained in this case.

Moreover, for regular chain-code grammars it is easy to construct equivalent collage grammars (that use only translations). This amounts to the following result.

Theorem 11.8.6

For every language L of line drawings the following are equivalent.

(1) L can be generated by a regular chain-code grammar;

(2) L can be generated by a linear collage grammar using only similarity transformations;

(3) L can be generated by a linear collage grammar using only translations. □

As a consequence, decidability and undecidability results known for regular chain-code grammars carry over to linear collage grammars using only similarity transformations. For details, the reader is referred to [6,5].

11.9 Software Systems

In the following, we shall briefly describe some software systems implementing collage grammars that have been developed at the University of Bremen.

11.9.1 Collage-One*, *Collage-Two*, and *Collage-VR

The systems Collage-One, Collage-Two, and Collage-VR constitute a line of development that was initiated by Stefan Taubenberger and continued by Nils Schwabe. All three systems allow to evaluate collage grammars in $\mathbb{R}^2$ and to visualize the derived collages. Furthermore, every derived collage can be saved as a PostScript file using a format which is especially suitable for inclusion in LaTeX documents such as this one. Collage grammars are defined in a textual way, using an easily understandable syntax.

Collage-One, the first one of these systems, is described in more detail in [16]. Collage-Two is an extension of Collage-One that was developed by Nils Schwabe as part of his diploma thesis. Its main advantage is that it can deal with context-sensitive productions (including those where the context contains hyperedges). Furthermore, collage grammars can be parameterized in Collage-Two. These parameters can be changed dynamically using tcl scripts, which makes it possible to generate animation sequences.

As an extension of Collage-Two, Collage-VR allows to evaluate and visualize collage grammars in $\mathbb{R}^3$ as well. The visualization of three-dimensional collages makes use of a standard VRML[11] viewer, so that the user can easily navigate through a displayed collage, turn it around, zoom in and out, etc. COLLAGE-VR can be obtained at http://www.informatik.uni-bremen.de/~ns/cs/Main.html.

11.9.2 The BIZARR Studio

The BIZARR studio is a system that was developed by BIZARR2, a students' project that was run from the winter term 1992/93 until summer 1994. The system makes it possible to generate three-dimensional animations by combining several methods of picture generation (L-systems with turtle geometry, map-L-systems, cellular automata, iterated function systems, and collage grammars) in a single animation. The generated animations are stored as files in MPEG format. Some animations that have been generated by this system are available at http://www.informatik.uni-bremen.de/~drewes/bizarr2.html.

11.9.3 TREEBAG

The newest development related to the topics discussed here is a Java implementation of a system called TREEBAG. It allows to generate collages and line drawings, but also numbers, strings, terms, trees, etc. using tree generators—devices that generate terms—whose output terms are then interpreted by appropriate algebras. The objects resulting from this interpretation can be visualized using suitable display components. For more information see [9,11, 10], the CDROM belonging to this volume, or http://www.informatik.uni-bremen.de/~drewes/treebag.

11.10 Conclusion

In this chapter, we have surveyed the collage grammar approach providing methods and tools to generate and analyse pictures and picture languages. Although the state of the art may show the potentials, much more work must be done in order to establish collage grammars as a picture processing concept that can be used in practice. Future studies will include the following topics.

(1) So far, the significance of collage grammars is illustrated by small examples only. In the future, the usefulness of collage grammars should be demonstrated by modelling, visualizing, or animating more realistic or practical situations, scenes, or developmental processes.

[11]VRML = *virtual reality modelling language*

(2) The results on the relationship between collage grammars, iterated function systems, and chain-code picture languages give some idea how collage grammars are placed among the variety of picture generating formalisms. To obtain a deeper understanding, collage grammars should be compared with other prominent approaches like L-systems with turtle geometry and map-L-systems (see [36]), or cellular automata (see [42,41]).

While the context-free case of collage grammars provides a nice theory, for practical reasons and purposes it may be necessary to increase the generative power even beyond the table-driven or part-sensitive mode of rewriting. In this respect it may be interesting to study other classes of transformations than affine ones, or to search for more sophisticated kinds of regulation.

But whichever serious goals one aims at in the investigation of collage grammars, they can also be fun.

Acknowledgement We thank Annegret Habel and Renate Klempien-Hinrichs for interesting discussions and a large number of constructive comments on a draft version of this survey.

References

1. Michael Barnsley. *Fractals Everywhere*. Academic Press, Boston, 1988.
2. Bruno Courcelle. The monadic second–order logic of graphs. I. Recognizable sets of finite graphs. *Information and Computation*, 85:12–75, 1990.
3. H.S.M. Coxeter. *Introduction to Geometry*. John Wiley & Sons, New York, Wiley Classics Library, second edition, 1989.
4. Karel Culik II and Simant Dube. Affine automata and related techniques for generation of complex images. *Theoretical Computer Science*, 116:373–398, 1993.
5. Jürgen Dassow, Annegret Habel, and Stefan Taubenberger. Chain-code pictures and collages generated by hyperedge replacement. In H. Ehrig, H.-J. Kreowski, and G. Rozenberg, editors, *Graph Grammars and Their Application to Computer Science*, volume 1073 of *Lecture Notes in Computer Science*, pages 412–427, 1996.
6. Jürgen Dassow and Friedhelm Hinz. Decision problems and regular chain code picture languages. *Discrete Applied Mathematics*, 45:29–49, 1993.
7. Frank Drewes. Language theoretic and algorithmic properties of d-dimensional collages and patterns in a grid. *Journal of Computer and System Sciences*, 53:33–60, 1996.

8. Frank Drewes. Some remarks on the generative power of collage grammars and chain-code grammars. In *Preliminary Proc. Sixth Intl. Workshop on Theory and Application of Graph Transformations (TAGT '98)*, pages 425–432, 1998.
9. Frank Drewes. TREEBAG—a tree-based generator for objects of various types. Report 1/98, Univ. Bremen, 1998.
10. Frank Drewes. Tree-based picture generation. Report 7/98, Univ. Bremen, 1998.
11. Frank Drewes. TREEBAG – Baum-basierte Generierung und Transformation von Objekten. In J. Dassow and R. Kruse, editors, *Proc. Informatik '98*, Informatik Aktuell, pages 47–56, 1998. In German.
12. Frank Drewes, Annegret Habel, and Hans-Jörg Kreowski. Hyperedge replacement graph grammars. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation. Vol. 1: Foundations*, chapter 2, pages 95–162. World Scientific, 1997.
13. Frank Drewes, Annegret Habel, Hans-Jörg Kreowski, and Stefan Taubenberger. Generating self-affine fractals by collage grammars. *Theoretical Computer Science*, 145:159–187, 1995.
14. Frank Drewes and Hans-Jörg Kreowski. (Un-)decidability of geometric properties of pictures generated by collage grammars. *Fundamenta Informaticae*, 25:295–325, 1996.
15. Frank Drewes, Hans-Jörg Kreowski, and Denis Lapoire. Criteria to disprove context-freeness of collage languages. In B.S. Chlebus and L. Czaja, editors, *Proc. Fundamentals of Computation Theory XI*, volume 1279 of *Lecture Notes in Computer Science*, pages 169–178, 1997.
16. Frank Drewes, Hans-Jörg Kreowski, and Nils Schwabe. COLLAGE-ONE: A system for evaluation and visualisation of collage grammars. In *Proc. 4th International Conference on Computer Graphics & Image Processing GKPO'96*, volume 5 of *Machine Graphics & Vision*, pages 393–402, Warsaw, Poland, 1996. Polish Academy of Sciences.
17. Gerald A. Edgar. *Measure, Topology, and Fractal Geometry.* Springer, New York, 1990.
18. Kenneth Falconer. *Fractal Geometry: Mathematical Foundations and Applications.* John Wiley & Sons, Chichester, 1990.
19. James Gips. *Shape Grammars and Their Uses, Artificial Perception, Shape Generation, and Computer Aesthetics.* Birkhäuser Verlag, Basel/Stuttgart, 1975.
20. Branko Grünbaum and G.C. Shephard. *Tilings and Patterns.* Freeman, New York, 1987.

21. Annegret Habel. *Hyperedge Replacement: Grammars and Languages*, volume 643 of *Lecture Notes in Computer Science*. Springer, 1992.
22. Annegret Habel and Hans-Jörg Kreowski. Collage grammars. In H. Ehrig, H.-J. Kreowski, and G. Rozenberg, editors, *Proc. Fourth Intl. Workshop on Graph Grammars and Their Application to Comp. Sci.*, volume 532 of *Lecture Notes in Computer Science*, pages 411–429, 1991.
23. Annegret Habel, Hans-Jörg Kreowski, and Stefan Taubenberger. Collages and patterns generated by hyperedge replacement. *Languages of Design*, 1:125–145, 1993.
24. Annegret Habel, Hans-Jörg Kreowski, and Walter Vogler. Metatheorems for decision problems on hyperedge replacement graph languages. *Acta Informatica*, 26:657–677, 1989.
25. Annegret Habel, Hans-Jörg Kreowski, and Walter Vogler. Decidable boundedness problems for sets of graphs generated by hyperedge-replacement. *Theoretical Computer Science*, 89:33–62, 1991.
26. John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison–Wesley, Reading, Massachusetts, 1979.
27. John E. Hutchinson. Fractals and self similarity. *Indiana University Mathematics Journal*, 30:713–747, 1981.
28. Lila Kari, Grzegorz Rozenberg, and Arto Salomaa. L systems. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages. Vol. I: Word, Language, Grammar*, chapter 5, pages 253–328. Springer, 1997.
29. Changwook Kim. Picture iteration and picture ambiguity. *Journal of Computer and System Sciences*, 40:289–306, 1990.
30. Thomas Lengauer and Egon Wanke. Efficient analysis of graph properties on context-free graph languages. In T. Leipistö and A. Salomaa, editors, *Proc. ICALP 88*, volume 317 of *Lecture Notes in Computer Science*, pages 379–393. Springer, 1988.
31. Benoit B. Mandelbrot. *The Fractal Geometry of Nature*. W.H. Freeman and Company, New York, 1983.
32. Hermann A. Maurer, Grzegorz Rozenberg, and Emo Welzl. Using string languages to describe picture languages. *Information and Control*, 54:155–185, 1982.
33. Rohit J. Parikh. On context-free languages. *Journal of the Association for Computing Machinery*, 13:570–581, 1966.
34. Heinz-Otto Peitgen, Hartmut Jürgens, and Dietmar Saupe. *Chaos and Fractals. New Frontiers of Science*. Springer, New York, 1992.

35. Heinz-Otto Peitgen, Hartmut Jürgens, and Dietmar Saupe. *Fractals for the Classroom. Part One: Introduction to Fractals and Chaos.* Springer, New York, 1992.
36. Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The Algorithmic Beauty of Plants.* Springer, 1990.
37. Grzegorz Rozenberg. Extensions of tabled 0L systems and languages. *International Journal of Computer and Information Sciences*, 2:311–334, 1973.
38. George Stiny. *Pictorial and Formal Aspects of Shapes and Shape Grammars.* Birkhäuser Verlag, Basel/Stuttgart, 1975.
39. Ivan Hal Sudborough and Emo Welzl. Complexity and decidability for chain-code picture languages. *Theoretical Computer Science*, 36:173–202, 1985.
40. Stefan Taubenberger. Correct translations of generalized iterated function systems to collage grammars. Report 7/94, Univ. Bremen, 1994.
41. Tommaso Toffoli and Norman Margolus. *Cellular Automata Machines.* MIT Press, Cambridge, Massachusetts, 1987.
42. Stephen Wolfram, editor. *Theory and Applications of Cellular Automata.* World Scientific, 1986.