Report 2/99, Univ. of Bremen

Tree-Based Generation of Languages of Fractals*

Frank Drewes
Department of Computer Science, University of Bremen,
P.O. Box 33 04 40, D–28334 Bremen, Germany
e-mail: drewes@informatik.uni-bremen.de

Abstract. The notion of $\mathcal{P}$-interpreted top-down tree generators is introduced, combining the nondeterministic nature of grammars as known from formal language theory with the infinite refinement of pictures studied in fractal geometry.

1 Introduction

This paper introduces an approach to generate languages of fractals, thus combining fractal geometry with one of the main ideas of formal language theory, namely to consider infinite sets of objects related by a common grammatical description.

The theory of formal languages is normally concerned with the grammatical generation of sets of discrete objects, these sets being called *languages*. Traditional questions ask for the properties of these languages and their recognition by various sorts of automata. While the objects of interest had initially been words, the ideas were soon taken up by researchers interested in more complex data structures like arrays, trees, and graphs (see the recently published books [RS97] for a representative overview of the field). Fractal geometry, on the other hand, usually focuses on single objects with an infinitely fine-grained structure, called *fractals* (see, e.g., [Man83, Edg90, PJS92, Bar93]).

The connections between fractal geometry and formal languages are manyfold. Just to mention a few examples, the approximation sequences of several fractals (especially curves like the Hilbert, dragon, or Koch curve) can be obtained by graphically interpreted Lindenmayer systems [PL90], the study of cellular automata is an important part of fractal geometry (see, e.g., [PJS92, Chapter 8]), and a suitable class of *collage grammars* [DK99] yields the approximation sequences of iterated function systems [KHKT99]. However, until now no one seems to have made a serious attempt to study devices which allow to generate languages of fractals, thus combining the views of fractal geometry and formal languages.

To some extend, the motivation to study fractals arose (and still arises) by the observation that the means of traditional Euclidean geometry are insufficient to describe the complex

*Partially supported by the EC TMR Network GETGRATS (General Theory of Graph Transformation Systems) and the ESPRIT Working Group APPLIGRAPH through the University of Bremen.

geometry of natural objects like plants, clouds, crystals, mountains, etc. This is because, as argued by Mandelbrot [Man83], these objects show details on any arbitrary scale (or at least on so many scales that they are "infinite in practice"), which prevents a faithful description by means of polygons, circles, etc. Fractal geometry can be used to model such objects in a more appropriate way. For example, in [Bar93] Barnsley uses an iterated function system in order to model a maple leaf. However, every iterated function system yields one particular fractal, which in this case means that it models a particular leaf. Depending on the application one may sometimes rather wish to model a whole variety of objects of a similar nature, like the *set* of all maple leaves—a language of fractals. From the point of view of formal language theory it is an obvious idea to achieve this by adding certain points of nondeterministic choice to the generation process, thus obtaining an infinity of generated fractals while retaining the finiteness of the generating device. As a matter of experience, the process of developing such a device for a particular set of objects will usually lead to a deeper understanding of the structure of these objects as it requires to find and express their common characteristics as well as the variation parameters.

Given a (device generating a) language of fractals, there immediately arise interesting theoretical questions which cannot sensibly be asked for single fractals. First, there are all the traditional questions of formal language theory: Is it possible to decide whether a given fractal (described by some iterated function system, say) is a member of the language? Do all the fractals in the language have a certain property? Is one type of generating device more powerful than the other? Which structural properties do the generated languages have, and which languages cannot be generated?

The approach presented in this paper is based on the notion of top-down tree generators (td generators, for short), continuing the work presented in [Dre00]. A td generator generates a language of trees—a *tree* being a term over some single-sorted signature Σ. By associating with such a td generator a Σ-algebra over pictures one obtains a language of pictures, given by the set of all values of trees in the generated tree language. Since every picture in the resulting language has a finite description, namely the tree whose value it is, this does not yield the desired language of fractals yet. In order to accomplish this, we turn from finite trees to infinite ones: The value of an infinite tree will be called a fractal (or, more precisely, a *syntactic fractal* since this notion of fractals refers to the syntactic description of a picture).

A td generator is composed of a regular tree grammar g and a finite sequence $td_1, \ldots, td_n$ of top-down tree transducers (td transducers), as introduced by Rounds and Thatcher [Rou70, Tha70] (see also [GS97] and the recent book [FV98]). It generates the set of trees obtained by applying $td_1, \ldots, td_n$ in succession to the language generated by g. So, the class of tree languages obtained is the closure of the class of regular tree languages under (the considered type of) td transductions. For the case of finite trees, this class was studied in many papers (see, for example, [Bak78, ERS80, Eng82]).

The advantage of using td generators for picture generation is that many properties and proofs can be formulated on the level of trees and tree languages without having to deal with the conceptually more complicated pictures all the time. In this respect it turns out to be very convenient that the evaluation of trees yields a continuous mapping from (possibly infinite) trees to pictures. The fact that an arbitrary number of td transducers can be used yields rather powerful devices, while proofs can usually be done by induction on the number of td transducers, so that their complexity remains limited.

The main results of this paper say that

(1) the language of all pictures generated by a so-called approximating td generator, which is a language consisting of fractals and finitely generated approximations, is the closure of the subset containing the finitely generated approximations,

(2) the fractals can be approximated by so-called refinement sequences, which yields a simple method to construct sequences converging to the fractals in the language,

(3) the languages containing only one fractal p are those for which p is *rational*, and p is rational if and only if it can be generated by a mutually recursive function system [CD93a, CD93b] (or, equivalently, a hierarchical IFS [PJS92]) with condensation sets, and

(4) in general, for every fractal in such a language, there is a sequence of rational fractals converging to the given one. In particular, if the generated language of fractals is finite, then it consists entirely of rational fractals.

In the next section some basic notions are recalled. Section 3 is devoted to regular tree grammars and top-down tree transducers, including their generalisation to the case of infinite trees. Top-down tree generators are introduced in Section 4, and Section 5 shows how to use them in order to generate picture languages. Some examples are presented in Section 6. In Section 7 the results concerning rational fractals are presented, and Section 8 contains some concluding remarks.

2 Preliminaries

In this section the basic notions and notations used throughout the paper are compiled. Mainly, these stem from the fields of term rewriting and metric spaces. Readers who are familiar with only one of these fields (or none of them) will hopefully find it convenient that I have tried to explain the standard notions as well. Others may perhaps prefer to have only a short glance at the main notations rather than reading everything in detail.

2.1 Basic mathematical notation

The sets of all natural numbers (including 0) and of all real numbers are denoted by $\mathbb{N}$ and $\mathbb{R}$, respectively. $\mathbb{N}_\infty$ denotes $\mathbb{N} \cup \{\infty\}$. For every $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, \ldots, n\}$. The cardinality of a set S is denoted by $|S|$. If $f : S \to T$ is a function then the canonical extension of f to the powerset of S is denoted by f, too: $f(S') = \{f(s) \mid s \in S'\}$ for all $S' \subseteq S$. The set of all finite sequences (also called strings or words) over a set S, including the empty string λ, is denoted by S^*, and $S^+ = S^* \setminus \{\lambda\}$. The length of a word w is denoted by $|w|$. An infinite sequence $(b_i)_{i\in\mathbb{N}}$ is a *subsequence* of $(a_i)_{i\in\mathbb{N}}$ if it equals $(a_{j_i})_{i\in\mathbb{N}}$ for a sequence $(j_i)_{i\in\mathbb{N}}$ of indices such that $j_i < j_{i+1}$ for all $i \in \mathbb{N}$.

If convenient, a binary relation $r \subseteq S \times T$ is considered as a function mapping S into the powerset of T, i.e., $r(s) = \{t \in T \mid (s,t) \in r\}$ for all $s \in S$. In this case, for every subset S' of S, $r(S')$ denotes the union of all $r(s)$ for which $s \in S'$. The composition of r with another binary relation $r' \subseteq T \times U$ is given by $r' \circ r$, where $(r' \circ r)(s) = r'(r(s))$ for all $s \in S$.

2.2 Trees

A (ranked) *symbol* is a pair (f,n) consisting of a symbol f and a number $n \in \mathbb{N}$, its *rank*. Instead of (f,n) we usually write $f^{(n)}$ or just f. The reader should keep in mind, however, that $f^{(m)}$ and $f^{(n)}$ are different for $m \neq n$, even though both may be denoted by f. A *signature* is a set Σ of symbols. By $\Sigma^{(n)}$ ($n \in \mathbb{N}$) we denote the set of all symbols in Σ whose rank equals n.

A (rooted, ordered, and node labelled) *tree* is a mapping $t\colon V(t) \to \Sigma$, where Σ is a signature and $V(t)$, the set of *nodes* or *vertices*, is a prefix-closed nonempty subset of $(\mathbb{N} \setminus \{0\})^*$ such that, for all $v \in V(t)$, $t(v) \in \Sigma^{(n)}$ implies $\{i \in \mathbb{N} \mid vi \in V(t)\} = [n]$. Thus, every node labelled with a symbol of rank n has exactly n children. A tree t is *finite* if $V(t)$ is finite; otherwise, it is *infinite*. The *depth* of t is given by $\operatorname{depth}(t) = \sup\{|v| \mid v \in V(t)\}$ (which equals ∞ if t is infinite).

For any label f and all trees $t_1,\dots,t_n$, $f[t_1,\dots,t_n]$ denotes the tree t such that $V(t) = \{\lambda\} \cup \bigcup_{i\in[n]}\{iv \mid v \in V(t_i)\}$, $t(\lambda) = f$ and $t(iv) = t_i(v)$ for all $i \in [n]$ and $v \in V(t_i)$. If $n = 0$ then we shall usually write f instead of $f[]$, thus identifying single-node trees with the symbol its root is labelled with. For $v \in V(t)$ the subtree of t rooted at v is denoted by t/v, i.e., $V(t/v) = \{w \mid vw \in V(t)\}$ where $t/v(w) = t(vw)$ for all $w \in V(t/v)$.

Given a signature Σ and a set T of trees, $\mathcal{T}_\Sigma(T)$ denotes the set of *trees over* Σ *with subtrees in* T. A tree t is in $\mathcal{T}_\Sigma(T)$ if $t(v) \in \Sigma$ or $t/v \in T$ for all $v \in V(t)$. $\mathcal{T}_\Sigma(\emptyset)$ is abbreviated by $\mathcal{T}_\Sigma$. Thus, $\mathcal{T}_\Sigma$ is simply the set of all trees of the form $t\colon V(t) \to \Sigma$. We shall furthermore use the notation $\mathcal{F}_\Sigma(T)$ in order to denote the set of all finite trees in $\mathcal{T}_\Sigma(T)$. Finally, $\Sigma(T)$ denotes the set of all trees $f[t_1,\dots,t_n]$ such that $f \in \Sigma^{(n)}$ and $t_1,\dots,t_n \in T$ for all $i \in [n]$.

2.3 Substitution

For the remainder of the paper, fix a countably infinite signature $X = \{x_1^{(0)}, x_2^{(0)}, \dots\}$ of pairwise distinct *variables* x_i. Variables are reserved for a special purpose and are not allowed to occur in ordinary signatures. For every $n \in \mathbb{N}$, X_n denotes $\{x_1,\dots,x_n\}$.

Let $n \in \mathbb{N}$ and consider trees $t, t_1,\dots,t_n$ such that (a) $\{t(v) \in X \mid v \in V(t)\} = X_n$ and (b) $t(u) = t(v) \in X_n$ implies $u = v$ for all $u, v \in V(t)$ (in other words, t contains each variable in X_n exactly once, and it does not contain any variable not in X_n). Then, we denote by $t[\![t_1,\dots,t_n]\!]$ the tree obtained from t by replacing the unique node labelled with x_i by t_i, for all $i \in [n]$. Formally, let $u_1,\dots,u_n \in V(t)$ be the nodes such that $t(u_i) = x_i$ for every

$i \in [n]$. Then $t[\![t_1, \ldots, t_n]\!]$ yields the tree t' such that $V(t') = V(t) \cup \bigcup_{i \in [n]} \{u_i v \mid v \in V(t_i)\}$ and, for all $u \in V(t')$,

$$t'(u) = \begin{cases} t(u) & \text{if } u \in V(t) \setminus \{u_1, \ldots, u_n\} \\ t_i(v) & \text{if } u = u_i v, \text{ where } i \in [n] \text{ and } v \in V(t_i). \end{cases}$$

In the following, the use of the notation $t[\![t_1, \ldots, t_n]\!]$ is always meant to imply the assumptions (a) and (b), even if they are not mentioned explicitly.

2.4 Metric spaces

A *distance function* or *metric* on a set $\mathbb{A}$ is a function $d\colon \mathbb{A} \times \mathbb{A} \to \mathbb{R}$ such that, for all $a, b, c \in \mathbb{A}$,

(i) $d(a, b) = d(b, a) \geq 0$,
(ii) $d(a, b) = 0$ if and only if $a = b$, and
(iii) $d(a, c) \leq d(a, b) + d(b, c)$.

If d is a metric on $\mathbb{A}$ then the pair $(\mathbb{A}, d)$ is a *metric space* (which may be identified with $\mathbb{A}$ if d is clear from the context). The elements of $\mathbb{A}$ are called the *points* of the space. As usual, a sequence $(a_i)_{i \in \mathbb{N}}$ of points in $\mathbb{A}$ is a *Cauchy sequence* if for every $\varepsilon > 0$ there is some $n \in \mathbb{N}$ such that $d(a_i, a_j) \leq \varepsilon$ for all $i, j \geq n$. The sequence *converges* to $a \in \mathbb{A}$ if for all $\varepsilon > 0$ there is some $n \in \mathbb{N}$ such that $i \geq n$ implies $d(a_i, a) \leq \varepsilon$. In this case, a is the *limit* of the sequence, denoted by $\lim_{i \to \infty} a_i$, or just $\lim a_i$ if the index in question is clear from the context. The metric space is *complete* if every Cauchy sequence $(a_i)_{i \in \mathbb{N}}$ in $\mathbb{A}$ converges to some point in $\mathbb{A}$. Thus, for a complete metric space a sequence is Cauchy if and only if it converges (as it is easy to see that every converging sequence is Cauchy), which means that the two terms can be understood as synonyms in this case.

$A \subseteq \mathbb{A}$ is *closed* if it contains every point a in $\mathbb{A}$ which is the limit of some converging sequence in A. A is *bounded* if $\sup\{d(a, b) \mid a, b \in A\} \neq \infty$, and *compact* if it is both bounded and closed. The *closure* $cl(A)$ of $A \subseteq \mathbb{A}$ is the smallest closed subset of $\mathbb{A}$ which contains A (i.e., $cl(A)$ is the set of all points $\lim a_i$ such that $(a_i)_{i \in \mathbb{N}}$ is a converging sequence in A).

A *transformation of* $\mathbb{A}$ is a continuous mapping $f\colon \mathbb{A} \to \mathbb{A}$. A function $f\colon \mathbb{A}^n \to \mathbb{A}$ is a *contraction* if there exists some $c < 1$, called a *contraction factor*, such that $d(f(a_1, \ldots, a_n), f(b_1, \ldots, b_n)) \leq c \cdot \max\{d(a_i, b_i) \mid i \in [n]\}$ for all $a_1, \ldots, a_n, b_1, \ldots, b_n \in \mathbb{A}$.

The set of all trees can be turned into a complete metric space in a well-known way: For all trees s, t, let $\mathrm{eq}(s, t) \in \mathbb{N}_\infty$ be the supremum of all $n \in \mathbb{N}$ such that the restrictions of s and t to $\{v \in V(s) \mid |v| < n\}$ and $\{v \in V(t) \mid |v| < n\}$, respectively, are equal. Intuitively, $\mathrm{eq}(s, t)$ is the maximum depth up to which the two trees coincide; in particular, $\mathrm{eq}(s, t) = \infty$ if (and only if) $s = t$. Now, define $d(s, t) = 2^{-\mathrm{eq}(s,t)}$ (where, by convention, $2^{-\infty} = 0$). It is easy to show that d is a metric on the set of all trees. In fact, d is even a *bounded ultrametric*: we have $d(s, t) \leq 1$ and $d(r, t) \leq max(d(r, s), d(s, t))$ for all trees r, s, t.

Most of the time we are going to deal with sets of trees over *finite* signatures. This is why the following lemma turns out to be quite useful.

2.1 Lemma Every sequence $(s_i)_{i\in\mathbb{N}}$ of trees in $\mathcal{T}_\Sigma$, where Σ is a finite signature, contains a subsequence $(t_i)_{i\in\mathbb{N}}$ which is Cauchy.

Proof. The assertion is trivial if the given sequence contains only finitely many pairwise distinct trees. Therefore, it may be assumed without loss of generality that the trees s_i are pairwise distinct (because repeated occurrences of a particular tree may be cancelled).

For a tree $s \in \mathcal{T}_\Sigma$ and a natural number i, denote by $[s]_i$ the equivalence class of all trees $t \in \mathcal{T}_\Sigma$ such that $\mathrm{eq}(s,t) \geq i$. Since Σ is finite, for every $i \in \mathbb{N}$ there are only finitely many pairwise distinct equivalence classes $[t]_i$ with $t \in \mathcal{T}_\Sigma$. This observation can be used to construct a sequence $T_0 \supseteq T_1 \supseteq T_2 \supseteq \cdots$ of infinite sets of trees in $(s_i)_{i\in\mathbb{N}}$ such that $\mathrm{eq}(t,t') \geq i$ for all $t,t' \in T_i$, as follows. To start with, let $T_0 = \{s_i \mid i \in \mathbb{N}\}$. By the assumption in the beginning, T_0 is infinite. Now, for $i \in \mathbb{N}$, define $T_{i+1} = T_i \cap [t]_{i+1}$, where $t \in T_i$ is some arbitrary tree such that $T_i \cap [t]_{i+1}$ is infinite. Since T_i is infinite and there are only finitely many equivalence classes $[t]_{i+1}$ such a tree must exist.

Now, the sequence $(T_i)_{i\in\mathbb{N}}$ can be used to define the trees t_i sought: $(t_i)_{i\in\mathbb{N}} = (s_{j_i})_{i\in\mathbb{N}}$ where $j_0 = 0$ and, for every $i \geq 1$, j_i is the smallest index greater than j_{i-1} such that $s_{j_i} \in T_i$ (which must exist because T_i is infinite). Clearly, $(t_i)_{i\in\mathbb{N}}$ is Cauchy because $t_i, t_j \in T_i$ for all $i,j \in \mathbb{N}$ with $i \leq j$, which implies $\mathrm{eq}(t_i,t_j) \geq i$. ■

Another useful property is the following one.

2.2 Lemma Let $(t_i)_{i\in\mathbb{N}}$ be a converging sequence of trees. If $t = \lim t_i$ is a finite tree, then there is some $i_0 \in \mathbb{N}$ such that $t_i = t$ for all $i \geq i_0$.

Proof. Let $\varepsilon = 2^{-\delta}$ where $\delta = \max\{|v| \mid v \in V(t)\}$. For every tree $t' \neq t$ we must necessarily have $\mathrm{eq}(t,t') \leq \delta$, and thus $d(t,t') \geq \varepsilon$. On the other hand, since $(t_i)_{i\in\mathbb{N}}$ converges there exists some $i_0 \in \mathbb{N}$ such that $d(t,t_i) < \varepsilon$, and therefore $t_i = t$, for all $i \geq i_0$. ■

3 Generation and transformation of trees

In this section, the concepts of a regular tree grammar and a top-down tree transducer are recalled (and generalised to the case of infinite trees), and some of their basic properties are shown.

3.1 Definition (*regular tree grammar*) A *regular tree grammar* is a quadruple $g = (N,\Sigma,P,S)$ consisting of a finite signature N of *nonterminals* of rank 0, a finite *output signature* Σ disjoint with N, a finite set P of *productions* (or *rules*) $A \to t$ such that $A \in N$ and $t \in \mathcal{F}_\Sigma(N) \setminus N$, and an *initial nonterminal* $S \in N$.

Let $s,s' \in \mathcal{T}_\Sigma(N)$. There is a *derivation step* from s to s', denoted by $s \Rightarrow_g s'$, if $V(s) \subseteq V(s')$ and for all $v \in V(s)$ either $s'(v) = s(v) \in \Sigma$ or $s(v) = A \in N$ and $s'(v) = t$ for some production $A \to t$ in P.

The *regular tree language generated by* g is given by

$$\mathrm{L}(g) = \{\lim t_i \mid t_0 \Rightarrow_g t_1 \Rightarrow_g \cdots \text{ where } t_0 = S\}$$

Furthermore, $\mathrm{L}_{\mathrm{fin}}(g) = \{t \in \mathrm{L}(g) \mid t \text{ finite}\}$ and $\mathrm{L}_{\mathrm{inf}}(g) = \{t \in \mathrm{L}(g) \mid t \text{ infinite}\}$ denote the sets of finite, respectively infinite trees generated by g. $\diamond$

As usual, the two parts A and t of a production $A \to t$ are said to be its *left-* and *right-hand side*, respectively. Sequences of derivation steps are called *derivations* (including infinite ones). A finite derivation $t_0 \Rightarrow_g t_1 \Rightarrow_g \cdots \Rightarrow_g t_n$ may, for the sake of brevity, be denoted by $t_0 \Rightarrow_g^* t_n$ or, if the number of steps matters, $t_0 \Rightarrow_g^n t_n$. Similarly, if $t_0 \Rightarrow_g t_1 \Rightarrow_g \cdots$ is an infinite derivation one may write $t_0 \Rightarrow_g^\infty t$ where $t = \lim t_i$.

Concerning the definition of regular tree grammars, there are some subtle points which should be noticed. First, for every tree $t \in \mathcal{T}_\Sigma$ it holds that $t \Rightarrow_g t$. Second, the right-hand side t of a production $A \to t$ is not allowed to be a single nonterminal (which is a well-known normal form in the case of finite trees, but is important if infinite trees are considered as well). Finally, derivations are maximum parallel in the sense that every derivation step is required to replace all nonterminals of a tree. The latter properties ensure that every infinite derivation $t_0 \Rightarrow_g t_1 \Rightarrow_g \cdots$ as in the definition of $\mathrm{L}(g)$ yields a sequence $(t_i)_{i \in \mathbb{N}}$ of trees such that $v \in V(t_i)$ and $t_i(v) \in N$ implies $|v| \geq i$ (which in particular yields $\mathrm{eq}(t_i, t_{i+1}) \geq i$) for all $i \in \mathbb{N}$. As a consequence, $\mathrm{L}(g)$ is a well-defined subset of $\mathcal{T}_\Sigma$. Moreover, by the first property, $\mathrm{L}(g)$ consists of *all* the terminal trees that can be derived, not only the infinite ones. Clearly, $\mathrm{L}_{\mathrm{fin}}(g)$ is the set of all trees $t \in \mathcal{T}_\Sigma$ such that there is a derivation $S \Rightarrow_g^* t$. Thus, for finite trees the definition does not deviate from those found in the literature.

A regular tree grammar $g = (N, \Sigma, P, S)$ is said to be *one-producing* if $P \subseteq N \times \Sigma(N)$ and *approximating* if there is at least one production $A \to t$ with $t \in \mathcal{F}_\Sigma$ for every $A \in N$.

3.2 Example (regular tree grammar) Consider the (approximating and one-producing) regular tree grammar $g = (\{A, B\}, \Sigma, P, A)$, where $\Sigma = \{a^{(2)}, b^{(2)}, \perp^{(0)}\}$, and

$$\begin{aligned} P = \{\ & A \to a[A, B], A \to a[B, A], A \to \perp, \\ & B \to b[B, B], B \to \perp\ \}. \end{aligned}$$

$\mathrm{L}(g)$ consists of all trees in $\mathcal{T}_\Sigma$ containing a unique a-labelled path. More precisely, a tree is in $\mathrm{L}(g)$ if and only if it equals $\perp$ or reads $a[t_1, t_2]$, where one of t_1, t_2 is in $\mathrm{L}(g)$ and the other one is a tree over $\{b^{(2)}, \perp^{(0)}\}$. A sample derivation is shown in Figure 1. If we delete the production $A \to \perp$, the grammar is not approximating any more, and the generated language consists entirely of infinite trees, namely all those where an infinite path is labelled by a's and the remaining binary nodes are labelled by b's. $\diamond$

Now, let us consider top-down tree transducers.

3.3 Definition (*top-down tree transducer*) A *top-down tree transducer* (td transducer) is a quintuple $td = (\Sigma, \Sigma', Q, R, q_0)$, where Σ and Σ' are finite signatures, the *input* and the *output signature*, Q is a finite signature of *states* of rank 1 with $Q \cap (\Sigma \cup \Sigma') = \emptyset$, R is a finite set of *rules*, and $q_0 \in Q$ is the *initial state*. Every rule in R has the form

$$q[f[x_1, \ldots, x_n]] \to t[\![q_1[x_{i_1}], \ldots, q_l[x_{i_l}]]\!],$$

where $q, q_1, \ldots, q_l \in Q$, $f \in \Sigma^{(n)}$ for some $n \in \mathbb{N}$, $t \in \mathcal{F}_{\Sigma'}(X_n) \setminus X_n$, and $x_{i_1}, \ldots, x_{i_l} \in X_n$.

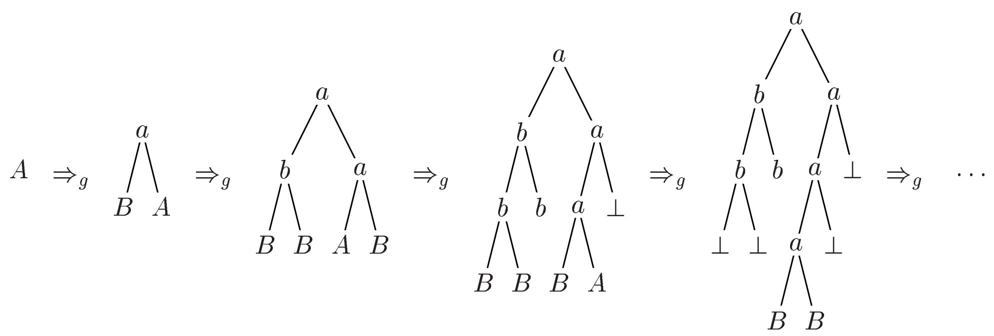

Figure 1: A derivation by a regular tree grammar

Consider two trees $s, s' \in \mathcal{T}_{\Sigma'}(Q(\mathcal{T}_\Sigma))$ and let $V = \{v \in V(s) \mid s(v') \notin Q$ for all proper prefixes v' of $v\}$.[1] There is a derivation step $s \Rightarrow_{td} s'$ if $V \subseteq V(s')$ and for all $v \in V$ either $s'(v) = s(v) \in \Sigma$ or $s'(v) = q[f[s_1, \ldots, s_n]] \in Q(\mathcal{T}_\Sigma)$, and there is a rule as above such that $s'/v = t[\![q_1[s_{i_1}], \ldots, q_l[s_{i_l}]]\!]$.

The set of output trees of td on input $t \in \mathcal{T}_\Sigma$, denoted by $td(t)$, is the set of all trees $\lim t_i$ such that $(t_i)_{i\in\mathbb{N}}$ is a sequence with $t_0 = q_0[t]$ and $t_i \Rightarrow_{td} t_{i+1}$ for all $i \in \mathbb{N}$. $\diamond$

Thus, a top-down tree transducer td as in the definition determines a binary relation between $\mathcal{T}_\Sigma$ and $\mathcal{T}_{\Sigma'}$, which is denoted by td as well, in the following. Henceforth, such a relation will be called a *top-down tree transduction*.

As in the case of regular tree grammars, only maximum parallel derivations are considered. Furthermore, deviating from the usual definition of td transducers, the one employed here forbids the use of rules whose right-hand sides are elements of $Q(X)$. This is important because a rule like $q[a[x_1]] \to q[x_1]$ would turn a tree t consisting of an infinite chain of a's into itself: $q[t] \Rightarrow_{td} q[t]$. The mentioned restriction prevents a behaviour like this. More precisely, for all trees $s, s' \in \mathcal{T}_{\Sigma'}(Q(\mathcal{T}_\Sigma))$ such that $s \Rightarrow_{td} s'$, it holds that $\min\{|v| \mid v \in V(s), s(v) \in Q\} < \min\{|v| \mid v \in V(s'), s'(v) \in Q\}$ (if s' contains a state at all). Therefore, $td(t)$ is a subset of $\mathcal{T}_{\Sigma'}$ for every tree $t \in \mathcal{T}_\Sigma$. Of course, for finite trees t we have $td(t) = \{t' \in \mathcal{F}_\Sigma \mid q_0[t] \Rightarrow_{td} \cdots \Rightarrow_{td} t'\}$, which is the definition of $td(t)$ one usually finds in the literature.

As before, derivations (i.e., sequences of derivation steps) can be denoted as $s \Rightarrow_{td}^{*} t$ or $s \Rightarrow_{td}^{n} t$ if they consist of a finite number n of steps, and as $s \Rightarrow_{td}^{\infty} t$ if $s = s_0 \Rightarrow_{td} s_1 \Rightarrow_{td} \cdots$ and $t = \lim s_i$. Notice that $t \Rightarrow_{td} t$ is a valid derivation step for $t \in \mathcal{F}_\Sigma$. This turns out to be convenient because it means that, for $s \in \mathcal{T}_\Sigma$, $td(s)$ is the set of all trees such that there exists a derivation $q_0[s] \Rightarrow_{td}^{\infty} t$. Thus, it is often not necessary to mention the finite case separately. A similar statement holds for every regular tree grammars $g = (N, \Sigma, P, S)$, of course: $\mathrm{L}(g)$ is the set of all trees $t \in \mathcal{T}_\Sigma$ such that $S \Rightarrow_{g}^{\infty} t$.

In order to simplify the denotation of trees and rules in connection with td transducers, the following conventions will be employed throughout the rest of this paper.

[1] If $\Sigma \cap \Sigma' = \emptyset$ one can just say $V = \{v \in V(s) \mid s(v) \in \Sigma \cup Q\}$.

1. A tree of the form $q[t]$, where q is a state, is denoted by qt.
2. The left-hand side $qf[x_1, \dots, x_n]$ of a rule is denoted by qf.
3. If the td transducer in question has a monadic input signature, the variable x_1 (the only one occurring in the rules) is omitted in the right-hand sides as well. Thus, in this case the rule $qf \to g[h[q'x_1, q''x_1], q'x_1]$ reads $qf \to g[h[q', q''], q']$, for example. (This is slightly ambiguous when there is a symbol $q^{(0)} \in \Sigma'$ such that $q^{(1)} \in Q$. Therefore, such a situation should be avoided.)

There are a few special cases of td transducers which turn out to be of interest in the context of this paper. A td transducer as in the definition is

- *total* (respectively *deterministic*) if R contains at least (respectively at most) one rule whose left-hand side is $qf[x_1, \dots, x_n]$, for every state $q \in Q$ and every input symbol $f \in \Sigma^{(n)}$ ($n \in \mathbb{N}$),
- *nondeleting* (respectively *linear*) if for every rule $qf \to t$ in R and every $i \in [n]$ there is at least (respectively at most) one node $v \in V(t)$ such that $t(v) = x_i$,
- *one-producing* if all right-hand sides of rules in R are elements of $\Sigma'(Q(X))$, and
- a (linear and nondeleting) *tree homomorphism* if it is total, deterministic, linear and nondeleting, and $|Q| = \{q_0\}$.

Notice that, in contrast to the definition of tree homomorphisms in [Eng75], the definition above requires them to be linear and nondeleting.

To see that the term *tree homomorphism* is indeed justified, observe that, for every symbol $f \in \Sigma^{(n)}$, the unique rule having f in its left-hand side has the form $qf \to t_f[\![qx_1, \dots, qx_n]\!]$, where $t_f \in \mathcal{F}_{\Sigma'}(X_n)$. In other words, td is uniquely determined by a mapping $h\colon \Sigma \to \mathcal{F}_{\Sigma'}(X)$ associating with every $f \in \Sigma^{(n)}$ a tree $h(f) \in \mathcal{F}_{\Sigma'}(X_n)$, so that every variable in X_n occurs in $h(f)$ exactly once. Then, $td(t) = h(f)[\![td(t_1), \dots, td(t_n)]\!]$ for every tree $t = f[t_1, \dots, t_n] \in \mathcal{T}_\Sigma$. This means that the tree transduction td is the canonical extension of h to trees in $\mathcal{T}_\Sigma$. In the following, if it is not of particular importance to emphasise that the tree homomorphism is realised by a top-down tree transducer, we shall therefore simply view it as the canonical extension of h to $\mathcal{T}_\Sigma$ and denote it by h as well.

3.4 Example (td transducer) Consider the td transducer $td = (\Sigma, \Sigma', \{q_0, q_1, q_1'\}, R, q_0)$, where $\Sigma = \{a^{(2)}, b^{(2)}, \bot^{(0)}\}$ is as in Example 3.2, $\Sigma' = \Sigma \cup \{\circ^{(2)}\}$, and R consists of the rules

$$
\begin{array}{lcll}
q_0\bot & \to & \circ[\bot, \bot], & \\
q_0 c & \to & \circ[c[q_1x_1, q_1x_2], c[q_1'x_2, q_1'x_1]] & \text{for } c \in \{a, b\}, \\
q_0 c & \to & \circ[c[q_1'x_2, q_1'x_1], c[q_1x_1, q_1x_2]] & \text{for } c \in \{a, b\}, \\
q\bot & \to & \bot & \text{for } q \in \{q_1, q_1'\}, \\
q_1 c & \to & c[q_1x_1, q_1x_2] & \text{for } c \in \{a, b\}, \text{ and} \\
q_1' c & \to & c[q_1'x_2, q_1'x_1] & \text{for } c \in \{a, b\}.
\end{array}
$$

In effect, td transforms every tree $t \in \mathcal{T}_\Sigma$ into the trees $\circ[t, t']$ and $\circ[t', t]$, such that, intuitively, t' is obtained by "mirroring" t at the vertical axis (where q_1 is the state

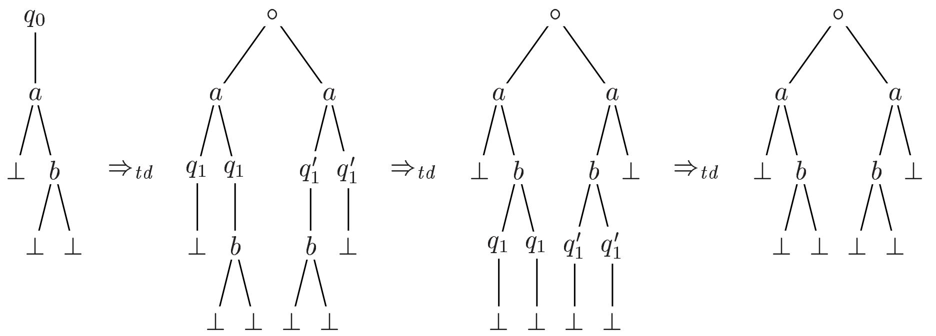

Figure 2: A derivation by a td transducer

producing t in the output by just copying it, and q_1' produces the mirror image of t by reversing the order of the two subtrees of every node which is not a leaf). Figure 2 shows a sample derivation on a small input tree. $\diamond$

Finally, let us discuss derivation trees. It should not come as a big surprise that the derivations of regular tree grammars and top-down tree transducers can be represented by derivation trees in a convenient way. In fact, these derivation trees can even be *generated* by regular tree grammars or top-down tree transducers, respectively. For this, let us assume from now on that every rule $s \to t$ (of a regular tree grammar or top-down tree transducer) is given a unique name r, denoted by $r\colon s \to t$, such that the names of distinct rules are distinct as well. One could, for example, use every rule as its own name. Therefore, in the following we usually do not distinguish between rules and their names.

For a regular tree grammar $g = (N, \Sigma, P, S)$ let $d\text{-}g = (N, \Sigma_P, P', S)$ be defined as follows. For every production $p\colon A \to t[\![A_1, \ldots, A_n]\!]$ in P (where $t \in \mathcal{F}_\Sigma(X_n)$), Σ_P contains the symbol $p^{(n)}$ and P' contains the production $A \to p[A_1, \ldots, A_n]$. Neither Σ_P nor P' contain any further elements. For $A \in N$, the trees $dt \in \mathcal{T}_{\Sigma_P}$ with $A \Rightarrow^\infty_{d\text{-}g} dt$ are the *A-derivation trees* of g. An S-derivation tree of g is called a *derivation tree* of g. The *result* of $dt \in \mathcal{T}_{\Sigma_P}(N)$ is given by $\mathrm{res}(dt) = dt$ if $dt \in N$ and $\mathrm{res}(dt) = t[\![\mathrm{res}(dt_1), \ldots, \mathrm{res}(dt_n)]\!]$ if $dt = p[dt_1, \ldots, dt_n]$ for some production $p\colon A \to t[A_1, \ldots, A_n]$ in P.

The respective definitions for top-down tree transducers are similar. Given a top-down tree transducer $td = (\Sigma, \Sigma', Q, R, q_0)$, $d\text{-}td = (\Sigma, \Sigma_R, Q, R', q_0)$ is defined as follows. For every rule $r\colon qf \to t[\![q_1 x_{i_1}, \ldots, q_m x_{i_m}]\!]$ in R (where $t \in \mathcal{F}_{\Sigma'}(X_m)$), Σ_R contains the symbol $r^{(m)}$ and R' contains the rule $qf \to r[q_1 x_{i_1}, \ldots, q_m x_{i_m}]$. Again, no further rules are included in Σ_R or R'. For $q \in Q$ and $t \in \mathcal{T}_\Sigma$, the trees $dt \in \mathcal{T}_{\Sigma_R}$ with $qt \Rightarrow^\infty_{d\text{-}td} dt$ are the *qt-derivation trees* of td. A q_0t-derivation tree of td is called a *t-derivation tree* of td. The *result* of $dt \in \mathcal{T}_{\Sigma_R}(Q(\mathcal{T}_\Sigma))$ is given by $\mathrm{res}(dt) = dt$ if $dt \in Q(\mathcal{T}_\Sigma)$ and $\mathrm{res}(dt) = t[\![\mathrm{res}(dt_1), \ldots, \mathrm{res}(dt_m)]\!]$ if $dt = r[dt_1, \ldots, dt_m]$ for some rule $r\colon qf \to t[\![q_1 x_{i_1}, \ldots, q_m x_{i_m}]\!]$ in R.

The following lemma proves that derivation trees are a faithful representation of derivations. Furthermore, res is compatible with taking limits. The proof is omitted because it is very straight forward.

3.5 Lemma

1. Let $g = (N, \Sigma, P, S)$ be a regular tree grammar and $s \in \mathcal{T}_{\Sigma_P}(N)$. If $s \Rightarrow_{d\text{-}g} t$ for some tree t then $\mathrm{res}(s) \Rightarrow_g \mathrm{res}(t)$. Conversely, if $\mathrm{res}(s) \Rightarrow_g t'$ for some tree t' then $t' = \mathrm{res}(t)$ for a tree t such that $s \Rightarrow_{d\text{-}g} t$.

   In particular, the equations $\mathrm{L_{fin}}(g) = \mathrm{res}(\mathrm{L_{fin}}(d\text{-}g))$, $\mathrm{L_{inf}}(g) = \mathrm{res}(\mathrm{L_{inf}}(d\text{-}g))$, and $\mathrm{L}(g) = \mathrm{res}(\mathrm{L}(d\text{-}g))$ are valid.
2. Let $td = (\Sigma, \Sigma', Q, R, q_0)$ be a top-down tree transducer and $s \in \mathcal{T}_{\Sigma_R}(Q(\mathcal{T}_\Sigma))$. If $s \Rightarrow_{d\text{-}td} t$ for some tree t then $\mathrm{res}(s) \Rightarrow_{td} \mathrm{res}(t)$. Conversely, if $\mathrm{res}(s) \Rightarrow_{td} t'$ for some tree t' then $t' = \mathrm{res}(t)$ for a tree t such that $s \Rightarrow_{d\text{-}td} t$.

   In particular, $td(t) = \mathrm{res}(d\text{-}td(t))$ for all trees $t \in \mathcal{T}_\Sigma$.
3. In both cases, if $(dt_i)_{i \in \mathbb{N}}$ is a sequence of derivation trees converging to dt, then dt is a derivation tree and $(\mathrm{res}(dt_i))_{i \in \mathbb{N}}$ converges to $\mathrm{res}(dt)$. $\diamond$

Consider a top-down tree transducer td and suppose we are given two trees s and t which which are equal up to depth $i+1$, a derivation of length i by $d\text{-}td$ on input s, yielding a tree s', and a t-derivation tree dt which coincides with s' up to depth i. Intuitively, the latter means that the derivation represented by dt and the given one (yielding s') coincide for the first i steps. However, since their respective input trees s and t coincide up to depth $i+1$, the input symbols for the next step coincide as well. Therefore, the considered derivation can be extended by another step in the same way as the one represented by dt. In other words, we obtain a tree s'' such that $\mathrm{eq}(s'', dt) \geq i+1$. This intuition is formalised by the following lemma, which will afterwards be used in order to construct infinite derivations from infinitely many finite ones.

3.6 Lemma Let $td = (\Sigma, \Sigma', Q, R, q_0)$ be a top-down tree transducer, $s, t \in \mathcal{T}_\Sigma$ such that $\mathrm{eq}(s,t) \geq i+1$ (where $i \in \mathbb{N}$), s' a tree such that $qs \Rightarrow^i_{d\text{-}td} s'$ for some $q \in Q$, and dt a qt-derivation tree such that $\mathrm{eq}(s', dt) \geq i$. Then there is a tree s'' with $\mathrm{eq}(s'', dt) \geq i+1$ and $s' \Rightarrow_{d\text{-}td} s''$.

Proof. Proceed by induction on i. Due to the assumption $\mathrm{eq}(s,t) \geq i+1$, s and t must have the form $s = f[s_1, \dots, s_n]$ and $t = f[t_1, \dots, t_n]$ for some $n \in \mathbb{N}$ and $f \in \Sigma^{(n)}$, where $\mathrm{eq}(s_j, t_j) \geq i$ for all $j \in [n]$. Let $dt = r[dt_1, \dots, dt_m]$ for some rule $r\colon qf \to u[\![q_1 x_{i_1}, \dots, q_m x_{i_m}]\!]$. For $i = 0$ one can immediately define $s'' = r[q_1 s_{i_1}, \dots, q_m s_{i_m}]$ in order to obtain $\mathrm{eq}(s'', dt) \geq 1$ and $s' = qs \Rightarrow_{d\text{-}td} s''$. For $i > 0$ the assumption $\mathrm{eq}(s', dt) \geq i$ implies that the derivation $qs \Rightarrow^i_{d\text{-}td} s'$ has the form

$$qs \Rightarrow_{d\text{-}td} r[q_1 s_{i_1}, \dots, q_m s_{i_m}] \Rightarrow^{i-1} r[s'_1, \dots, s'_m] = s',$$

where $\mathrm{eq}(s'_j, dt_j) \geq i-1$ for all $i \in [m]$. Therefore, the induction hypothesis yields trees s''_j ($j \in [m]$) with $\mathrm{eq}(s''_j, dt_j) \geq i$ such that $s'_j \Rightarrow^*_{d\text{-}td} s''_j$, which means that we have $\mathrm{eq}(s'', dt) \geq i+1$ and $s' \Rightarrow_{d\text{-}td} s''$ for $s'' = r[s''_1, \dots, s''_m]$. ∎

3.7 Lemma Let $td = (\Sigma, \Sigma', Q, R, q_0)$ be a top-down tree transducer and $t_i \in td(s_i)$ for all $i \in \mathbb{N}$, where $(s_i)_{i \in \mathbb{N}}$ is any sequence of trees in $\mathcal{T}_\Sigma$. If $(t_i)_{i \in \mathbb{N}}$ converges to a tree t, then there is a subsequence $(s'_i)_{i \in \mathbb{N}}$ of $(s_i)_{i \in \mathbb{N}}$ converging to a tree s such that $t \in td(s)$.

Proof. For every $i \in \mathbb{N}$, let dt_i be an s_i-derivation tree such that $\mathrm{res}(dt_i) = t_i$. By Lemma 2.1 there is a subsequence $(s'_i)_{i\in\mathbb{N}}$ of $(s_i)_{i\in\mathbb{N}}$ which converges to some tree s. Let $(dt'_i)_{i\in\mathbb{N}}$ be the corresponding subsequence of $(dt_i)_{i\in\mathbb{N}}$. Since $(\mathrm{res}(dt'_i))_{i\in\mathbb{N}}$ is a subsequence of $(t_i)_{i\in\mathbb{N}}$, it still converges to t. By a second application of Lemma 2.1, now to the derivation trees dt'_i themselves, it may additionally be assumed that $(dt'_i)_{i\in\mathbb{N}}$ converges (for if it does not, it can be replaced by a subsequence that does).

Let $dt = \lim dt'_i$ and $u_0 = q_0 s$. We are going to use Lemma 3.6 in order to construct a derivation $u_0 \Rightarrow_{d\text{-}td} u_1 \Rightarrow_{d\text{-}td} \cdots$ such that $\mathrm{eq}(u_i, dt) \geq i$ for all $i \in \mathbb{N}$, which means that $\lim u_i = dt$, showing that $t = \mathrm{res}(dt) \in td(s)$. Suppose $u_0 \Rightarrow^i_{d\text{-}td} u_i$ with $\mathrm{eq}(u_i, dt) \geq i$ (which clearly holds for the induction basis $i = 0$). Then there is some $j \in \mathbb{N}$ satisfying $\mathrm{eq}(s'_j, s) \geq i+1$ and $\mathrm{eq}(dt_j, dt) \geq i+1$. Since $\mathrm{eq}(u_i, dt) \geq i$ the latter implies $\mathrm{eq}(u_i, dt_j) \geq i$, which by Lemma 3.6 means that $u_i \Rightarrow_{d\text{-}td} u_{i+1}$ for some tree u_{i+1} with $\mathrm{eq}(u_{i+1}, dt_j) \geq i + 1$. Together with the fact that $\mathrm{eq}(dt_j, dt) \geq i + 1$ this yields $\mathrm{eq}(u_{i+1}, dt) \geq i + 1$, as required. ∎

From the previous lemma the main result of this section is obtained.

3.8 Theorem For every top-down tree transducer $td = (\Sigma, \Sigma', Q, R, q_0)$ and every set $T \subseteq \mathcal{T}_\Sigma$ the set $td(cl(T))$ is closed, and if td is total then $td(cl(T)) = cl(td(T))$.

Proof. The first assertion is a consequence of Lemma 3.7. Suppose $(t_i)_{i\in\mathbb{N}}$ converges to a tree t, where $t_i \in td(cl(T))$ for all $i \in \mathbb{N}$, and let $s_i \in cl(T)$ be such that $t_i \in td(s_i)$ for all $i \in \mathbb{N}$. By Lemma 3.7 there is a converging subsequence $(s'_i)_{i\in\mathbb{N}}$ of $(s_i)_{i\in\mathbb{N}}$ such that $t \in td(\lim s'_i)$. Since $s'_i \in cl(T)$ for all $i \in \mathbb{N}$, we have $\lim s_i \in cl(T)$, i.e., $t \in td(cl(T))$, showing that $td(cl(T))$ is closed.

Now, suppose td is total. The proof of the inclusion $cl(td(T)) \subseteq td(cl(T))$ is quite the same as the one above: If $(t_i)_{i\in\mathbb{N}}$ converges to a tree t, where $t_i \in td(T)$ for all $i \in \mathbb{N}$, then by Lemma 3.7 there is a converging sequence $(s'_i)_{i\in\mathbb{N}}$ in T such that $t \in td(\lim s'_i)$, which yields $t \in td(cl(T))$.

It remains to check the inclusion $td(cl(T)) \subseteq cl(td(T))$, assuming that td is total. Let $s = \lim s_i$ and $t \in td(s)$ for some Cauchy sequence $(s_i)_{i\in\mathbb{N}}$ in T, and let dt be an s-derivation tree of td with $\mathrm{res}(dt) = t$. For every $i \in \mathbb{N}$, by the totality of td, $\mathrm{eq}(s_i, s) \geq n$ means that there is an s_i-derivation tree dt_i of td such that $\mathrm{eq}(dt_i, dt) \geq n$. This observation yields a sequence $(dt_i)_{i\in\mathbb{N}}$ converging to dt, where each dt_i is an s_i-derivation tree ($i \in \mathbb{N}$). Thus, $(\mathrm{res}(dt_i))_{i\in\mathbb{N}}$ is a sequence in $td(T)$ converging to t. ∎

Observe that, due to this theorem, the range of a total td transducer $td = (\Sigma, \Sigma', Q, R, q_0)$ is nothing but the closure of $td(\mathcal{F}_\Sigma)$, except for the pathological case of a nonempty input signature containing only symbols of rank strictly greater than 0 (which is the only case in which $\mathcal{T}_\Sigma \neq cl(\mathcal{F}_\Sigma)$). If td is not total, then its range is still closed (even in the mentioned pathological case), but it may not be the closure of $td(\mathcal{F}_\Sigma)$. In particular, $td(\mathcal{F}_\Sigma)$ may be empty while $td(\mathcal{T}_\Sigma)$ is not.

4 Top-down tree generators

It is now possible to define the notion of top-down tree generators, the type of tree-generating device which will be considered throughout the rest of this paper. A top-down generator consists of a regular tree grammar and a finite sequence of top-down tree transducers. It generates the language of trees which is obtained by taking the language generated by the regular tree grammar, and then transforming it by applying the tree transducers one after another.

4.1 Definition (*top-down tree generator*) A *top-down tree generator* (td generator) is a pair $G = (g, td_1 \cdots td_n)$ consisting of a regular tree grammar and a finite sequence of td transducers $td_1, \ldots, td_n$ (for some $n \in \mathbb{N}$) such that

(i) g and $td_1, \ldots, td_n$ are one-producing and

(ii) for all $i \in [n]$, the input signature of td_i coincides with the output signature of td_{i-1} for $i > 1$ and of g for $i = 1$.

G generates the languages

$$\begin{aligned} &\mathrm{L}(G) = td_n(\cdots(td_1(\mathrm{L}(g)))\cdots), \\ &\mathrm{L}_{\mathrm{fin}}(G) = \{t \in \mathrm{L}(G) \mid t \text{ finite}\}, \text{ and} \\ &\mathrm{L}_{\mathrm{inf}}(G) = \{t \in \mathrm{L}(G) \mid t \text{ infinite}\}. \end{aligned} \qquad \blacksquare$$

In the following, the regular tree grammar g of a td generator $G = (g, td_1 \cdots td_n)$ is denoted by g_G if it is not explicitly named, and the composition of $td_1, \ldots, td_n$ is denoted by τ_G, i.e., $\tau_G = td_n \circ \cdots \circ td_1$ (if $n = 0$ then τ_G is the identity on $\mathcal{T}_\Sigma$, where Σ is the output signature of g). The *output signature* of G is the output signature of td_n. G is called *approximating* if g is approximating and all of $td_1, \ldots, td_n$ are total.

4.2 Example (td generator) Let $G = (g, td)$, where g is the regular tree grammar discussed in Example 3.2 and td is the td transducer of Example 3.4. Then, $\mathrm{L}(G)$ is the set of all trees $\circ[t, t']$ such that t contains one path of a's from the root to some leaf (labelled $\perp$), all other binary nodes of t being labelled by b's, and t' is t mirrored at the vertical axis. Notice that this is something which cannot be achieved by a regular tree grammar because regular tree grammars do not provide any means to "synchronise" the derivation of distinct occurrences of nonterminals. $\diamond$

By definition, the class of tree languages which can be generated by td generators is the closure of the regular tree languages under one-producing td transductions. For the case of finite trees, this class of tree languages (without the restriction to one-producing td transducers) has been studied before by several authors (see, e.g., [Bak78, ERS80, Eng82]). In [Eng82] a hierarchy result is shown which proves that the language generating power strictly increases with the number of td transducers used. As one can easily see, this result remains valid for one-producing td transducers, and also if approximating td generators are considered. More precisely, we have the following theorem.

4.3 Theorem ([Eng82]) For every $n \in \mathbb{N}$ let $\mathcal{T}_n$ denote the class of all tree languages T such that $T = \mathrm{L}(G)$ for some td generator $G = (g, td_1 \cdots td_n)$. Then $\mathcal{T}_n$ is properly contained in $\mathcal{T}_{n+1}$ for all $n \in \mathbb{N}$. $\diamond$

The following lemma shows that, intuitively, the regular tree grammar in a td generator could be replaced by an additional top-down tree transducer. For this, and for future use as well, let Σ_{succ} be the signature consisting of a symbol s of rank 1 and a symbol 0 of rank 0. Furthermore, let $s^0[0] = 0$ and $s^{n+1}[0] = s[s^n[0]]$ for $n \in \mathbb{N}$, and denote by s^∞ the unique infinite tree in $\mathcal{T}_{\Sigma_{\mathrm{succ}}}$.

4.4 Lemma For every regular tree grammar $g = (N, \Sigma, P, S)$ there is a top-down tree transducer $td = (\Sigma_{\mathrm{succ}}, \Sigma, N, R, S)$ such that $td(\mathcal{F}_{\Sigma_{\mathrm{succ}}}) = \mathrm{L}_{\mathrm{fin}}(g)$ and $td(s^\infty) = td(\mathcal{T}_{\Sigma_{\mathrm{succ}}}) = \mathrm{L}(g)$. If g is approximating then td is total and if g is one-producing, so is td.

Proof. The construction is straight forward. For every production $A \to t[\![A_1, \ldots, A_n]\!]$ in P, where $t \in \mathcal{F}_\Sigma(X_n)$, R contains the rule $As \to t[\![A_1, \ldots, A_n]\!]$. Furthermore, if $n = 0$ then R also contains the rule $A0 \to t$.

Since g is approximating, td is total. Furthermore, if g is one-producing, so is td. It follows by a straightforward induction on i that, for $A, A_1, \ldots, A_n \in N$ and $t \in \mathcal{F}_\Sigma(X_n)$, we have $A \Rightarrow_g^i t[\![A_1, \ldots, A_n]\!]$ if and only if $As^{i+j}[0] \Rightarrow_{td}^i t[\![A_1 s^j[0], \ldots, A_n s^j[0]]\!]$, for all $i, j \in \mathbb{N}$. This implies $\mathrm{L}_{\mathrm{fin}}(g) = td(\mathcal{F}_{\Sigma_{\mathrm{succ}}})$ and $\mathrm{L}_{\mathrm{inf}}(g) = td(s^\infty)$, which completes the proof. ■

The most interesting td generators are the approximating ones. This is because it turns out that $\mathrm{L}(G)$ is simply the closure of $\mathrm{L}_{\mathrm{fin}}(G)$, which is mainly a consequence of Theorem 3.8. Furthermore, despite the fact that τ_G may transform infinite trees into finite ones by deleting infinite subtrees of its input, $\mathrm{L}_{\mathrm{fin}}(G)$ equals $\tau_G(\mathrm{L}_{\mathrm{fin}}(g_G))$. For td generators which are not approximating both statements do not generally hold, but one can at least say that $\mathrm{L}(G)$ is always closed.

4.5 Theorem For every td generator G, $\mathrm{L}(G)$ is closed. Furthermore, if G is approximating then $\mathrm{L}_{\mathrm{fin}}(G) = \tau_G(\mathrm{L}_{\mathrm{fin}}(g_G))$ and $\mathrm{L}(G) = cl(\mathrm{L}_{\mathrm{fin}}(G))$.

Proof. Let $\mathrm{L}^0_{\mathrm{fin}} = \mathcal{F}_{\Sigma_{\mathrm{succ}}}$ and $\mathrm{L}^0 = \mathcal{T}_{\Sigma_{\mathrm{succ}}}$. Lemma 4.4 states that there are td transducers $td_1, \ldots, td_n$ such that $\mathrm{L}_{\mathrm{fin}}(G) = \tau(\mathrm{L}^0_{\mathrm{fin}})$ and $\mathrm{L}(G) = \tau(\mathrm{L}^0)$, where $\tau = td_n \circ \cdots \circ td_1$. Moreover, if G is approximating then it may be assumed that $td_1, \ldots, td_n$ are total.

Let $\mathrm{L}^i_{\mathrm{fin}} = td_i(\cdots td_1(\mathrm{L}^0_{\mathrm{fin}}) \cdots)$ and $\mathrm{L}^i = td_i(\cdots td_1(\mathrm{L}^0) \cdots)$ for all $i \in [n]$. The closedness of L^i follows by induction on i, using Theorem 3.8 and the fact that L^0 is closed. Thus, in order to complete the proof it suffices to show by induction on i that $\{t \in \mathrm{L}^i \mid t \text{ finite}\} \subseteq \mathrm{L}^i_{\mathrm{fin}}$ and $\mathrm{L}^i = cl(\mathrm{L}^i_{\mathrm{fin}})$ (which yields claimed equations by taking $i = n$).

For $i = 0$ both equations obviously hold. Now, consider some $i \geq 1$ and suppose the equations are correct with respect to $i - 1$. In order to verify the first equation, let s be an infinite tree in L^{i-1} such that there is a finite tree $t \in td_i(s)$. By the second part of the induction hypothesis, $s \in cl(\mathrm{L}^{i-1}_{\mathrm{fin}})$, which yields $t \in td_i(cl(\mathrm{L}^{i-1}_{\mathrm{fin}}))$ and thus, by Theorem 3.8, $t \in cl(\mathrm{L}^i_{\mathrm{fin}})$. However, since t is finite this implies $t \in \mathrm{L}^i_{\mathrm{fin}}$, using Lemma 2.2.

The second equation follows directly from Theorem 3.8 and the induction hypothesis: $\mathrm{L}^i = td_i(\mathrm{L}^{i-1}) = td_i(cl(\mathrm{L}_{\mathrm{fin}}^{i-1})) = cl(td_i(\mathrm{L}_{\mathrm{fin}}^{i-1})) = cl(\mathrm{L}_{\mathrm{fin}}^{i})$. ∎

For approximating td generators, Theorem 4.5 tells us that the infinite trees in $\mathrm{L}_{\mathrm{inf}}(G)$ can be approximated by considering converging sequences in $\mathrm{L}_{\mathrm{fin}}(G)$. However, is there a simple way to find a set of converging sequences which is sufficient in order to get all the elements of $\mathrm{L}_{\mathrm{inf}}(G)$? It is shown below that there is indeed a rather natural procedure to obtain approximations of all trees in $\mathrm{L}_{\mathrm{inf}}(G)$. For this, the following definition is needed.

4.6 Definition Let $G = (g, td_1 \cdots td_n)$ be an approximating td generator.

1. A tuple $D = (dt_0, \ldots, dt_n)$ is a *derivation in* G if dt_0 is a derivation tree of g and dt_i is a $\mathrm{res}(dt_{i-1})$-derivation tree of td_i for every $i \in [n]$. The *result* $\mathrm{res}(D)$ of D is $\mathrm{res}(dt_n)$ and its *depth* is $\mathrm{depth}(D) = \mathrm{depth}(dt_0)$.
2. Let $D = (dt_0, \ldots, dt_n)$ be a derivation in G, where $\mathrm{depth}(D) = m \in \mathbb{N}$. A derivation $D' = (dt'_0, \ldots, dt'_n)$ *extends* D if $\mathrm{depth}(D') = m + 1$ and $\mathrm{eq}(dt_i, dt'_i) = m$ for all $i \in \{0, \ldots, m\}$.

   If $(D_i)_{i\in\mathbb{N}}$ is a sequence of derivations such that $\mathrm{depth}(D_0) = 0$ and D_{i+1} extends D_i for all $i \in \mathbb{N}$, then $(\mathrm{res}(D_i))_{i\in\mathbb{N}}$ is a *refinement sequence* in G. ◇

Obviously, due to Lemma 3.5, $\mathrm{L}(G)$ is the set of all trees $\mathrm{res}(D)$ such that D is a derivation in G. We can now show that it is sufficient to concentrate on refinement sequences if one wants to approximate the elements of $\mathrm{L}_{\mathrm{inf}}(G)$.

4.7 Theorem For every approximating td generator G,

$$\mathrm{L}_{\mathrm{inf}}(G) = \{\lim t_i \mid (t_i)_{i\in\mathbb{N}} \text{ is a refinement sequence in } G\}.$$

Proof.

($\Leftarrow$) It follows directly from Definition 4.6 that, given a refinement sequence $(t_i)_{i\in\mathbb{N}}$ in G, we have $t_i \in \mathrm{L}_{\mathrm{fin}}(G)$ and $\mathrm{eq}(t_i, t_{i+1}) = i$ for all $i \in \mathbb{N}$. Consequently, $t = \lim t_i$ exists and is infinite (since $\mathrm{eq}(t_i, t_{i+1}) = i$ implies $\mathrm{depth}(t_i) \geq i$). By Theorem 4.5 this means $t \in \mathrm{L}_{\mathrm{inf}}(G)$.

($\Rightarrow$) Let $G = (g, td_1 \cdots td_n)$ and proceed by induction on n. In order to prove the statement for $n = 0$, let $t \in \mathrm{L}_{\mathrm{inf}}(g)$ and consider a derivation $dt_0 \Rightarrow_{d\text{-}g} dt_1 \Rightarrow_{d\text{-}g} dt_2 \cdots$ such that $\mathrm{res}(dt) = t$, where $dt = \lim dt_i$. By assumption, g is approximating. Therefore, for all $i \in \mathbb{N}$ there is some $dt'_i \in \mathcal{F}_{\Sigma_P}$ such that $dt_i \Rightarrow_{d\text{-}g} dt'_i$ (where P is the set of productions of g). Since t is infinite, so is dt, which means that $\mathrm{depth}(dt_i) = i$ and thus $\mathrm{depth}(dt'_i) = i$ for all $i \in \mathbb{N}$. This implies $\mathrm{eq}(dt'_i, dt'_{i+1}) \leq i$. On the other hand, by construction we have $\mathrm{eq}(dt'_i, dt'_{i+1}) \geq i$, yielding the required equation $\mathrm{eq}(dt'_i, dt'_{i+1}) = i$. This shows that dt'_{i+1} extends dt'_i for all $i \in \mathbb{N}$. Furthermore, $(\mathrm{res}(dt'_i))_{i\in\mathbb{N}}$ converges to t as $(dt'_i)_{i\in\mathbb{N}}$ converges to dt.

Now, suppose $n > 0$, let $td_n = (\Sigma, \Sigma', Q, R, q_0)$, and assume that the statement is valid for the td generator $G' = (g, td_1 \cdots td_{n-1})$. The proof is mainly based on the following observation, which is an obvious consequence of the fact that td_n is total (by the definition of td generators).

> Let $s, s' \in \mathcal{T}_\Sigma$ and let dt be an s-derivation tree of td_n. For all $m \in \mathbb{N}$ such that $\mathrm{eq}(s, s') \geq m$, there is an s'-derivation tree dt' of td_n such that $\mathrm{eq}(dt, dt') \geq m$.

Now, let $t \in \mathrm{L}_{\mathrm{inf}}(G)$ and consider an s-derivation tree dt of td_n such that $s \in \mathrm{L}_{\mathrm{inf}}(G')$ and $\mathrm{res}(dt) = t$. By the induction hypothesis there exists a refinement sequence $(s_i)_{i \in \mathbb{N}}$ in G' such that $s = \lim s_i$. Furthermore, $\mathrm{depth}(s_i) = i = \mathrm{eq}(s_i, s)$ for all $i \in \mathbb{N}$, by the definition of refinement sequences and the fact that $g, td_1, \ldots, td_{n-1}$ are one-producing. Consequently, by the observation above, for every $i \in \mathbb{N}$ there is an s_i-derivation tree dt_i of td_n such that $\mathrm{eq}(dt_i, dt) \geq i$. Since we have $\mathrm{depth}(dt_i) \leq \mathrm{depth}(s_i) = i$ the inequality $\mathrm{eq}(dt_i, dt) > i$ would imply $dt_i = dt$, which is impossible since s is infinite. Therefore, $\mathrm{eq}(dt_i, dt) = i$ and thus $\mathrm{eq}(dt_i, dt_{i+1}) = i$ for all $i \in \mathbb{N}$, which proves that $(\mathrm{res}(dt_i))_{i \in \mathbb{N}}$ is a refinement sequence in G (that converges to t, as required, since $(dt_i)_{i \in \mathbb{N}}$ converges to dt and $\mathrm{res}(dt) = t$). ■

5 Languages of pictures and fractals

In this section the results on td generators are applied in order to obtain devices that generate languages of pictures and fractals. The basic idea is to consider algebras over pictures and to use td generators in order to generate tree languages which are then evaluated in such an algebra. This way of generating sets is of course applicable to any sort of domain, not just to pictures. Therefore, the basic definition is not at all restricted to the case of pictures.

Let us first make precise what an algebra is. For this, let Σ be a signature such that $\Sigma^{(0)} \neq \emptyset$. A Σ-*algebra* $\mathcal{A}$ (which we shall just call an *algebra* if Σ is of minor importance) is a pair $(\mathbb{A}, (f_\mathcal{A})_{f \in \Sigma})$ such that $\mathbb{A}$ is a set, the *domain* of $\mathcal{A}$, and $f_\mathcal{A} \colon \mathbb{A}^n \to \mathbb{A}$ is a function for every $f \in \Sigma^{(n)}$ $(n \in \mathbb{N})$. The functions $f_\mathcal{A}$ are called the *operations* of $\mathcal{A}$. Trees in $\mathcal{F}_\Sigma$ are evaluated by the homomorphism $\mathrm{val}_\mathcal{A}$ in the usual way: $val_\mathcal{A}(f[t_1, \ldots, t_n]) = f_\mathcal{A}(\mathrm{val}_\mathcal{A}(t_1), \ldots, \mathrm{val}_\mathcal{A}(t_n))$ for all trees $f[t_1, \ldots, t_n] \in \mathcal{F}_\Sigma$. We shall consider $\mathrm{val}_\mathcal{A}$ as a partial function from $\mathcal{T}_\Sigma$ to $\mathbb{A}$ (which is total on $\mathcal{F}_\Sigma$). This turns out to be useful because sometimes infinite trees can be assigned a value in a meaningful way, too—a fact that will soon be exploited. In this respect it is important to be aware of the general requirement $\Sigma^{(0)} \neq \emptyset$. Obviously, this ensures that $\mathcal{T}_\Sigma$ is the closure of $\mathcal{F}_\Sigma$, so that every infinite tree can be approximated by a sequence of finite ones. In the following, whenever a signature Σ appears in connection with a Σ-algebra, $\Sigma^{(0)} \neq \emptyset$ is assumed without mentioning this fact explicitly.

As remarked above, the idea developed in this section is to consider td generators whose generated trees are interpreted in an algebra $\mathcal{A}$ in order to obtain a subset of $\mathbb{A}$. However, in order to increase the flexibility of such devices, a slight generalisation turns out to be useful: Rather than interpreting trees in $\mathcal{A}$ itself, one may choose an algebra derived from $\mathcal{A}$. For this, the following definitions are necessary.

Let $\mathcal{A}$ be a Σ-algebra. Then every tree $t \in \mathcal{F}_\mathcal{A}(X_n)$ (where $n \in \mathbb{N}$) defines a function

$\mathrm{op}_{\mathcal{A}}(t)\colon \mathbb{A}^n \to \mathbb{A}$ which, applied to $a_1, \ldots, a_n \in \mathbb{A}$, simply evaluates t while interpreting x_i as a_i. Formally, for $a_1, \ldots, a_n \in \mathbb{A}$ let $\mathcal{B}$ be the $\Sigma \cup X_n$-algebra with domain $\mathbb{A}$ such that $f_{\mathcal{B}} = f_{\mathcal{A}}$ for all $f \in \Sigma$ and $x_{i\mathcal{B}} = a_i$ for all $i \in [n]$. Now, $\mathrm{op}_{\mathcal{A}}(t)$ is the function $g\colon \mathbb{A}^n \to \mathbb{A}$ such that $g(a_1, \ldots, a_n) = \mathrm{val}_{\mathcal{B}}(t)$ for all $a_1, \ldots, a_n \in \mathbb{A}$.

A Σ'-algebra $\mathcal{B}$ is said to be *derived from* $\mathcal{A}$ if $\mathbb{B} = \mathbb{A}$ and there is a tree homomorphism h such that $f_{\mathcal{B}} = \mathrm{op}_{\mathcal{A}}(h(f))$ for all $f \in \Sigma'$. The operations of $\mathcal{B}$ are said to be *derived from the operations of* $\mathcal{A}$. Notice that, intuitively, according to the definition of tree homomorphisms a derived operation can neither delete nor copy its arguments since each of the variables in X_n must occur exactly once in the tree $h(f)$ defining $f_{\mathcal{B}}$ for $f \in \Sigma'^{(n)}$. As mentioned above, we shall also consider algebras in which certain infinite trees have a defined value. If $\mathcal{A}$ is such an algebra then the definedness of values carries over to $\mathcal{B}$ via h: For all infinite trees $t \in \mathcal{T}_{\Sigma'}$ such that $\mathrm{val}_{\mathcal{A}}(h(t))$ is defined, we set $\mathrm{val}_{\mathcal{B}}(t) = \mathrm{val}_{\mathcal{A}}(h(t))$. (Notice that this is compatible with the finite case, since we obviously have $\mathrm{val}_{\mathcal{B}}(t) = \mathrm{val}_{\mathcal{A}}(h(t))$ for all $t \in \mathcal{F}_{\Sigma'}$.)

Now, the central definition of this paper can be given.

5.1 Definition (*$\mathcal{A}$-interpreted td generator*) Let $\mathcal{A}$ be an algebra. An *$\mathcal{A}$-interpreted td generator* is a pair $\mathcal{G} = (G, \mathcal{B})$ consisting of a td generator G and a Σ-algebra $\mathcal{B}$ derived from $\mathcal{A}$, such that Σ is the output signature of G.

$\mathcal{G}$ defines the language $\mathrm{L}_{\mathrm{fin}}(\mathcal{G}) = \mathrm{val}_{\mathcal{B}}(\mathrm{L}_{\mathrm{fin}}(G))$. Moreover, if $\mathrm{val}_{\mathcal{B}}(t)$ is defined for all $t \in \mathrm{L}_{\mathrm{inf}}(G)$ then $\mathrm{L}_{\mathrm{inf}}(\mathcal{G}) = \mathrm{val}_{\mathcal{B}}(\mathrm{L}_{\mathrm{inf}}(G))$ and $\mathrm{L}(\mathcal{G}) = \mathrm{val}_{\mathcal{B}}(\mathrm{L}(G)) = \mathrm{L}_{\mathrm{fin}}(\mathcal{G}) \cup \mathrm{L}_{\mathrm{inf}}(\mathcal{G})$. ◇

The following lemma is an immediate consequence of these definitions.

5.2 Lemma Let $\mathcal{G} = (G, \mathcal{B})$ be an $\mathcal{A}$-interpreted td generator, for some Σ-algebra $\mathcal{A}$, where $\mathcal{B}$ is a Σ'-algebra given by a tree homomorphism $h\colon \mathcal{T}_{\Sigma'} \to \mathcal{T}_{\Sigma}$. Then $\mathrm{L}_{\mathrm{fin}}(\mathcal{G}) = \mathrm{val}_{\mathcal{A}}(h(\mathrm{L}_{\mathrm{fin}}(G)))$ and, if $\mathrm{L}_{\mathrm{inf}}(\mathcal{G})$ is defined, $\mathrm{L}_{\mathrm{inf}}(\mathcal{G}) = \mathrm{val}_{\mathcal{A}}(h(\mathrm{L}_{\mathrm{inf}}(G)))$. ◇

Regarding the previous lemma, it seems worth mentioning that a tree language is of the form $h(\mathrm{L}(G))$ for some td generator G and a tree homomorphism h if and only if it has the form $td_n(\cdots td_1(\mathrm{L}(g))\cdots)$ for some regular tree grammar g and td transducers $td_1, \ldots, td_n$. In other words, when designing a td generator to be interpreted in an algebra $\mathcal{A}$, one can use regular tree grammars and td transducers which are not necessarily one-producing. The reason is that td_i ($i \in [n]$) can be written as $\mathrm{res} \circ d\text{-}td_i$, which yields a decomposition of td_i into a one-producing td transducer and a tree homomorphism. Similarly, $\mathrm{L}(g) = \mathrm{res}(\mathrm{L}(d\text{-}g))$ and $d\text{-}g$ is one-producing, too. For a td transducer td and a homomorphism h, $td \circ h$ is again a td transducer (see [Eng75]). Furthermore, the composition of tree homomorphisms is a tree homomorphism, which yields the claimed equivalence by induction on the number of td transducers involved, as all the homomorphisms can be shifted to the left. Moreover, if g is approximating and $td_1, \ldots, td_n$ are total, then the resulting td generator G is approximating. This follows from the known fact that $td \circ h$ can be realised by a total td transducer if td is total.

In the following, we shall be interested in the case where $\mathcal{A}$ is a certain algebra over pictures. There are, in fact, several such algebras one may consider, dealing with different sorts of pictures. For example, it is shown in [Dre00] that Lindenmayer systems with turtle interpretation [PL90] and context-free chain-code grammars [MRW82, DH93] are special cases of $\mathcal{A}$-interpreted td generators, where $\mathcal{A}$ is an algebra over line drawings, its central operation being the concatenation of line drawings. Another possibility (also studied in [Dre00]) is to view subsets of $\mathbb{R}^d$ as pictures and use operations which are based on the geometric transformation of such pictures. This yields a generalisation of iterated function systems [Bar93, Edg90, PJS92] and collage grammars [HK91, DK99].

The second case is the one studied in the remainder of this paper. In order to define the respective algebra, let us fix an arbitrary complete metric space $(\mathbb{S}, d)$ such as the Euclidean plane. A *picture* is a nonempty compact subset of $\mathbb{S}$. The set of all pictures is henceforth denoted by $\mathbb{P}$. It is well-known (see, e.g., [Bar93]) that $\mathbb{P}$ can be turned into a complete metric space $(\mathbb{P}, d_{\mathrm{H}})$ by means of the *Hausdorff metric* d_{H}, which is obtained from d as follows. For every point $x \in \mathbb{S}$ and every picture p let $d_1(x,p) = \min\{d(x,y) \mid y \in p\}$. Then, for $p, p' \in \mathbb{P}$,

$$d_{\mathrm{H}}(p,p') = \max(\{d_1(x,p') \mid x \in p\} \cup \{d_1(x,p) \mid x \in p'\}).$$

Intuitively, $d_{\mathrm{H}}(p,p')$ is the maximum distance of a point in one of the pictures to the nearest point in the other picture. It should be clear that every transformation f of $\mathbb{S}$, remains continuous (i.e., a transformation) if it is viewed as a mapping $f\colon \mathbb{P} \to \mathbb{P}$ (which can be done because f is continuous and, therefore, maps compact sets to compact sets). Furthermore, if f is contracting as a mapping on $\mathbb{S}$ then it is also contracting as a mapping on $\mathbb{P}$, with the same contraction factor.

If we are given n transformations $f_1, \ldots, f_n$ of $\mathbb{S}$, we can turn them into a *picture operation* $\langle f_1 \cdots f_n \rangle\colon \mathbb{P}^n \to \mathbb{P}$ by defining

$$\langle f_1 \cdots f_n \rangle(p_1, \ldots, p_n) = f_1(p_1) \cup \cdots \cup f_n(p_n)$$

for all pictures $p_1, \ldots, p_n \in \mathbb{P}$. These picture operations are the main operations of the algebra $\mathcal{P}$ considered in the remainder of this paper. All other operations are constants—operations of arity 0 denoting pictures.

Let us fix an arbitrary nonempty subset $\mathbb{P}_0$ of $\mathbb{P}$ (not necessarily finite or countable) whose elements are called the *primitive pictures*. Now, $\mathcal{P}$ is the $\Sigma_{\mathcal{P}}$-algebra with domain $\mathbb{P}$, such that $\Sigma_{\mathcal{P}}$ consists of

- all symbols $p^{(0)}$, where $p \in \mathbb{P}_0$ and $p_{\mathcal{P}} = p$ (i.e., every primitive picture is a constant of the algebra, and it denotes itself) and
- all symbols $\langle f_1 \cdots f_n \rangle^{(n)}$, where $n \geq 1$, $f_1, \ldots, f_n$ are contracting transformations of $\mathbb{S}$, and $\langle f_1 \cdots f_n \rangle_{\mathcal{A}} = \langle f_1 \cdots f_n \rangle$ (i.e., every picture operation consisting of $n \geq 1$ contracting transformations is an operation of $\mathcal{P}$ and is denoted by itself).

Remark. Notice that it is intentionally left unspecified what a primitive picture is. In fact, the results in this paper do not depend on the choice of $\mathbb{P}_0$. One can even choose $\mathbb{P}_0 = \mathbb{P}$, but this seems to be somewhat inappropriate from the point of view of formal language theory, as in this case there are $\mathcal{P}$-interpreted td generators which do not have a finite description. Another somewhat unnatural effect of this choice would be that every picture could be derived by an appropriate $\mathcal{P}$-interpreted td generator in one step, just by using the picture in question as a constant. Therefore, one should preferably think of $\mathbb{P}_0$ as a set of pictures with finite descriptions in classical geometry (if $(\mathbb{S}, d)$ is the Euclidean plane, for example), like finite unions of polygons, circles, parabola sections etc. $\diamond$

According to the definition of $\mathcal{P}$, every operation $\langle f_1 \cdots f_n \rangle$ is required to consist of contracting transformations. In the theory of iterated function systems there exists a well-known basic result saying that an iterated function system consisting of contracting transformations is itself a continuous contraction. Using similar arguments, we obtain the following lemma saying that the operations of $\mathcal{P}$ are continuous contractions.

5.3 Lemma Let $f_1, \ldots, f_n$ be contracting transformations of $\mathbb{S}$ for some $n \geq 1$, and let c_i be the contractivity factor of f_i for every $i \in [n]$. Then $\langle f_1 \cdots f_n \rangle$ is a continuous contraction with contraction factor $\max\{c_1, \ldots, c_n\}$.

Proof. Let $F = \langle f_1 \cdots f_n \rangle$ and $c = \max\{c_1, \ldots, c_n\}$. It is well known that the inequality $d_{\mathrm{H}}(p_1 \cup p_2, p'_1 \cup p'_2) \leq \max(d_{\mathrm{H}}(p_1, p'_1), d_{\mathrm{H}}(p_2, p'_2))$ holds for all pictures $p_1, p_2, p'_1, p'_2 \in \mathbb{P}$ (see, e.g., [Bar93, Lemma 7.4]). Using this, we obtain the inequality required by the definition of contractivity, for all $p_1, \ldots, p_n, p'_1, \ldots, p'_n \in \mathbb{P}$:

$$\begin{array}{rcl} d_{\mathrm{H}}(F(p_1, \ldots, p_n), F(p'_1, \ldots, p'_n)) & = & d_{\mathrm{H}}(\bigcup_{i \in [n]} f_i(p_i), \bigcup_{i \in [n]} f_i(p'_i)) \\ & \leq & \max_{i \in [n]} d_{\mathrm{H}}(f_i(p_i), f_i(p'_i)) \\ & \leq & \max_{i \in [n]} c_i \cdot d_{\mathrm{H}}(p_i, p'_i) \\ & \leq & c \cdot \max_{i \in [n]} d_{\mathrm{H}}(p_i, p'_i). \end{array}$$

The claimed continuity of F follows directly from the continuity of $f_1, \ldots, f_n$. ■

Hence, all operations of $\mathcal{P}$ are contractions. This enables us to prove the following lemma, which makes it possible to extend $\mathrm{val}_{\mathcal{P}}$ to suitable infinite trees.

5.4 Lemma For every finite signature $\Sigma \subseteq \Sigma_{\mathcal{P}}$ there are $c, C_0 \in \mathbb{R}$, $0 \leq c < 1$, such that $\mathrm{eq}(s, t) \geq n$ implies $d_{\mathrm{H}}(\mathrm{val}_{\mathcal{P}}(s), \mathrm{val}_{\mathcal{P}}(t)) \leq c^n \cdot C_0$ for all $s, t \in \mathcal{F}_\Sigma$ and $n \in \mathbb{N}$.

Proof. Let $p_0 = \bigcup \Sigma^{(0)}$ (which, by the finiteness of Σ, is compact), and let Θ be the set of all transformations which appear in the remaining operations of Σ, i.e., $\Theta = \bigcup \{\{f_1, \ldots, f_n\} \mid \langle f_1 \cdots f_n \rangle \in \Sigma\}$. Define c to be the maximum of the contraction factors of transformations in Θ. By Banach's fixed-point theorem every $f \in \Theta$ possesses a unique fixed point $x_f \in \mathbb{S}$ (as it is contracting). Let $B \supseteq p_0$ be any closed ball such that $x_f \in B$ for all $f \in \Theta$. Clearly, such a ball exists because Θ is finite. Let z be the centre of B and r its radius, and consider some $f \in \Theta$. Let $\Delta = d(z, x_f)$. Then, for every point $x \in B$ we get

$$\begin{array}{rcl} d(f(x), z) & \leq & d(f(x), x_f) + \Delta \\ & = & d(f(x), f(x_f)) + \Delta \\ & \leq & c \cdot d(x, x_f) + \Delta \\ & \leq & c \cdot (d(x, z) + d(z, x_f)) + \Delta \\ & \leq & c \cdot (r + \Delta) + \Delta. \end{array}$$

In other words, if we choose r large enough to satisfy $r \geq c \cdot (r + \Delta) + \Delta$, i.e., $r \geq \Delta \cdot (1 + c)/(1 - c)$, then $f(B) \subseteq B$. As Θ is finite, this proves the following claim.

Claim. There is a closed ball $B \subseteq \mathbb{S}$ such that $\mathrm{val}_{\mathcal{P}}(t) \subseteq B$ for all $t \in \mathcal{F}_{\Sigma}$.

Let B be a ball as in the claim and define C_0 to be the diameter of B: $C_0 = \max\{d(x, y) \mid x, y \in B\}$. By Lemma 5.3 every operation $F_{\mathcal{P}}$, $F \in \Sigma$, is contracting, with contraction factor $\leq c$. Using this, it follows by induction on $n \in \mathbb{N}$ that $\mathrm{eq}(s, t) \geq n$ implies $d_{\mathrm{H}}(\mathrm{val}_{\mathcal{A}}(s), \mathrm{val}_{\mathcal{A}}(t)) \leq c^n \cdot C_0$ for all trees $s, t \in \mathcal{F}_{\Sigma}$, as claimed in the lemma. This is because, for $n = 0$, by the choice of C_0 and the fact that $\mathrm{val}_{\mathcal{P}}(s), \mathrm{val}_{\mathcal{P}}(t) \subseteq B$, we have $d_{\mathrm{H}}(\mathrm{val}_{\mathcal{P}}(s), \mathrm{val}_{\mathcal{P}}(t)) \leq C_0$. For $n > 0$, let $s = F[s_1, \ldots, s_k]$ and $t = F[t_1, \ldots, t_k]$. By the induction hypothesis the inequality $d_{\mathrm{H}}(\mathrm{val}_{\mathcal{P}}(s_i), \mathrm{val}_{\mathcal{P}}(t_i)) \leq c^{n-1} \cdot C_0$ holds for all $i \in [k]$, which yields $d_{\mathrm{H}}(\mathrm{val}_{\mathcal{P}}(s), \mathrm{val}_{\mathcal{P}}(t)) \leq c^n \cdot C_0$, due to the contractivity of $F_{\mathcal{P}}$. This completes the proof of the lemma. ■

Using Lemma 5.4, we can now define the value of those infinite trees over $\Sigma_{\mathcal{P}}$ which contain only finitely many pairwise distinct symbols. Let $t \in \mathcal{T}_{\Sigma}$ be infinite, where $\Sigma \subseteq \Sigma_{\mathcal{P}}$ is finite, and consider a Cauchy sequence $(t_i)_{i \in \mathbb{N}}$ in $\mathcal{F}_{\Sigma}$ with $\lim t_i = t$ (without loss of generality we may assume that $\Sigma^{(0)} \neq \emptyset$, which ensures that such a sequence exists). By Lemma 5.4 the sequence $(\mathrm{val}_{\mathcal{P}}(t_i))_{i \in \mathbb{N}}$ is Cauchy in $(\mathbb{P}, d_{\mathrm{H}})$. Since $(\mathbb{P}, d_{\mathrm{H}})$ is complete, $\lim \mathrm{val}_{\mathcal{P}}(t_i)$ is a picture in $\mathbb{P}$ as well. We can thus define $\mathrm{val}_{\mathcal{P}}(t) = \lim \mathrm{val}_{\mathcal{P}}(t_i)$.

In order to see that this definition is sound, it suffices to notice that any other sequence $(s_i)_{i \in \mathbb{N}}$ in $\mathcal{F}_{\Sigma}$ which converges to t satisfies $\lim \mathrm{val}_{\mathcal{P}}(s_i) = \lim \mathrm{val}_{\mathcal{P}}(t_i)$ because, according to Lemma 5.4, $d_{\mathrm{H}}(\mathrm{val}_{\mathcal{P}}(s_i), \mathrm{val}_{\mathcal{P}}(t_i))$ converges to 0.

As a consequence of this definition, for every $\mathcal{P}$-interpreted td generator $\mathcal{G} = (G, \mathcal{A})$ the mapping $\mathrm{val}_{\mathcal{A}}$ is now total. This is because, by definition, $\mathcal{A}$ is a Σ-algebra, where Σ is the finite output signature of G. It is furthermore worthwhile to notice that, since $\mathrm{val}_{\mathcal{A}}(t) = \mathrm{val}_{\mathcal{P}}(h(t))$ for a tree homomorphism h (by definition) and tree homomorphisms are continuous, it follows that $\mathrm{val}_{\mathcal{A}}$ is continuous as well.

Due to the totality of $\mathrm{val}_{\mathcal{A}}$, the language $\mathrm{L}_{\mathrm{inf}}(\mathcal{G})$ is defined for every $\mathcal{P}$-interpreted td generator $\mathcal{G} = (G, \mathcal{A})$. The elements of $\mathrm{L}_{\mathrm{inf}}(\mathcal{G})$ will be called the *syntactic fractals* (or just *fractals*, for short) generated by $\mathcal{G}$. The attribute 'syntactic' is used because this notion of fractals depends on $\mathcal{G}$, i.e., on the way a picture is generated by the particular td generator at hand. By contrast, in the literature most (if not all) attempts to provide a formal definition for the term "fractal" are *semantic* ones which are based, for example, on the fractal dimension of a picture. The syntactic variant proposed here focuses on the finiteness or infinity of the *description* of a picture. Of course, there may sometimes be pictures in $\mathrm{L}_{\mathrm{fin}}(\mathcal{G}) \cap \mathrm{L}_{\mathrm{inf}}(\mathcal{G})$. According to the definition above, these are called fractals as well, but it is certainly a matter of taste how to decide this question.

Let us call a $\mathcal{P}$-interpreted td generator $\mathcal{G} = (G, \mathcal{A})$ *approximating* if G is approximating. Making use of the continuity of $\mathrm{val}_{\mathcal{A}}$, the results of the previous section carry over to $\mathrm{L}_{\mathrm{fin}}(\mathcal{G})$ and $\mathrm{L}_{\mathrm{inf}}(\mathcal{G})$. This is stated in the following theorem, where a sequence of pictures is called a *refinement sequence in* $\mathcal{G}$ if it has the form $(\mathrm{val}_{\mathcal{A}}(t_i))_{i \in \mathbb{N}}$ for some refinement sequence $(t_i)_{i \in \mathbb{N}}$ in G.

5.5 Theorem For every $\mathcal{P}$-interpreted td generator $\mathcal{G}$ the language $\mathrm{L}(\mathcal{G})$ is closed. Furthermore, if $\mathcal{G}$ is approximating then

(1) $\mathrm{L}(\mathcal{G})$ is the closure of $\mathrm{L}_{\mathrm{fin}}(\mathcal{G})$ and

(2) a picture $p \in \mathbb{P}$ is an element of $\mathrm{L}_{\mathrm{inf}}(\mathcal{G})$ if and only if $p = \lim p_i$ for some refinement sequence $(p_i)_{i\in\mathbb{N}}$ in $\mathrm{L}_{\mathrm{fin}}(\mathcal{G})$.

Proof. This follows directly from Theorems 4.5 and 4.7, and the fact that $\mathrm{val}_{\mathcal{A}}$ is continuous for every Σ-algebra $\mathcal{A}$ derived from $\mathcal{P}$ (where Σ is finite). ∎

6 Examples

The purpose of this section is to present some examples, all of which having as their underlying space the Euclidean plane. Furthermore, all operations which appear in the examples are derived from constants which are polygons or filled polygons, and picture operations consisting of affine transformations. As a convenient side effect of the latter, operations can be suitably represented in a graphical way, which is probably easier to grasp than an explicit numerical definition.

Recall that an affine transformation f of $\mathbb{R}^2$ is uniquely determined by any three pairs $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ such that $y_i = f(x_i)$ for $i = \{1, 2, 3\}$, provided that the x_i do not lie on the same straight line (or, equivalently, that $x_2 - x_1$ and $x_3 - x_1$ are linearly independent). Therefore, f can be described by fixing the image of some geometric object p which is not a subset of a straight line (a polygon, say) under f. In fact, to be precise we must also say which point in p is mapped to which one in $f(p)$. If p is a polygon, this can be done by saying which of its vertices is the first, which one is the second, etc. In the following such a geometric object p, whose sole purpose is to visualise transformations, is called a *sample*. Like the constants, all samples in this section are polygons. The order of the vertices of a sample is indicated by an arrow.

As an example, Figure 3 depicts the operations of a Σ-algebra $\mathcal{H}$ derived from $\mathcal{P}$, where $\Sigma = \{F^{(3)}, F'^{(3)}, G^{(1)}, H^{(0)}\}$, the sample being the hexagon shown at the top. It should be rather obvious how to interpret the bottom row of Figure 3. The first picture says that $F_{\mathcal{H}}$ is given by $\langle f_1 f_2 f_3\rangle$, where f_1 is the unique affine transformation mapping the sample to the topmost one of the three small hexagons (without any rotation or reflexion). Similarly, f_2 and f_3 map the sample to the two remaining hexagons. The derived operation $F'_{\mathcal{H}}$ makes use of the same transformations, but in addition its definition involves a constant of $\mathcal{P}$. More precisely, $F'_{\mathcal{H}}(p_1, p_2, p_3) = p_0 \cup \langle f_1 f_2 f_3\rangle(p_1, p_2, p_3)$ for all $p_1, p_2, p_3 \in \mathbb{P}$, where p_0 is the outermost hexagonal polygon. Notice that $F'_{\mathcal{H}}$ is indeed an allowed operation as it can be defined by $\langle f_0 f_1 f_2 f_3\rangle[p'_0, x_1, x_2, x_3] \in \mathcal{F}_{\Sigma_{\mathcal{P}}}$, where f_0 is any injective affine contraction such that $p'_0 = f_0^{-1}(p_0) \in \mathbb{P}_0$ (assuming that $\mathbb{P}_0$ is rich enough).

The operation $G_{\mathcal{H}}$ is the unary derived operation such that $G_{\mathcal{H}}(p) = p_0 \cup g_1(p)$ for all $p \in \mathbb{P}$, where p_0 is as before and g_1 is the transformation indicated in the figure, involving a slight scaling and a rotation about the centre of the sample. Finally, $H_{\mathcal{H}}$ is the filled hexagon whose dimensions are those of the sample.

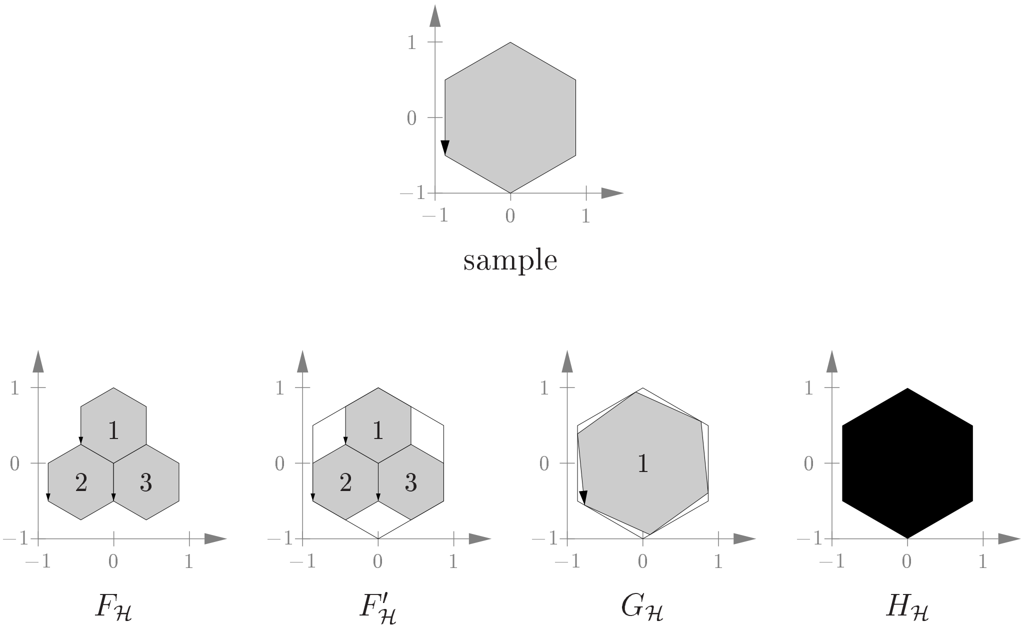

Figure 3: Graphical representation of operations used in the first example

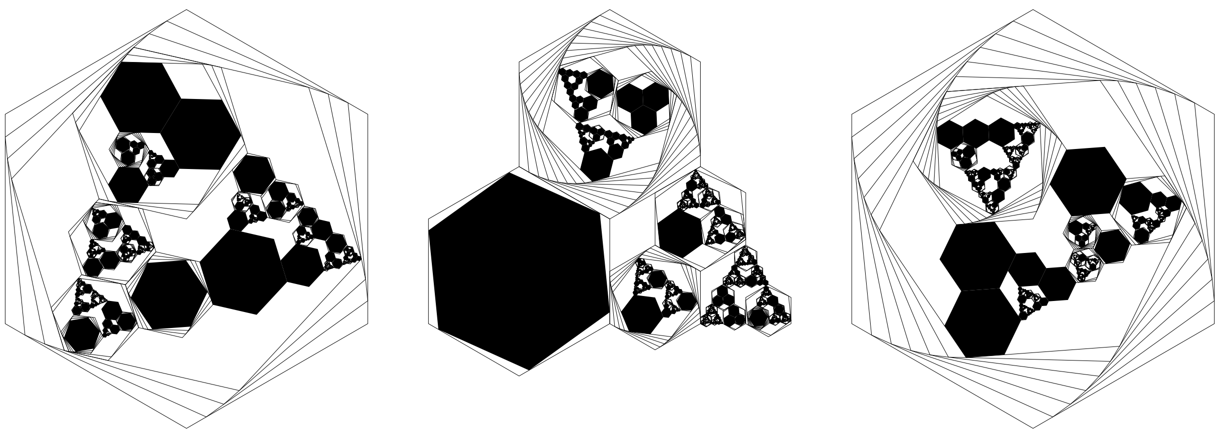

Figure 4: Fractals generated by $\mathcal{G}_{\text{HEXAGONS}_1}$

Now, let HEXAGONS_1 be the td generator (g, λ) simply consisting of the regular tree grammar $g = (\{S, A\}, \Sigma, P, S)$, where

$$\begin{aligned} P = \{ \; & S \to F[S,S,S], \quad S \to G[A], \quad S \to H, \\ & A \to F'[S,S,S], \quad A \to G[A], \quad A \to H \; \}. \end{aligned}$$

Then the $\mathcal{P}$-interpreted td generator $\mathcal{G}_{\text{HEXAGONS}_1} = (\text{HEXAGONS}_1, \mathcal{H})$ yields fractals like those in Figure 4.[2] An initial segment of a refinement sequence is depicted in Figure 5. It was shown in [Dre00] that $\mathcal{P}$-interpreted td generators of this kind, i.e., those which just consist of a regular tree grammar, are equivalent to collage grammars, as introduced by

[2]Of course, strictly speaking, these are only good approximations instead of "real" fractals.

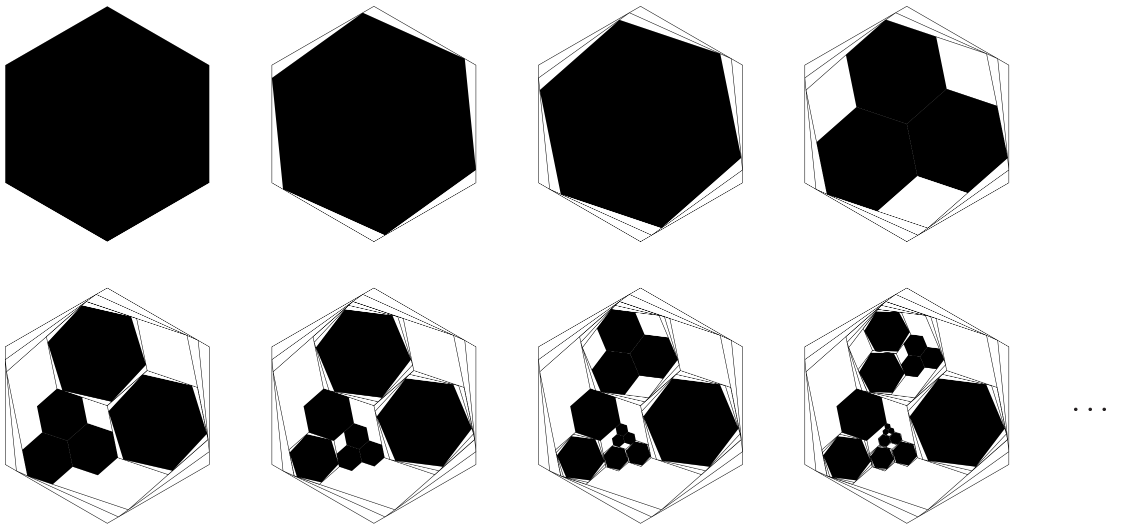

Figure 5: A refinement sequence in $\mathcal{G}_{\text{HEXAGONS}_1}$

Habel and Kreowski in [HK91] (see also [DK99]). To be precise, it must be mentioned that the usual definition of collage grammars neither contains the restriction to contracting transformations and compact pictures, nor does it define $\mathrm{L}_{\mathrm{inf}}(\mathcal{G})$. Thus, in this paper we obtain a notion of fractals generated by collage grammars at the expense of imposing the restriction to contracting transformations and compactness.

Using a td generator which consists of a regular tree grammar and a td transducer, a language of more symmetric pictures can be obtained. For this, let $\Sigma_0 = \{a^{(1)}, b^{(1)}, c^{(0)}\}$ and define $\textsc{Hexagons}_2 = (g', td)$ where $g' = (\{S\}, \Sigma_0, P', S)$ and $td = (\Sigma_0, \Sigma, \{q, q'\}, R, q)$. Here, Σ is as before and the productions of g' and rules of td are the following ones:[3]

$$\begin{array}{rcl} P' &=& \{\ S \to a[S],\quad S \to b[S],\quad S \to c\ \}, \text{ and} \\ R &=& \{\ qa \ \to F[q,q,q],\quad qb \ \to G[q'],\quad qc \ \to H, \\ && \quad q'a \to F'[q,q,q],\quad q'b \to G[q'],\quad q'c \to H\ \}. \end{array}$$

Notice the similarity between the productions of g above and the rules of td. The states q and q' correspond to the nonterminals S and A, respectively, the only difference being that the formerly nondeterministic choice of rules is now determined by the input symbols read. In fact, it was shown in [ERS80] that, for a td transducer having a monadic input signature, the input symbols determine a partition of the set of rules into "tables" in the sense of T0L-systems [Roz73]. Rules having the same input symbol in their left-hand sides belong to the same table. Since the input tree is monadic, in each step all states process the same input symbol, which ensures that rules from the same table are chosen.

As for the present example, this means that every tree t generated by $\textsc{Hexagons}_2$ has a rather special form. For all nodes $v, v' \in V(t)$ such that $|v| = |v'|$ it holds that $t(v) = t(v')$.

[3]Recall the convention that, since Σ_0 is monadic, the variable x_1 is omitted in the right-hand sides of the rules in R.

Regarding $\mathrm{L}(\mathcal{G}_{\textsc{Hexagons}_2})$ (where $\mathcal{G}_{\textsc{Hexagons}_2} = (\textsc{Hexagons}_2, \mathcal{H})$) this guarantees a high degree of symmetry, as shown in Figure 6. Part of the refinement sequence of the first fractal in Figure 6 can be seen in Figure 7.

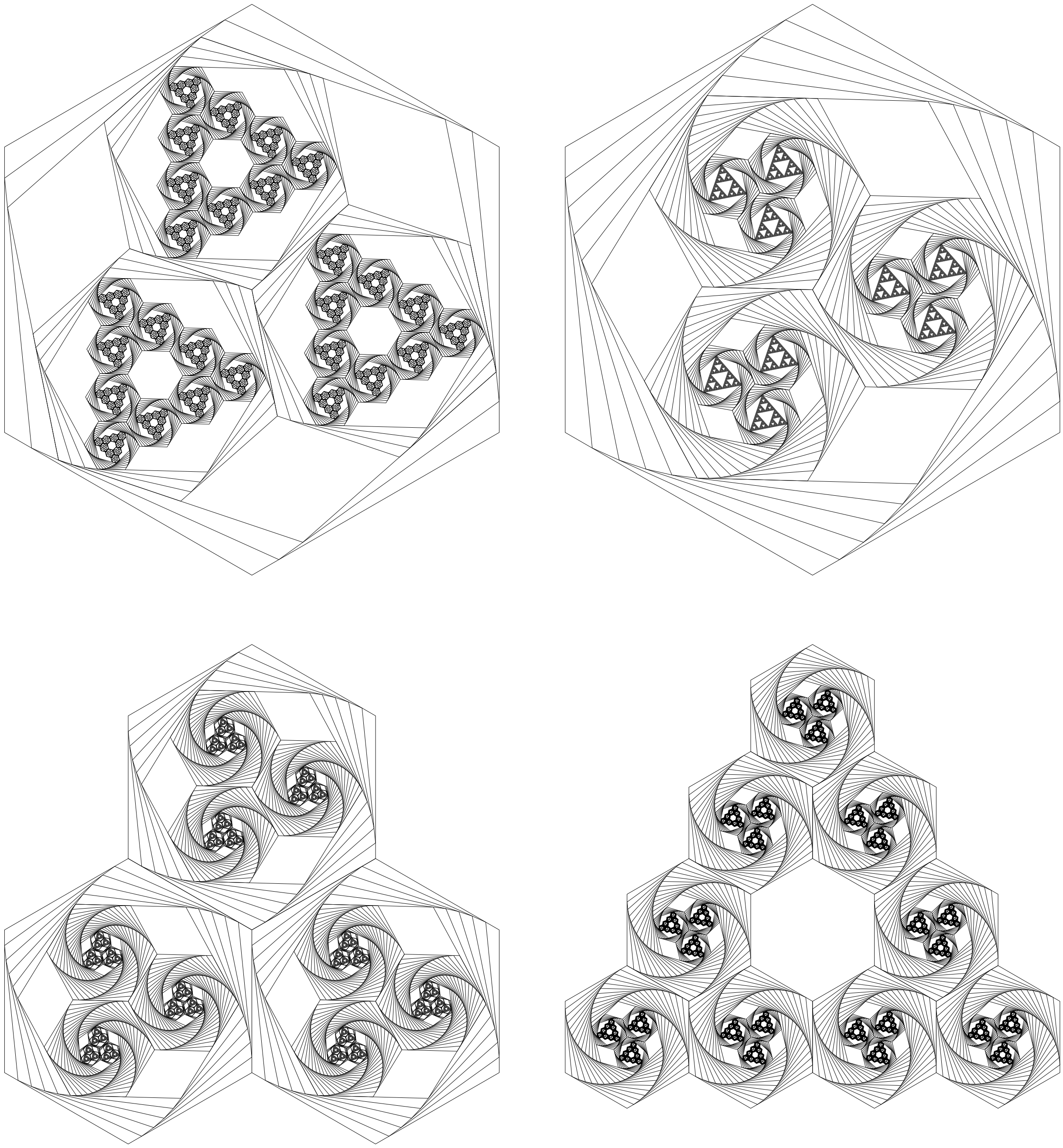

Figure 6: Fractals generated by $\mathcal{G}_{\textsc{Hexagons}_2}$

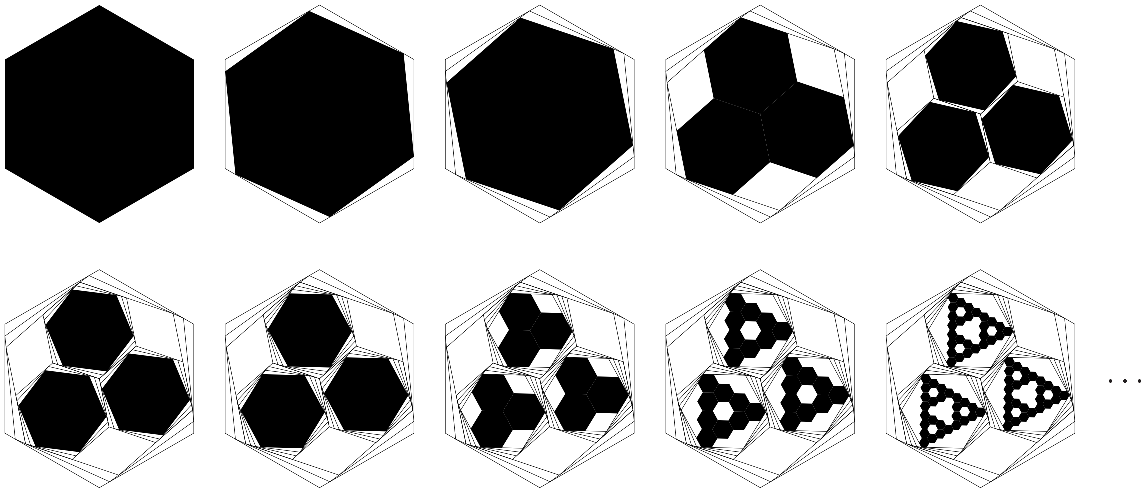

Figure 7: A refinement sequence in $\mathcal{G}_{\textsc{Hexagons}_2}$

To discuss another example, let $\Sigma = \{F^{(2)}, F'^{(1)}, F''^{(1)}, G^{(2)}, G'^{(1)}, G''^{(1)}, C^{(0)}\}$ and consider the Σ-algebra $\mathcal{S}$ whose operations are depicted (in a similar way as before) in Figure 8.

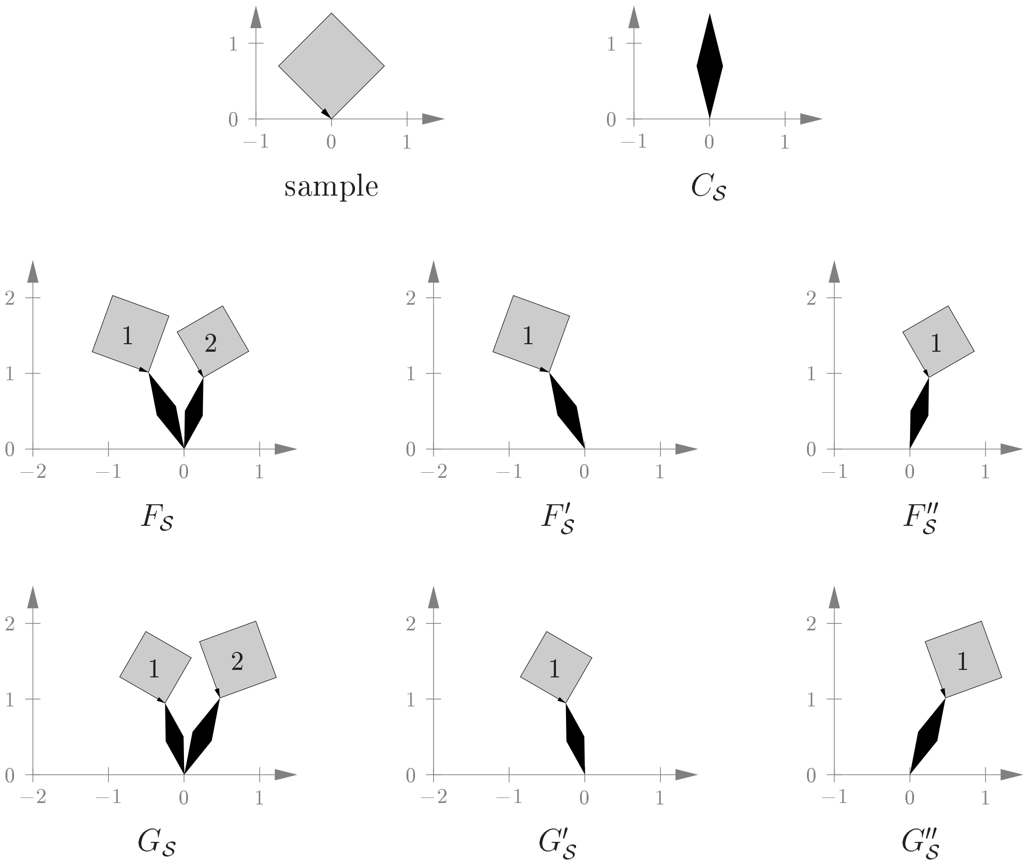

Figure 8: The operations of $\mathcal{S}$

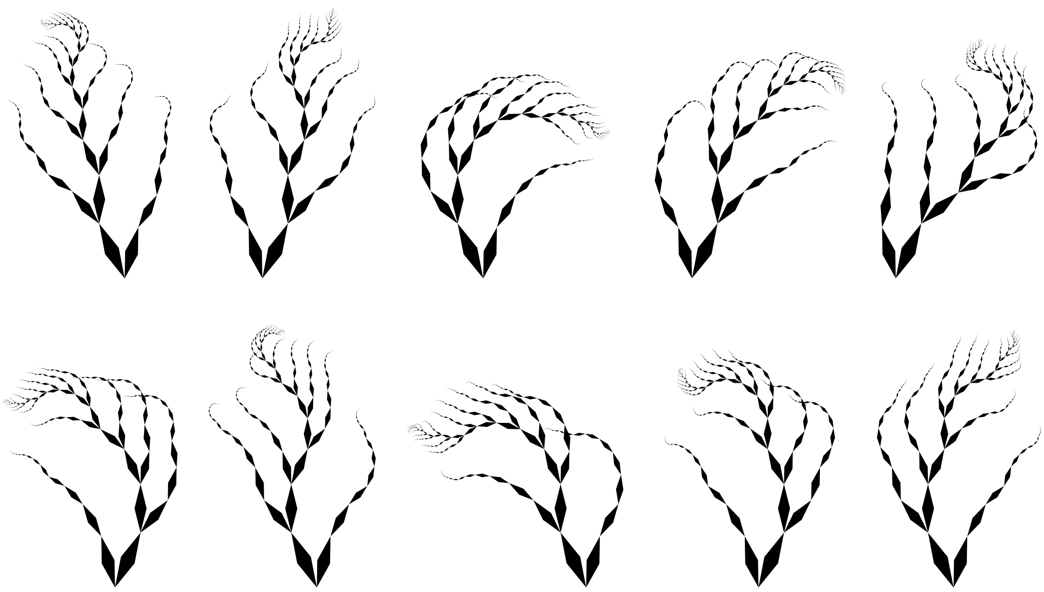

Figure 9: Fractals generated by $\mathcal{G}_{\text{SEAWEED}}$

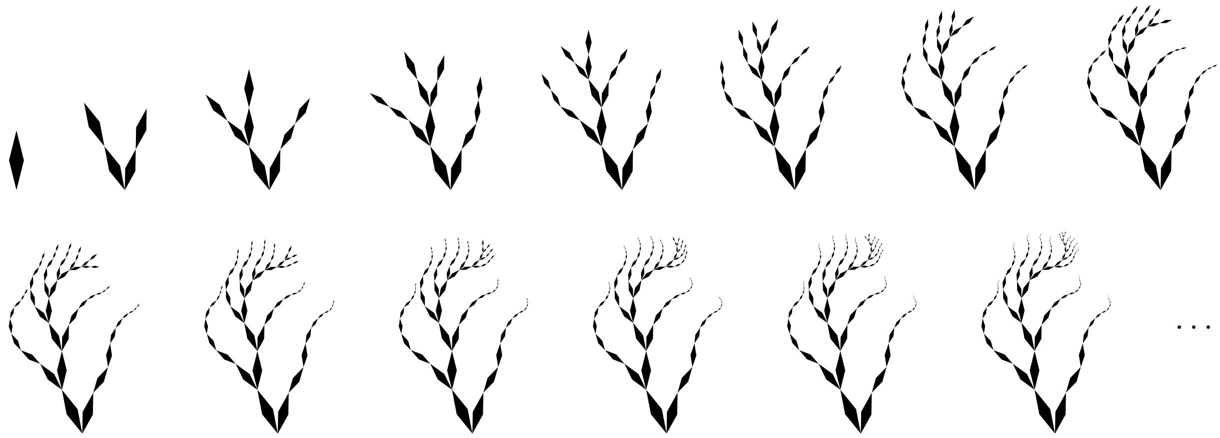

Figure 10: A refinement sequence in $\mathcal{G}_{\text{SEAWEED}}$

Notice that, intuitively, $F'_{\mathcal{S}}$ and $F''_{\mathcal{S}}$ ($G'_{\mathcal{S}}$ and $G''_{\mathcal{S}}$) are the left and right part of $F_{\mathcal{S}}$ ($G_{\mathcal{S}}$, respectively). The rotation angles used in $F_{\mathcal{S}}$ are 25° and −15°. Conversely, in $G_{\mathcal{S}}$ the angles are 15° and −25°.

Let SEAWEED $= (g', td)$, where g' is as in the previous example, generating all trees over $\Sigma_0 = \{a^{(1)}, b^{(1)}, c^{(0)}\}$, and $td = (\Sigma_0, \Sigma, \{q, r, l\}, R, q)$ where

$$\begin{aligned} R = \{\ & qa \to F[q,r], && qb \to G[l,q], && qc \to C, \\ & ra \to F'[r], && rb \to F''[r], && rc \to C \\ & la \to G'[l], && lb \to G''[l], && lc \to C\ \}. \end{aligned}$$

Some of the generated fractals and a refinement sequence are shown in Figures 9 and 10. Intuitively, the state q of td produces the main branch of each generated picture, turning

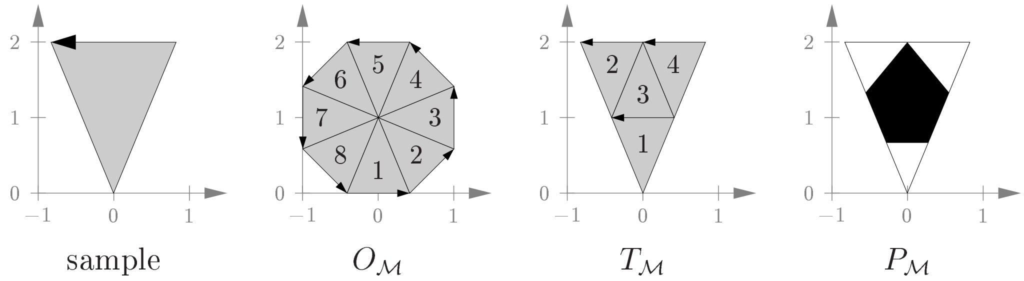

Figure 11: The operations of $\mathcal{M}$

to the left if the input symbol b is encountered, and to the right otherwise. In both cases it produces a smaller side-branch (which is not ramifying any further) at the opposite side. Left side-branches are produced by means of the state l whereas right ones are produced by means of r. Both of them generally behave like the main branch, but l uses the smaller one of the two rotation angles whenever it turns left. Similarly, r uses the small angle when turning right.

The next example shows how one can make use of a td generator consisting, as in the two examples before, of a regular tree grammar and a td transducer, but where the trees generated by the regular tree grammar are not monadic. Let $\Sigma = \{O^{(8)}, T^{(4)}, P^{(0)}\}$ and let $\mathcal{M}$ be the Σ-algebra whose operations are depicted in Figure 11. Consider the td generator $\mathcal{G}_{\text{Mosaic}} = (\text{Mosaic}, \mathcal{M})$, where $\text{Mosaic} = (g, td)$ is given by $g = (\{S, A\}, \Sigma_0, P, S)$ and $td = (\Sigma_0, \Sigma, \{q, q'\}, R, q)$ with

$$\begin{array}{rcl} P & = & \{\ S \to \mathit{init}[A],\ \ S \to b,\ \ A \to a[A, A],\ \ A \to b\ \}\ \text{and} \\ R & = & \{\ q\,\mathit{init} \to O[qx_1, qx_1, qx_1, qx_1, qx_1, qx_1, qx_1, qx_1], \\ & & \quad qa \quad\ \to T[qx_1, qx_2, q'x_2, qx_2], \\ & & \quad q'a \quad\ \to T[q'x_2, q'x_1, qx_1, q'x_1], \\ & & \quad qb \quad\ \to P, \\ & & \quad q'b \quad\ \to P\ \}. \end{array}$$

Figure 12 shows some of the fractals generated by $\mathcal{G}_{\text{Mosaic}}$; a refinement sequence is pictured in Figure 13. Intuitively, the octagon is divided into concentric rings of triangles using $T_{\mathcal{M}}$. The input tree processed by td determines whether or not a ring is subdivided. In order to see how this works, consider a derivation of td on input $t \in \mathcal{T}_{\Sigma_0}$. After some derivation steps, every state in the derived tree corresponds to a triangle on one of the mentioned rings. What is more, all states referring to the same ring process the same subtree of t. In the next step, if the input in state q is $a[t_1, t_2]$, the triangle is replaced with four smaller ones: an inner triangle whose further refinement is determined by t_1, and three outer triangles refined along t_2. Because of the fact that one of the triangles is mirrored, we need a second state in order to keep track of the orientation. The refinement of a ring stops if the input symbol b is encountered.

Finally, let us consider an example which demonstrates the benefit of using more than just one td transducer. The previous examples should have made clear that a single nonlinear

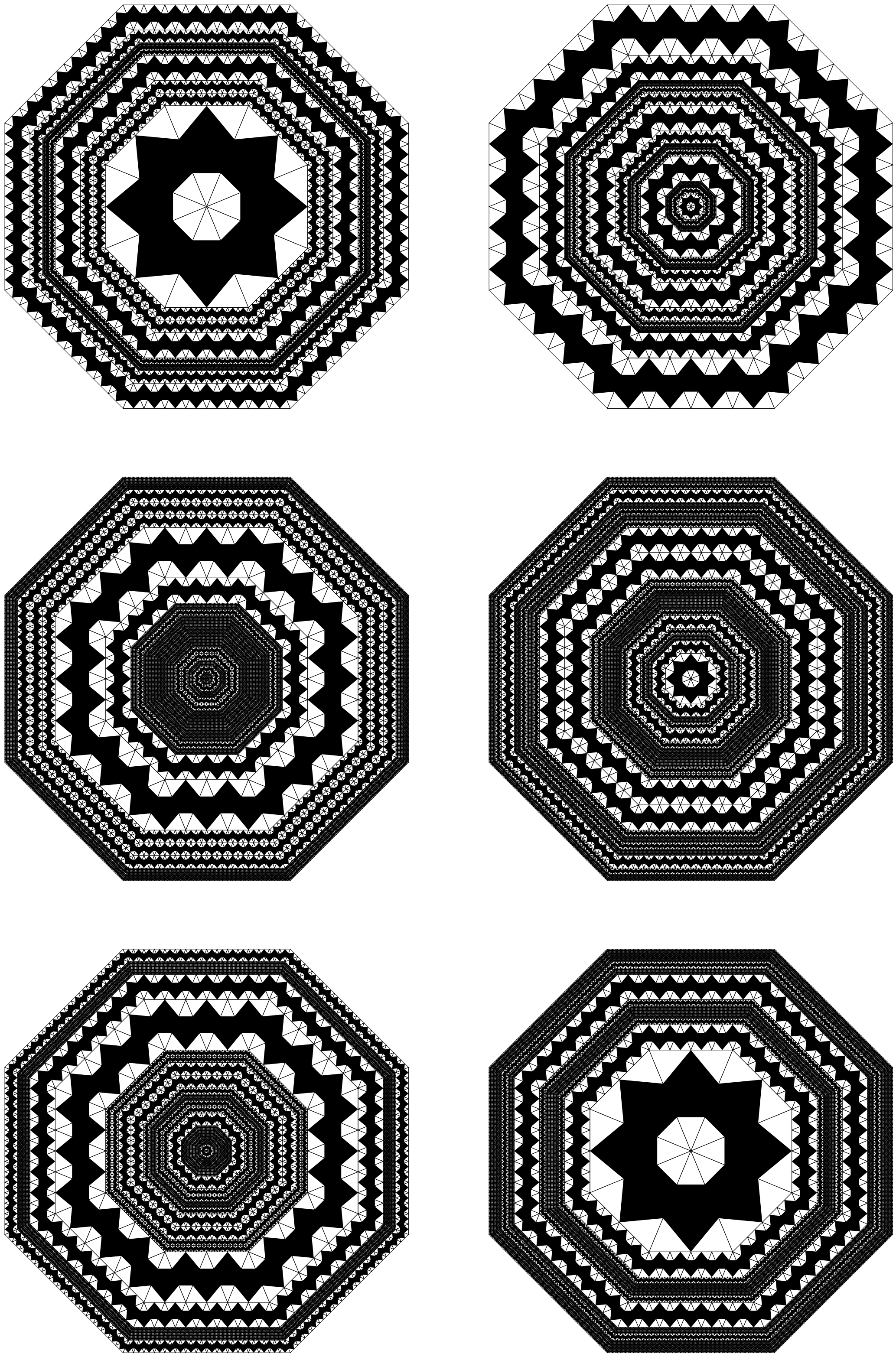

Figure 12: Fractals generated by $\mathcal{G}_{\textsc{Mosaic}}$

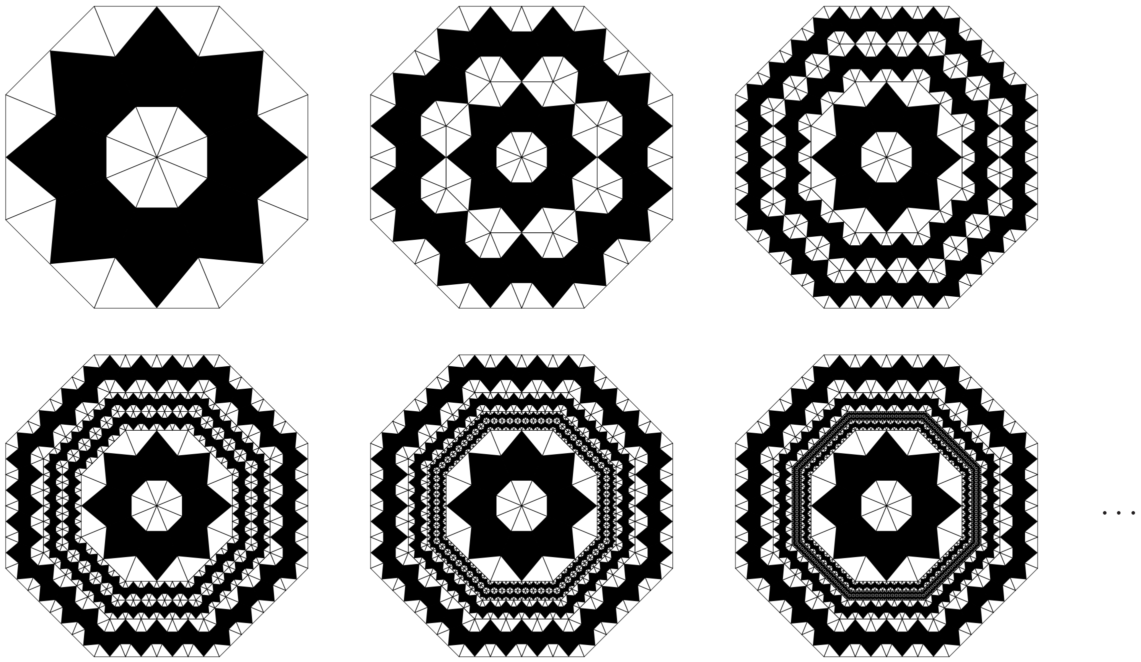

Figure 13: A refinement sequence in $\mathcal{G}_{\text{Mosaic}}$ (the first picture, $P_{\mathcal{M}}$, being omitted)

td transducer can be employed in order to create certain dependencies between different parts of a generated picture. The simplest case is a td transducer whose input signature is Σ_{succ} (or a similar one). Here, the input $s^n[0]$ can be used as a kind of counter in order to ensure that all parts of the picture are refined to the same degree. This cannot be achieved by a regular tree grammar (provided that the output trees are not essentially monadic ones) because the latter allows to replace nonterminal symbols independently from each other (see the pictures generated by $\mathcal{G}_{\text{Hexagons}_1}$ in the beginning of this section). However, a td transducer can also use nondeterminism. If this is combined with nonlinearity, two copies of a subtree of the input tree may yield output trees which show certain similarities (by taking advantage of the fact that the same input is processed), yet they are different because of the nondeterministic choice of rules. Now, a second td transducer can be used in order to repeat this game on the next level: By copying subtrees of its input (which the first td transducer produced in a nondeterministic way) it can use them to produce dependencies between different parts of a picture.

Here is an example, using the Σ-algebra $\mathcal{L}$ whose operations are shown in Figure 14 (in the way the reader should by now be familiar with). Thus, $\Sigma = \{E^{(3)}, F^{(2)}, G^{(1)}, G'^{(1)}, T^{(0)}\}$. The aim is to construct a $\mathcal{P}$-interpreted td generator $\mathcal{G}_{\text{Leaves}} = (\text{Leaves}, \mathcal{L})$ such that $\mathrm{L}_{\text{inf}}(\mathcal{G}_{\text{Leaves}})$ consists of "leaves" like those shown in Figure 15. Every such leaf consists of a stem with several leaflets on either side. Each pair of opposite leaflets is exactly symmetric: The leaflet on the right is the mirror image of its counterpart on the left. On the other hand, the individual leaflets themselves are not symmetric with respect to their middle axis. Intuitively, this is achieved by generating the two halves

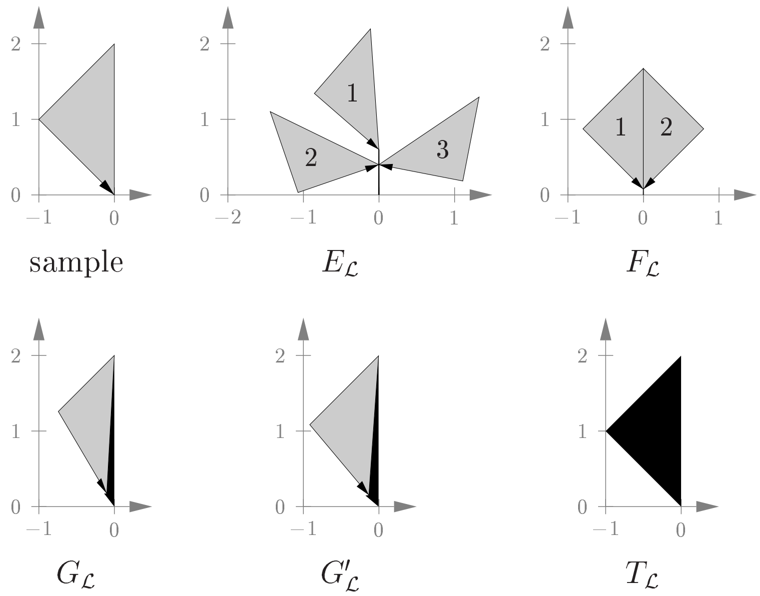

Figure 14: The operations of $\mathcal{L}$

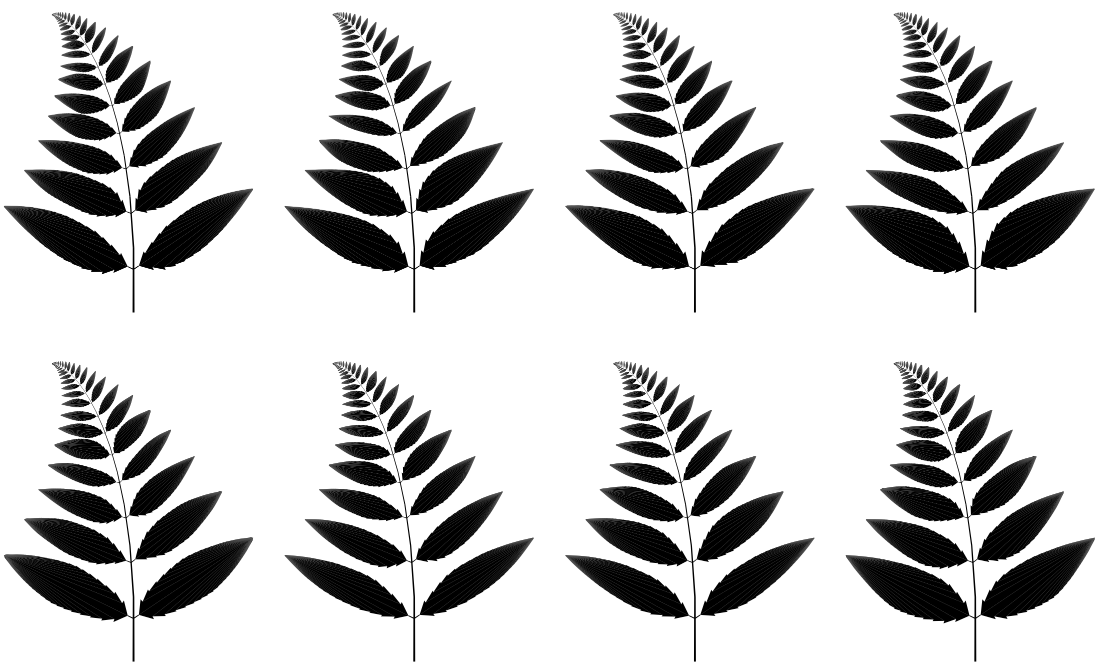

Figure 15: Fractals generated by $\mathcal{G}_{\text{LEAVES}}$

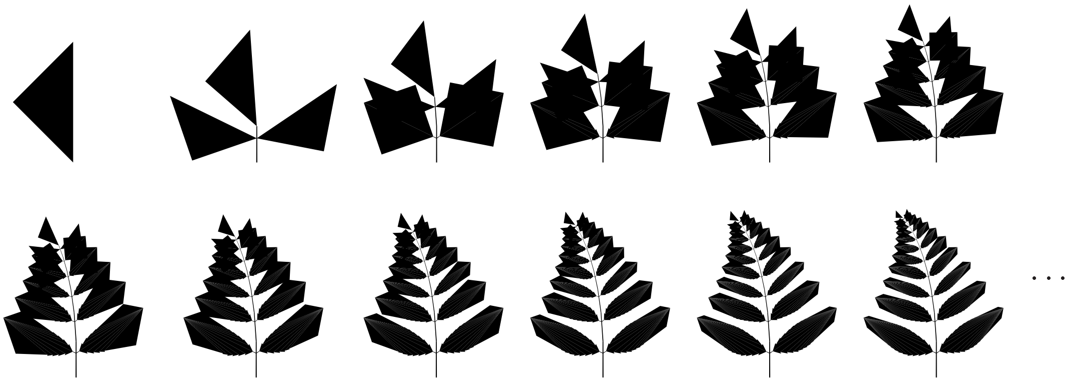

Figure 16: A refinement sequence in $\mathcal{G}_{\text{LEAVES}}$

of a leaflet in a nondeterministic way using a td transducer, and then duplicating the resulting leaflets by another td transducer in order to obtain two copies of each. For this, let LEAVES $= (g_{\text{succ}}, td_0\, td_1)$, where g_{succ} is as in Lemma 4.4 and

$$td_0 = (\Sigma_{\text{succ}}, \Sigma_0, \{q_0, q, q'\}, R_0, q_0) \quad \text{and} \quad td_1 = (\Sigma_0, \Sigma, \{q\}, R, q).$$

Here, $\Sigma_0 = \{E^{(2)}, F^{(2)}, G^{(1)}, G'^{(1)}, T^{(0)}\}$ (which deviates from Σ only with respect to the rank of E) and

$$\begin{aligned} R_0 &= \{\ q_0 s \to E[q_0, q],\ qs \to F[q', q'],\ q's \to G[q'],\ q's \to G'[q'], \\ &\quad\ q_0 0 \to T,\ q0 \to T,\ q'0 \to T\ \}, \text{ and} \\ R &= \{\ qE \to E[qx_1, qx_2, qx_2],\ qF \to F[qx_1, qx_2], \\ &\quad\ qG \to G[qx_1],\ qG' \to G'[qx_1], \\ &\quad\ qT \to T\ \}. \end{aligned}$$

Thus, td_1 simply copies its input tree, except that the second subtree of every E-labelled node is duplicated. In Figure 16 one of the refinement sequences in $\mathcal{G}_{\text{LEAVES}}$ is depicted.

7 Rational fractals

$\mathcal{P}$-interpreted td generators can simulate the so-called mutually recursive function systems (MRFS's), which generalise the well-known iterated function systems. In the literature, MRFS's can be found under different names, with slightly different but obviously equivalent definitions regarding their picture generating capabilities. They are called MRFS's by Cŭlik and Dube [CD93a, CD93b] and hierarchical IFS's by Peitgen, Jürgens, and Saupe [PJS92]. As a slight extension, one may allow an MRFS to make use of the *condensation sets* known from Barnsley's famous book [Bar93], which yields *MRFS's with condensation sets* (cMRFS's).

The following definition of cMRFS's in terms of td generators is taken from [Dre00] (where they are called tree-based IFS's).

7.1 Definition (*tree-based cMRFS*) A *tree-based cMRFS* is a $\mathcal{P}$-interpreted td generator $(G, \mathcal{A})$ such that $G = (td, g_{\mathrm{succ}})$ for a total, deterministic, and nondeleting td transducer td. $\diamond$

Notice that every tree-based cMRFS as in the definition is approximating. Since the regular tree grammar g_{succ} just generates $\mathcal{T}_{\Sigma_{\mathrm{succ}}}$, $\mathrm{L}(G)$ is simply the range of the td transducer td. As the latter is required to be total, deterministic, and nondeleting, it is clear that $td(s^n[0])$ consists of a single tree of depth n, for every $n \in \mathbb{N}$. Consequently, $td(s^\infty) = \mathrm{L}_{\mathrm{inf}}(G)$ is a singleton, i.e., $\mathrm{L}_{\mathrm{inf}}(\mathcal{G})$ consists of a single fractal.

It was shown in [Dre00] that tree-based cMRFS's generate the same fractals as cMRFS's, provided the latter use only condensation sets which can be expressed using the constants in $\mathbb{P}_0$. Intuitively, the condensation sets are the constants used in the definition of the derived operations in $\mathcal{A}$. Thus, if we restrict ourselves to the case where $\mathcal{A}$ contains only operations of $\mathcal{P}$ (rather than derived ones), tree-based MRFS's are obtained which are equal in power to MRFS's. Finally, if the state set of td is required to be a singleton, then we get tree-based IFS's (with or without condensation, the difference being the same as above).

As, in general, a $\mathcal{P}$-interpreted td generator yields a whole language of fractals, one may immediately expect that some of its fractals can be generated by cMRFS's whereas others cannot. The study of this relationship is the purpose of the present section.

The next definition, together with Theorem 7.4, provides a name for the sort of fractals generated by cMRFS's.

7.2 Definition (*rational fractal*) A tree t is *rational* if it has only finitely many subtrees, i.e., if the set $\{t/v \mid v \in V(t)\}$ is finite. A picture p is a $\mathbb{P}_0$*-rational fractal* (*rational fractal*, for short) if $p = \mathrm{val}_{\mathcal{P}}(t)$ for some infinite rational tree $t \in \mathcal{T}_{\Sigma_{\mathcal{P}}}$. $\diamond$

Let us call a tree or fractal *irrational* if it is not rational. (Notice that irrational trees must necessarily be infinite.) The following lemma states that a td transducer cannot derive any irrational tree from a rational one without deriving a rational tree from this input tree, too. Furthermore, if the irrational output tree is taken from a regular tree language L, then the rational one can also be taken from L.

7.3 Lemma Let $td = (\Sigma, \Sigma', Q, R, q_0)$ be a td transducer, $L \subseteq \mathcal{T}_{\Sigma'}$ a regular tree language, and $t \in \mathcal{T}_\Sigma$ a rational tree. If $td(t) \cap L \neq \emptyset$ then it contains a rational tree.

Proof. It is well known that the regularity of L implies the existence of a td transducer $td' = (\Sigma, \Sigma', Q', R', q_0')$ such that $td'(s) = td(s) \cap L$ for all $s \in \mathcal{T}_\Sigma$ (see, e.g., [Eng82, Lemma 2.2]). Therefore, it suffices to prove the following claim.

Claim. For every rational tree $t \in \mathcal{T}_\Sigma$ such that $td(t) \neq \emptyset$, $td(t)$ contains a rational tree.

The claim is proved by constructing a rational t-derivation tree. For this, if dt is a t-derivation tree of td, denote the tree in $Q(\mathcal{T}_\Sigma)$ from which dt/v is derived by $orig_{dt}(v)$: if $|v| = l$ and $q_0 t \Rightarrow^l_{d\text{-}td} t' \Rightarrow^\infty_{d\text{-}td} dt$ is the unique derivation yielding dt, then $orig_{dt}(v) = t'/v$. The rational t-derivation tree dt to be constructed will be obtained as the limit of a converging sequence $(dt_i)_{i\in\mathbb{N}}$ of t-derivation trees. For this, associate with every tree $s \in Q(\mathcal{T}_\Sigma)$ an arbitrary but fixed tree $\delta(s)$ such that $s \Rightarrow^{\infty}_{d\text{-}td} \delta(s)$, if such a tree exists.

The sequence $(dt_i)_{i\in\mathbb{N}}$ is constructed inductively. To start with, let $dt_0 = \delta(q_0 t)$ (which is defined, by the assumption that $td(t) \neq \emptyset$). For $i \geq 1$, dt_i is the unique tree such that

(1) $dt_i(v) = dt_{i-1}(v)$ for all $v \in V(dt_{i-1})$ with $|v| < i$, and

(2) $dt_i/v = \delta(\mathit{orig}_{dt_{i-1}}(v))$ for all $v \in V(dt_{i-1})$ with $|v| = i$.

The tree $\delta(\mathit{orig}_{dt_{i-1}}(v))$ in (2) exists since the derivation $\mathit{orig}_{dt_{i-1}}(v) \Rightarrow^*_{DTtd} dt_{i-1}/v$ exists. Therefore, each dt_i is indeed a t-derivation tree of td. Furthermore, $\lim dt_i$ (which, by the fact that $\mathrm{eq}(dt_i, dt_{i+1}) > i$ for all $i \in \mathbb{N}$, is defined) yields an infinite t-derivation tree dt.

It remains to be shown that dt is rational. Due to (1) and (2), for all $v \in V(dt_i)$ with $|v| \leq i$ it holds that $dt_i(v) = \delta(\mathit{orig}_{dt_i}(v))(\lambda)$ (i.e., $dt_i(v)$ is the root symbol of $\delta(\mathit{orig}_{dt_i}(v))$). Consequently, $dt(v) = \delta(\mathit{orig}_{dt}(v))(\lambda)$ for all $v \in V(dt)$. However, by the definition of derivation trees, $\mathit{orig}_{dt}(v)$ and $dt(v)$ together uniquely determine $\mathit{orig}_{dt}(vj)$ for all $j \in \mathbb{N}$ such that $vj \in V(td)$, which means that dt/v is uniquely determined by $\mathit{orig}_{dt}(v)$. Therefore, $dt/u = dt/v$ for all $u, v \in V(dt)$ satisfying $\mathit{orig}_{dt}(u) = \mathit{orig}_{dt}(v)$. This shows that dt is rational, because t is rational and every tree $\mathit{orig}_{dt}(v)$ ($v \in V(dt)$) has the form qt', where $q \in Q$ and $t' = t/v'$ for some $v' \in V(t)$. Of course, the rationality of dt implies that $\mathrm{res}(dt)$ is rational, which completes the proof of the claim and, hence, the proof of the lemma. ■

We can now show that rationality of a fractal means that it is generated by a tree-based cMRFS, and vice versa.

7.4 Theorem A picture p is a rational fractal if and only if there is a tree-based cMRFS $\mathcal{G}$ such that $\mathrm{L}_{\mathrm{inf}}(\mathcal{G}) = \{p\}$.

Proof.

(⇒) Let $t \in \mathcal{T}_{\Sigma_\mathcal{P}}$ be an infinite rational tree such that $p = \mathrm{val}_\mathcal{P}(t)$, and let $\{t_1, \dots, t_k\} = \{t' \in \mathcal{T}_{\Sigma_\mathcal{P}} \setminus \mathcal{F}_{\Sigma_\mathcal{P}} \mid t' = t/v \text{ for some } v \in V(t)\}$, where $t_1 = t$. In order to construct an appropriate tree-based cMRFS $\mathcal{G} = (G, \mathcal{A})$, define a signature Σ and a Σ-algebra $\mathcal{A}$ as follows. $\Sigma^{(0)} = \{a\}$ where $a_\mathcal{A} = p_0$ for some arbitrary element p_0 of $\mathbb{P}_0$. Furthermore, for every $i \in [k]$, Σ contains a symbol F_i whose rank and interpretation in $\mathcal{A}$ are determined as follows. Obviously, t_i can be written in the form $t_i^0[\![t_{\mu_i(1)}, \dots, t_{\mu_i(l_i)}]\!]$ for some tree $t_i^0 \in \mathcal{F}_{\Sigma_\mathcal{P}}(X_{l_i})$, where $l_i \in \mathbb{N}$ and $\mu_i(j) \in [k]$ for all $j \in [l_i]$. (Notice that $l_i \geq 1$ since t_i is infinite.) Now, let the rank of F_i be l_i and define $F_{i\mathcal{A}} = \mathrm{op}_\mathcal{P}(t_i^0)$.

Let $td = (\Sigma_{\mathrm{succ}}, \Sigma, \{q_1, \dots, q_k\}, R, q_1)$ where R contains the rules

$$\begin{aligned} q_i 0 &\to a \\ q_i s &\to F_i[q_{\mu_i(1)}, \dots, q_{\mu_i(l_i)}] \end{aligned}$$

for every $i \in [k]$. It follows immediately from this construction that $td(s^\infty) = \{t'\}$ for some tree $t' \in \mathcal{F}_\Sigma$ satisfying $h(t') = t$, where h is the homomorphism defining $\mathcal{A}$, i.e., the one given by $h(F_i) = t_i^0$ and $h(a) = p_0$. Consequently, $\mathcal{G} = (G, \mathcal{A})$ with $G = (td, g_{\mathrm{succ}})$ satisfies $\mathrm{L}_{\mathrm{inf}}(\mathcal{G}) = \{p\}$. Furthermore, td is deterministic. It is also nondeleting since $l_i \geq 1$

(as mentioned above), which proves that $\mathcal{G}$ is a tree-based cMRFS, as required.

($\Leftarrow$) Let $\mathrm{L}_{\mathrm{inf}}(\mathcal{G}) = \{p\}$ for some tree-based cMRFS $\mathcal{G}$. By definition, this means $p = \mathrm{val}_{\mathcal{P}}(h(td(s^{\infty})))$ for some total, deterministic, and nondeleting td transducer td and a tree homomorphism h (which, by the very definition of tree homomorphisms, is also a total, deterministic, and nondeleting td transduction). By the nondeleting property of td and h, $h(td(s^{\infty}))$ is infinite since s^{∞} is, and by Lemma 7.3 (applied twice, where L is the set of all trees over the relevant output signature) $h(td(s^{\infty}))$ is rational, so p is a rational fractal, as claimed. ∎

The next lemma directly leads to the main theorem of this section. The lemma states that every fractal in $\mathrm{L}_{\mathrm{inf}}(\mathcal{G})$ (where $\mathcal{G}$ is a $\mathcal{P}$-interpreted td generator) lies near to a rational fractal.

7.5 Lemma Let $\mathcal{G}$ be a $\mathcal{P}$-interpreted td generator and let $p \in \mathrm{L}_{\mathrm{inf}}(\mathcal{G})$. For every $\varepsilon > 0$ there is a rational fractal $p' \in \mathrm{L}_{\mathrm{inf}}(\mathcal{G})$ such that $d_{\mathrm{H}}(p, p') \leq \varepsilon$.

Proof. The proof of the lemma consists mainly of the verification of two claims. The first claim states that the inverse image of a regular tree language under a td transduction is regular (see also [Eng77, Lemma 1.2] and the remark following its proof).

Claim 1. Let $td = (\Sigma, \Sigma', Q, R, q_0)$ be a td transducer and $L \subseteq \mathcal{T}_{\Sigma'}$ a regular tree language. Then the tree language

$$td^{-1}(L) = \{s \in \mathcal{T}_{\Sigma} \mid td(s) \cap L \neq \emptyset\}$$

is regular.

In order to prove Claim 1 it suffices to note the following well-known facts (the first one of which was already used in the proof of Lemma 7.3):

(1) There is a td transducer td' such that $td'(s) = td(s) \cap L$ for all $s \in \mathcal{T}_{\Sigma}$.

(2) For every td transducer td', the set $dom(td')$ of all trees s such that $td'(s) \neq \emptyset$, is regular.

By (1) and (2) $dom(td') = td^{-1}(L)$ is regular, as claimed.

The second claim is already almost everything which is needed in order to prove the lemma itself.

Claim 2. Let τ be a composition of finitely many td transductions, and let L be a regular tree language. If $\tau(\mathcal{T}_{\Sigma_{\mathrm{succ}}}) \cap L \neq \emptyset$ then $\tau(\mathcal{T}_{\Sigma_{\mathrm{succ}}}) \cap L$ contains a rational tree.

In order to prove Claim 2, proceed by induction on the number n of td transductions τ is composed of. For $n = 0$ there is nothing to show because $\mathcal{T}_{\Sigma_{\mathrm{succ}}}$ contains only rational trees. Now, let $\tau = td \circ \tau'$ for some td transducer td and assume that the claim is known to hold for τ'. If $\tau(\mathcal{T}_{\Sigma_{\mathrm{succ}}}) \cap L \neq \emptyset$ then there is a tree $t_0 \in \tau'(\mathcal{T}_{\Sigma_{\mathrm{succ}}})$ such that $td(t_0) \cap L \neq \emptyset$. By Claim 1 the tree language $td^{-1}(L)$ is regular. Since $t_0 \in td^{-1}(L)$, the induction hypothesis yields a rational tree $t_0' \in \tau'(\mathcal{T}_{\Sigma_{\mathrm{succ}}}) \cap td^{-1}(L)$. This implies that $td(t_0') \cap L \neq \emptyset$. As t_0' is rational, Lemma 7.3 says that $td(t_0') \cap L$ contains a rational tree. This finishes the proof of Claim 2 because $td(t_0') \subseteq \tau(\mathcal{T}_{\Sigma_{\mathrm{succ}}})$.

Now, in order to prove the lemma, let $\mathcal{G} = (G, \mathcal{A})$ for some td generator G and a Σ-algebra $\mathcal{A}$. By the continuity of $\mathrm{val}_{\mathcal{A}}$ (see Lemma 5.4) it suffices to show that, for every tree $t \in \mathrm{L}_{\mathrm{inf}}(G)$ and every $m \in \mathbb{N}$, there is an infinite rational tree $t' \in \mathrm{L}_{\mathrm{inf}}(G)$ satisfying $\mathrm{eq}(t, t') \geq m$. By Lemma 4.4 it holds that $\mathrm{L}(G) = \tau(\mathcal{T}_{\Sigma_{\mathrm{succ}}})$ for a composition τ of finitely many td transducers. In other words, we have $t \in \tau(\mathcal{T}_{\Sigma_{\mathrm{succ}}}) \cap I$, where $I = \mathcal{T}_\Sigma \setminus \mathcal{F}_\Sigma$. The set I is in fact regular: it is generated by the regular tree grammar $g = (\{S, A\}, \Sigma, P, S)$, where P contains, for every $f \in \Sigma^{(n)}$ ($n \in \mathbb{N}$), all rules

$$S \to f[\underbrace{A, \ldots, A}_{i-1}, S, \underbrace{A, \ldots, A}_{n-i}]$$

such that $i \in [n]$, as well as the rule $A \to f[A, \ldots, A]$. Another regular tree language is $L_{t,m} = \{t' \in \mathcal{T}_\Sigma \mid \mathrm{eq}(t, t') \geq m\}$. To see this, decompose t into $t_0[\![t_1, \ldots, t_l]\!]$, where $V(t_0) = \{v \in V(t) \mid |v| \leq m\}$ and $t_0(v) \in X \iff |v| = m$ for all $v \in V(t_0)$ (i.e., $t_1, \ldots, t_l$ are the subtrees of t rooted at depth m). Then, $L_{t,m}$ is generated by the regular tree grammar $g_{t,m} = (\{S, A\}, \Sigma, \{S \to t_0[\![A, \ldots, A]\!]\} \cup \{A \to f[A, \ldots, A] \mid f \in \Sigma\}, S)$.

Due to the regularity of I and $L_{t,m}$ their intersection is regular as well. Furthermore, $t \in \tau(\mathcal{T}_{\Sigma_{\mathrm{succ}}}) \cap I \cap L_{t,m}$, which by Claim 2 implies that $\tau(\mathcal{T}_{\Sigma_{\mathrm{succ}}}) \cap I \cap L_{t,m}$ contains a rational tree t'. In other words, t' is a rational tree in $\mathrm{L}_{\mathrm{inf}}(G)$ such that $\mathrm{eq}(t, t') \geq m$, as required. ■

As a direct consequence of Lemma 7.5 the main theorem of this section, which extends Theorem 7.4, is obtained.

7.6 Theorem Let $\mathcal{G}$ be a $\mathcal{P}$-interpreted td generator. For every irrational fractal $p \in \mathrm{L}_{\mathrm{inf}}(\mathcal{G})$ there is a converging sequence $(p_i)_{i\in\mathbb{N}}$ of rational fractals $p_i \in \mathrm{L}_{\mathrm{inf}}(\mathcal{G})$ such that $\lim p_i = p$. In particular, if $\mathrm{L}_{\mathrm{inf}}(\mathcal{G})$ is finite then every picture $p \in \mathrm{L}_{\mathrm{inf}}(\mathcal{G})$ is a rational fractal.

Proof. The main statement is an immediate consequence of Lemma 7.5. Furthermore, if $p = \lim p_i$ is irrational for a converging sequence $(p_i)_{i\in\mathbb{N}}$ of rational fractals, then $\{p_i \mid i \in \mathbb{N}\}$ must necessarily be infinite (since $p \neq p_i$ implies $d_{\mathrm{H}}(p, p_i) > 0$ for all $i \in \mathbb{N}$, yet $(p_i)_{i\in\mathbb{N}}$ converges to p). Therefore, $\mathrm{L}_{\mathrm{inf}}(\mathcal{G})$ cannot contain an irrational fractal unless it is infinite. ■

8 Conclusion

In this paper, an approach has been proposed which combines formal language theory and fractal geometry through the notion of $\mathcal{P}$-interpreted top-down tree generators. These devices generate languages of fractals by deriving infinite trees and interpreting them as expressions which denote pictures.

It was shown in [Dre00] that $\mathcal{P}$-interpreted td generators are able to simulate collage grammars [HK91, DK99] and mutually recursive function systems ([CD93a, CD93b], see also the previous section). However, while collage grammars consist of pictures with finite descriptions (i.e., they do not contain fractals), a mutually recursive function system is a deterministic device which generates just a single fractal. The approach presented here combines the nondeterministic nature of grammars with the fractal detailedness of generated objects.

The so-called approximating td generators turned out to be particularly well behaved as the fractals they generate are the limits of refinement sequences—converging sequences in the generated language which can be constructed by a simple syntactic procedure. As a consequence, the generated fractals can be approximated by pictures which are themselves elements of the generated language.

It remains to be mentioned that the examples shown in this paper were produced using TREEBAG, a Java implementation of the ideas presented here, which can be downloaded from http://www.informatik.uni-bremen.de/~drewes/treebag. The distribution contains many examples, including those presented in Section 6 in this paper.

References

[Bak78] Brenda S. Baker. Tree transducers and tree languages. *Information and Control*, 37:241–266, 1978.

[Bar93] Michael Barnsley. *Fractals Everywhere*. Academic Press, Boston, 2nd edition, 1993.

[CD93a] Karel Culik II and Simant Dube. Affine automata and related techniques for generation of complex images. *Theoretical Computer Science*, 116:373–398, 1993.

[CD93b] Karel Culik II and Simant Dube. L-systems and mutually recursive function systems. *Acta Informatica*, 30:279–302, 1993.

[DH93] Jürgen Dassow and Friedhelm Hinz. Decision problems and regular chain code picture languages. *Discrete Applied Mathematics*, 45:29–49, 1993.

[DK99] Frank Drewes and Hans-Jörg Kreowski. Picture generation by collage grammars. In H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, editors, *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 2, chapter 11, pages 397–457. World Scientific, 1999.

[Dre00] Frank Drewes. Tree-based picture generation. *Theoretical Computer Science*, 2000. To appear.

[Edg90] Gerald A. Edgar. *Measure, Topology, and Fractal Geometry*. Springer, New York, 1990.

[Eng75] Joost Engelfriet. Bottom-up and top-down tree transformations—a comparison. *Mathematical Systems Theory*, 9(3):198–231, 1975.

[Eng77] Joost Engelfriet. Top-down tree transducers with regular look-ahead. *Mathematical Systems Theory*, 10:289–303, 1977.

[Eng82] Joost Engelfriet. Three hierarchies of transducers. *Mathematical Systems Theory*, 15:95–125, 1982.

[ERS80] Joost Engelfriet, Grzegorz Rozenberg, and Giora Slutzki. Tree transducers, L systems, and two-way machines. *Journal of Computer and System Sciences*, 20:150–202, 1980.

[FV98] Zoltán Fülöp and Heiko Vogler. *Syntax-Directed Semantics: Formal Models Based on Tree Transducers*. Springer, 1998.

[GS97] Ferenc Gécseg and Magnus Steinby. Tree languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages. Vol. III: Beyond Words*, chapter 1, pages 1–68. Springer, 1997.

[HK91] 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. Springer, 1991.

[KHKT99] Renate Klempien-Hinrichs, Hans-Jörg Kreowski, and Stefan Taubenberger. Correct translation of mutually recursive function systems into TOL collage grammars. In G. Ciobanu and Gh. Paun, editors, *Proc. Fundamentals of Computation Theory (FCT'99)*, volume 1684 of *Lecture Notes in Computer Science*, pages 350–361, 1999.

[Man83] Benoit B. Mandelbrot. *The Fractal Geometry of Nature*. W.H. Freeman and Company, New York, 1983.

[MRW82] Hermann A. Maurer, Grzegorz Rozenberg, and Emo Welzl. Using string languages to describe picture languages. *Information and Control*, 54:155–185, 1982.

[PJS92] Heinz-Otto Peitgen, Hartmut Jürgens, and Dietmar Saupe. *Chaos and Fractals. New Frontiers of Science*. Springer-Verlag, New York, 1992.

[PL90] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag, New York, 1990.

[Rou70] William C. Rounds. Mappings and grammars on trees. *Mathematical Systems Theory*, 4:257–287, 1970.

[Roz73] Grzegorz Rozenberg. Extensions of tabled 0L systems and languages. *International Journal of Computer and Information Sciences*, 2:311–334, 1973.

[RS97] Grzegorz Rozenberg and Arto Salomaa, editors. *Handbook of Formal Languages*, volume 1–3. Springer, 1997.

[Tha70] James W. Thatcher. Generalized^2 sequential machine maps. *Journal of Computer and System Sciences*, 4:339–367, 1970.