Journal of Universal Computer Science, vol. 5, no. 9 (1999), 599-609
submitted: 10/6/99, accepted: 15/7/99, appeared: 28/9/99

Splicing on Trees: the Iterated Case

George Rahonis
(30 Kyprou st. 57019 Perea (Thessaloniki), Greece,
grahonis@ccf.auth.gr)

Abstract: The closure under the splicing operation with finite and recognizable sets of rules, is extended to the family of generalized synchronized forests. Moreover, we investigate the application of the iterated splicing on known families of forests. Interesting properties of this operation are established.

Key Words: DNA Computing, Trees languages

Category: F.1.1, F.4.0, F.4.3

1 Introduction

L. Adleman in his seminal paper [Adl. 94] in 1994, gave birth to a new direction in research, as well as to a new way of thinking on computer science. Currently, DNA computing is one of the most interesting research fields in order to construct new types of computers. From the practical and also the theoretical point of view, central role in the DNA computing plays the splicing operation.This operation has been widely studied for formal languages [Head et al. 97, Păun et al. 98]. On the other hand, the importance of the tree structure in computer science is very well-known, thus the investigation of splicing-like operations on trees is of great interest.

In [Boz., Rah. 98], we defined a splicing operation on trees and we proved the closure of the families REC of recognizable, OCF of one counter and ALG of algebraic forests, under this operation with finite and recognizable sets of rules.

In this paper, we extend the above result to the family GST of generalized synchronized forests. Next, we define the iterated splicing on trees which has an unexpected property: it terminates after a finite number of steps. The families REC, OCF, ALG and GST are proved to be closed under this operation with finite and recognizable sets of rules. Moreover, we show that the height and the branches of each forest remain unchanged under iterated splicing.

2 Preliminaries

We assume the reader to be familiar with basics on formal languages [Sal. 73], as well as on DNA computing [Head et al. 97, Păun et al. 98], and tree language theory [Géc., Ste. 84, Géc., Ste. 97]. Here, we recall some notations and definitions that will be used in the sequel.

The power set of a set V, is written $P(V)$. V^* denotes the free monoid generated by V and λ is the empty word. If n is a positive natural number, then $[n] = \{1, ..., n\}$.

We write Σ for a *finite ranked alphabet* and we denote by Σ_n, the set of symbols of Σ whose rank equals to n. A symbol $\sigma \in \Sigma$ might have more than one ranks. The degree of Σ is denoted by $\deg(\Sigma)$ and is the maximal number n such that $\Sigma_n \neq \emptyset$ and $\Sigma_k = \emptyset$ for each $k > n$.

Let $X = \{x_1, x_2, ...\}$ be a countably infinite set of variables and $X_m = \{x_1, ..., x_m\}$, for $m \geq 0$. The set of all trees over Σ, indexed by the variables $x_1, ..., x_m$ is denoted by $T_\Sigma(X_m)$. A *forest* F is a set of trees over an alphabet Σ, possibly with variables, that is $F \subseteq T_\Sigma(X_m)$, $m \geq 0$.

For $t \in T_\Sigma(X_n)$ and $t_1, ..., t_n \in T_\Sigma(X_m)$, we write $t(t_1, ..., t_n)$ for the result of substituting t_i for x_i in t.

There is also another common way to define trees. A *tree domain* D [Sal. 94], is a subset of N_+^* (N denotes the set of natural numbers), which satisfies the following conditions:

(i) If $u \in D$ then $v \in D$ for each prefix v of u.

(ii) For each $u \in D$, there exists $i \in N$, such that $uj \in D$ for each $j \in [i]$ (if u has no sons then $i = 0$).

For a set V, a $V-$*labeled tree* is a mapping $t : D \to V$, where D is a tree domain. We call the elements of D the *nodes* of the tree and we denote D by $dom(t)$. A node $u \in D$ is labeled by $t(u) \in V$. If v is a proper prefix of $u \in dom(t)$, then u is called a *successor* of v.

In this way, each tree $t \in T_\Sigma(X)$ can be considered as a $\Sigma \cup X$-labeled tree $t : dom(t) \to \Sigma \cup X$, such that each node of t labeled by an element of rank $n \geq 0$, has exactly n immediate successors (sons) and variables don't have successors. The nodes of a tree t labeled by constants are called *leaves,* and the set of all leaves of a tree t is denoted by $leaf(t)$.

For a V-labeled tree, a *path* of t is a word

$path(t, u_m) = t(u_1)...t(u_m) \in V^+$

where $u_1 = \lambda$, $u_m \in leaf(t)$ and u_{i+1} is an immediate successor of u_i, $i = 1, ..., m-1$. Then

$path(t) = \{path(t, u) \ / \ u \in leaf(t)\}$.

Let now, Σ, Γ be ranked alphabets and $\deg(\Sigma) = n$. Assume that for each $k \in [n]$, there is a mapping $h_k : \Sigma_k \to T_\Gamma(X_k)$. Then the mappings h_k, $k \in [n]$ are organized into a *tree homomorphism* $h : T_\Sigma \to T_\Gamma$, defined inductively by:

$h(\sigma(t_1, ..., t_k)) = h_k(\sigma)(h(t_1), ..., h(t_k))$, for $k \geq 0$, $\sigma \in \Sigma_k$ and $t_1, ..., t_k \in T_\Sigma$.

A tree homomorphism $h : T_\Sigma \to T_\Gamma$ is called *linear,* if for each $\sigma \in \Sigma_k$, $k \geq 0$, $h_k(\sigma)$ is a linear tree, that is each variable from X_k appears at most

once in $h_k(\sigma)$. Moreover, a linear tree homomorphism $h : T_\Sigma \to T_\Gamma$ is called *alphabetic,* if for each $\sigma \in \Sigma_k$, $k \geq 0$, either

$h_k(\sigma) = \gamma(x_{i_1}, ..., x_{i_m})$, $\gamma \in \Gamma_m$, or

$h_k(\sigma) = x_n$, $1 \leq n \leq k$.

A *nondeterministic top-down finite tree automaton* is a four-tuple $\mathcal{A} = (\Sigma, Q, Q_0, \alpha)$, with Σ the finite ranked alphabet of input symbols, Q the finite set of states, $Q_0 \subseteq Q$ is the set of initial states and α is the family of state transitions which is defined by the mappings $\alpha_\sigma : Q \to P(\ Q^n)$, where $\sigma \in \Sigma_n$, $n \geq 0$. If $\sigma \in \Sigma_0$, then $\alpha_\sigma \subseteq Q$.

A *computation* of $\mathcal{A} = (\Sigma, Q, Q_0, \alpha)$ on an input tree $t \in T_\Sigma$, is a Q-labeled tree $r : dom(t) \to Q$, satisfying the following conditions:

(i) $r(\lambda) \in Q_0$.

(ii) Suppose that $u \in dom(t)$ has m successors $u_1, ..., u_m$ and $t(u) = \sigma \in \Sigma_m$. Then $(r(u_1), ..., r(u_m)) \in \alpha_\sigma(r(u))$.

(iii) If $u \in leaf(t)$ and $t(u) = \sigma\ (\in \Sigma_0)$, then $r(u) \in \alpha_\sigma$.

The set of all computations of $\mathcal{A}$ on t, is denoted by $com_\mathcal{A}(t)$ and the forest recognized by $\mathcal{A}$ is

$$L(\mathcal{A}) = \{t \in T_\Sigma \ / \ com_\mathcal{A}(t) \neq \emptyset\}.$$

It is well known, that the family of forests recognized by nondeterministic top-down tree automata, is the family of recognizable forests and is denoted by REC [Géc., Ste. 84, Géc., Ste. 97].

A *nondeterministic synchronized tree automaton (nsta)* [Sal. 94] is a top-down finite tree automaton $\mathcal{A} = (\Sigma, Q, Q_0, \alpha)$, where Q is of the form

$$Q = (Q_1 \times \{\lambda\}) \cup (Q_2 \times S)$$

(recall that λ denotes the empty word).

The set S is the *synchronization alphabet* and elements of S are called *synchronizing symbols* (abbreviated as sync-symbols).We define a morphism $h_\mathcal{A} : Q^* \to S^*$ by setting $h_\mathcal{A}((q_1, \lambda)) = \lambda$ and $h_\mathcal{A}((q_2, s)) = s$, for $q_i \in Q_i$, $i = 1, 2$ and $s \in S$. The set of *synchronized computations* of $\mathcal{A}$ on a tree $t \in T_\Sigma$, is

$$scom_\mathcal{A}(t) = \{r \in com_\mathcal{A}(t) \ / \ \forall u, w \in path(r),\ h_\mathcal{A}(u) \approx_{pr} h_\mathcal{A}(w)\}$$

where $h_\mathcal{A}(u) \approx_{pr} h_\mathcal{A}(w)$ means that one of the words $h_\mathcal{A}(u)$ and $h_\mathcal{A}(w)$ is a prefix of the other.

The *nondeterminitic synchronized forest* recognized by $\mathcal{A}$, is

$$L_S(\mathcal{A}) = \{t \in T_\Sigma \ / \ scom_\mathcal{A}(t) \neq \emptyset\}.$$

The family of all nondeterministic synchronized forests is denoted by NST.

We also define the notion of a *generalized synchronized tree automaton (gsta)* [Rah., Sal. 98]. It is an nsta which is allowed to make $\lambda - transitions$, that is to change the state in a fixed node. Let $\mathcal{A}$ be a gsta with state set $Q = (Q_1 \times \{\lambda\}) \cup (Q_2 \times S)$ and let $t \in T_\Sigma$. A *computation of* $\mathcal{A}$ *on* t is a tree τ, where each node $u \in dom(\tau) = dom(t)$ is labeled by an element of $Q_1 \times \{\lambda\}$ or by a pair

$$(q, s_1 s_2 ... s_n),$$

where q is the last element of $Q_1 \cup Q_2$ appearing at u in the computation τ and $s_1 s_2 ... s_n$, $n \geq 1$, is the sequence of sync-symbols appearing at node u in τ. (If $\mathcal{A}$ makes λ-transitions at node u, it is possible that $n \geq 2$). Now, the morphism $h_\mathcal{A}$ is defined by $h_\mathcal{A}((q, \lambda)) = \lambda$, and $h_\mathcal{A}((q, s_1 s_2 ... s_n)) = s_1 s_2 ... s_n$, for $q \in Q_1 \cup Q_2$, $s_j \in S$, $j \in [n]$. The forest $L_S(\mathcal{A})$ recognized by $\mathcal{A}$, is defined as for nondeterministic synchronized tree automata. The family of forests which are recognized by all gsta is denoted by GST.

Finally, we denote by FIN the family of finite forests, by ALG the family of *algebraic forests* [Gue. 83], and by OCF the family of *one counter forests* [Boz., Rah. 94].

In [Boz. 92], the notion of an alphabetic tree transduction was defined, and used as the basis to build the AFL theory to the tree case [Boz., Rah. 94, Rah. 200x].

Definition 1. A tree transduction $\tau : T_\Sigma \to P(T_\Gamma)$ is called *alphabetic,* if there exists a ranked alphabet Δ, a recognizable forest $R \subseteq T_\Delta$, and two alphabetic homomorphisms $\varphi : T_\Delta \to T_\Sigma$, $\psi : T_\Delta \to T_\Gamma$, such that $\#\tau = \{(\varphi(t), \psi(t)) \ / \ t \in R\}$, where $\#\tau$ denotes the graph of τ.

Then, an *alphabetic cone* is a family of forests closed under alphabetic transductions. Moreover, *a sheaf of forests* is an alphabetic cone closed under the rational tree operations: *union*, *top-catenation*, α-*product* and *a-star* [Boz., Rah. 94].

Proposition 2. *[Boz., Rah. 94, Rah., Sal. 98] The classes REC of recognizable, ALG of algebraic, OCF of one counter, and GST of generalized synchronized forests are sheaves.*□

3 H schemes on trees

The material of this section can be found in [Boz., Rah. 98].

Firstly, we recall the notion of the shift operation.

Consider a finite ranked alphabet Σ and let $\Gamma = \Sigma \cup \{\delta, \sigma_1, ..., \sigma_n\}$, with $rank(\delta) = n$ and $rank(\sigma_i) = m_i$, $i \in [n]$. The (n+1)-tuple $(\delta, \sigma_1, ..., \sigma_n)$ is called *a connected list in a forest* $F \subseteq T_\Gamma$, if whenever one of the above symbols appears in a tree $t \in F$, then it appears in the fork $\delta(\sigma_1, ..., \sigma_n)$, that is none of these symbols appears in trees of F out of the above fork.

Let now $\Delta = \Sigma \cup \{\sigma\}$, with $rank(\sigma) = \sum_{i=1}^{n} m_i = m$.

The *shift operation on the connected list* $(\delta, \sigma_1, ..., \sigma_n)$, denoted $sh : T_\Gamma \to T_\Delta$, is the inverse linear tree homomorphism $h : T_\Delta \to T_\Gamma$, which is defined by
-$h_k(\gamma(x_1, ..., x_k)) = \gamma(x_1, ..., x_k)$, for each $\gamma \in \Sigma_k, k \geq 0$,
-$h_m(\sigma(x_1, ..., x_m)) =$
$\delta(\sigma_1(x_1, ..., x_{m_1}), \sigma_2(x_{m_1+1}, ..., x_{m_1+m_2}), ..., \sigma_n(x_{m_1+...+m_{n-1}+1}, ..., x_m))$.
Thus, we write
$sh(\delta(\sigma_1(x_1, ..., x_{m_1}), \sigma_2(x_{m_1+1}, ..., x_{m_1+m_2}), ..., \sigma_n(x_{m_1+...+m_{n-1}+1}, ..., x_m))) =$
$\sigma(x_1, ..., x_m)$
and
$sh(\gamma(x_1, ..., x_k)) = \gamma(x_1, ..., x_k)$ for each $\gamma \in \Sigma_k$, $k \geq 0$.

The shift operation can obviously be extended to more than one connected lists.

We give now the definition of an *H tree scheme.*

Definition 3. Let Σ be a finite ranked alphabet and $@, \#, \$$ be three new symbols not belonging to Σ, with $rank(@) = 3, 4, 5, 6, 7$ and $rank(\#) = rank(\$) = 0$. An *H tree scheme* is a pair $f = (\Sigma, R)$, with R the forest of *rules,* $R \subseteq @(\omega, \#, \omega, \$, \omega, \#, \omega)$, where ω stands for T_Σ or for nothing.

Thus a tree $r \in R$ is of the form $@(s_1, \#, s_2, \$, s_3, \#, s_4)$, $s_i \in T_\Sigma$, $i = 1, 2, 3, 4$, where some s_i may be missed, for example $@(s_1, \#, s_2, \$, \#, s_4)$, $@(\#, s_2, \$, \#)$ can be trees in R.

R can be either finite or infinite.

An H tree scheme $f = (\Sigma, R)$ is called of $\mathcal{F}$ *type,* if R belongs to the family of forests $\mathcal{F}$.

Let now $t, p, z \in T_\Sigma$. We say that z is obtained by *splicing* t, p and we write $(t, p) \to_r z$, if $t = \sigma(t_1, ..., t_{i-1}, s_1, s_2, t_{i+2}, ..., t_n)$, $p = \sigma(p_1, ..., p_{i-1}, s_3, s_4, p_{i+2}, ..., p_n)$, $z = \sigma(t_1, ..., t_{i-1}, s_1, s_4, p_{i+2}, ..., p_n)$, with $\sigma \in \Sigma_n$, $n > 0$, $t_1, ..., t_{i-1}, t_{i+2}$, $..., t_n, p_1, ..., p_{i-1}, p_{i+2}, ..., p_n \in T_\Sigma$ and $r = @(s_1, \#, s_2, \$, s_3, \#, s_4)$ is a rule in R.

Thus splicing on trees, means "vertically cutting" of two trees with the same root, at certain indices under the presence of a rule, and "connecting" the leftmost part of the first tree with the rightmost part of the second tree.

For $F \subseteq T_\Sigma$, and $f = (\Sigma, R)$ an H tree scheme, we write

$$f(F) = \{z \in T_\Sigma \ / \ \exists\ t, p \in F, \exists\ r \in R, \text{ such that } (t,p) \rightarrow_r z\}$$

and

$$f^{(1)}(F) = F \cup f(F).$$

If $\mathcal{F}_1, \mathcal{F}_2$ are two families of forests, we define the families

$$S(\mathcal{F}_1, \mathcal{F}_2) = \{f(F_1) \ / \ F_1 \in \mathcal{F}_1,\ f = (\Sigma, R) \text{ is an H tree scheme with } R \in \mathcal{F}_2\}$$

and

$$S_1(\mathcal{F}_1, \mathcal{F}_2) = \{f^{(1)}(F_1) \ / \ F_1 \in \mathcal{F}_1,\ f = (\Sigma, R) \text{ is an H tree scheme with } R \in \mathcal{F}_2\}.$$

Theorem 4. *If $\mathcal{F}$ is a sheaf of forests closed under the shift operation on connected lists, then $S_1(\mathcal{F}, FIN) = \mathcal{F}$ and $S_1(\mathcal{F}, REC) = \mathcal{F}$.*□

The next lemma gave an interesting application of the above result.

Lemma 5. *The families of forests REC, ALG and OCF are closed under the shift opeartion on connected lists.*□

Corollary 6. *It holds*
$S_1(REC, FIN) = REC,\ \ S_1(ALG, FIN) = ALG,\ \ S_1(OCF, FIN) = OCF$ *and*
$S_1(REC, REC) = REC,\ \ S_1(ALG, REC) = ALG,\ \ S_1(OCF, REC) = OCF$.□

The reader can easily understand that the result of theorem 4 cannot be "linearized" for the word case, because of the shift operation, which has no meaning on strings.

Let now, $f = (\Sigma, R)$ be an H tree scheme. By passing to the yield of R, we obtain an H string scheme $\sigma = (\Sigma_0, yield(R))$ [Head et al. 97].

For $F \subseteq T_\Sigma$, the inclusion below is in general strict

$$yield(f(F)) \subseteq \sigma(yield(F)).$$

In the case of recognizable forests, the next proposition holds

Proposition 7. $yield(S_1(REC, FIN)) = S_1(yield(REC), yield(FIN))$, *where the symbol S_1 has an analogous meaning in the word case.*□

4 Results

As a first result in this paper, we extend lemma 5 to the class GST of generalized synchronized forests.

Lemma 8. *GST is closed under the shift operation on connected lists.*

Proof. Consider a generalized synchronized forest $F \subseteq T_\Gamma$, with $\Gamma = \Sigma \cup \{\delta, \sigma_1, ..., \sigma_n\}$, $rank(\delta) = n$, $rank(\sigma_i) = m_i$, $i \in [n]$, such that $(\delta, \sigma_1, ..., \sigma_n)$ is a connected list in F.

Let $sh : T_\Gamma \to T_\Delta$, $\Delta = \Sigma \cup \{\sigma\}$, $rank(\sigma) = \sum_{i=1}^{n} m_i = m$, be the shift operation on the connected list $(\delta, \sigma_1, ..., \sigma_n)$, and $\mathcal{A} = (\Gamma, Q, Q_0, \alpha)$ be a gsta, with $Q = (Q_1 \times \{\lambda\}) \cup (Q_2 \times S)$, and $F = L_S(\mathcal{A})$.

For $(q, s) \in Q$, let $((q_1, s_1), ..., (q_n, s_n)) \in \alpha_\delta((q, s))$, with $(q_i, s_i) \in Q$, $i \in [n]$. Now, for each (q_i, s_i), $i \in [n]$ there is a finite number of finite sequences of λ-moves

$\alpha_\lambda((q_i, s_i)) \ni (q_{i_1}, s_{i_1})$, $\alpha_\lambda((q_{i_1}, s_{i_1})) \ni (q_{i_2}, s_{i_2})$, ..., $\alpha_\lambda((q_{i_{k_i-1}}, s_{i_{k_i-1}})) \ni (q_{i_{k_i}}, s_{i_{k_i}})$,

such that $\alpha_{\sigma_i}((q_{i_{k_i}}, s_{i_{k_i}})) \neq \emptyset$, and all the states that appear at each sequence are pairwise disjoint.

Because of the definition of the connected list in F, we can assume that $\alpha_\gamma((q_j, s_j)) = \emptyset$, for each $j \in \{i, i_1, ..., i_{k_i}\}$, $i \in [n]$, and $\gamma \in \Sigma$.

We substitute each state which appears in each of the above sequences, with m_i, $i \in [n]$ different copies i.e.

$$(q_i^1, s_i), (q_{i_1}^1, s_{i_1}), ..., (q_{i_{k_i}}^1, s_{i_{k_i}}),$$
$$\vdots$$
$$(q_i^{m_i}, s_i), (q_{i_1}^{m_i}, s_{i_1}), ..., (q_{i_{k_i}}^{m_i}, s_{i_{k_i}}).$$

Let Q' be the so obtained set of states, and α' the new family of state transitions, where

- $(q_{i_1}^{l_i}, s_{i_1}) \in \alpha'_\lambda((q_i^{l_i}, s_i))$, whenever $(q_{i_1}, s_{i_1}) \in \alpha_\lambda((q_i, s_i))$,

for each $i \in [n]$, $l_i \in [m_i]$,

- $(q_{i_{j+1}}^{l_i}, s_{i_{j+1}}) \in \alpha'_\lambda((q_{i_j}^{l_i}, s_{i_j}))$, whenever $(q_{i_{j+1}}, s_{i_{j+1}}) \in \alpha_\lambda((q_{i_j}, s_{i_j}))$,

for each $i \in [n]$, $j \in [k_i - 1]$, $l_i \in [m_i]$,

- $\alpha'_{\sigma_i}((q_{i_{k_i}}^{l_i}, s_{i_{k_i}})) = \alpha_{\sigma_i}((q_{i_{k_i}}, s_{i_{k_i}}))$, for each $i \in [n]$, $l_i \in [m_i]$,
- $\alpha'_\gamma((q, s)) = \alpha_\gamma((q, s))$, for each $(q, s) \in Q$, $\gamma \in \Sigma \cup \{\lambda\}$.

Obviously, for the gsta $\mathcal{A}' = (\Gamma, Q', Q_0, \alpha')$, it holds $L_S(\mathcal{A}') = L_S(\mathcal{A})$.

Next, we consider the gsta $\mathcal{B} = (\Delta, Q', Q_0, \beta)$, with the family β defined in the following way:
- $\beta_\gamma((q,s)) = \alpha'_\gamma((q,s))$, for $(q,s) \in Q'$, $\gamma \in \Sigma \cup \{\lambda\}$,
- $((q_1^1, s_1), ..., (q_1^{m_1}, s_1), ..., (q_n^1, s_n), ..., (q_n^{m_n}, s_n)) \in \beta_\sigma((q,s))$,
for $(q,s), (q_i^{l_i}, s_i) \in Q'$, $i \in [n]$, $l_i \in [m_i]$, whenever $((q_1,s_1), ..., (q_n,s_n)) \in \alpha_\delta((q,s))$,
- $(q,s) \in \beta_\lambda((q_{i_{k_i}}^{l_i}, s_{i_{k_i}}))$, for $(q,s), (q_{i_{k_i}}^{l_i}, s_{i_{k_i}}) \in Q'$, $i \in [n]$, $l_i \in [m_i]$,
whenever $(..., \underbrace{(q,s)}_{l_i - place}, ...) \in \alpha_{\sigma_i}((q_{i_{k_i}}, s_{i_{k_i}}))$.

The reader will have no difficulties to verify that $L_S(\mathcal{B}) = sh(F)$, and thus $sh(F) \in GST$.□

A combination of the above lemma, proposition 2 and theorem 4 gives

Theorem 9. *$S_1(GST, FIN) = GST$ and $S_1(GST, REC) = GST$.*□

We define now the iterated splicing on H tree schemes.
Let $f = (\Sigma, R)$ be an H tree scheme and $F \subseteq T_\Sigma$. Then

$$f^{(*)}(F) = \bigcup_{k \geq 0} f^{(k)}(F)$$

where

$$f^{(0)}(F) = F, \qquad \text{and} \qquad f^{(k+1)}(F) = f^{(k)}(F) \cup f(f^{(k)}(F)), \quad k \geq 0.$$

We call *index of F in f*, and we denote by $ind(f,F)$, the quantity $n \in N \cup \{\infty\}$, such that

$$f^{(*)}(F) = \bigcup_{k=0}^{n} f^{(k)}(F).$$

We say that the forest $F \subseteq T_\Sigma$ *has finite index* in the H tree scheme $f=(\Sigma, R)$, if $ind(f,F) \in N$.

From the definition of the splicing operation on trees, we can easily understand that for each rule $r = @(s_1, \#, s_2, \$, s_3, \#, s_4) \in R$, the number of appearances of the site (s_1, s_2) or (s_3, s_4) in the trees of F is bounded (in fact this number is $\leq \deg(\Sigma)$), so the application of the splicing operation on F, after a finite number of steps, produces always the same trees. Thus

Proposition 10. *The index of each forest $F \subseteq T_\Sigma$, in each H tree scheme $f = (\Sigma, R)$ is finite.*□

For two families of forests $\mathcal{F}_1, \mathcal{F}_2$, we define the family

$$H(\mathcal{F}_1, \mathcal{F}_2) = \{f^{(*)}(F_1) \ / \ F_1 \in \mathcal{F}_1,\ f = (\Sigma, R) \text{ is an H tree scheme with } R \in \mathcal{F}_2\}.$$

Then the next theorem holds

Theorem 11. *If $\mathcal{F}$ is a sheaf of forests closed under the shift operation on connectd lists, then $H(\mathcal{F}, FIN) = \mathcal{F}$, and $H(\mathcal{F}, REC) = \mathcal{F}$.*

Proof. Let $F \in \mathcal{F}$, $F \subseteq T_\Sigma$, and $f = (\Sigma, R)$ be an H tree scheme of $\mathcal{R}$ type, with $\mathcal{R} \in \{FIN, REC\}$. Then

$$f^{(*)}(F) = \bigcup_{k=0}^{ind(f,F)} f^{(k)}(F).$$

By theorem 4, the forest $f^{(1)}(F) = F \cup f(F)$ belongs to $\mathcal{F}$. Thus by induction $\bigcup_{k=0}^{ind(f,F)} f^{(k)}(F) \in \mathcal{F}$, that is $H(\mathcal{F}, \mathcal{R}) \subseteq \mathcal{F}$.

Conversely, for $F \in \mathcal{F}$, $F \subseteq T_\Sigma$, we consider the H tree scheme $f = (\Sigma, \emptyset)$. Then $f^{(*)}(F) = F$ which means that $\mathcal{F} \subseteq H(\mathcal{F}, \mathcal{R})$.□

Combining the last theorem with lemmas 5 and 8, we obtain

Corollary 12. *It holds $H(\mathcal{F}, \mathcal{R}) = \mathcal{F}$, with $\mathcal{F} \in \{REC, ALG, OCF, GST\}$ and $\mathcal{R} \in \{FIN, REC\}$.*□

Let $F \subseteq T_\Sigma$, and $f = (\Sigma, R)$ be an H tree scheme. For the H string scheme $\sigma = (\Sigma_0, yield(R))$ obtained by f [Boz., Rah. 98], the next inclusion is generally strict

$$yield(f^{(*)}(F)) \subseteq \sigma^{(*)}(yield(F).$$

We shall investigate now a property of the iterated splicing.
Recall that the *height* of a tree $t \in T_\Sigma$, where Σ is a finite ranked alphabet, is defined by
- $hg(t) = 0$, if $t \in \Sigma_0$,
- $hg(\sigma(t_1, ..., t_n)) = 1 + \max\{hg(t_1), ..., hg(t_n)\}$.

If $F \subseteq T_\Sigma$, then its *height* is the number $hg(F) = \sup\{hg(t) \ / \ t \in F\}$.

For each ranked alphabet Σ, we define a monadic alphabet $\Gamma(\Sigma)$ by
- $\Gamma(\Sigma)_0 = \Sigma_0$,
- $\Gamma(\Sigma)_1 = \{\sigma_i \ / \ \sigma \in \Sigma_n,\ n > 0,\ i \in [n]\}$.

The transduction *branches* $br : T_\Sigma \to P(T_{\Gamma(\Sigma)})$ is defined inductively by
- $br(t) = t$, if $t \in \Sigma_0$,
- $br(\sigma(t_1, ..., t_n)) = \bigcup_{i=1}^{n} \sigma_i(br(t_i))$.

For a forest $F \subseteq T_\Sigma$, we write $br(F) = \bigcup_{t \in F} br(t)$.

Let now, $F \subseteq T_\Sigma$, $f = (\Sigma, R)$ be an H tree scheme, $t, p \in F$, $r \in R$, and $(t, p) \rightarrow_r z$. From the definition of the splicing operation, we have that $hg(z) = hg(t)$ or $hg(z) = hg(p)$, that is $hg(z) \leq \max\{hg(t), hg(p)\}$, so

$$hg(f(F)) \leq hg(F).$$

On the other hand $br(z) \subseteq br(t) \cup br(p)$, and thus

$$br(f(F)) \subseteq br(F).$$

We conclude

Proposition 13. *The iterated splicing on a forest $F \subseteq T_\Sigma$, with an H tree scheme $f = (\Sigma, R)$, preserves the height and the branches of F, that is*

$$hg(f^{(*)}(F)) = hg(F) \text{ and } br(f^{(*)}(F)) = br(F).\square$$

5 Conclusion

We defined iterated splicing on trees, and we proved that this operation terminates after a finite number of steps, whereas it leaves unchanged two characteristics of the tree structure. It is not difficult to see that this does not happen for the other known way for splicing trees [Sak., Fer. 99]. Moreover, this does not happen in general, for the word case. For further research we state the following open problem:

Given a finite alphabet A, a language $L \in A^*$, and an H string scheme $\sigma = (A, R)$, find whether L has finite index in σ.

References

[Adl. 94] Adleman M. L.: "Molecular computation of solutions to combinatorial problems"; Science 226, (1994), 1021-1024.

[Boz. 92] Bozapalidis S.: "Alphabetic tree relations"; Theoretical Computer Science 99, (1992), 177-211.

[Boz., Rah. 94] Bozapalidis S., Rahonis G.: "On two families of forests"; Acta Informatica 31, (1994), 235-260.

[Boz., Rah. 98] Bozapalidis S., Rahonis G.: "H tree schemes with finite and recognizable sets of rules"; Romanian Journal of Information Science and Technology 4, 1, (1998), 307-318.

[Géc., Ste. 84] Gécseg F., Steinby M.: "Tree Automata"; Akadémiai Kiadó / Budapest (1984).

[Géc., Ste. 97] Gécseg F., Steinby M.: "Tree Languages", in: Handbook of Formal Languages, Vol III, Rozenberg G., Salomaa A. eds.; Springer-Verlag, (1997), 1-68.

[Gue. 83] Guessarian I.: "Pushdown tree automata"; Mathematical Systems Theory 16, (1983), 237-263.

[Head et al. 97] Head T., Păun Gh., Pixton D.: "Language Theory and Molecular Genetics: Generative Mechanisms Suggested by DNA Recombination", in: Handbook of Formal Languages, Vol II, Rozenberg G., Salomaa A. eds.; Springer-Verlag, (1997), 295-360.

[Păun et al. 98] Păun Gh, Rozenberg G, Salomaa A.: "DNA Computing. New Computing Paradigms"; Springer-Verlag / Berlin, (1998).

[Rah., Sal. 98] Rahonis G., Salomaa K.: "On the size of stack and synchronization alphabets of tree automata"; Fundamenta Informaticae 36, (1998), 57-69.

[Rah. 200x] Rahonis G.: "Alphabetic and synchronized tree transducers"; Theoretical Computer Science (to appear).

[Sak., Fer. 99] Sakakibara Y., Ferretti C.: "Splicing on Tree-like Structures"; Theoretical Computer Science 210, 2, (1999), 227-243.

[Sal. 73] Salomaa A.: "Formal Languages"; Academic Press / New York, (1973).

[Sal. 94] Salomaa K.: "Synchronized tree automata"; Theoretical Computer Science 127, (1994), 25-51.