Technical Report No. 2006–513
Iterated TGR Languages: Membership Problem and Effective Closure Properties

Ian McQuillan
Department of Computer Science, University of Saskatchewan,
Saskatoon, Saskatchewan, Canada S7N 5A9, email: mcquillan@cs.usask.ca

Kai Salomaa
School of Computing, Queen's University,
Kingston, Ontario, Canada K7L 3N6, email: ksalomaa@cs.queensu.ca

Mark Daley
Department of Computer Science and Department of Biology,
University of Western Ontario, London, Ontario, Canada N6A 5B7,
email: daley@csd.uwo.ca

April 30, 2006

Abstract

We show that membership is decidable for languages defined by iterated template-guided recombination systems when the set of templates is regular and the initial language is context-free. Using this result we show that when the set of templates is regular and the initial language is context-free (respectively, regular) we can effectively construct a pushdown automaton (respectively, finite automaton) for the corresponding iterated template-guided recombination language.

Keywords: template-guided recombination systems, formal languages, decision problems, finite automata, regular languages

1 Introduction

The spirotrichous ciliates are a type of unicellular protozoa which possess a unique and fascinating genetic behaviour. Each ciliate cell contains two types of nuclei, *macronuclei* which are responsible for the day-to-day "genetic housekeeping" of the cell, and *micronuclei* which are functionally inert, but used in reproduction. This is in contrast to, e.g., mammalian cells which have only one micronucleus. Although they reproduce asexually, ciliates are also capable of sexual activity in which they exchange haploid micronuclear genomes. This results in each ciliate getting a "genetic facelift" by combining its own genes with those of a mate. After creating a new, hybrid, micronucelus, each ciliate will then regenerate its macronucleus. It is this process of macronuclear regeneration that is of principle interest to us here.

In the spirotrichous ciliates in particular, this macronuclear regeneration involves an intricate process of genetic gymnastics. Suppose that a functional gene in the macronucleus can be divided into 5 sections and written as follows: 1-2-3-4-5. In many cases, the micronuclear form of the same gene may have the segments in a completely different order and include additional segments not found in the macronucleus. For the example given above, a micronuclear gene may appear as: 3-x-5-y-1-z-4-2. For the ciliate to produce a functional macronucleus and continue living, it must *descramble* these micronuclear genes. (See, e.g., [13] for further detail).

A biological model for this descrambling process, based on template-guided DNA recombination, was proposed in [14]. This model was formalized as an operation on words and languages in [4] which also introduced the notion of a template-guided recombination system (TGR system). It was then shown in [5] that a TGR system with a regular set of templates preserves regularity, that is, for a regular initial language, the language resulting from iterated application of the TGR system is always regular. This is in striking contrast to splicing systems since the splicing language generated by a regular set of rules and a finite initial language need not be recursive [9]. In fact, [5] shows much more generally that the operation defined by a TGR system with a regular set of templates preserves any language family that is a full AFL [15]. Thus if the initial language is context-free, a TGR system with a regular set of templates defines a context-free language.

However, the above results are non-constructive and, in particular, do not give an algorithm to decide the membership problem for the language defined by a TGR system, even in the case where the initial language is finite and the set of templates is regular. Here we show that the uniform membership problem for the language defined by a TGR system is decidable when the initial language is context-free and the set of templates is regular. The nonuniform membership problem (where the TGR system is fixed) can be decided in polynomial time. The decidability result can be extended for initial languages that are extensions of the context-free languages, such as the indexed languages, or, more generally, for languages that belong to an arbitrary full AFL satisfying certain natural effectiveness conditions.

Moreover, we use this result to positively solve the main open problem from [5]. That is, given a context-free (respectively, regular) initial language and a regular set of templates, we can effectively construct a pushdown automaton (respectively, a finite automaton) for the language defined by the TGR system. Using a variant of the decision algorithm for the membership problem, we effectively find a DFA for the subset of templates that can be used in some recombination operation and this, together with the results of [5], enables us to construct the pushdown automaton (respectively, the finite automaton) for the language defined by the TGR system. This result also holds for regular sets of templates and initial languages from an arbitrary full AFL that satisfies certain effectiveness conditions.

Both the algorithm for the membership problem and the method for finding the set of useful templates use expensive brute-force techniques. It remains an open question, whether it is possible to find a more efficient algorithm, at least in the case where both the initial language and the set of templates are regular.

2 Preliminaries

Here we recall some basic definitions needed in the next section. For all unexplained notions related to formal languages we refer the reader e.g. to [10, 15]. Recent work on language classes motivated by bio-operations can be found in [2, 3] and in the references listed there.

In the following Σ is a finite alphabet and the set of all words over Σ is Σ^*. The length of a word $w \in \Sigma^*$ is $|w|$. The ith symbol of a word $w \in \Sigma^*$ is denoted $w[i]$, $i = 1, \ldots |w|$.

A language is a subset of Σ^*. The sets of all prefixes, all suffixes and all subwords of words in L are denoted, respectively, $\mathrm{pref}(L)$, $\mathrm{suf}(L)$, $\mathrm{subw}(L)$. An *inner subword* of a word w is a subword that is not a prefix and not a suffix. By a *proper subword* of w we mean any subword that is not w. An inner subword is a proper subword, but not vice versa, in general.

A deterministic finite automaton (DFA) is a tuple $A = (\Sigma, Q, q_0, F, \delta)$ where Σ is the input alphabet, Q is the finite set of states, $q_0 \in Q$ is the start state, $F \subseteq Q$ is the set of accepting states and $\delta : Q \times \Sigma \to Q$ is the transition function. In the well known way δ is extended as a function $\delta : Q \times \Sigma^* \to Q$ and the language recognized by A is $L(A) = \{w \in \Sigma^* \mid \delta(q_0, w) \in F\}$. The languages recognized by DFA's are exactly the regular languages.

A family of languages is said to be a *full abstract family of languages* (full AFL) [8, 15] if it contains a nonempty language and is closed under the following operations: union, Kleene plus, homomorphism, inverse homomorphism, and intersection with regular languages. The operations in the above list are called the AFL operations.

We recall definitions concerning template-guided recombination systems [4, 5].

Definition 2.1 *A* template-guided recombination system *(TGR system) is a tuple $\varrho = (T, \Sigma, n_1, n_2)$, where Σ is a finite alphabet, $T \subseteq \Sigma^*$ is the template language, and $n_1, n_2 \in \mathbb{N}$.*

Let $x, y \in \Sigma^*$ and $t \in T$. The recombination operation defined by ϱ is given by: $(x, y) \vdash_t^{\varrho} w$ if and only if we can write

$$x = u\alpha\beta d, \quad y = e\beta\gamma v, \quad t = \alpha\beta\gamma \quad \textit{and} \quad w = u\alpha\beta\gamma v \tag{1}$$

for some $u, v, d, e \in \Sigma^*$, $\alpha, \gamma \in \Sigma^{\geq n_1}$ and $\beta \in \Sigma^{n_2}$.

For $L \subseteq \Sigma^*$ we define:

$$\varrho(L) = \{w \in \Sigma^* \mid (x, y) \vdash_t^{\varrho} w \textit{ for some } x, y \in L, t \in T\}$$

The original definition of the recombination operation $\vdash_t^{\varrho}$ uses a weaker condition that requires only $\beta \in \Sigma^{\geq n_2}$. It is shown in [4] that without loss of generality we can assume that β has length exactly n_2.

Let $\varrho = (T, \Sigma, n_1, n_2)$ be a TGR system and let $L \subseteq \Sigma^*$. We define the iteration $\varrho^{(*)}$ of the operation ϱ by setting $\varrho^{(0)}(L) = L$, and defining

$$\varrho^{(i+1)}(L) = \varrho^{(i)}(L) \cup \varrho(\varrho^{(i)}(L)) \text{ for all } i \geq 0. \tag{2}$$

Denote

$$\varrho^{(*)}(L) = \bigcup_{i=0}^{\infty} \varrho^{(i)}(L).$$

Let $\varrho = (T, \Sigma, n_1, n_2)$ be a TGR system and let $L \subseteq \Sigma^*$. A word $t \in T$ is said to be *useful* on (L, ϱ) if t can be used in iterated application of ϱ on the initial language L. It is shown in [5] that $t \in T$ is useful on (L, ϱ) if and only if $|t| \geq 2n_1 + n_2$ and t is a subword of some word in $\varrho^{(*)}(L)$. The TGR system ϱ is said to be *useful on L* if every word of T is useful on (L, ϱ). The *useful subset of ϱ on L* is the set of all words in T which are useful on (L, ϱ).

3 Membership problem

Here we show that for a context-free language L and a TGR system $\varrho = (T, \Sigma, n_1, n_2)$ where T is regular, the uniform membership problem for the language $\varrho^{(*)}(L)$ is decidable.

We want to be able to establish properties concerning how many recombination operations are required to produce some subword of a word w when it is known that w requires a given number of recombination operations. For this purpose it turns out to be useful to consider "marked variants" of words over Σ. The marked variants associate states of a DFA recognizing the set of templates T and length information with certain positions in the word. This additional control information is used to keep track of the templates (or strictly speaking equivalence classes of templates) that can be used in the recombination operations.

For the above purpose we next introduce some technical notation. Let $\varrho = (T, \Sigma, n_1, n_2)$ be a TGR system and

$$A = (\Sigma, Q, q_0, F, \delta) \tag{3}$$

be a DFA that recognizes T.

Denote

$$\overrightarrow{Q} = \{\overrightarrow{q} \mid q \in Q\}, \quad \overleftarrow{Q} = \{\overleftarrow{q} \mid q \in Q\}.$$

For $n \in \mathbb{N}$ let $[n] = \{0, 1, \ldots, n\}$. We define the extended alphabet $\Sigma[\varrho]$ as

$$\Sigma[\varrho] = \Sigma \times \mathcal{P}((\overrightarrow{Q} \cup \overleftarrow{Q}) \times [n_1]). \tag{4}$$

The first component of elements of $\Sigma[\varrho]$ is an element of Σ and the second component consists of a set of states of Q each marked with a "right arrow" or a "left arrow". Additionally, each state is associated with an index from $\{0, 1, \ldots, n_1\}$.

The projections from $\Sigma[\varrho]$ to Σ and to $\mathcal{P}((\overrightarrow{Q} \cup \overleftarrow{Q}) \times [n_1])$ are denoted, respectively, π_1^ϱ and π_2^ϱ. When ϱ is clear from the context, we denote the projections simply as π_1 and π_2. The projection π_1 is in the natural way extended to a morphism $\Sigma[\varrho]^* \longrightarrow \Sigma^*$.

Let $L \subseteq \Sigma^*$. The *T-controlled marked variant* of L is the largest language $C_T(L) \subseteq \Sigma[\varrho]^*$ such that the below conditions (i) and (ii) hold[1]. The notations refer to (3) that gives a DFA for the language T.

(i) For every $w \in C_T(L)$, $\pi_1(w) \in L$.

(ii) Assume that $w \in C_T(L)$ and $(p, j) \in \pi_2(w[i])$, $1 \leq i \leq |w|$, $p \in \overrightarrow{Q} \cup \overleftarrow{Q}$, $j \in [n_1]$.

   (a) If $p \in \overrightarrow{Q}$, then $\pi_1(w)$ has a subword u starting at the $(i+1)$th position such that $|u| \geq j$ and $\delta(p, u) \in F$.

   (b) If $p \in \overleftarrow{Q}$, then $\pi_1(w)$ has a subword u ending at the $(i-1)$th position such that $|u| \geq j$ and $\delta(q_0, u) = p$.

Note that for any $w \in L$, the word w' is in $C_T(L)$ where w' is obtained from w by replacing each symbol $c \in \Sigma$ by $(c, \emptyset) \in \Sigma[\varrho]$. We identify words w and w' and in this way

$$\text{we can view } L \text{ to be a subset of } C_T(L). \tag{5}$$

[1]Note that the union of languages satisfying this property also satisfies this property, and so the largest language must exist.

According to (i) and (ii) above, the elements (p, j), $p \in \overrightarrow{Q} \cup \overleftarrow{Q}$ occurring in symbols of a word $w \in C_T(L)$ place conditions on what kind of subwords w must have starting directly after or ending directly before that position. If $p \in \overrightarrow{Q}$, this means that $\pi_1(w)$ must have a subword u starting from the next position that is a suffix of a word in T, u is of length at least j, and p is the state of A that corresponds to this suffix (that is, $\delta(p, u) \in F$). If $p \in \overleftarrow{Q}$, this means that $\pi_1(w)$ must have a subword u ending at the previous position that is a prefix of a word in T, u has length at least j, and p is the state of A that corresponds to this prefix.

The following technical property will be needed later.

Lemma 3.1 *Let $L \subseteq \Sigma^*$ and $w \in C_T(L)$. Assume that $w' \in \Sigma[\varrho]^*$ has the properties that $\pi_1(w') = \pi_1(w)$ and, for all $i = 1, \ldots, |w|$, $\pi_2(w'[i]) \subseteq \pi_2(w[i])$.*[2]

Then $w' \in C_T(L)$.

Proof. This follows directly by the definition of the marked variant $C_T(L)$ of L. The elements in the second components of symbols of $\Sigma[\varrho]$ are used to restrict what can be the π_1-images of subwords to the left or to the right of the current position. Since w' is obtained from w by removing some of the elements in the second components, if w is in $C_T(L)$ the same has to hold for w'. ■

We still need the following notation to manipulate words over the alphabet $\Sigma[\varrho]$. Let $w \in \Sigma[\varrho]^*$, $1 \leq i \leq w$, $p \in (\overrightarrow{Q} \cup \overleftarrow{Q})$ and $j \in [n_1]$. Then

$$w[i \leftarrow (p, j)] \tag{6}$$

denotes the word obtained from w by adding (p, j) to the second component of the ith symbol, that is, the second component of the ith symbol is changed to be $\pi_2(w[i]) \cup \{(p, j)\}$.

We say that a word $w \in \Sigma[\varrho]^*$ is *well formed* if $|w| \geq 2$ and the following conditions hold:

(i) $\pi_2(w[1]) \subseteq \overleftarrow{Q} \times [n_1]$,

(ii) $\pi_2(w[|w|]) \subseteq \overrightarrow{Q} \times [n_1]$, and

(iii) $\pi_2(w[j]) = \emptyset$ when $1 < j < |w|$.

In a well formed marked word the first symbol contains only elements of the type $(\overleftarrow{p}, j)$ as markers, and the last symbol contains only elements of the type $(\overrightarrow{p}, j)$ as markers, $p \in Q$, $j \in [n_1]$. Symbols of w other than the first or the last symbol have $\emptyset$ as the second component. Note that any word $w \in \Sigma^*$, $|w| \geq 2$, is well formed when w is viewed as an element of $\Sigma[\varrho]^*$ according to convention (5).

The set of all well formed words over $\Sigma[\varrho]$ is denoted by $\mathcal{WF}(\Sigma[\varrho])$

The following lemma says, very roughly speaking, that if w is a subword of $\varrho^{(k+1)}(L)$ but w is not a subword of $\varrho^{(k)}(L)$, then w has a proper subword that is a subword of $\varrho^{(k)}(L)$ but not a subword of $\varrho^{(k-1)}(L)$. The statement in the previous sentence is oversimplified and does not hold as such. To be precise, in order to be able to establish the required property we need to add to the subwords information on the states of the DFA for T associated with the templates used in the recombination operations, that is, we need to consider subwords of the T-controlled marked variant of $\varrho^{(k)}(L)$, $k \geq 1$.

[2]Note that $\pi_1(w') = \pi_1(w)$ implies $|w'| = |w|$.

Below when referring to a recombination operation that uses a template t, by the *overlapping part* of the template we mean the subword occurrence β in the word w if the notations are as in (1) and the recombination operation is $(x,y) \vdash_t^\varrho w$.

For $m, n \in \mathbb{N}$ we define the non-negative difference of m and n as

$$m \ominus n = \begin{cases} m - n \text{ if } m \geq n, \\ 0 \text{ otherwise.} \end{cases}$$

Lemma 3.2 *Let $\varrho = (T, \Sigma, n_1, n_2)$ where T is regular and let A as in (3) be a DFA that recognizes T. Let $k \geq 1$ and $L \subseteq \Sigma^*$.*

We claim that if $w \in \mathcal{WF}(\Sigma[\varrho])$ and

$$w \in \mathrm{subw}(C_T(\varrho^{(k+1)}(L))) - \mathrm{subw}(C_T(\varrho^{(k)}(L))) \tag{7}$$

then one of the below cases (8)–(11) holds:

$$\begin{aligned} w &= u\alpha\beta\gamma v, \pi_1(\alpha\beta\gamma) \in T, |\beta| = n_2, |\alpha|, |\gamma| \geq n_1, \\ &\quad u\alpha\beta \in \mathrm{subw}(C_T(\varrho^{(k)}(L))) \cap \mathcal{WF}(\Sigma[\varrho]), \beta\gamma v \in \mathrm{subw}(C_T(\varrho^{(k)}(L))) \cap \mathcal{WF}(\Sigma[\varrho]), \end{aligned} \tag{8}$$

or,

$$\begin{aligned} w &= u\alpha\beta\gamma', |\beta| = n_2, |\alpha| \geq n_1, |\gamma'| \geq 1, u\alpha\beta \in \mathrm{subw}(C_T(\varrho^{(k)}(L))) \cap \mathcal{WF}(\Sigma[\varrho]), \\ &\quad \beta\gamma'[|\beta\gamma'| \leftarrow (\overrightarrow{p}, n_1 \ominus |\gamma'|) \in \mathrm{subw}(C_T(\varrho^{(k)}(L))) \cap \mathcal{WF}(\Sigma[\varrho]), \text{ where } p = \delta(q_0, \alpha\beta\gamma'). \end{aligned} \tag{9}$$

or,

$$\begin{aligned} w &= \alpha'\beta\gamma v, |\beta| = n_2, |\gamma| \geq n_1, |\alpha'| \geq 1, \\ &\quad \alpha'\beta[1 \leftarrow (\overleftarrow{p}, n_1 \ominus |\alpha'|)] \in \mathrm{subw}(C_T(\varrho^{(k)}(L))) \cap \mathcal{WF}(\Sigma[\varrho]), p \in Q, \\ &\quad \beta\gamma v \in \mathrm{subw}(C_T(\varrho^{(k)}(L))) \cap \mathcal{WF}(\Sigma[\varrho]), \text{ where } \delta(p, \alpha'\beta\gamma) \in F. \end{aligned} \tag{10}$$

or,

$$\begin{aligned} w &= \alpha'\beta\gamma', |\beta| = n_2, |\alpha'|, |\gamma'| \geq 1, \\ &\quad \alpha'\beta[1 \leftarrow (\overleftarrow{p}, n_1 \ominus |\alpha'|)] \in \mathrm{subw}(C_T(\varrho^{(k)}(L))) \cap \mathcal{WF}(\Sigma[\varrho]), p \in Q, \\ &\quad \beta\gamma'[|\beta\gamma'| \leftarrow (\overrightarrow{p_1}, n_1 \ominus |\gamma'|)] \in \mathrm{subw}(C_T(\varrho^{(k)}(L))) \cap \mathcal{WF}(\Sigma[\varrho]), \text{ where } \delta(p, \alpha'\beta\gamma') = p_1. \end{aligned} \tag{11}$$

Furthermore, in any decomposition of w as in (8)–(11) at most one of the two mentioned marked subwords of $\mathrm{subw}(C_T(\varrho^{(k)}(L)))$ *can be in* $\mathrm{subw}(C_T(\varrho^{(k-1)}(L)))$*.*

Proof. If for some words x_i, y_i, $i = 1, 2, 3$, $x_1x_2x_3 = y_1y_2y_3$ and $|x_1| \geq |y_1|$, $|x_3| \geq |y_3|$ we say that the subword occurrence x_2 is *contained* in the subword occurrence y_2. When it is understood that we are talking about subword occurrences, we may say simply that "a subword is contained in another subword".

Consider a particular subword occurrence x in a word y, where $x, y \in \Sigma[\varrho]^*$. Then by the occurrence of the subword $\pi_1(x)$ in $\pi_1(y)$ we mean the subword of $\pi_1(y)$ at the position where x occurs in y (although $\pi_1(x)$ naturally could occur also at other positions).

Let w be as in (7). Since $w \notin \text{subw}(C_T(\varrho^{(k)}(L)))$, according to (2), w must be a subword of $w_1 \in C_T(\varrho^{(k+1)}(L))$ such that

$$(x, y) \vdash_t^\varrho \pi_1(w_1), \;\; x, y \in \varrho^{(k)}(L), \tag{12}$$

and the overlapping part of the template t is an inner subword of the subword occurrence $\pi_1(w)$ in $\pi_1(w_1)$.

Now (8) corresponds to the situation where the entire template t (when viewed as a subword occurrence of $\pi_1(w_1)$) is contained in $\pi_1(w)$, (9) corresponds to the situation where the end of the template t lies outside of $\pi_1(w)$, (10) corresponds to the situation where the beginning of the template t lies outside of $\pi_1(w)$, and (11) corresponds to the situation where both the beginning and end of the template t lie outside of $\pi_1(w)$. It is immediate that one of these situations must occur.

In each of the cases (8)–(11), in the resulting two words of $\text{subw}(C_T(\varrho^{(k)}(L)))$ we add new state markers only to the first and/or the last symbol, and any state added to the first symbol has a "left arrow" and any state added to the last symbol has a "right arrow". Since w is well formed, it follows that also the resulting new words are well formed. Note that for the purpose of finding the two words in $\text{subw}(C_T(\varrho^{(k)}(L)))$ with the required properties we, of course, would not need any of the newly added state markers. The new markers are needed below to argue that, in each case (8)–(11), both of the resulting words cannot be in $\text{subw}(C_T(\varrho^{(k-1)}(L)))$.

Note that the decomposition of w as in (8)–(11) is, in general, not effective, e.g., in (10) or (11) we do not need to worry how the state p would be found.

Next we argue that if (11) is true, at least one of the two subwords must be in the set $\text{subw}(C_T(\varrho^{(k)}(L))) - \text{subw}(C_T(\varrho^{(k-1)}(L)))$. The first three cases (8)–(10) are similar and simpler. If in (11) both subwords are in $\text{subw}(C_T(\varrho^{(k-1)}(L)))$, then the control elements added to the first symbol of $\alpha'\beta$ and the last symbol of $\beta\gamma'$ guarantee that there exist $\alpha'', \gamma'' \in \Sigma[\varrho]^*$ such that $|\alpha''\alpha'|, |\gamma'\gamma''| \geq n_1$, $\delta(q_0, \pi_1(\alpha'')) = p$ and $\delta(p_1, \pi_1(\gamma'')) \in F$. Thus, $\pi_1(\alpha''\alpha'\beta\gamma'\gamma'') \in T$ and $\alpha''\alpha'\beta$ and $\beta\gamma'\gamma''$ are subwords of $x, y \in C_T(\varrho^{(k-1)}(L))$, respectively, i.e., we can write

$$x = x_1\alpha''\alpha'\beta x_2, \quad y = y_1\beta\gamma'\gamma''y_2. \tag{13}$$

Now

$$(\pi_1(x), \pi_1(y)) \vdash_{t_1}^\varrho \pi_1(x_1\alpha''\alpha'\beta\gamma'\gamma''y_2), \text{ where } t_1 = \pi_1(\alpha''\alpha'\beta\gamma'\gamma'') \in T. \tag{14}$$

By Lemma 3.1 we can choose in (13) the words x and y so that they do not contain any marker elements other than the elements occurring in α' and γ'. Note that since $w = \alpha'\beta\gamma'$ is well formed, it can have marker states only in the first symbol of α' and in the last symbol of γ'. That is, in all symbols of x and y except the first symbol of α' and the last symbol of γ' the second component is $\emptyset$.

Since $x \in \text{subw}(C_T(\varrho^{(k-1)}(L)))$, the prefix $x_1\alpha''$ has to satisfy the requirements imposed by the "left-arrow" states in the first symbol of α'. Similarly, since $y \in \text{subw}(C_T(\varrho^{(k-1)}(L)))$ the suffix $\gamma''y_2$ has to satisfy the requirements imposed by all "right arrow" states in the last symbol of γ'.

By (14), $\pi_1(x_1\alpha''\alpha'\beta\gamma'\gamma''y_2) \in \varrho^{(k)}(L)$ and since above it was established that all control requirements imposed by the marker elements are satisfied it follows that

$$x_1\alpha''\alpha'\beta\gamma'\gamma''y_2 \in C_T(\varrho^{(k)}(L))$$

and, consequently,

$$w = \alpha'\beta\gamma' \in \text{subw}(C_T(\varrho^{(k)}(L))).$$

This contradicts (7). ■

We should note that in (9), (10) and (11) in Lemma 3.2 it is essential that we add the new marker states to the resulting subwords. For example, using the notations of (11), it is quite possible that $\pi_1(\alpha'\beta) \in \mathrm{subw}(\varrho^{(k-1)}(L))$ <u>and</u> $\pi_1(\beta\gamma') \in \mathrm{subw}(\varrho^{(k-1)}(L))$ because $\alpha'\beta$ could be part of a word that does not allow recombination using any template of T with the words where $\beta\gamma'$ occurs as a subword. The marked variants of the words prevent this possibility by storing the appropriate states and length information in the first symbol of α' and in the last symbol of γ'. The marker information forces that $\alpha'\beta$ (respectively, $\beta\gamma'$) must occur in a position where the immediately preceding (respectively, immediately following) subword contains a suffix (respectively, a prefix) that allows us to complete $\alpha'\beta\gamma'$ into a template of T.

Using Lemma 3.2 we get the following property that will be essential for deciding the membership problem. Also we note that Lemma 3.3 (i) is not a special case of (ii) (although their proofs are similar) and hence we include both statements.

Lemma 3.3 *Let $\varrho = (T, \Sigma, n_1, n_2)$ be a TGR-system where T is regular and $L \subseteq \Sigma^*$.*

(i) *If $w \in \varrho^{(k)}(L) - \varrho^{(k-1)}(L)$, $k \geq 1$, then $|w| - n_2 - 1 \geq k$.*

(ii) *If $w \in \mathrm{subw}(\varrho^{(k)}(L)) - \mathrm{subw}(\varrho^{(k-1)}(L))$, then $|w| - n_2 - 1 \geq k$.*

Proof. For (i) we want to argue that if $w \in \varrho^{(k+1)}(L) - \varrho^{(k)}(L)$, then w has to have length at least k (plus some constant). This is done by applying the conditions of Lemma 3.2 recursively. When dealing with prefixes or suffixes of the original word w, some of the possible ways to perform the next recombination step listed in (8)–(11) may not be applicable.

When applying Lemma 3.2 to $w \in \varrho^{(k+1)}(L) - \varrho^{(k)}(L)$, in the first recombination step the entire template must lie inside w and necessarily we get $u\alpha\beta$ and $\beta\gamma v$ as in (8). We say that $w_1 = u\alpha\beta$ is an *uncut prefix* to indicate that it contains the initial part of the original word. Similarly $w_2 = \beta\gamma v$ is called an *uncut suffix.*

Now according to Lemma 3.2, there is $i \in \{1, 2\}$ such that

$$w_i \in \mathrm{subw}(C_T(\varrho^{(k)}(L))) - \mathrm{subw}(C_T(\varrho^{(k-1)}(L))). \tag{15}$$

After the first step the words do not have any marker elements, so above w_i is in fact a word over Σ.

Next the process is continued from w_i. When applying Lemma 3.2 to an uncut prefix, only cases (8) or (9) may apply since it is not possible that the begin of the template used in the next operation would lie outside of w_i. The first of the resulting words is again an uncut prefix and the second is a subword of $C_T(\varrho^{(k-1)}(L))$.

Similarly, when applying Lemma 3.2 to an uncut suffix, only cases (8) or (10) may apply since the end of the template may not lie outside of the uncut suffix, and in this case the first resulting word is a subword of $C_T(\varrho^{(k-1)}(L))$ and the second is an uncut suffix. When applying Lemma 3.2 to words other than an uncut prefix or an uncut suffix, any of the cases (8)–(11) may apply.

At each step the length of each of the resulting words is strictly shorter than the previous word, and one of the two resulting words requires only one less iteration of (2) than the original word. At the end we obtain w_i as in (15) with $k = 1$. Directly from the definition of the recombination operation (1) it follows that if $u \in \mathrm{subw}(\varrho^{(*)}(L))$ and $|u| \leq n_2 + 1$ then $u \in \mathrm{subw}(L)$. Since, according to (15), $w_i \notin \mathrm{subw}(C_T(\varrho^{(0)}(L))) = \mathrm{subw}(C_T(L))$ it follows that

$$|w_i| \geq n_2 + 2.$$

For (ii) we use exactly the same argument. Now we start with a word $w \in \mathrm{subw}(\varrho^{(k+1)}(L)) - \mathrm{subw}(\varrho^{(k)}(L))$, and the only difference to the above argument is that when applying Lemma 3.2 at the start of the process any of the cases (8)–(11) may apply. ■

Using a more general definition of $C_T(L)$ the result of Lemma 3.3 can be extended for sets of templates that are not regular. For our main results here it is sufficient to consider regular sets of templates.

Theorem 3.1 *Given a TGR system $\varrho = (T, \Sigma, n_1, n_2)$ with T regular, a context-free language L and a word $w \in \Sigma^*$, it is decidable whether or not $w \in \varrho^{(*)}(L)$.*

Furthermore, it is decidable whether or not $w \in \mathrm{subw}(\varrho^{(*)}(L))$.

Proof. Let $A = (\Sigma, Q, q_0, F, \delta)$ be a DFA that recognizes T. Given a pushdown automaton B_i for $\varrho^{(i)}(L)$, $i \geq 0$, we can construct a pushdown automaton B_{i+1} for $\varrho^{(i+1)}(L)$ as follows.

Let $\beta \in \Sigma^{n_2}$ and $q \in Q$. We define

$$L_1(B_i, \beta, q) = \{ \, w \in \mathrm{pref}(L(B_i)) \mid w = u\alpha\beta, |\alpha| \geq n_1, \delta(q_0, \alpha\beta) = q \, \},$$

and,

$$L_2(B_i, \beta, q) = \{ \, w \in \beta^{-1}\mathrm{suf}(L(B_i)) \mid w = \gamma v, |\gamma| \geq n_1, \delta(q, \gamma) \in F \, \}.$$

Now it is clear that

$$\varrho^{(i+1)}(L) = \varrho^{(i)}(L) \cup \bigcup_{\beta \in \Sigma^{n_2}, q \in Q} L_1(B_i, \beta, q) \cdot L_2(B_i, \beta, q). \tag{16}$$

Since context-free languages are effectively closed under prefix, suffix, union, and quotient and intersection with a regular language, using (16) we can construct a pushdown automaton B_{i+1} for $\varrho^{(i+1)}(L)$.

By Lemma 3.3, it is sufficient to construct the pushdown automaton $B_{|w|-n_2-1}$ and decide whether or not $B_{|w|-n_2-1}$ accepts w. The latter can be done effectively since membership is decidable for context-free languages.

Also, context-free languages are effectively closed under subword. Thus, we can test whether $w \in \mathrm{subw}(\varrho^{(|w|-n_2-1)}(L))$ and, by Lemma 3.3 (ii), this holds if and only if $w \in \mathrm{subw}(\varrho^{(*)}(L))$. ■

The operation (16) uses union indexed over all words of length n_2 and consequently the algorithm given by Theorem 3.1 for the uniform membership problem requires exponential time. However, if ϱ is fixed, i.e., if we consider the non-uniform membership problem then the algorithm given by Theorem 3.1 uses polynomial time. The same is true even if only the value of n_2 is fixed. Note that the number of iterations of (16) is upper bounded by the length of w, i.e., the number of iterations is given in unary notation.

Corollary 3.1 *Let n_2 be fixed. Given a TGR system $\varrho = (T, \Sigma, n_1, n_2)$ with T regular, a context-free language L and a word $w \in \Sigma^*$, it is decidable in polynomial time whether or not $w \in \varrho^{(*)}(L)$.*

Lemma 3.3 does not make any assumptions on the initial language. The proof of Theorem 3.1 uses certain closure and decidability properties of context-free languages. An arbitrary full AFL satisfies the required conditions, assuming that membership is decidable and closure under the AFL operations is effective, and a corresponding extended result is stated below in Corollary 3.2. Before that we introduce some terminology dealing with AFL's consisting of recursive languages. The terminology will be useful also in the next section in order to be able to rely in a uniform way on results from [5] that are formulated in terms of AFL's.

Definition 3.1 *We say that a property P of Turing machines is* syntactic *if given an arbitrary Turing machine M it is decidable whether or not M has property P. The class of Turing machines satisfying a property P is denoted* $\mathrm{TM}[P]$.

A language family $\mathcal{L}$ is said to be a constructive full AFL *if $\mathcal{L}$ contains a nonempty language and there exists a syntactic property of Turing machines $P_{\mathcal{L}}$ such that*

(i) *a language L is in $\mathcal{L}$ if and only if L is recognized by some Turing machine in* $\mathrm{TM}[P_{\mathcal{L}}]$,

(ii) *given $M \in \mathrm{TM}[P_{\mathcal{L}}]$ and an input word w, it is decidable whether or not $w \in L(M)$, and*

(iii) *languages recognized by machines in $\mathrm{TM}[P_{\mathcal{L}}]$ are effectively closed under the AFL operations. That is, there is an algorithm that for given $M_1, M_2 \in \mathrm{TM}[P_{\mathcal{L}}]$ constructs $M_{\mathrm{union}} \in \mathrm{TM}[P_{\mathcal{L}}]$ such that $L(M_{\mathrm{union}}) = L(M_1) \cup L(M_2)$, and for any AFL operation σ other than union there is an algorithm to construct $M \in \mathrm{TM}[P_{\mathcal{L}}]$ such that $L(M) = \sigma(L(M_1))$.*

Well known examples of constructive full AFL's are the regular and the context-free languages. An example of a more general constructive full AFL is the family of languages recognized by (one-way, single head) k-iterated pushdown automata, $k \geq 1$, [7]. It is easy to verify that any (k-iterated) pushdown automaton can be simulated by a Turing machine where the transition relation satisfies a suitably defined syntactic property that forces the work tape to simulate a (k-iterated) pushdown store. It seems that any full AFL consisting only of recursive languages that is defined by a "reasonable" machine model could be characterized in the above way.

The family of recursively enumerable languages is a full AFL that is not a constructive full AFL.

Corollary 3.2 *Let $\mathcal{L}$ be a constructive full AFL. Given a TGR system $\varrho = (T, \Sigma, n_1, n_2)$ where T is regular and $L \in \mathcal{L}$, the membership problem for $\varrho^{(*)}(L)$ is decidable.*

The set of useful templates of a TGR system $\varrho = (T, \Sigma, n_1, n_2)$ with an initial language L is the set $T \cap \mathrm{subw}(\varrho^{(*)}(L)) \cap \Sigma^{\geq 2n_1+n_2}$ [5]. Thus from Theorem 3.1 we get:

Corollary 3.3 *Given a TGR system $\varrho = (T, \Sigma, n_1, n_2)$ where T is regular and a context-free initial language L, we can effectively decide whether or not a given template is useful on (L, ϱ).*

This holds again for an arbitrary constructive full AFL.

Corollary 3.4 *Let $\mathcal{L}$ be a constructive full AFL. Given a TGR system $\varrho = (T, \Sigma, n_1, n_2)$ where T is regular and $L \in \mathcal{L}$, we can effectively decide whether or not a given template is useful on (L, ϱ).*

To conclude this section we make a couple of remarks on limitations in attempting to extend the previous results. The 2-iterated pushdown automata recognize the indexed languages [1] and, thus, from Corollary 3.2 we get a decidability result for the membership problem when the initial language is an indexed language. However, there is no known polynomial time parsing algorithm for general indexed languages and Corollary 3.1 cannot be extended for the case where the initial language is indexed.

Finally we observe that Theorem 3.1 cannot be extended for the context-sensitive languages or the languages generated by Boolean grammars [11]. Note that the context-sensitive languages or the languages generated by Boolean grammars [11] have a decidable membership problem. However, these language families are not closed under the prefix operation. The below construction shows that Theorem 3.1 could not be extended for context-sensitive or Boolean languages. Let $L \subseteq \Sigma^*$ be arbitrary and $c \notin \Sigma$. By choosing $\varrho = (\Sigma^2 \cdot \{c\}, \Sigma \cup \{c\}, 1, 1)$, we note that $\varrho^{(*)}(L \cup \Sigma \cdot \{c\}) \cap (\Sigma^{\geq 2} \cdot \{c\})$ consists of exactly the prefixes of L of length at least two followed by the endmarker c. Using a variant of Theorem 11 of [12] it can be shown that membership in the set of prefixes of a language generated by a Boolean grammar is undecidable, and consequently also membership in $\varrho^{(*)}(L)$ is undecidable when L is generated by a Boolean grammar and the set of templates is finite. Languages generated by Boolean grammars are a subfamily of the context-sensitive languages.

4 Effective closure properties

We would now like to attack the question of, given $\varrho = (T, \Sigma, n_1, n_2)$, with T regular, and L recognized by a pushdown automaton (respectively, a finite automaton), can we effectively construct a pushdown automaton (respectively, a finite automaton) which recognizes $\varrho^{(*)}(L)$? Note that in the former case it is known that $\varrho^{(*)}(L)$ is context-free (and in the latter case regular) [5] but the results are non-constructive.

We first need to provide some details from [5]. The main non-constructive proof from this paper shows that, for an arbitrary TGR system $\varrho = (T, \Sigma, n_1, n_2)$ with T regular, and an arbitrary full AFL $\mathcal{L}$ the following holds: If $L \in \mathcal{L}$, then $\varrho^{(*)}(L) \in \mathcal{L}$. The proof of this result relies on two auxiliary results, the first one of which is the following:

Proposition 4.1 (Theorem 4.2 of [5]) *Let $\varrho = (T, \Sigma, n_1, n_2)$ be a TGR system and let $L \subseteq \Sigma^*$. Let T_u be the useful subset of ϱ on L. If T is a regular language, then T_u is also regular.*

The proof of the above result [5] is not constructive, even in the case where we have some effective representation for L. However, the proof does give some information as to the structure of the DFA which accepts T_u. If Q is the state set of a DFA which accepts T, then the proof creates a finite set of automata $\mathcal{X}_{T,L}$, each automaton with a state set of size $q_{T,L} = (|Q|+1)^n \cdot (|\Sigma|+1)^{n-1}$ where $n = 2n_1 + n_2 - 1$. Moreover, the proof establishes that one of these automata accepts T_u, but does not tell us which one is the correct automaton.

Indeed, let $\varrho = (T, \Sigma, n_1, n_2)$ be a TGR system where T is regular and let $\mathcal{L}$ be a constructive full AFL, and let $L \in \mathcal{L}$. Then, by Corollary 3.4, we can decide whether or not a given template is useful on (L, ϱ). Consider $T_u \cap \Sigma^{\leq 2 \cdot q_{T,L}}$, the finite set of all words which are useful on (L, ϱ) and which are of length less than or equal to $2 \cdot q_{T,L}$. Using Corollary 3.4 we can now effectively determine this set. In addition, for each automaton $M = (Q, \Sigma, q_0, F, \delta) \in \mathcal{X}_{T,L}$, we can check whether or not

$$T_u \cap \Sigma^{\leq 2 \cdot q_{T,L}} = L(M) \cap \Sigma^{\leq 2 \cdot q_{T,L}}.$$

We claim that this is true if and only $L(M) = T_u$.

Claim 4.1 $T_u \cap \Sigma^{\leq 2 \cdot q_{T,L}} = L(M) \cap \Sigma^{\leq 2 \cdot q_{T,L}}$ *if and only if* $T_u = L(M)$.

Proof. It is sufficient to show the implication from left to right. According to Proposition 6.3 of [6], the following is true: Let M_1, M_2 be two deterministic finite automata with state sets Q_1, Q_2 respectively. Then $L(M_1) = L(M_2)$ whenever for all $s \in \Sigma^*$ such that $|s| < |Q_1| + |Q_2|$ we have

$$s \in L(M_1) \Leftrightarrow s \in L(M_2).$$

Assume by contradiction that $T_u \neq L(M)$. But there exists $M' \in \mathcal{X}_{T,L}$ (also with a state set of size $q_{T,L}$) such that $L(M') = T_u$, and hence

$$L(M') \cap \Sigma^{\leq 2q_{T,L}} = T_u \cap \Sigma^{\leq 2q_{T,L}} = L(M) \cap \Sigma^{\leq 2q_{T,L}}.$$

However, according to the proposition from [6], this implies $L(M') = L(M)$, a contradiction. ■

By Claim 4.1, we can find from $\mathcal{X}_{T,L}$ the correct automaton which accepts T_u. Hence, we can effectively construct a deterministic finite automaton which accepts T_u. Thus we have shown that the following holds:

Lemma 4.1 *Let $\mathcal{L}$ be a constructive full AFL. Given $\varrho = (T, \Sigma, n_1, n_2)$ with T regular and $L \in \mathcal{L}$, we can effectively construct a DFA for the useful subset of ϱ on L.*

The definition of useful templates now directly gives the following corollary.

Corollary 4.1 *Let $\mathcal{L}$ be a constructive full AFL. Given $\varrho = (T, \Sigma, n_1, n_2)$ with T regular and $L \in \mathcal{L}$, we can effectively find a regular set of templates T_1 such that if $\varrho_1 = (T_1, \Sigma, n_1, n_2)$ then $\varrho_1^{(*)}(L) = \varrho^{(*)}(L)$ and ϱ_1 is useful on L.*

The second auxiliary result from [5] that turns out to be useful is the following:

Proposition 4.2 (Theorem 4.1 of [5]) *If $\mathcal{L}$ is a full AFL, $\varrho = (T, \Sigma, n_1, n_2)$ is a TGR system and $L, T \in \mathcal{L}$, $L \subseteq \Sigma^*$, are such that ϱ is useful on L, then $\varrho^{(*)}(L) \in \mathcal{L}$.*

The proof of Proposition 4.2 in [5] establishes that $\varrho^{(*)}(L)$ is in $\mathcal{L}$ by showing that $\varrho^{(*)}(L)$ is obtained from L using a finite number of operations each of which can be expressed as a composition of AFL operations. Thus the proof gives the following corollary for constructive full AFL's.

Corollary 4.2 *Let $\mathcal{L}$ be a constructive full AFL. Given a TGR system $\varrho = (T, \Sigma, n_1, n_2)$ where $T \in \mathcal{L}$, an intial language $L \in \mathcal{L}$, $L \subseteq \Sigma^*$, such that ϱ is useful on L, we can effectively construct (a Turing machine in* TM$[P_{\mathcal{L}}]$ *for) $\varrho^{(*)}(L) \in \mathcal{L}$.*

Now we are ready to prove the main result of this section.

Theorem 4.1 *Let $\mathcal{L}$ be a constructive full AFL. Given $L \in \mathcal{L}$ and a TGR system $\varrho = (T, \Sigma, n_1, n_2)$ where T is regular, we can effectively construct (a Turing machine in* TM$[P_{\mathcal{L}}]$ *for) the language $\varrho^{(*)}(L)$ (which is always in $\mathcal{L}$).*

Proof. By Corollary 4.1 we can effectively find a regular set of templates T_1 ($\subseteq T$) such that if $\varrho_1 = (T_1, \Sigma, n_1, n_2)$ then ϱ_1 is useful on L and

$$\varrho_1^{(*)}(L) = \varrho^{(*)}(L). \tag{17}$$

Since any full AFL contains all regular languages, we have $T_1 \in \mathcal{L}$. Now, by Corollary 4.2, given L and T_1 we can effectively construct a Turing machine for $\varrho_1^{(*)}(L)$ and we are done by (17). ■

Since the regular and the context-free languages are examples of constructive full AFL's, as particular cases Theorem 4.1 implies that if ϱ is a TGR system with a regular set of templates, given a finite automaton (respectively, a pushdown automaton) for a language L, we can effectively construct a finite automaton (respectively, a pushdown automaton) for the language $\varrho^{(*)}(L)$.

Finally, it can be noted that Theorem 4.1 relies on Corollary 4.1 and Corollary 3.4 (that in turn relies on Corollary 3.2), and these results use brute-force constructions that basically enumerate all words up to a given length. It would be interesting to know whether for a regular initial language L and a regular set of templates there is some reasonably efficient algorithm to construct a (not necessarily deterministic) finite automaton for $\varrho^{(*)}(L)$.

References

[1] A. V. Aho, Indexed grammars – an extenion of context-free grammars, *Journal of the ACM* **15** (1968) 647–671.

[2] M. Daley, O. Ibarra and L. Kari, Closure properties and decision questions concerning some language classes under ciliate bio-operations. *Theoretical Computer Science* **306** (2003) 19–38.

[3] M. Daley, O. Ibarra, L. Kari, I. McQuillan and K. Nakano, Closure and decision properties of some language classes under ld and dlad bio-operations. *Journal of Automata, Languages and Combinatorics* **8** (2003) 477–498.

[4] M. Daley and I. McQuillan, Template-guided DNA recombination, *Theoretical Computer Science* **330** (2005) 237–250.

[5] M. Daley and I. McQuillan, Useful templates and iterated template-guided DNA recombination in ciliates, *Theory of Computing Systems,* in press. E-print available doi:10.1007/s00224-005-1206-6

[6] S. Eilenberg, *Automata, Languages, and Machines, volume A.* Academic Press, Inc., New York, NY, 1974.

[7] J. Engelfriet, Iterated stack automata and complexity classes, *Information and Computation* **95** (1991) 21–75.

[8] S. Ginsburg, *Algebraic and Automata-Theoretic Properties of Formal Languages,* North-Holland, Amsterdam 1975.

[9] T. Head and D. Pixton, Splicing and regularity, to appear. (Available at `www.math.binghamton.edu/dennis/Papers`.)

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

[11] A. Okhotin, Boolean grammars, Proceedings of Developments in Language Theory, DLT 2003 (Szeged, Hungary, July 7–11, 2003), Lecture Notes in Computer Science **2710**, Springer, 2003, pp. 398–410. Full version to appear in *Information and Computation.*

[12] A. Okhotin, On the closure properties of linear conjunctive languages, *Theoretical Computer Science* **299** (2003) 663–685.

[13] D.M. Prescott. Genome Gymnastics: Unique modes of DNA evolution and processing in ciliates, *Nature Reviews Genetics* **1** (2000) 191–198.

[14] D.M. Prescott, A. Ehrenfeucht and G. Rozenberg, Template guided recombination for IES elimination and unscrambling of genes in stichotrichous ciliates, *J. Theoretical Biology* **222** (2003) 323-330.

[15] G. Rozenberg and A. Salomaa (Eds.), *Handbook of Formal Languages,* Vols. 1–3, Springer Verlag, 1997.