The Formal Language Theory Column

by

Arto Salomaa
Turku Centre for Computer Science
University of Turku
Lemminkäisenkatu 14, 20520 Turku, Finland
asalomaa@it.utu.fi

More Words on Trajectories

Michael Domaratzki
Jodrey School of Computer Science, Acadia University,
Wolfville, NS B4P 2R6 Canada
mike.domaratzki@acadiau.ca

Abstract

We survey recent results on the use of trajectories as a tool for modelling language operations and other, related objects. Many applications of the concept of trajectories have been developed since their introduction by Mateescu, Rozenberg and Salomaa in 1996. Areas which have seen activity include the theory of codes, language equations, modelling noisy channels, grammar models and DNA code-word design. We survey each of these areas.

1 Introduction

Trajectories, introduced by Mateescu, Rozenberg and Salomaa [79], are a manner in which a language operation is defined by a fixed language, used as a parameter.

In this way, we define infinitely many language operations as each language over the trajectory alphabet defines a binary language operation.

The simplicity and elegance of the concept of trajectories has lead to applications and generalizations in a number of research areas, including language equations, theory of codes, DNA code-word design, modelling noisy channels and other applied areas of formal language theory.

In this survey, we review several recent research areas related to the use of trajectories. For a survey of early results on trajectories, we refer the reader to the Formal Language Theory column "Words on Trajectories" by Mateescu [78].

2 Shuffle on Trajectories

We begin with the fundamental definition for the investigation of trajectories, that of *shuffle on trajectories*. The shuffle on trajectories operation is a method for specifying the ways in which two input words may be merged, while preserving the order of symbols in each word, to form a result. Each trajectory $t \in \{0, 1\}^*$ with $|t|_0 = n$ and $|t|_1 = m$ specifies the manner in which we can form the shuffle on trajectories of two words of length n (as the left input word) and m (as the right input word). The word resulting from the shuffle along t will have a letter from the left input word in position i if the i-th symbol of t is 0, and a letter from the right input word in position i if the i-th symbol of t is 1.

We now give the formal definition of shuffle on trajectories, originally due to Mateescu *et al.* [79]. Shuffle on trajectories is defined by first defining the shuffle of two words x and y over an alphabet Σ on a trajectory t, a word over $\{0, 1\}$. We denote the shuffle of x and y on trajectory t by $x \sqcup\!\sqcup_t y$.

If $x = ax'$, $y = by'$ (with $a, b \in \Sigma$) and $t = et'$ (with $e \in \{0, 1\}$), then

$$x \sqcup\!\sqcup_{et'} y = \begin{cases} a(x' \sqcup\!\sqcup_{t'} by') & \text{if } e = 0; \\ b(ax' \sqcup\!\sqcup_{t'} y') & \text{if } e = 1. \end{cases}$$

If $x = ax'$ ($a \in \Sigma$), $y = \epsilon$ and $t = et'$ ($e \in \{0, 1\}$), then

$$x \sqcup\!\sqcup_{et'} \epsilon = \begin{cases} a(x' \sqcup\!\sqcup_{t'} \epsilon) & \text{if } e = 0; \\ \emptyset & \text{otherwise.} \end{cases}$$

If $x = \epsilon$, $y = by'$ ($b \in \Sigma$) and $t = et'$ ($e \in \{0, 1\}$), then

$$\epsilon \sqcup\!\sqcup_{et'} y = \begin{cases} b(\epsilon \sqcup\!\sqcup_{t'} y') & \text{if } e = 1; \\ \emptyset & \text{otherwise.} \end{cases}$$

We let $x \sqcup\!\sqcup_\epsilon y = \emptyset$ if $\{x, y\} \neq \{\epsilon\}$. Finally, if $x = y = \epsilon$, then $\epsilon \sqcup\!\sqcup_t \epsilon = \epsilon$ if $t = \epsilon$ and $\emptyset$ otherwise.

It is not difficult to see that if $t = \prod_{i=1}^{n} 0^{j_i} 1^{k_i}$ for some $n \geq 0$ and $j_i, k_i \geq 0$ for all $1 \leq i \leq n$, then we have that

$$x \sqcup\!\sqcup_t y = \{\prod_{i=1}^{n} x_i y_i \ : \ x = \prod_{i=1}^{n} x_i, y = \prod_{i=1}^{n} y_i, \\ \text{with } |x_i| = j_i, |y_i| = k_i \text{ for all } 1 \leq i \leq n\}$$

if $|x| = |t|_0$ and $|y| = |t|_1$ and $x \sqcup\!\sqcup_t y = \emptyset$ if $|x| \neq |t|_0$ or $|y| \neq |t|_1$.

We extend shuffle on trajectories to *sets* $T \subseteq \{0,1\}^*$ *of trajectories* as follows:

$$x \sqcup\!\sqcup_T y = \bigcup_{t \in T} x \sqcup\!\sqcup_t y.$$

Further, for $L_1, L_2 \subseteq \Sigma^*$, we define

$$L_1 \sqcup\!\sqcup_T L_2 = \bigcup_{\substack{x \in L_1 \\ y \in L_2}} x \sqcup\!\sqcup_T y.$$

Consider the following examples. We can see that if $T = 0^*1^*$, we have that $L_1 \sqcup\!\sqcup_T L_2 = L_1 L_2$, i.e., $T = 0^*1^*$ gives the concatenation operation. If $T = (0+1)^*$, then $L_1 \sqcup\!\sqcup_T L_2 = L_1 \sqcup\!\sqcup L_2$, i.e., $T = \{0,1\}^*$ gives the shuffle operation. This is the least restrictive set of trajectories. If $T = 0^*1^*0^*$, then $\sqcup\!\sqcup_T$ is the insertion operation $\leftarrow$ (see, e.g, Kari [50]) which is defined by $x \leftarrow y = \{x_1 y x_2 \ : \ x_1, x_2 \in \Sigma^*, x_1 x_2 = x\}$ for all $x, y \in \Sigma^*$. See Mateescu *et al.* [79] for the fundamental study of shuffle on trajectories.

3 Deletion along Trajectories

We now consider deletion on trajectories, an important addition to the study of trajectories and related areas. The concept of deletion along trajectories was independently introduced by the author [17, 22] and Kari and Sosík [55, 56]. The primary motivation for the introduction of deletion on trajectories is to define an "inverse" to shuffle on trajectories, in a sense we will see below. Intuitively, deletion on trajectories uses the trajectories to model language operations which delete an occurrence of the right argument from the left argument in a controlled, scattered way.

Let $x, y \in \Sigma^*$ be words with $x = ax'$, $y = by'$ $(a, b \in \Sigma)$. Let t be a word over $\{i, d\}$ such that $t = et'$ with $e \in \{i, d\}$. Then we define $x \rightsquigarrow_t y$, the deletion of y from x along trajectory t, as follows:

$$x \rightsquigarrow_t y = \begin{cases} a(x' \rightsquigarrow_{t'} by') & \text{if } e = i; \\ x' \rightsquigarrow_{t'} y' & \text{if } e = d \text{ and } a = b; \\ \emptyset & \text{otherwise.} \end{cases}$$

Also, if $x = ax'$ ($a \in \Sigma$) and $t = et'$ ($e \in \{i, d\}$), then

$$x \rightsquigarrow_t \epsilon = \begin{cases} a(x' \rightsquigarrow_{t'} \epsilon) & \text{if } e = i; \\ \emptyset & \text{otherwise.} \end{cases}$$

If $x \neq \epsilon$, then $x \rightsquigarrow_\epsilon y = \emptyset$. Further, $\epsilon \rightsquigarrow_t y = \epsilon$ if $t = y = \epsilon$. Otherwise, $\epsilon \rightsquigarrow_t y = \emptyset$.

Let $T \subseteq \{i, d\}^*$. Then

$$x \rightsquigarrow_T y = \bigcup_{t \in T} x \rightsquigarrow_t y.$$

We extend this to languages as expected: Let $L_1, L_2 \subseteq \Sigma^*$ and $T \subseteq \{i, d\}^*$. Then

$$L_1 \rightsquigarrow_T L_2 = \bigcup_{\substack{x \in L_1 \\ y \in L_2}} x \rightsquigarrow_T y.$$

We consider the following examples of deletion along trajectories:

(a) if $T = i^* d^*$, then $\rightsquigarrow_T = /$, the right-quotient operation;
(b) if $T = d^* i^*$, then $\rightsquigarrow_T = \backslash$, the left-quotient operation;
(c) if $T = i^* d^* i^*$, then $\rightsquigarrow_T = \longrightarrow$, the deletion operation (see, e.g., Kari [49, 50]);
(d) if $T = (i + d)^*$, then $\rightsquigarrow_T = \rightsquigarrow$, the scattered deletion operation (see, e.g., Ito *et al.* [41]);
(e) if $T = d^* i^* d^*$, then $\rightsquigarrow_T = \rightleftharpoons$, the bi-polar deletion operation (see, e.g., Kari [50]);
(f) let $k \geq 0$ and $T_k = i^* d^* i^{\leq k}$. Then $\rightsquigarrow_{T_k} = \longrightarrow^k$, the k-deletion operation (see, e.g., Kari and Thierrin [57]).

We now recall some of the closure properties of deletion along trajectories.

Theorem 3.1. *Let Σ be an alphabet. There exist weak codings $\rho_1, \rho_2, \tau, \varphi$ and a regular language R such that for all $L_1, L_2 \subseteq \Sigma^*$ and all $T \subseteq \{i, d\}^*$,*

$$L_1 \rightsquigarrow_T L_2 = \varphi\left(\rho_1^{-1}(L_1) \cap \rho_2^{-1}(L_2) \cap \tau^{-1}(T) \cap R\right).$$

Corollary 3.2. *Let $\mathcal{L}$ be a cone. Then for all L_1, L_2, T such that two are regular languages and the third is from $\mathcal{L}$, $L_1 \rightsquigarrow_T L_2 \in \mathcal{L}$.*

3.1 Non-regular Trajectories Preserving Regularity

Consider the following result of Mateescu *et al.* [79, Thm. 5.1]: if $L_1 \sqcup\!\sqcup_T L_2$ is regular for all regular languages L_1, L_2, then T is regular. This result is clear upon noting that for all T, $0^* \sqcup\!\sqcup_T 1^* = T$.

However, we note that the same result does not hold if we replace "shuffle on trajectories" by "deletion along trajectories". As motivation, we begin with a basic example. Let Σ be an alphabet and $H = \{i^n d^n \ : \ n \geq 0\}$. Note that

$$R_1 \rightsquigarrow_H R_2 = \{x \in \Sigma^* \ : \ \exists y \in R_2 \text{ such that } xy \in R_1 \text{ and } |x| = |y|\}.$$

We can establish directly (by constructing an NFA) that for all regular languages $R_1, R_2 \subseteq \Sigma^*$, the language $R_1 \rightsquigarrow_H R_2$ is regular. However, H itself is not regular.

We remark that $R_1 \rightsquigarrow_H R_2$ is similar to proportional removals studied by Stearns and Hartmanis [90], Amar and Putzolu [1, 2], Seiferas and McNaughton [86], Kosaraju [60, 61, 62], Kozen [63], Zhang [96], the author [16], Berstel *et al.* [4], and others. In particular, we note the case of $\frac{1}{2}(L)$, given by

$$\frac{1}{2}(L) = \{x \in \Sigma^* : \exists y \in \Sigma^* \text{ such that } xy \in L \text{ and } |x| = |y|\}.$$

The operation $\frac{1}{2}(L)$ is one of a class of operations which preserve regularity. Seiferas and McNaughton completely characterize those binary relations $r \subseteq \mathbb{N}^2$ such that the operation

$$P(L, r) = \{x \in \Sigma^* : \exists y \in \Sigma^* \text{ such that } xy \in L \text{ and } r(|x|, |y|)\}$$

preserves regularity.

Recall that a set A is ultimately periodic (u.p.) if there exist $n_0, p \in \mathbb{N}$, $p > 0$, such that for all $x \geq n_0$, $x \in I \iff x + p \in I$. Call a binary relation $r \subseteq \mathbb{N}^2$ *u.p.-preserving* if A u.p. implies $r^{-1}(A) = \{i \ : \ \exists j \in A \text{ such that } r(i, j)\}$ is also u.p. Then, the binary relations r such that $P(\cdot, r)$ preserves regularity are precisely the u.p.-preserving relations [86]. This was extended to deletion on trajectories [17, 22]:

Theorem 3.3. *Let $r \subseteq \mathbb{N}^2$ be a binary relation and $H_r = \{i^n d^m \ : \ r(n, m)\}$. The operation $\rightsquigarrow_{H_r}$ is regularity-preserving if and only if r is u.p.-preserving.*

Theorem 3.3 has been extended by the author [17, 22] to cover other bounded sets of trajectories.

3.2 Algebraic Properties

Kari and Sosík [56] show the following results concerning algebraic properties of deletion along trajectories:

Theorem 3.4. *Let $T \subseteq \{i, d\}^*$. The following three conditions are equivalent:*

(a) *the operation* $\rightsquigarrow_T$ *is commutative;*
(b) $L_1 \rightsquigarrow_T L_2 \subseteq \{\epsilon\}$ *for all languages* L_1, L_2*;*
(c) $T \subseteq d^*$.

Corollary 3.5. *Given a context-free set of trajectories* $T \subseteq \{i, d\}^*$*, it is decidable whether* $\rightsquigarrow_T$ *is a commutative operation.*

Theorem 3.6. *For a set of trajectories* $T \subseteq \{i, d\}^*$*, the following two conditions are equivalent.*

(i) *For all* $t_1 \in 1^m \sqcup\!\sqcup 0^n$, $t_2 \in 1^n \sqcup\!\sqcup 0^i$ *and* $t_3 \in 1^j \sqcup\!\sqcup 0^{m+n}$, $i, j, m, n \in \mathbb{N}$,

(a) $m > 0$ *and* $t_1 \in T$ *implies* $t_2 \notin T$.
(b) $m > 0$ *and* $t_1 \in T$ *implies* $t_3 \notin T$.
(c) $t_1 \in T$ *and* $0^m \in T$ *implies* $0^n \in T$.
(d) $t_1 \in T$ *and* $0^n \in T$ *implies* $0^m \in T$.

(ii) $\rightsquigarrow_T$ *is an associative operation.*

However, the associated decidability problem is apparently open:

Open Problem 3.7. *Given a regular set of trajectories* $T \subseteq \{i, d\}^*$*, is it decidable whether* $\rightsquigarrow_T$ *is an associative operation? What if* T *is context-free?*

3.3 Commutative Closure

Recall that the commutative closure of a word $w \in \Sigma^*$ is the set $com(w) = \{x : \forall a \in \Sigma, |w|_a = |x|_a\}$. The commutative closure of a language L is the union of the commutative closure of each word in L. Note that the operation *com* does not preserve regularity: $com((ab)^*) = \{x \in \{a, b\}^* : |x|_a = |x|_b\}$.

The author, Mateescu, K. Salomaa and Yu [23] have shown that if T is complete and associative, then for all languages L, we can define an a congruence relation $\sim_{L,T}$. Then the factor monoid of $\mathcal{L}_T = (\mathcal{P}(\Sigma^*), \sqcup\!\sqcup_T, \epsilon)$ with respect to $\sim_{L,T}$ is finite if and only if $com(L)$ is a regular language [23].

4 Language Equations

The immediate consequence of the introduction of deletion on trajectories is its application to language equations. A language equation is an equality involving constant languages, language operations and unknowns. We refer the reader to Leiss [67] for an introduction to the theory of language equations.

There are several facets to research on language equations. For instance, given a language equation or a system of language equations, we can seek to characterize the solutions to the equation, or simply decide if a solution exists. In the research on language equations involving shuffle on trajectories, the focus of research has been the latter task.

4.1 Zero Variable Equations

The most simple language equations are those without any variables at all. It follows from the closure properties of $\sqcup\!\sqcup_T$ and $\leadsto_T$ that it is decidable if either $R_1 \sqcup\!\sqcup_T R_2 = R_3$ or $R_1 \leadsto_T R_2 = R_3$ holds if all of R_1, R_2 and R_3 are regular languages and T is regular. Below, we review decidability results when the languages and set of trajectories are not all regular.

Let $\Psi_a(T) = \{|t|_a : t \in T\}$, where a is a letter and $|t|_a$ denotes the number of zeroes in t. Kari and Sosík [56] establish the following elegant result:

Theorem 4.1. *Let T be an arbitrary but fixed regular set of trajectories. The problem "Is $L_1 \sqcup\!\sqcup_T L_2 = R$?" is decidable for a CFL L_1 and regular languages L_2, R if and only if $\Psi_0(T)$ is finite.*

The analogous result for deletion on trajectories (i.e., $\Psi_i(T)$ is finite) also holds [56]. Further, if L_1 is taken to be a regular language and L_2 is taken to be a CFL, then the necessary and sufficient conditions are that $\Psi_1(T)$ is finite [56].

We can contrast the above results with the following surprising result which was established by the author and K. Salomaa [26].

Theorem 4.2. *Let $T = \{0^i 1^r 0^j 1^s 0^k : r \neq s, r, s \text{ are odd }, i, j, k \geq 0\}$. Then for a given alphabet Σ and given regular languages $S, R_1, R_2 \subseteq \Sigma$, it is undecidable whether or not $S = R_1 \sqcup\!\sqcup_T R_2$.*

The result is interesting since it is an undecidability result about regular languages: given three regular languages, it is undecidable whether the given equality holds; the context-free set of trajectories T if fixed (and is not part of the input).

4.2 One Variable Equations

One variable equations involving shuffle and deletion on trajectories have been studied by the author [17, 22], Kari and Sosík [55, 56] and the author and K. Salomaa [25]. We summarize these results below.

By general results of Kari on the inverse of language operations [50], the following result on one variable language equations involving shuffle and deletion along trajectories is immediate.

Theorem 4.3. *Let L, R be regular languages. Then it is decidable whether any of the following equations have a solution X, and if so, the maximal solution (under inclusion) is an effectively constructible regular language:*

(a) $X \sqcup\!\sqcup_T L = R$ where $T \subseteq \{0,1\}^*$ is regular.
(b) $L \sqcup\!\sqcup_T X = R$ where $T \subseteq \{0,1\}^*$ is regular.
(c) $X \rightsquigarrow_T L = R$ where $T \subseteq \{i,d\}^*$ is regular.
(d) $L \rightsquigarrow_T X = R$ where $T \subseteq \{i,d\}^*$ is regular.

We can also examine the question of if it is decidable whether a set of trajectories can be found to satisfy a fixed language equation:

Theorem 4.4. *Let $L_1, L_2, R \subseteq \Sigma^*$ be regular languages. Then it is decidable whether*

(a) there exists a set $T \subseteq \{0,1\}^*$ of trajectories with $L_1 \sqcup\!\sqcup_T L_2 = R$.
(b) there exists a set $T \subseteq \{i,d\}^*$ of trajectories with $L_1 \rightsquigarrow_T L_2 = R$.

We now turn to undecidability. Let $\Pi_0, \Pi_1 : \{0,1\}^* \to \{0,1\}^*$ be the projections given by $\Pi_0(0) = 0, \Pi_0(1) = \epsilon$ and $\Pi_1(1) = 1, \Pi_1(0) = \epsilon$. We say that $T \subseteq \{0,1\}^*$ is *left-enabling* (resp., *right-enabling*) if $\Pi_0(T) = 0^*$ (resp., $\Pi_1(T) = 1^*$). It is undecidable whether the corresponding equation has a solution:

Theorem 4.5. *Fix $T \subseteq \{0,1\}^*$ to be a regular set of left-enabling (resp., right-enabling) trajectories. For a given LCFL L and regular language R, it is undecidable whether or not $L \sqcup\!\sqcup_T X = R$ (resp., $X \sqcup\!\sqcup_T L = R$) has a solution X.*

Say that a set of trajectories is left-preserving (resp., right-preserving) if $T \supseteq 0^*$ (resp., $T \supseteq 1^*$). We can give an incomparable result which removes the condition that T must be regular, but must strengthen the conditions on words in T:

Theorem 4.6. *Fix a left-preserving (resp., right-preserving) set of trajectories $T \subseteq \{0,1\}^*$. Given an LCFL L and a regular language R, it is undecidable whether there exists a language X such that $L \sqcup\!\sqcup_T X = R$ (resp., $X \sqcup\!\sqcup_T L = R$).*

By an extension of Theorem 4.2, we can also show that there exists a fixed context-free set of trajectories T such that, on input R_1, R_2 (regular languages), deciding whether or not there exists a language X such that $R_1 = X \sqcup\!\sqcup_T R_2$ is undecidable [26]. We also have the following undecidability results concerning the existence of sets of trajectories [25, 22]:

Theorem 4.7. *Given an LCFL L and regular languages R_1, R_2, it is undecidable whether there exists $T \subseteq \{0,1\}^*$ such that (a) $R_1 \sqcup\!\sqcup_T R_2 = L$, (b) $R_1 \sqcup\!\sqcup_T L = R_2$ or (c) $L \sqcup\!\sqcup_T R_1 = R_2$.*

Given an LCFL L and regular languages R_1, R_2, it is undecidable whether there exists $T \subseteq \{i,d\}^*$ such that (a) $R_1 \rightsquigarrow_T R_2 = L$, (b) $L \rightsquigarrow_T R_1 = R_2$, or (c) $R_1 \rightsquigarrow_T L = R_2$.

4.3 Two Variable Equations

We can also consider language equations with two variables. As an example, consider the well-studied language equation

$$L = X_1 X_2 \tag{1}$$

where L is a fixed language and X_1, X_2 are unknown. Study of this equation has been undertaken by Conway [10], Kari and Thierrin [59], Salomaa and Yu [85] and Choffrut and Karhumäki [9]. In particular, it is known that if L is a regular language, we can determine if a solution to (1) exists. Furthermore, it is known that if a solution X_1, X_2 exists, there exists a maximal regular solution; i.e., there exists R_1, R_2 such that $X_i \subseteq R_i$ for $i = 1, 2$ and $L = R_1 R_2$.

For language equations of the form $L = X_1 \sqcup\!\sqcup_T X_2$, unlike the case of one-variable equations, we do not have a general result which applies to all regular sets of trajectories T. However, a large class of trajectories are covered by the author and K. Salomaa [25, 22].

Recall that a language $L \subseteq \Sigma^*$ is *bounded* if there exist $w_1, w_2, \ldots, w_n \in \Sigma^*$ such that $L \subseteq w_1^* w_2^* \cdots w_n^*$. We say that L is *letter-bounded* if $w_i \in \Sigma$ for all $1 \leq i \leq n$. The following result is due to the author and K. Salomaa [25, 22].

Theorem 4.8. *Let $T \subseteq \{0, 1\}^*$ be a letter-bounded regular set of trajectories. Then given a regular language R, it is decidable whether there exist X_1, X_2 such that $X_1 \sqcup\!\sqcup_T X_2 = R$.*

As we have mentioned, this result was known for catenation, $T = 0^*1^*$. However, it also holds for, e.g., the following operations: insertion ($0^*1^*0^*$), k-insertion ($0^*1^*0^{\leq k}$ for fixed $k \geq 0$), and bi-catenation ($1^*0^* + 0^*1^*$).

We also note that if the equation $X_1 \sqcup\!\sqcup_T X_2 = R$ has a solution, where R is a regular language and T is a letter-bounded regular set of trajectories, then the equation also has solution $Y_1 \sqcup\!\sqcup_T Y_2 = R$ where Y_1, Y_2 are regular languages. This result is well-known for $T = 0^*1^*$ (see, e.g., Choffrut and Karhumäki [9]).

For $T = (0 + 1)^*$, the two-variable decomposition problem is open [7]:

Open Problem 4.9. *Given a regular language R, is it decidable whether there exist X_1, X_2 (with $X_1, X_2 \neq \{\epsilon\}$) such that $R = X_1 \sqcup\!\sqcup X_2$?*

Recall that a language L is *k-thin* if $|L \cap \Sigma^n| \leq k$ for all $n \geq 0$. The following problem is also open:

Open Problem 4.10. *Given a k-thin set of trajectories $T \subseteq \{0, 1\}^*$, is it decidable, given a regular language R, whether R has a shuffle decomposition with respect to T?*

The author and K. Salomaa [26] have shown that if $T \subseteq \{0,1\}^*$ is a fixed 1-thin set of trajectories, given a regular language R, it is decidable whether R has a shuffle decomposition with respect to T. However, even for 2-thin sets of trajectories, the problem remains open [26].

We can also note that if the left and right operand are restricted to be the same, the resulting language equation problem remains decidable if the set of trajectories is regular and letter-bounded [25, 22]:

Theorem 4.11. *Fix a letter-bounded regular set of trajectories $T \subseteq \{0,1\}^*$. Then it is decidable whether there exists a solution X to the equation $X \sqcup\!\sqcup_T X = R$ for a given regular language R.*

We now turn to undecidability. It has been shown [7] that it is undecidable whether a context-free language has a nontrivial shuffle decomposition with respect to the set of trajectories $\{0,1\}^*$. This result can be extended for arbitrary complete regular sets trajectories [25]. (Note that if T is a complete set of trajectories, then any language L has decompositions $L \sqcup\!\sqcup_T \{\epsilon\}$ and $\{\epsilon\} \sqcup\!\sqcup_T L$. Below we exclude these trivial decompositions; all other decompositions of L are said to be nontrivial.)

Theorem 4.12. *Let T be any fixed complete regular set of trajectories. For a given context-free language L it is undecidable whether or not there exist languages $X_1, X_2 \neq \{\epsilon\}$ such that $L = X_1 \sqcup\!\sqcup_T X_2$.*

We also note the following open problem [26]:

Open Problem 4.13. *Is it possible to construct a fixed context-free set of trajectories T such that it is undecidable whether there exist languages $X_1, X_2 \neq \{\epsilon\}$ such that $L = X_1 \sqcup\!\sqcup_T X_2$?*

Also open are other forms of language equation. We mention only two here:

Open Problem 4.14. *Find necessary and sufficient conditions on sets of trajectories T_1, T_2 so that, given a regular language R, it is decidable whether there exist nontrivial languages X_1, X_2, X_3 satisfying $(X_1 \sqcup\!\sqcup_{T_1} X_2) \sqcup\!\sqcup_{T_2} X_3 = R$.*

Open Problem 4.15. *Given regular languages R_1, R_2, is it decidable whether there exists nontrivial languages X_1, T (with $T \subseteq \{0,1\}^*$) such that $X_1 \sqcup\!\sqcup_T R_1 = R_2$ or $R_1 \sqcup\!\sqcup_T X_1 = R_2$?*

5 Splicing on Routes

The notion of shuffle on trajectories was extended by Mateescu [77] to encompass certain splicing operations. This extension is called *splicing on routes*. Splicing on routes is a proper extension of shuffle on trajectories, and also encompasses several unary operations. Bel-Enguix *et al.* also use the concept of splicing on routes to model dialog in natural language [3].

Splicing on routes was introduced by Mateescu [77] to model generalizations of the crossover splicing operation (see Mateescu [77] for a definition of the crossover splicing operation). Splicing on routes generalizes the crossover splicing operation by specifying a set T of routes which restricts the way in which splicing can occur. The result is that specific sets of routes can simulate not only the crossover operation, but also such operations on DNA such as the *simple splicing* and the *equal-length crossover* operations (see Mateescu for details and definitions of these operations [77]).

We now define the concept of *splicing on routes*, and note the difference between deletion along trajectories from splicing on routes, which allows discarding letters from either input word. In particular, a *route* is a word t specified over the alphabet $\{0, \overline{0}, 1, \overline{1}\}$, where, informally, $0, 1$ means insert the letter from the appropriate word, and $\overline{0}, \overline{1}$ means discard that letter and continue.

Formally, let $x, y \in \Sigma^*$ and $t \in \{0, \overline{0}, 1, \overline{1}\}^*$. We define the splicing of x and y, denoted $x \bowtie_t y$ recursively as follows: if $x = ax'$, $y = by'$ $(a, b \in \Sigma)$ and $t = ct'$ $(c \in \{0, \overline{0}, 1, \overline{1}\})$, then

$$x \bowtie_{ct'} y = \begin{cases} a(x' \bowtie_{t'} y) & \text{if } c = 0; \\ (x' \bowtie_{t'} y) & \text{if } c = \overline{0}; \\ b(x \bowtie_{t'} y') & \text{if } c = 1; \\ (x \bowtie_{t'} y') & \text{if } c = \overline{1}. \end{cases}$$

If $x = ax'$ and $t = ct'$, where $a \in \Sigma$ and $c \in \{0, \overline{0}, 1, \overline{1}\}$, then

$$x \bowtie_{ct'} \epsilon = \begin{cases} a(x' \bowtie_{t'} \epsilon) & \text{if } c = 0; \\ (x' \bowtie_{t'} \epsilon) & \text{if } c = \overline{0}; \\ \emptyset & \text{otherwise.} \end{cases}$$

If $y = by'$ and $t = ct'$, where $a \in \Sigma$ and $c \in \{0, \overline{0}, 1, \overline{1}\}$, then

$$x \bowtie_{ct'} \epsilon = \begin{cases} a(x' \bowtie_{t'} \epsilon) & \text{if } c = 0; \\ (x' \bowtie_{t'} \epsilon) & \text{if } c = \overline{0}; \\ \emptyset & \text{otherwise.} \end{cases}$$

If $y = by'$ and $t = ct'$, where $a \in \Sigma$ and $c \in \{0, \overline{0}, 1, \overline{1}\}$, then

$$\epsilon \bowtie_{ct'} y = \begin{cases} b(\epsilon \bowtie_{t'} y') & \text{if } c = 1; \\ (\epsilon \bowtie_{t'} y') & \text{if } c = \overline{1}; \\ \emptyset & \text{otherwise.} \end{cases}$$

We have $x \bowtie_\epsilon y = \emptyset$ if $\{x, y\} \neq \{\epsilon\}$. Finally, we set $\epsilon \bowtie_t \epsilon = \epsilon$ if $t = \epsilon$ and $\emptyset$ otherwise. We extend $\bowtie_t$ to sets of routes and languages as expected:

$$\begin{aligned} x \bowtie_T y &= \bigcup_{t \in T} x \bowtie_t y \quad \forall T \subseteq \{0, \overline{0}, 1, \overline{1}\}^*, x, y \in \Sigma^*; \\ L_1 \bowtie_T L_2 &= \bigcup_{x \in L_1, y \in L_2} x \bowtie_T y. \end{aligned}$$

For example, if $x = abc$, $y = cbc$ and $T = \{010011, 0\overline{1}0\overline{0}11\}$, then $x \bowtie_T y = \{acbcbc, abbc\}$.

It turns out that splicing on routes can be simulated by a combination of shuffle on trajectories and deletion along trajectories [22]:

Theorem 5.1. *There exist weak codings $\pi_1, \pi_2 : \{0, 1, \overline{0}, \overline{1}\}^* \to \{i, d\}^*$ and a weak coding $\pi_3 : \{0, 1, \overline{0}, \overline{1}\}^* \to \{0, 1\}^*$ such that for all $t \in \{0, \overline{0}, 1, \overline{1}\}^*$, and for all $x, y \in \Sigma^*$, we have*

$$x \bowtie_t y = (x \rightsquigarrow_{\pi_1(t)} \Sigma^*) \sqcup\!\sqcup_{\pi_3(t)} (y \rightsquigarrow_{\pi_2(t)} \Sigma^*).$$

Corollary 5.2. *There exist weak codings $\pi_1, \pi_2 : \{0, 1, \overline{0}, \overline{1}\}^* \to \{i, d\}^*$ and $\pi_3 : \{0, 1, \overline{0}, \overline{1}\}^* \to \{0, 1\}^*$ such that for all $T \subseteq \{0, \overline{0}, 1, \overline{1}\}^*$ and $L_1, L_2 \subseteq \Sigma^*$,*

$$L_1 \bowtie_T L_2 = \bigcup_{t \in T} (L_1 \rightsquigarrow_{\pi_1(t)} \Sigma^*) \sqcup\!\sqcup_{\pi_3(t)} (L_2 \rightsquigarrow_{\pi_2(t)} \Sigma^*).$$

Unfortunately, the identity

$$L_1 \bowtie_T L_2 = (L_1 \rightsquigarrow_{\pi_1(T)} \Sigma^*) \sqcup\!\sqcup_{\pi_3(T)} (L_2 \rightsquigarrow_{\pi_2(T)} \Sigma^*)$$

does not hold in general, even if L_1, L_2 are singletons and $|T| = 2$. For example, if $L_1 = \{ab\}, L_2 = \{cd\}$ and $T = \{\overline{0}01\overline{1}, 0\overline{0}\overline{1}1\}$, then

$$\begin{aligned} L_1 \bowtie_T L_2 &= \{bc, ad\}; \\ (L_1 \rightsquigarrow_{\pi_1(T)} \Sigma^*) \sqcup\!\sqcup_{\pi_3(T)} (L_2 \rightsquigarrow_{\pi_2(T)} \Sigma^*) &= \{ac, ad, bc, bd\}. \end{aligned}$$

However, if T is a *unary* set of routes, by which we mean that $T \subseteq \{0, \overline{0}\}^* \overline{1}^*$, then we have the following result [22]:

Corollary 5.3. *Let $T \subseteq \{0, \overline{0}\}^* \overline{1}^*$. Then for all $L \subseteq \Sigma^*$,*

$$L \bowtie_T \Sigma^* = L \rightsquigarrow_{\pi_1(T)} \Sigma^*.$$

We refer the reader to Mateescu [77] for a discussion of unary operations defined by splicing on routes. As an example, consider that with $T = \{0^n\overline{0}^n : n \geq 0\}\overline{1}^*$, $L \bowtie_T \Sigma^* = \frac{1}{2}(L)$, where $\frac{1}{2}(L)$ was given in Section 3.1.

Bel-Enguix *et al.* [3] have used splicing on routes to model dialogue in natural language. A *dialogue* is the result of two or more actors who alternately communicate. The alternating nature of trajectory-based operations makes it natural to apply them to modeling dialogue. Routes, and in particular the ability to not insert portions of certain actor's input, is important for selecting different pieces of information for different dialogues: "Depending on the conversational goal and on the behaviour of the agents, different acts will be selected " [3]. Bel-Enguix *et al.* [3] note three important classes of routes used in dialogue:

(a) *Imbrication* in a dialogue is the result of applying a route from the set of routes defined by the regular expression $((01) + (\overline{01}))^+$. Imbrication represents the "perfect conversation" where actors strictly alternate.

(b) *Concatenation* in a dialogue is the result of applying a route from the set of route defined by the regular expression $(0 + \overline{0})^+(1 + \overline{1})^+$. According to Bel-Enguix, concatenation has applications to more formal dialogues such as debates and round tables [3].

(c) *Merging* is the result of applying a route which does not fall into the categories of imbrication or concatenation. Merging is more common than imbrication and concatenation in informal dialogues.

Bel-Enguix *et al.* [3] mention several open areas of research relating to the use of splicing on routes for modelling dialogue. In particular, they ask about establishing a mechanism in order to select routes depending on context. This would establish a more semantic operation, in the terminology of Section 6 below.

6 Semantic Shuffle on Trajectories

In the paper which introduced shuffle on trajectories, Mateescu *et al.* make a distinction between *syntactic* and *semantic* operations on words:

> [Shuffle on trajectories is] based on syntactic constraints on the shuffle operations. The constraints are referred to as syntactic constraints since they do not concern properties of the words that are shuffled, or properties of the letters that occur in these words.
>
> Instead, the constraints involve the general strategy to switch from one word to another word. Once such a strategy is defined, the structure of the words that are shuffled does not play any role.

However, constraints that take into consideration the inner structure of the words that are shuffled together are referred to as semantic constraints. [79, p. 2]

The author [19] has introduced a semantic variant of shuffle on trajectories, naturally called *semantic shuffle on trajectories* (SST). The corresponding notion for deletion on trajectories is called *semantic deletion on trajectories* (SDT). The advantages of SST and SDT are that they preserves many of the desirable properties of the usual, syntactic shuffle on trajectories, while being capable of simulating more operations of interest.

The two semantic constructs we introduce are *synchronization* and *content restriction*. Synchronization allows for only one letter to be output for two corresponding, identical symbols in the input words. Content restriction allows a trajectory to specify that a particular letter must appear at a specific point. This is inspired by bio-informatical operations, where operations occur only in the context of certain subsequences of the DNA strand.

Before we define SST, we define the trajectory alphabet. Let $\Gamma = \{0, 1, \sigma\}$. For any alphabet Σ, let $\Gamma_\Sigma = \Gamma \cup (\Gamma \times \Sigma)$. For ease of readability, we denote $[c, a]$ by $\overset{a}{c}$ for all $a \in \Sigma$ and $c \in \Gamma$.

We can now define the SST operation. Let Σ be an alphabet, $t \in \Gamma_\Sigma^*$ and $x, y \in \Sigma^*$. Then the SST of x and y along t, denoted $x \sqcap\!\sqcap_t y$, is defined as follows: If $x = ax'$, $y = by'$ (where $a, b \in \Sigma$, $x', y' \in \Sigma^*$), and $t = ct'$, where $c \in \Gamma_\Sigma$ and $t' \in \Gamma_\Sigma^*$, then

$$x \sqcap\!\sqcap_t y = \begin{cases} a(x' \sqcap\!\sqcap_{t'} y) & \text{if } c \in \{0, \overset{a}{0}\}, \\ b(x \sqcap\!\sqcap_{t'} y') & \text{if } c \in \{1, \overset{b}{1}\}, \\ a(x' \sqcap\!\sqcap_{t'} y') & \text{if } a = b \text{ and } c \in \{\sigma, \overset{a}{\sigma}\}, \\ \emptyset & \text{otherwise.} \end{cases}$$

If $x = ax'$, $y = \epsilon$ and $t = ct'$ then

$$x \sqcap\!\sqcap_t \epsilon = \begin{cases} a(x' \sqcap\!\sqcap_{t'} \epsilon) & \text{if } c \in \{0, \overset{a}{0}\}, \\ \emptyset & \text{otherwise.} \end{cases}$$

If $x = \epsilon$, $y = by'$ and $t = ct'$ then

$$\epsilon \sqcap\!\sqcap_t y = \begin{cases} b(\epsilon \sqcap\!\sqcap_{t'} y') & \text{if } c \in \{1, \overset{b}{1}\}, \\ \emptyset & \text{otherwise.} \end{cases}$$

If $x = y = \epsilon$, then $x \sqcap\!\sqcap_t y = \epsilon$ if $t = \epsilon$ and $\emptyset$ otherwise. Finally, if $\{x, y\} \neq \{\epsilon\}$, then $x \sqcap\!\sqcap_\epsilon y = \emptyset$. If $x, y \in \Sigma^*$ and $T \subseteq \Gamma_\Sigma^*$, then $x \sqcap\!\sqcap_T y = \cup_{t \in T} x \sqcap\!\sqcap_T y$. If $L_1, L_2 \subseteq \Sigma^*$ and

$T \subseteq \Gamma_\Sigma^*$, then $L_1 \sqcap\!\!\sqcap_T L_2 = \cup_{x \in L_1, y \in L_2} x \sqcap\!\!\sqcap_T y$. The associated deletion operation, over the alphabet $\Delta_\Sigma = \Delta \cup (\Delta \times \Sigma)$ (where $\Delta = \{i, d, \sigma\}$), is defined in the natural way [19].

We now consider, given Σ and two sets of trajectories $T_1, T_2 \subseteq \Gamma_\Sigma^*$, whether the operations $\sqcap\!\!\sqcap_{T_1}, \sqcap\!\!\sqcap_{T_2}$ coincide, that is, whether $L_1 \sqcap\!\!\sqcap_{T_1} L_2 = L_1 \sqcap\!\!\sqcap_{T_2} L_2$ for all languages $L_1, L_2 \subseteq \Sigma^*$. If $\sqcap\!\!\sqcap_{T_1}, \sqcap\!\!\sqcap_{T_2}$ represent, in this sense, the same operation, we say that T_1, T_2 are *equivalent* sets of trajectories.

We note that it is possible for two distinct sets of trajectories $T_1, T_2 \subseteq \Gamma_\Sigma^*$ to be equivalent. As a simple example, consider $T_1 = \{\overset{a}{0}\overset{a}{1}\}$ and $T_2 = \{\overset{a}{1}\overset{a}{0}\}$. Note that for $i = 1, 2$,

$$L_1 \sqcap\!\!\sqcap_{T_i} L_2 = \begin{cases} \{aa\} & \text{if } L_1 \cap L_2 \supseteq \{a\}; \\ \emptyset & \text{otherwise.} \end{cases}$$

Thus, T_1, T_2 are equivalent, but not equal.

By using a special case of partial commutation and trace languages (see Diekert and Métivier [15]), we can show that two sets of trajectories $T_1, T_2 \subseteq \Gamma_\Sigma^*$ are equivalent if and only if their corresponding trace languages (under the natural morphism) are equivalent. This implies the following important decidability result:

Theorem 6.1. *Let Σ be an alphabet and $T_1, T_2 \subseteq \Gamma_\Sigma^*$. If T_1, T_2 are regular, it is decidable whether T_1 and T_2 are equivalent.*

We can now consider some examples of the power of SST and SDT. We first note that SST consists of a valid extension of the shuffle on trajectories: if $T \subseteq \{0, 1\}^*$, then $L_1 \sqcap\!\!\sqcap_T L_2 = L_1 \sqcup\!\!\sqcup_T L_2$, the syntactic shuffle on trajectories operation [79]. We also note that if $T = \sigma^*$, then $\sqcap\!\!\sqcap_T = \cap$.

Given the very natural set of trajectories $T = (0 + 1 + \sigma)^*$, $\sqcap\!\!\sqcap_T$ denotes the *infiltration product*, $\uparrow$, see, e.g., Pin and Sakarovitch [81]. The infiltration product is defined as follows: if $x = x_1 \ldots x_n$ is a word of length n and $I = (i_1, i_2, \ldots, i_r)$ is a subsequence of $(1, 2, \ldots, n)$, let $x_I = x_{i_1} x_{i_2} \cdots x_{i_r}$. Then given $x, y \in \Sigma^*$,

$$x \uparrow y = \{z \in \Sigma^* \ : \ \exists I, J \subseteq [|z|] \text{ such that } I \cup J = [|z|], z_I = x \text{ and } z_J = y\}.$$

For example, $ab \uparrow ba = \{aba, bab, baab, baba, abba, abab\}$.

We now show how to use SST and SDT to simulate ciliate bio-operations which have been the subject of recent research in the literature. A model of ciliate bio-operations without circular variants were introduced by Daley *et al.* [11] to mimic the manner in which DNA is unscrambled in the DNA of certain unicellular ciliates in the process of asexual reproduction. Ciliate bio-operations are also investigated by Ehrenfeucht *et al.* [28], Prescott *et al.* [82], and Daley and McQuillan [12] using various approaches.

Daley *et al.* [11] define several language operations which simulate ciliate bio-operations, including synchronized insertion, deletion and bi-polar deletion. Synchronized insertion can be given as follows:

$$\alpha \oplus \beta = \{uavaw \; : \; a \in \Sigma, \alpha = uaw, \beta = va\}.$$

The operation is extended to languages as usual. Let $T = \bigcup_{a \in \Sigma} 0^* \overset{a}{0} 1^* \overset{a}{1} 0^*$. Then for all $L_1, L_2 \subseteq \Sigma^*$, $L_1 \oplus L_2 = L_1 \sqcap_T L_2$. The operations of synchronized deletion and synchronized bi-polar deletion (also defined by Daley *et al.* [11]) can also be simulated by SDT.

Contextual insertion and deletion were introduced by Kari and Thierrin as a simple set of operations which are capable for modelling DNA computing [58]. Let Σ be an alphabet and $[x, y] \in (\Sigma^*)^2$. We call $[x, y]$ a *context*. Then given $v, u \in \Sigma^*$, the $[x, y]$-*contextual insertion* of v into u is given by $u \overleftarrow{\scriptstyle [x,y]} v = \{u_1 xvyu_2 \; : \; u = u_1 xyu_2, u_1, u_2 \in \Sigma^*\}$. Let $C \subseteq (\Sigma^*)^2$. Then

$$u \overleftarrow{\scriptstyle C} v = \bigcup_{[x,y] \in C} u \overleftarrow{\scriptstyle [x,y]} v.$$

The operation $\overleftarrow{\scriptstyle C}$ is extended to languages monotonically as expected. Let $x = x_1 \cdots x_n$ and $y = y_1 \cdots y_m$ be arbitrary words over Σ. Then define

$$T_{[x,y]} = 0^* \prod_{i=1}^{n} \overset{x_i}{0} \, 1^* \prod_{i=1}^{m} \overset{y_i}{0} \, 0^*.$$

We naturally extend this to $T_C = \cup_{[x,y] \in C} T_{[x,y]}$ for all $C \subseteq (\Sigma^*)^2$. Under this definition it is clear that $\sqcap_{T_C} = \overleftarrow{\scriptstyle C}$ for all $C \subseteq (\Sigma^*)^2$.

Kari and Thierrin note that if $C \subseteq (\Sigma^*)^2$ is finite, then the regular and context-free languages are closed under $\overleftarrow{\scriptstyle C}$. As T_C is regular for all finite C, we note that the closure of the regular languages under $\overleftarrow{\scriptstyle C}$ is a consequence of the closure properties of SST. It is known that the CFLs are closed under $\overleftarrow{\scriptstyle C}$ [58]. However, in general, the CFLs are not closed under $\sqcap_T$. This leads to the following open problem:

Open Problem 6.2. *Find necessary and sufficient (language-theoretic) conditions on a set of trajectories $T \subseteq \Gamma_\Sigma^*$ such that the CFLs over Σ are closed under $\sqcap_T$.*

We note that $[x, y]$-contextual deletion and $[x, y]$-contextual bi-polar deletion [58] can be simulated by SDT [19].

Recall that *synchronized shuffle* (see, e.g., Latteux and Roos [66]) is defined as follows. Let Σ_1, Σ_2 be alphabets, not necessarily disjoint. Let $\rho_i : (\Sigma_1 \cup \Sigma_2) \to \Sigma_i$ be the projection onto Σ_i given by $\rho_i(a) = a$ for all $a \in \Sigma_i$ and $\rho_i(a) = \epsilon$ for all $a \in (\Sigma_1 \cup \Sigma_2) - \Sigma_i$.

Let $L_i \subseteq \Sigma_i$ for $i = 1, 2$. Then the synchronized shuffle of L_1 and L_2, denoted $L_1 \parallel L_2$, is given by

$$L_1 \parallel L_2 = \rho_1^{-1}(L_1) \cap \rho_2^{-1}(L_2).$$

Lemma 6.3. *Let Σ_1, Σ_2 be alphabets. Let $T \subseteq \Gamma^*_{\Sigma_1 \cup \Sigma_2}$ be given by*

$$T = ((\bigcup_{a \in \Sigma_1 \cap \Sigma_2} \overset{a}{\sigma}) + (\bigcup_{a \in \Sigma_1 - \Sigma_2} \overset{a}{0}) + (\bigcup_{a \in \Sigma_2 - \Sigma_1} \overset{a}{1}))^*.$$

Then for all L_1, L_2 such that $L_i \subseteq \Sigma_i$ for $i = 1, 2$, $L_1 \parallel L_2 = L_1 \sqcap_T L_2$.

Further examples of operations simulated by SST are described by the author [19].

We note that many of the results on language equations hold for SST. In particular, the following are new results on bio-operations which have not been noted before [19]:

Corollary 6.4. *Let R be a regular language. Then it is decidable whether there exist languages X_1, X_2 such that $R = X_1 \oplus X_2$.*

Corollary 6.5. *Let $C \subseteq (\Sigma^*)^2$ be a finite set of contexts. Let R be a regular language. Then it is decidable whether there exist languages X_1, X_2 such that $R = X_1 \overleftarrow{\scriptstyle C} X_2$.*

7 Descriptional Complexity

Descriptional complexity of formal languages deals with the problems of concise descriptions of languages in terms of generative or accepting devices. For instance, the *(deterministic) state complexity* of a regular language L is the minimal number of states in any deterministic finite automaton accepting L [94]. Nondeterministic state complexity of a regular language is similarly defined (see, e.g., Holzer and Kutrib [36]).

For shuffle on trajectories, Mateescu *et al.* [79] and Harju *et al.* [34] both give proofs that, given a regular set of trajectories T and regular languages L_1, L_2, the operation $L_1 \sqcup\!\sqcup_T L_2$ always yields a regular language. Thus, it is reasonable to consider the state complexity of shuffle on trajectories. The author and K. Salomaa [24] have obtained results in this area. We state an upper bound in terms of nondeterministic state complexity:

Lemma 7.1. *Let L_1, L_2 be regular languages over Σ^* and $T \subseteq \{0, 1\}^*$ be a regular set of trajectories. Then*

$$sc(L_1 \sqcup\!\sqcup_T L_2) \leq 2^{nsc(L_1)nsc(L_2)nsc(T)}.$$

The following open problem appears to be very challenging:

Open Problem 7.2. *For what regular sets of trajectories $T \subseteq \{0,1\}^*$ does the construction given by Lemma 7.1 give a construction which is best possible?*

Consider unrestricted shuffle, given by the set of trajectories $T = (0+1)^*$. The bound of Lemma 7.1 in this case is $2^{nsc(L_1)nsc(L_2)}$. Câmpeanu *et al.* [8] have shown that there exist languages L_1 and L_2 accepted by incomplete DFAs having, respectively, n and m states such that any incomplete DFA accepting $L_1 \sqcup\!\sqcup L_2$ has at least $2^{nm} - 1$ states. This bound is optimal for incomplete DFAs, however; for complete DFAs it gives only the lower bound $2^{(sc(L_1)-1)(sc(L_2)-1)}$. However, we regard this as near enough to our goal of Lemma 7.1 for our purposes, i.e., we regard $T = (0+1)^*$ as an example of a set of trajectories T satisfying Open Problem 7.2.

The *density function* of a language $L \subseteq \Sigma^*$ is defined by $p_L : \mathbb{N} \to \mathbb{N}$ as $p_L(n) = |L \cap \Sigma^n|$ for all $n \geq 0$. That is, $p_L(n)$ gives the number of words of length n in L. By the density of a language L, we informally mean the asymptotic behaviour of p_L. The following important result of Szilard *et al.* [91, Thm. 3] characterizes the density of regular languages:

Theorem 7.3. *A regular language R over Σ satisfies $p_R(n) \in O(n^k)$, $k \geq 0$ if and only if R can be represented as a finite union of regular expressions of the following form: $xy_1^*z_1 \cdots y_t^*z_t$ where $x, y_1, z_1, \cdots, y_t, z_t \in \Sigma^*$, and $0 \leq t \leq k+1$.*

Call a language L *slender* if $p_L(n) \in O(1)$ [83]. Let R be a regular language which has polynomial density $O(n^k)$, and let t be the smallest integer such that $R = \cup_{i=1}^{t} x_i y_{i,1}^* z_{i,1} \cdots y_{i,k_i}^* z_{i,k_i}$, $0 \leq k_i \leq k+1$, $i = 1, \ldots, t$. Then call t the *UkL-index* of L. If $k = 0$, we call t the *USL-index* of L (languages with USL index t are called t-thin by Păun and Salomaa [83]; slender regular languages were also characterized independently by Shallit [87, Lemma 3, p. 336]).

Lemma 7.4. *Let $T = uv^*$ where $u, v \in \{0,1\}^*$. Let L_i be regular languages over Σ, with $sc(L_i) = n_i$, $i = 1, 2$. Let $L = L_1 \sqcup\!\sqcup_T L_2$. Then*

$$sc(L) \leq |uv| n_1 n_2. \tag{2}$$

We now give a bound for sets of trajectories $T = uv^*w$ with $w \neq \epsilon$.

Lemma 7.5. *Let $T = uv^*w$ where $u, v, w \in \{0,1\}^*$ and $w \neq \epsilon$. Let L_i be regular languages over Σ, with $sc(L_i) = n_i$, $i = 1, 2$. Let $L = L_1 \sqcup\!\sqcup_T L_2$. Then*

$$sc(L) \leq n_1 n_2 \left(|u| + 1 + |v| \frac{(n_1 n_2)^{\lceil \frac{|w|}{|v|} \rceil + 1} - n_1 n_2}{n_1 n_2 - 1} \right). \tag{3}$$

Our aim is to obtain a lower bound for the shuffle operation on trajectories with USL index 1. It seems likely that the bound (3) cannot be reached for any fixed set of trajectories (and for all values of $sc(L_i)$,$i = 1, 2$). In particular, if $|w|$ is fixed and $sc(L_i)$ can grow arbitrarily, then it seems impossible that the $\lceil \frac{|w|}{|v|} \rceil$ parallel computations on the suffix w could simultaneously reach all combinations of states of the DFAs for L_1 and L_2. Note that if the computation of M contains parallel branches that simulate the computations of M_i ($1 \leq i \leq 2$), in states $P_i \subseteq Q_i$, then all the states of P_i need to be reachable from a single state of M_i with inputs of length at most $|w|$.

For the above reason, we consider a lower bound for sets of trajectories uv^*w where the length of v and of w can depend on the sizes of the minimal DFAs for the component languages L_1 and L_2. Furthermore, to simplify the notations below we give lower bound results for sets of trajectories of the form v^*w, i.e., $u = \epsilon$. It would be straightforward to modify the construction for prefixes u of arbitrary length to include the additive term $n_1n_2 \cdot (|u| + 1)$ from (3).

Lemma 7.6. *Let $\Sigma = \{a, b, c\}$. For any $n_1, n_2 \in \mathbb{N}$ there exist regular languages $L_i \subseteq \Sigma^*$ with $sc(L_i) = n_i$, $i = 1, 2$, and a set of trajectories $T = v^*w$, where $v, w \in \{0, 1\}^*$, such that*

$$sc(L_1 \sqcup\!\sqcup_T L_2) \geq (n_1n_2)^{\lceil \frac{|w|}{|v|} \rceil + 1}.$$

The ratio $|w|/|v|$ above can be chosen to be arbitrarily large.

By extending Lemma 7.6 slightly, we obtain the following result:

Theorem 7.7. *The upper bound (3) is asymptotically optimal if $sc(T)$ (that is, $|v|$) can be arbitrarily large compared to $sc(L_i)$, $i = 1, 2$.*

We conclude with some open problems. Recall that the example of arbitrary shuffle, shown by Câmpeanu *et al.* to have state complexity no better than our construction in Lemma 7.1, uses the set of trajectories $T = (0 + 1)^*$ of density 2^n. We also note that, by Szilard *et al.* [91], the density of a regular language over Σ is either $O(p(n))$, where p is a polynomial, or $\Omega(|\Sigma|^n)$.

Thus, we may conjecture that a set of trajectories T yields an operation which is, in the worst case, no better than Lemma 7.1 if and only if $p_T(n) \in \Omega(2^n)$, i.e. T has exponential density.

Our constructions in Lemma 7.6 use three-letter alphabets. Can these constructions be improved to two-letter alphabets? The problem of restricting the alphabet size to be as small as possible is often challenging. For example, in the case of concatenation, the state complexity problem was solved for a three-letter alphabet by Yu *et al.* [94], but the case of a two-letter alphabet was open until recently [44, 43].

8 Substitution on Trajectories

We now recall the definition of substitution on trajectories, originally given by Kari *et al.* [53, 52]. A trajectory t is a word over $\{0,1\}$. Given a trajectory t and $u, v \in \Sigma^*$, the substitution of v into u (or the substitution in u of v) is given by

$$\begin{aligned} u \bowtie_t v \;\; = \;\; & \{(\prod_{i=1}^{n} u_i v_i) u_{n+1} \; : \; n \geq 0, u = (\prod_{i=1}^{n} u_i a_i) u_{n+1}, v = \prod_{i=1}^{n} v_i \\ & t = \prod_{i=1}^{n} 0^{a_i} 1) 0^{a_{n+1}}, a_i, v_i \in \Sigma, \forall i, 1 \leq i \leq k, u_i \in \Sigma^*, \forall i, 1 \leq i \leq k+1, \\ & j_i = |u_i| \forall 1 \leq i \leq k\}. \end{aligned}$$

Note that if $|u| \neq |t|$ or $|v| \neq |t|_1$ then $u \bowtie_t v = \emptyset$.

We extend this to sets of trajectories $T \subseteq \{0,1\}^*$ as expected: $u \bowtie_T v = \cup_{t \in T} u \bowtie_T v$ Further, if L_1, L_2 are languages, then $L_1 \bowtie_T L_2 = \cup_{\substack{x \in L_1 \\ y \in L_2}} u \bowtie_T v$. We note that the notation $\bowtie_T$ was also used by Mateescu [77] for the splicing on routes. This concept is unrelated to substitution on trajectories except in that they both are based on the concept of trajectories.

We consider some examples:

(i) If $T = \{0,1\}^*$, the resulting operation $\bowtie_T$ is known as the substitution operation. In this operation, substitutions are permitted in any possible position.

(ii) If $T_k = 0^*(10^*)^k 0^*$, the language $x \bowtie_T \Sigma^k$ contains all possible words obtained by substituting exactly k symbols into x.

One motivation for substitution on trajectories is the close associations between insertion, deletion and substitution in models of channels with can generate errors while transmitting data. Indeed, so-called SID channels (for substitution, insertion and deletion) are a strong model of transmission media where these three types of errors may occur. Thus, it is natural to consider a substitution-based operation using trajectories.

Formally, a channel γ is a binary relation on words which defines a set of possible outputs from a channel given an input word: if $u \,\gamma\, y$, then y is a possible output of the channel on input u. A language L is error-detecting for a channel γ if, for all $u, v \in L \cup \{\epsilon\}$, $u \,\gamma\, v$ implies $u = v$. For a given set of trajectories T, the channel defined by T is given by the relation $\{(u, v) \; : \; v \in u \bowtie_T \Sigma^*\}$. This is a very natural definition: it is completely analogous to the definition of the binary relation defined by shuffle on trajectories defined independently by the author and described in Section 9.1 below).

Kari *et al.* [53, 52] show that given a language and a substitution channel (defined by a set of trajectories), it is decidable in polynomial time whether the language is error-detecting for a channel. Kari *et al.* [53, 52] have also investigated language equations involving substitution on trajectories. In particular, for one-variable equations of the form $X \bowtie_T R_1 = R_2$ or $R_1 \bowtie_T X = R_2$, it is decidable whether there exists a solution if R_1, R_2 and T are all regular. The problem of examining two-variable equations for substitution on trajectories, however, is apparently open:

Open Problem 8.1. *Find necessary and sufficient conditions on T such that, given a regular language R, it is decidable whether there exist X_1, X_2 such that $R = X_1 \bowtie_T X_2$.*

9 Theory of Codes

Prefix codes are fundamental objects in formal language theory, and are likely one of the most well-studied classes of languages which are not defined by their relation to a generating or accepting device. A language L is a prefix code if no word $x \in L$ is a prefix of another word y in L. The use of prefix is intricately linked to concatenation—which is itself a particular case of an operation defined by shuffle on trajectories. This is reflected by the following well-known identify for prefix codes: L is a prefix code if and only if $L \cap L\Sigma^+ = \emptyset$ (We refer the reader to Berstel and Perrin [5], Jürgensen and Konstantinidis [46] or Shyr [88] for an introduction to the theory of codes and prefix codes).

The question now arises: is concatenation the only shuffle-on-trajectories operation for which the associated "prefix-like" property defines a class of languages related to the theory of codes? It turns out that the answer is no: several well-studied language classes related to the theory of codes are a particular case of the concept of T-codes, which we review now. The results in this section are due to the author [20, 21, 22].

Let $L \subseteq \Sigma^+$ be a language. Then, for any $T \subseteq \{0,1\}^*$, we say that L is a *T-code* if L is non-empty and $(L \sqcup_T \Sigma^+) \cap L = \emptyset$. If Σ is an alphabet and $T \subseteq \{0,1\}^*$, let $\mathcal{P}_T(\Sigma)$ denote the set of all T-codes over Σ.

There has been much research into the idea of T-codes for particular $T \subseteq \{0,1\}^*$, including

(a) prefix, suffix and biprefix (or bifix) codes, corresponding to $T = 0^*1^*$, $T = 1^*0^*$ and $T = 0^*1^* + 1^*0^*$, respectively;
(b) outfix and infix codes, corresponding to $T = 0^*1^*0^*$ and $T = 1^*0^*1^*$, respectively;
(c) shuffle-codes, corresponding to bounded sets of trajectories such as $T = (0^*1^*)^n$ for fixed $n \geq 1$ (prefix codes of index n), $T = (1^*0^*)^n$ for fixed $n \geq 1$ (suffix codes of index n), $T = 1^*(0^*1^*)^n$ for fixed $n \geq 1$

(infix codes of index n), or $T = (0^*1^*)^n 0^*$ for fixed $n \geq 1$ (outfix codes of index n).
(d) hypercodes, corresponding to $T = (0+1)^*$;
(e) k-codes, corresponding to $T = 0^*1^*0^{\leq k}$ (see Kari and Thierrin [57]) for fixed $k \geq 0$; and
(f) for arbitrary $k \geq 1$, codes defined by the sets of trajectories $PP_k = 0^* + (0^*1^*)^{k-1}0^*1^+$, $PS_k = 0^* + 1^+0^*(1^*0^*)^{k-1}$, $PI_k = 0^* + (1^*0^*)^k 1^+$, $SI_k = 0^* + 1^+(0^*1^*)^k$, $PB_k = PP_k \cup PS_k$ and $BI_k = PI_k \cup SI_k$, see Long [69], or Ito *et al.* [40] for PI_1, SI_1.

For a list of references related to (a)–(d), see Jürgensen and Konstantinidis [46, pp. 549–553].

9.1 The Binary Relation Defined by Shuffle on Trajectories

We can also define T-codes by appealing to a definition based on binary relations. In particular, for $T \subseteq \{0,1\}^*$, define ω_T as follows: for all $x, y \in \Sigma^*$,

$$x \, \omega_T \, y \iff y \in x \sqcup\!\sqcup_T \Sigma^*.$$

It is clear that $L \subseteq \Sigma^+$ is a T-code if and only if L is an anti-chain under ω_T (i.e, $x, y \in L$ and $x \, \omega_T \, y$ implies $x = y$).

We note that the relation analogous to ω_T for infinite words and ω-trajectories was defined by Kadrie *et al.* [48]. In what follows, we will refer to T having a property P if and only if ω_T has property P. Recall that a binary relation ρ on Σ^* is said to be *positive* if $\epsilon \, \rho \, x$ for all $x \in \Sigma^*$.

Lemma 9.1. *Let $T \subseteq \{0,1\}^*$.*

(a) The relation ω_T is anti-symmetric.
(b) T is reflexive if and only if $0^* \subseteq T$.
(c) T is positive if and only if $1^* \subseteq T$.

Let ρ be a binary relation on Σ^*. Then we say that ρ is *left-compatible* (resp., *right-compatible*) if, for all $u, v, w \in \Sigma^*$, $u \, \rho \, v$ implies that $wu \, \rho \, wv$ (resp., $uw \, \rho \, vw$). If ρ is both left- and right-compatible, we say it is *compatible*.

Lemma 9.2. *Let $T \subseteq \{0,1\}^*$. Then T is right-compatible (resp., left-compatible, compatible) if and only if $T0^* \subseteq T$ (resp., $0^*T \subseteq T$, $0^*T0^* \subseteq T$).*

We now consider conditions on T which will ensure that ω_T is a transitive relation. Transitivity is often, but not always, a property of the binary relations defining the classic code classes. For instance, both bi-prefix and outfix codes are defined by binary relations which are not transitive, and hence not a partial order.

First, we define three morphisms we will need. Let $D = \{x, y, z\}$ and $\varphi, \sigma, \psi : D^* \to \{0,1\}^*$ be the morphisms given by

$$\begin{array}{rclrclrcl} \varphi(x) & = & 0, & \sigma(x) & = & 0, & \psi(x) & = & 0, \\ \varphi(y) & = & 0, & \sigma(y) & = & 1, & \psi(y) & = & 1, \\ \varphi(z) & = & 1, & \sigma(z) & = & \epsilon, & \psi(z) & = & 1. \end{array}$$

Note that these morphisms are similar to the substitutions defined by Mateescu *et al.* [79], whose purpose is to give necessary and sufficient conditions on a set T of trajectories defining an associative operation.

Theorem 9.3. *Let $T \subseteq \{0,1\}^*$. Then T is transitive if and only if $\psi(\varphi^{-1}(T) \cap \sigma^{-1}(T)) \subseteq T$.*

As an alternate formulation for Theorem 9.3, we note that, for all $T \subseteq \{0,1\}^*$, T is transitive if and only if $T \sqcup_T 1^* \subseteq T$.

Corollary 9.4. *Given a regular set $T \subseteq \{0,1\}^*$ of trajectories, it is decidable whether T is transitive.*

For undecidability, we naturally find that deciding whether a context-free set of trajectories is transitive is undecidable:

Theorem 9.5. *Given a CF set $T \subseteq \{0,1\}^*$ of trajectories, it is undecidable whether T is transitive.*

It is easy to see that if $\{T_i\}_{i\in I}$ is a family of transitive sets of trajectories, then the set $\cap_{i\in I} T_i$ is also transitive. Thus, we can define the transitive closure of a set T of trajectories as follows: for all $T \subseteq \{0,1\}^*$, let $tr(T) = \{T' \subseteq \{0,1\}^* \ : \ T \subseteq T', T' \text{ transitive}\}$. Note that $tr(T) \neq \emptyset$, as $\{0,1\}^* \in tr(T)$ for all $T \subseteq \{0,1\}^*$. Define $\widehat{T}$ as

$$\widehat{T} = \bigcap_{T' \in tr(T)} T'. \tag{4}$$

Then note that $\widehat{T}$ is transitive and is the smallest transitive set of trajectories containing T. The operation $\widehat{\cdot} : 2^{\{0,1\}^*} \to 2^{\{0,1\}^*}$ is indeed a closure operator (much like the closure operators on sets of trajectories constructed by Mateescu *et al.* [79] for, e.g., associativity and commutativity) in the algebraic sense, since $T \subseteq \widehat{T}$, and $\widehat{\cdot}$ preserves inclusion and is idempotent.

Consider the operator $\Omega_T : 2^{\{0,1\}^*} \to 2^{\{0,1\}^*}$ given by

$$\Omega_T(T') = T \cup T' \cup \psi(\sigma^{-1}(T') \cap \varphi^{-1}(T')).$$

It is not difficult to see that, given T, we can find $\widehat{T}$ by iteratively applying Ω_T to T, and in fact $\widehat{T} = \bigcup_{i\geq 0} \Omega_T^i(T)$. This observation allows us to construct $\widehat{T}$, and, for instance, gives us the following result (a similar result for ω-trajectories is given by Kadrie *et al.* [48]):

Lemma 9.6. *There exists a regular set of trajectories $T \subseteq \{0,1\}^*$ such that $\widehat{T}$ is not a CFL.*

In particular, consider $T = (01)^*$, corresponding to perfect or balanced literal shuffle. Then we note that $\widehat{T} \cap 01^* = \{01^{2^n-1} \ : \ n \geq 1\}$.

Open Problem 9.7. *Given $T \in$ REG (or $T \in$ CF), is it decidable whether $\widehat{T} \in$ CF?*

We now examine the relationship between T-codes and $\widehat{T}$-codes for arbitrary $T \subseteq \{0,1\}^*$. We call a language $L \subseteq \Sigma^*$ *T-convex* if, for all $y \in \Sigma^*$ and $x, z \in L$, $x \, \omega_T \, y$ and $y \, \omega_T \, z$ implies $y \in L$.

We now characterize when a language is T-convex using shuffle and deletion along trajectories. Define the morphism $\tau : \{0,1\}^* \to \{i,d\}^*$ by $\tau(0) = i$ and $\tau(1) = d$. This morphism defines the relationship between shuffle and deletion along trajectories.

Lemma 9.8. *Let $T \subseteq \{0,1\}^*$. Then $L \subseteq \Sigma^*$ is T-convex if and only if $(L \sqcup\!\sqcup_T \Sigma^*) \cap (L \rightsquigarrow_{\tau(T)} \Sigma^*) \subseteq L$.*

We now turn to decidability:

Corollary 9.9. *Let $T \subseteq \{0,1\}^*$ be a regular set of trajectories. Given a regular language L, it is decidable whether L is T-convex.*

These results lead to the following general relationship between T-codes and $\widehat{T}$-codes:

Theorem 9.10. *Let Σ be an alphabet and $T \subseteq \{0,1\}^*$. For all languages $L \subseteq \Sigma^+$, the following two conditions are equivalent:*

(i) L is a $\widehat{T}$-code;
(ii) L is a $\widehat{T}$-convex T-code.

Theorem 9.10 was known for the case $O = 0^*1^*0^*$, which corresponds to outfix codes, see, e.g., Shyr and Thierrin [89, Prop. 2]. In this case, $\widehat{O} = H = (0+1)^*$, which corresponds to hypercodes. Theorem 9.10 was known to Guo *et al.* [31, Prop. 2] in a slightly weaker form for $B = 0^*1^*+1^*0^*$. In this case, $\widehat{B} = I = 1^*0^*1^*$, and the convexity is with respect to the factor (or subword) ordering. See also Long [70, Sect. 5] for the case of shuffle codes.

We conclude this section with some research directions. A binary relation ρ on Σ^* is said to be *left-cancellative* (resp., *right-cancellative*) if $uv \, \rho \, ux$ implies $v \, \rho \, x$ (resp., $vu \, \rho \, xu$ implies $v \, \rho \, x$) for all $u, v, x \in \Sigma^*$. The relation ρ is *cancellative* if it is both left- and right-cancellative.

Open Problem 9.11. *What are necessary and sufficient language-theoretic conditions on a set of trajectories T so that T is left-cancellative, right-cancellative or cancellative?*

Another research direction would be to consider the class of codes defined not by shuffle on trajectories, but by splicing on routes. We note that additional interesting binary relations from the literature can be modeled using splicing on routes. For instance, we leave it to the reader to verify that if $T = 0^* + (\overline{0}1)^*1^+$ then the associated binary relation (defined in the same way as for shuffle on trajectories) is the length ordering, given by

$$x \leq y \iff (|x| < |y|) \text{ or } x = y.$$

9.2 Maximal T-codes

Let $T \subseteq \{0,1\}^*$. We say that $L \in \mathcal{P}_T(\Sigma)$ is a *maximal* T-code if, for all $L' \in \mathcal{P}_T(\Sigma)$, $L \subseteq L'$ implies $L = L'$. Denote the set of all maximal T-codes over an alphabet Σ by $\mathcal{M}_T(\Sigma)$. Note that the alphabet Σ is crucial in the definition of maximality. By Zorn's Lemma, we can easily establish that every $L \in \mathcal{P}_T(\Sigma)$ is contained in some element of $\mathcal{M}_T(\Sigma)$. The proof is a specific instance of a result from dependency theory [46].

Unlike showing that every T-code can be embedded in a maximal T-code, to our knowledge, dependency theory has not addressed the problem of deciding whether a language is a maximal code under some dependence system. We address this problem for T-codes now. We first require the following technical lemma, which is interesting in its own right (specific cases were known for, e.g., prefix codes [5, Prop. 3.1, Thm. 3.3], hypercodes [89, Cor. to Prop. 11], as well as biprefix and outfix codes [68, Lemmas 3.3 and 3.5]). Let $\tau : \{0,1\}^* \to \{i,d\}^*$ be again given by $\tau(0) = i$ and $\tau(1) = d$.

Lemma 9.12. *Let $T \subseteq \{0,1\}^*$. Let Σ be an alphabet. For all $L \in \mathcal{P}_T(\Sigma)$, $L \in \mathcal{M}_T(\Sigma)$ if and only if*

$$L \cup (L \sqcup\!\sqcup_T \Sigma^+) \cup (L \rightsquigarrow_{\tau(T)} \Sigma^+) = \Sigma^+. \tag{5}$$

Corollary 9.13. *Let $T \subseteq \{0,1\}^*$ be a regular set of trajectories. Given a regular language $L \subseteq \Sigma^+$, it is decidable whether $L \in \mathcal{M}_T(\Sigma)$.*

Similar results were also obtained by Kari *et al.* [51, Sect. 5].

9.3 Embedding and Finiteness

Given a class of codes $\mathcal{C}$, and a language $L \in \mathcal{C}$ of given complexity, there has been much research into whether or not L can be *embedded in* (or *completed to*) a maximal element $L' \in \mathcal{C}$ of the same complexity, i.e., a maximal code $L' \in \mathcal{C}$ with $L \subseteq L'$. Finite and regular languages in these classes of codes are of particular interest. For instance, we note that every regular code can be completed to a maximal regular code, while the same is not true for finite codes or finite biprefix codes.

We now show an interesting result on embedding T-codes in maximal T-codes while preserving complexity. Our construction is a generalization of a result due to Lam [65]. In particular, we define two transformations on languages. Let T be a set of trajectories and $L \subseteq \Sigma^+$ be a language. Then define $U_T(L), V_T(L) \subseteq \Sigma^+$ as

$$\begin{aligned} U_T(L) &= \Sigma^+ - (L \sqcup_T \Sigma^+ \cup L \rightsquigarrow_{\tau(T)} \Sigma^+); \\ V_T(L) &= U_T(L) - (U_T(L) \sqcup_T \Sigma^+). \end{aligned}$$

Recall that $\tau : \{0,1\}^* \to \{i,d\}^*$ is given by $\tau(0) = i$ and $\tau(1) = d$.

Theorem 9.14. *Let $T \subseteq \{0,1\}^*$ be transitive. Let Σ be an alphabet. Then for all $L \in \mathcal{P}_T(\Sigma)$, the language $V_T(L)$ contains L and $V_T(L) \in \mathcal{M}_T(\Sigma)$.*

We note one consequence of Theorem 9.14:

Corollary 9.15. *Let $T \subseteq \{0,1\}^*$ be transitive and regular. Then every regular (resp., recursive) T-code is contained in a maximal regular (resp., recursive) T-code.*

Corollary 9.15 was given for $T = 1^*0^*1^*$ and regular T-codes by Lam [65, Prop. 3.2]. Further research into the case when T is not transitive is necessary (for example, the proofs of Zhang and Shen [97] and Bruyère and Perrin [6] on embedding regular biprefix codes are much more involved than the above construction, and do not seem to be easily generalized).

Open Problem 9.16. *Characterize those $T \subseteq \{0,1\}^*$ for which every regular (resp., finite, recursive) T-code can be embedded in a maximal regular (resp., finite recursive) T-code.*

We can extend our embedding results to finite languages with one additional constraint on T, namely completeness.

Corollary 9.17. *Let $T \subseteq \{0,1\}^*$ be transitive and complete. Let Σ be an alphabet. Then for all finite $F \in \mathcal{P}_T(\Sigma)$, there exists a finite language $F' \in \mathcal{M}_T(\Sigma)$ such that $F \subseteq F'$. Further, if T is effectively regular, and F is effectively given, we can effectively construct F'.*

In practice, the condition that T be complete is not very restrictive, since natural operations seem to typically be defined by a complete set of trajectories.

Also of interest are those $T \subseteq \{0,1\}^*$ such that all T-codes are finite. It is a well-known result that all hypercodes ($T = \{0,1\}^*$) are finite, which can be concluded from a result due to Higman [35]. We define the class $\mathfrak{F}_H$ as

$$\mathfrak{F}_H = \{T \in \{0,1\}^* \ : \ \mathcal{P}_T(\Sigma) \subseteq \textsc{fin}\}.$$

Also studied by the author are the classes of trajectories such that every *regular* (or context-free) T-code is finite [20, 22].

The class $\mathfrak{F}_H$ is related to a large amount of research in the literature. If T is a partial order and $T \in \mathfrak{F}_H$, then T is a *well partial order*. We define $\mathfrak{F}_H^{(po)}$ to be the set of all T which are well partial orders. Without trying to be exhaustive, we note the work of Jullien [45], Haines [32], van Leeuwen [93], Ehrenfeucht *et al.* [29], Ilie [37, 38], Ilie and Salomaa [42] and Harju and Ilie [33] on well partial orders relating to words. We also refer the reader to the survey of results presented by de Luca and Varricchio [14, Sect. 5].

We now consider the question of the existence of arbitrary infinite languages in a class of T-codes. We first show that if T is bounded, then there is an infinite T-code.

Theorem 9.18. *Let $T \subseteq \{0,1\}^*$ be a bounded set of trajectories. Then for all Σ with $|\Sigma| > 1$, $\mathcal{P}_T(\Sigma)$ contains an infinite language, i.e., $T \notin \mathfrak{F}_H$.*

Further, there exist uncountably many unbounded trajectories T such that $\mathcal{P}_T(\Sigma)$ contains infinite–even infinite *regular*–languages. Infinitely many of these are unbounded regular sets of trajectories.

Theorem 9.19. *Let $T \subseteq \{0,1\}^*$ be a set of trajectories such that there exists $n \geq 0$ such that $T \subseteq 0^{\leq n}1(0+1)^*$. Then for all Σ with $|\Sigma| > 1$, $\mathcal{P}_T(\Sigma)$ contains an infinite regular language.*

We now turn to defining sets T of trajectories such that all T-codes are finite. The following proof is generalized from the case $H = (0+1)^*$ found in, e.g., Lothaire [74] or Conway [10, pp. 63–64].

Lemma 9.20. *Let $n, m \geq 1$ be such that $m \mid n$. Let $T_{n,m} = (0^n + 1^m)^* 0^{\leq n-1}$. Then $T_{n,m} \in \mathfrak{F}_H$.*

As another class of examples, Ehrenfeucht *et al.* [29, p. 317] note that $\{1^n, 0\}^* \in \mathfrak{F}_H$ for all $n \geq 1$. Ilie [38, Sect. 7.7] also gives a class of partial orders which we may phrase in terms of sets of trajectories. In particular, define the set of functions

$$\mathcal{G} = \{g : \mathbb{N} \to \mathbb{N} \ : \ g(0) = 0 \text{ and } 1 \leq g(n) \leq n \text{ for all } n \geq 1\}.$$

Then for all $g \in \mathcal{G}$, we define

$$T_g = \{1^* \prod_{k=1}^{m} (0^{i_k} 1^*) \ : \ i_k \geq 0 \ \forall 1 \leq k \leq m; \ m = g(\sum_{k=1}^{m} i_k)\}.$$

We denote the *upper limit* of a sequence $\{s_n\}_{n \geq 1}$ by $\overline{\lim}_{n \to \infty} s_n$. We have the following result [38, Thm. 7.7.8]:

Theorem 9.21. *Let $g \in \mathcal{G}$. Then $T_g \in \mathfrak{F}_H \iff \overline{\lim}_{n \to \infty} \frac{n}{g(n)} < \infty$.*

However, a complete characterization is still open:

Open Problem 9.22. *Give necessary and sufficient language-theoretic conditions on a set of trajectories T so that $T \in \mathfrak{F}_H$.*

We now turn to the complexity of T-convex languages:

Theorem 9.23. *Let C be a cone. Let $T \in \mathfrak{F}_H^{(po)}$ be an element of C. Then every T-convex language is an element of $C \wedge$ co-C.*

Corollary 9.24. *Let $T \in$ REG (resp., REC) be such that $T \in \mathfrak{F}_H^{(po)}$. If L is a T-convex language, then $L \in$ REG (resp., REC).*

Corollary 9.24 was known for the case of $H = (0+1)^*$ and $L \in$ REG, see Thierrin [92, Cor. to Prop. 3].

9.4 Codes defined by Multiple Sets of Trajectories

When studying T-codes, we note that if $T_1, T_2 \subseteq \{0,1\}^*$ are sets of trajectories, there is not necessarily a set of trajectories T such that $\omega_T = \omega_{T_1} \cap \omega_{T_2}$, i.e., such that $x \, \omega_T \, y \iff (x \, \omega_{T_1} \, y) \wedge (x \, \omega_{T_2} \, y)$. For instance, for $P = 0^*1^*$ and $S = 1^*0^*$, the relation $\omega_P \cap \omega_S$ is given by $\leq_d$, where $x \leq_d y$ if and only if there exist $u, v \in \Sigma^*$ such that $y = xu = vx$. This relation cannot be represented by a set of trajectories. For a discussion of $\leq_d$, see Shyr [88, Ch. 8].

In fact, there exist many natural classes of languages studied in connection to the theory of codes which are not T-codes. Classes and their associated binary relations studied by Day and Shyr [13], Fan *et al.* [30], Ito *et al.* [39], Long [71], Long *et al.* [73, 72], Shyr [88], Yu [95] and the author [18] are instead defined by a binary relation dependent on multiple sets of trajectories. The author and K. Salomaa [27] have studied the properties of classes of languages defined by multiple sets of trajectories.

Let $\mathbf{T} \subseteq 2^{\{0,1\}^*}$. We call such a set of sets of trajectories $\mathbf{T}$ a *hyperset of trajectories*; such a hyperset of trajectories is always assumed to be a finite set of sets of trajectories. Define $\omega_{\mathbf{T}}$ as

$$x \, \omega_{\mathbf{T}} \, y \iff \bigwedge_{T \in \mathbf{T}} x \, \omega_T \, y.$$

That is, $x \, \omega_{\mathbf{T}} \, y$ if and only if $x \, \omega_T \, y$ for all $T \in \mathbf{T}$.

The class $\mathcal{P}_{\mathbf{T}}^{(\wedge)}(\Sigma)$ is defined as follows: for all non-empty languages $L \subseteq \Sigma^+$, $L \in \mathcal{P}_{\mathbf{T}}^{(\wedge)}(\Sigma)$ if and only if L is an anti-chain under $\omega_{\mathbf{T}}$. That is, for all $x, y \in L$, if $x \, \omega_{\mathbf{T}} \, y$, then $x = y$.

The definition of $\mathcal{P}_{\mathbf{T}}^{(\wedge)}(\Sigma)$ is motivated by the interest in the class $\mathcal{P}_{\mathbf{T}_{\text{ps}}}^{(\wedge)}(\Sigma)$ for $\mathbf{T}_{\text{ps}} = \{0^*1^*, 1^*0^*\}$. Note that $x \, \omega_{\mathbf{T}_{\text{ps}}} \, y$, i.e., $x \leq_d y$, implies that x is both a prefix and a suffix of y. We refer the reader to Jürgensen and Konstantinidis [46, pp. 550–551] for references and a discussion of $\mathcal{P}_{\mathbf{T}_{\text{ps}}}^{(\wedge)}(\Sigma)$.

We also define a second class of languages, indexed by an integer m, which is considered in conjunction with $\mathcal{P}_{\mathbf{T}}^{(\wedge)}(\Sigma)$ for particular $\mathbf{T}$. For all $m \geq 0$, let $\mathcal{P}_{\mathbf{T}}^{(m)}(\Sigma)$ be defined as follows: for all non-empty languages $L \subseteq \Sigma^+$, $L \in \mathcal{P}_{\mathbf{T}}^{(m)}(\Sigma)$ if and only if for all $L' \subseteq L$ with $|L'| \leq m$, $L' \in \cup_{T \in \mathbf{T}} \mathcal{P}_T(\Sigma)$.

The following hypersets of trajectories have been studied in connection with the associated class $\mathcal{P}_{\mathbf{T}}^{(m)}(\Sigma)$:

(i) $\mathbf{T}_{\text{ps}} = \{0^*1^*, 0^*1^*\}$. The class $\mathcal{P}_{\mathbf{T}_{\text{ps}}}^{(m)}(\Sigma)$ is known as the class of *m-prefix-suffix codes* (or m-ps-codes). See Ito *et al.* [39] for details;

(ii) $\mathbf{T}_{\text{io}} = \{0^*1^*0^*, 1^*0^*1^*\}$ [73, 18]. The class $\mathcal{P}_{\mathbf{T}_{\text{io}}}^{(m)}(\Sigma)$ is known as the class of *m-infix-outfix codes*;

(iii) $\mathbf{T}_{k-\text{io}} = \{(1^*0^*)^k 1^*, (0^*1^*)^k 0^*\}$ and $\mathbf{T}_{k-\text{ps}} = \{(0^*1^*)^k, (1^*0^*)^k\}$ for $k \geq 1$. The class $\mathcal{P}_{\mathbf{T}_{k-\text{io}}}^{(m)}(\Sigma)$ (resp., $\mathcal{P}_{\mathbf{T}_{k-\text{ps}}}^{(m)}(\Sigma)$) is known as the class of *m-k-infix-outfix codes* (resp., *m-k-prefix-suffix codes*). For results on these classes, see Long *et al.* [72, Sect. 4] or Long [71, Sect. 2.3])

The following lemma [27] states that $\mathcal{P}_{\mathbf{T}}^{(\wedge)}(\Sigma)$ and $\mathcal{P}_{\mathbf{T}}^{(2)}(\Sigma)$ always coincide:

Lemma 9.25. *Let $\mathbf{T} \subseteq 2^{\{0,1\}^*}$ be a hyperset of trajectories. Then*

$$\mathcal{P}_{\mathbf{T}}^{(\wedge)}(\Sigma) = \mathcal{P}_{\mathbf{T}}^{(2)}(\Sigma).$$

Lemma 9.25 was previously observed for, e.g., the case $\mathbf{T}_{\text{ps}} = \{0^*1^*, 1^*0^*\}$, see Ito *et al.* [39]. The following equations detail the hierarchies induced by varying m in $\mathcal{P}_{\mathbf{T}}^{(m)}(\Sigma)$, and their collapse. These equations, which hold for all $\mathbf{T} \subseteq 2^{\{0,1\}^*}$, can be proven using dependency theory [46] (in the following, $|\mathbf{T}|$ is the cardinality of $\mathbf{T}$ as a subset of $2^{\{0,1\}^*}$):

$$\begin{aligned} \mathcal{P}_{\mathbf{T}}^{(n)}(\Sigma) &\supseteq \mathcal{P}_{\mathbf{T}}^{(n+1)}(\Sigma) \quad \forall n \geq 0; & (6)\\ \mathcal{P}_{\mathbf{T}}^{(2|\mathbf{T}|)}(\Sigma) &= \mathcal{P}_{\mathbf{T}}^{(2|\mathbf{T}|+i)}(\Sigma) = \bigcup_{T \in \mathbf{T}} \mathcal{P}_T(\Sigma) \quad \forall i \geq 0. & (7) \end{aligned}$$

See, e.g., Ito *et al.* [39, Cor. 3.2] for (7) in the particular case of $\mathbf{T}_{\mathrm{ps}} = \{0^*1^*, 1^*0^*\}$.

The positive decidability of membership in $\mathcal{P}_{\mathbf{T}_{\mathrm{ps}}}^{(\wedge)}(\Sigma)$ for regular languages (see Ito *et al.* [39] or Jürgensen *et al.* [47]) relies intrinsically on the nature of the members of $\mathbf{T}_{\mathrm{ps}}$. The corresponding positive decidability problem for $\mathbf{T}_{\mathrm{io}}$ also relies on the nature of the sets of trajectories involved [18]. Kari *et al.* [51, Thm. 4.7] have resolved the decidability of a somewhat similar decision problem for two sets of trajectories in their framework of *bond-free property* (see Section 10). However, their approach is not applicable to our formalism. We recall a particular case of their result, translated into our framework:

Theorem 9.26. *Let $\mathbf{T} = \{T_1, T_2\}$ be a hyperset of trajectories where $T_i \in$ REG for $i = 1, 2$. Given a regular language $R \subseteq \Sigma^*$, we can determine whether there exist $w_1, w_2 \in R$, and $w \in \Sigma^+$ such that $w_i \neq w$ and $w_i \,\omega_{T_i}\, w$ for $i = 1, 2$.*

However, the following surprising undecidability result holds for $\mathcal{P}_{\mathbf{T}}^{(\wedge)}(\Sigma)$ [27]:

Theorem 9.27. *Given $\boldsymbol{T} = \{T_1, T_2\}$, where $T_i \in$ REG, for $i = 1, 2$ and a regular language R it is undecidable whether or not $R \in \mathcal{P}_{\mathbf{T}}^{(\wedge)}(\Sigma)$.*

The following undecidability result also holds [27]:

Theorem 9.28. *There exists a fixed hyperset of trajectories $\mathbf{T} = \{T_1, T_2\}$ where $T_i \in$ REG for $i = 1, 2$, such that the following problem is undecidable: "Given $L \in$ CF, is $L \in \mathcal{P}_{\mathbf{T}}^{(\wedge)}(\Sigma)$?"*

However, the decidability of membership in $\mathcal{P}_{\mathbf{T}}^{(\wedge)}(\Sigma)$ for a fixed $\mathbf{T}$ remains open:

Open Problem 9.29. *For which hypersets of trajectories $\mathbf{T} \subseteq 2^{\{0,1\}^*}$ is the following problem decidable: "Given $L \in$ REG, is $L \in \mathcal{P}_{\mathbf{T}}^{(\wedge)}(\Sigma)$?"*

It is conceivable that the question stated in Open Problem 9.29 could be decidable for all hypersets $\mathbf{T} = \{T_1, \ldots, T_n\}$ where $T_i \in$ REG for $1 \leq i \leq n$, in particular, if the alphabet Σ is fixed. If this is the case, by Theorem 9.27, given $\mathbf{T}$, the corresponding algorithm cannot be found effectively.

The author and K. Salomaa have also studied the equivalence problem for hypersets of trajectories. In particular, given $\mathbf{T_1}, \mathbf{T_2} \subseteq 2^{\{0,1\}^*}$, we say that $\mathbf{T_1}$ and $\mathbf{T_2}$ are $\wedge$-equivalent with respect to Σ if $\mathcal{P}_{\mathbf{T}_1}^{(\wedge)}(\Sigma) = \mathcal{P}_{\mathbf{T}_2}^{(\wedge)}(\Sigma)$. We simply say that $\mathbf{T_1}, \mathbf{T_2}$ are $\wedge$-equivalent if they are $\wedge$-equivalent with respect to every finite alphabet Σ. We use the notation $\mathbf{T}_1 \equiv_\wedge \mathbf{T}_2$ to indicate that $\mathbf{T}_1, \mathbf{T}_2$ are $\wedge$-equivalent.

Open Problem 9.30. *Given* $\mathbf{T}_1, \mathbf{T}_2$*, each consisting of regular sets of trajectories, can we determine whether* $\mathbf{T}_1$ *and* $\mathbf{T}_2$ *are* $\wedge$*-equivalent?*

We can restate Open Problem 9.30 as follows:

Open Problem 9.31. *For given regular languages* $T_1, \cdots, T_k \subseteq \{0,1\}^*$ *and* $U \subseteq \{0,1\}^*$ *is it decidable whether or not there exist an alphabet* Σ *and* $x, y \in \Sigma^+$ *such that for all* $1 \leq i \leq k$, $y \in x \sqcup\!\sqcup_{T_i} \Sigma^+$ *but* $y \notin x \sqcup\!\sqcup_U \Sigma^+$.

This problem seems very challenging. Similarly, we say that $\mathbf{T_1}, \mathbf{T_2}$ are m-equivalent with respect to Σ if $\mathcal{P}^{(m)}_{\mathbf{T}_1}(\Sigma) = \mathcal{P}^{(m)}_{\mathbf{T}_2}(\Sigma)$. Again, we say that $\mathbf{T}_1, \mathbf{T}_2$ are m-equivalent if they are m-equivalent with respect to every finite alphabet Σ. We use the notation $\mathbf{T}_1 \equiv_m \mathbf{T}_2$ to indicate that $\mathbf{T}_1, \mathbf{T}_2$ are m-equivalent. Open Problems 9.30 and 9.31 also remain open where $\wedge$-equivalence is replaced by m-equivalence.

10 DNA Code-word Design

The design of DNA code-words is a crucial step in employing DNA for computing purposes. DNA code-words are strands of DNA which allow only desired bonding to occur; careful consideration must be taken when designing code-words. Kari *et al.* [54] have used trajectories to investigate DNA bonding and code-word design.

An involution $\theta : \Sigma \to \Sigma$ is any function such that θ^2 is equal to the identity mapping. Any involution can be extended to a morphism via the rule $\theta(xy) = \theta(x)\theta(y)$ for all $x, y \in \Sigma^*$, or an antimorphism via the rule $\theta(xy) = \theta(y)\theta(x)$ for all $x, y \in \Sigma^*$.

Kari *et al.* [54] define the concept of the *bond-free properties* of languages. In particular, the bond-free property with respect to the sets of trajectories $T_{\mathrm{lo}}, T_{\mathrm{up}}$ is given by the following condition (for some involution θ):

$$\forall w \in \Sigma^+, x, y \in \Sigma^*(w \sqcup\!\sqcup_{T_{\mathrm{lo}}} x \cap L \neq \emptyset, w \sqcup\!\sqcup_{T_{\mathrm{up}}} y \cap \theta(L) \neq \emptyset) \Rightarrow xy = \epsilon.$$

Several previously studied DNA coding-inspired conditions are particular cases of the bond-free properties for particular pairs of sets of trajectories. This again demonstrates the power of trajectories–by approaching a problem from a uniform, trajectory-based viewpoint, many particular cases can be unified and studied systematically. The following decidability result shows that the bond-free property is decidable if all languages involved are regular [54]:

Theorem 10.1. *Let* T_{lo}, T_{up} *be regular sets of trajectories. Given a regular language L, it is decidable in quadratic time whether L satisfies the bond-free property with respect to* T_{lo}, T_{up}.

11 Contextual Grammars

Contextual grammar with contexts shuffled along trajectories, or *CST grammars* are an interesting application of trajectories to the study of the generative capacity of grammar forms. CST grammars were introduced by Martin-Vide *et al.* [75]. We recall the definition here: A CST grammar is a 4-tuple $G = (\Sigma, B, C, \mathcal{T})$, where Σ is an alphabet, $B, C \subseteq \Sigma^*$ are finite languages, and $\mathcal{T} = (T_c)_{c \in C}$ is a finite family of sets of trajectories indexed by words c from C. The language B is called the base of G and C is called the contexts of G. Derivations in G are given by $x \Rightarrow_G y$ if and only if there exists $c \in C$ such that $y \in x \sqcup\!\sqcup_{T_c} c$. The reflexive, transitive closure of $\Rightarrow_G$ is denoted by $\Rightarrow_G^*$. The language of G is the set of all words which are derivable from a word in the base of G:

$$L(G) = \{w \in \Sigma^* \ : \ \exists x \in B \text{ such that } x \Rightarrow_G^* w\}.$$

Recently, Okhotin and K. Salomaa [80] have investigated uniform CST grammars. A CST grammar $G = (\Sigma, B, C, \mathcal{T})$ is said to be *uniform* if $T_c = T_{c'}$ for all $c, c' \in C$. In this case, we denote the uniform CST by $G = (\Sigma, B, C, T)$ for some $T \subseteq \Sigma^*$. Okhotin and K. Salomaa demonstrate several results relating to uniform CST grammars:

Theorem 11.1. *There exists a language $L \subseteq \Sigma^*$ where $|\Sigma| \geq 2$ that cannot be generated by any uniform CST grammar.*

Theorem 11.1 does not depend in any way on the complexity of the set of trajectories: the language L cannot be generated by a uniform CST grammar G regardless of the complexity of T. However, Okhotin and K. Salomaa show that this same language L can be generated by a non-uniform CST $G = (\Sigma, B, C, \mathcal{T})$ where each $T_c \in \mathcal{T}$ is a context-sensitive set of trajectories.

Theorem 11.2. *Non-uniform CST grammars with context-sensitive sets of trajectories are strictly more powerful than uniform CST grammars with context-sensitive sets of trajectories.*

The following questions remain open [80]:

Open Problem 11.3. *Are non-uniform CST grammars with context-free (resp. regular) sets of trajectories more powerful than uniform CST grammars with context-free (resp., regular) sets of trajectories?*

We also note that Mateescu has also extended the notion of co-operating distributed grammars (CD grammars) to encompass the notion of trajectories [76]. A CD grammar on trajectory T is a six-tuple $\Gamma = (V, \Sigma, S, P_0, P_1, T)$ where V is a finite set of non-terminals, Σ is a finite alphabet, $S \in V$ is a distinguished start state, $P_0, P_1 \subseteq V \times (V \cup \Sigma)^*$ are two finite sets of productions, and $T \subseteq \{0,1\}^*$ is the set of trajectories.

Let $\Rightarrow_i$ denote the relation defined by the CFG $\Gamma_i = (V, \Sigma, S, P_i)$, for $i = 0, 1$. Then a word $w \in \Sigma^*$ is generated by Γ if there exist $t \in T$ of length n and $\alpha_i \in (V \cup \Sigma)^*$ for $1 \leq i \leq n$ such that if $t = t_1 t_2 \cdots t_n$ with $t_i \in \{0,1\}$ then for all $1 \leq i \leq n-1$ $\alpha_i \Rightarrow_{t_i} \alpha_{i+1}$, with $S = \alpha_1$ and $w = \alpha_n$. The usual notion of a CD grammar corresponds to $T = 0^*1^*$. The notion of CD grammars on trajectories is also generalized to grammars with n sets of productions $P_0, P_1, \ldots, P_{n-1}$, and a set of trajectories $T \subseteq \{0, \ldots, n-1\}^*$.

12 Conclusion

The notion of trajectories can seem deceptively simple: languages are used to parameterize language operations. This provides a basis for uniform results on language operations. However, in the ten years since their introduction, trajectories have seen much use beyond modelling language operations; we have surveyed these areas. The use of trajectories in many, varied areas is a testament to the elegance of the concept. We are confident that interest in trajectories will remain high as researchers continue to find new applications for the concept.

Acknowledgments

I am grateful to Kai Salomaa for reading a version of this survey.

References

[1] Amar, V., and Putzolu, G. On a family of linear grammars. *Inf. and Cont. 7* (1964), 283–291.

[2] Amar, V., and Putzolu, G. Generalizations of regular events. *Inf. and Cont. 8* (1965), 56–63.

[3] Bel-Enguix, G., Martín-Vide, C., and Mateescu, A. Dialog and splicing on routes. *Romanian Journal of Information Science and Technology 6*, 1-2 (2003), 45–59.

[4] Berstel, J., Boasson, L., Carton, O., Petazzoni, B., and Pin, J.-E. Operations preserving recognizable languages. In *Fundamentals of Computation Theory (FCT 2003)* (2003), A. Lingas and B. Nilsson, Eds., vol. 2751 of *Lecture Notes in Computer Science*, Springer, pp. 343–354.

[5] Berstel, J., and Perrin, D. *Theory of Codes*. Available at `http://www-igm.univ-mlv.fr/%7Eberstel/LivreCodes/Codes.html`, 1996.

[6] Bruyère, V., and Perrin, D. Maximal bifix codes. *Theor. Comp. Sci. 218* (1999), 107–121.

[7] Câmpeanu, C., Salomaa, K., and Vágvölgyi, S. Shuffle decompositions of regular languages. *Int. J. Found. Comp. Sci. 13*, 6 (2002), 799–816.

[8] Câmpeanu, C., Salomaa, K., and Yu, S. Tight lower bound for the state complexity of shuffle of regular languages. *J. Automata, Languages and Combinatorics 7*, 3 (2002), 303–310.

[9] Choffrut, C., and Karhumäki, J. On Fatou properties of rational languages. In *Where Mathematics, Computer Science, Linguistics and Biology Meet* (2000), C. Martin-Vide and V. Mitrana, Eds., Kluwer, pp. 227–235.

[10] Conway, J. *Regular Algebra and Finite Machines*. Chapman and Hall, 1971.

[11] Daley, M., Ibarra, O., and Kari, L. Closure properties and decision questions of some language classes under ciliate bio-operations. *Theor. Comp. Sci. 306*, 1 (2003), 19–38.

[12] Daley, M., and McQuillan, I. Template-guided DNA recombination. *Theor. Comp. Sci. 330* (2005), 237–250.

[13] Day, P.-H., and Shyr, H. Languages defined by some partial orders. *Soochow J. Math. 9* (1983), 53–62.

[14] de Luca, A., and Varricchio, S. Regularity and finiteness conditions. pp. 747–810. In [84], volume I.

[15] Diekert, V., and Métivier, Y. Partial commutation and traces. pp. 457–534. In [84], volume III.

[16] Domaratzki, M. State complexity of proportional removals. *J. Automata, Languages and Combinatorics 7*, 4 (2002), 455–468.

[17] Domaratzki, M. Deletion along trajectories. *Theor. Comp. Sci. 320*, 2–3 (2004), 293–313.

[18] Domaratzki, M. On the decidability of 2-infix-outfix codes. Tech. Rep. 2004-479, School of Computing, Queen's University, 2004.

[19] Domaratzki, M. Semantic shuffle on and deletion along trajectories. In *Developments in Language Theory* (2004), C. Calude, E. Calude, and M. Dineen, Eds., vol. 3340 of *LNCS*, Springer, pp. 163–174.

[20] Domaratzki, M. Trajectory-based codes. *Acta Inf. 40*, 6–7 (2004), 491–527.

[21] Domaratzki, M. Trajectory-based embedding relations. *Fund. Inf. 59*, 4 (2004), 349–363.

[22] Domaratzki, M. *Trajectory-Based Operations*. PhD thesis, Queen's University, 2004.

[23] Domaratzki, M., Mateescu, A., Salomaa, K., and Yu, S. Deletion along trajectories and commutative closure. In *Proceedings of WORDS'03: 4th International Conference on Combinatorics on Words* (2003), T. Harju and J. Karhumäki, Eds., pp. 309–319.

[24] Domaratzki, M., and Salomaa, K. State complexity of shuffle on trajectories. In *Descriptional Complexity of Formal Systems (DCFS)* (2002), J. Dassow, M. Hoeberechts, H. Jürgensen, and D. Wotschke, Eds., pp. 95–109. To appear, J. Automata, Languages and Combinatorics.

[25] Domaratzki, M., and Salomaa, K. Decidability of trajectory-based equations. In *Mathematical Foundations of Computer Science 2004* (2004), J. Fiala, V. Koubek, and J. Kratochvil, Eds., vol. 3153 of *LNCS*, Springer, pp. 723–734. To appear, Theor. Comp. Sci.

[26] Domaratzki, M., and Salomaa, K. Restricted sets of trajectories and decidability of shuffle decompositions. In *Descriptional Complexity of Formal Systems (DCFS 2004)* (2004), L. Ilie and D. Wotschke, Eds., pp. 37–51. To appear, Int. J. Found. Comp. Sci.

[27] Domaratzki, M., and Salomaa, K. Codes defined by multiple sets of trajectories. To appear, *AFL 2005* (2005).

[28] Ehrenfeucht, A., Harju, T., Petre, I., Prescott, D., and Rozenberg, G. *Computation in Living Cells: Gene assembly in ciliates*. Springer, 2004.

[29] Ehrenfeucht, A., Haussler, D., and Rozenberg, G. On regularity of context-free languages. *Theor. Comp. Sci. 23* (1983), 311–332.

[30] Fan, C.-M., Shyr, H., and Yu, S. S. *d*-Words and *d*-languages. *Acta Inf. 35* (1998), 709–727.

[31] Guo, Y., Shyr, H., and Thierrin, G. E-Convex infix codes. *Order 3* (1986), 55–59.

[32] Haines, L. On free monoids partially ordered by embedding. *J. Comb. Th. 6* (1969), 94–98.

[33] Harju, T., and Ilie, L. On quasi orders of words and the confluence property. *Theor. Comp. Sci. 200* (1998), 205–224.

[34] Harju, T., Mateescu, A., and Salomaa, A. Shuffle on trajectories: The Schützenberger product and related operations. In *Mathematical Foundations of Computer Science 1998* (1998), L. Brim, J. Gruska, and J. Zlatuska, Eds., no. 1450 in Lecture Notes in Computer Science, Springer, pp. 503–511.

[35] Higman, G. Ordering by divisibility in abstract algebras. *Proc. London Math. Soc. 2*, 3 (1952), 326–336.

[36] Holzer, M., and Kutrib, M. State complexity of basic operations on nondeterministic finite automata. In *Implementation and Application of Automata: 7th International Conference, CIAA 2002* (2003), J.-M. Champarnaud and D. Maurel, Eds., vol. 2608 of *Lecture Notes in Computer Science*, Springer, pp. 148–157.

[37] Ilie, L. Remarks on well quasi orders of words. In *Proceedings of the 3rd DLT* (1997), S. Bozapalidis, Ed., pp. 399–411.

[38] Ilie, L. *Decision Problems on Orders of Words*. PhD thesis, University of Turku, 1998.

[39] Ito, M., Jürgensen, H., Shyr, H., and Thierrin, G. n-prefix-suffix languages. *Intl. J. Comp. Math. 30* (1989), 37–56.

[40] Ito, M., Jürgensen, H., Shyr, H., and Thierrin, G. Outfix and infix codes and related classes of languages. *J. Comp. Sys. Sci. 43* (1991), 484–508.

[41] Ito, M., Kari, L., and Thierrin, G. Shuffle and scattered deletion closure of languages. *Theor. Comp. Sci. 245* (2000), 115–133.

[42] Ito, M., and Silva, P. Remark on deletions, scattered deletions and related operations on languages. In *Semigroups and Applications* (1998), J. Howie and N. Ruskuc, Eds., World Scientific, pp. 97–105.

[43] Jirásek, J., Jirásková, G., and Szabari, A. State complexity of concatenation and complementation of regular languages. In *Implementation and Application of Automata* (2004), M. Domaratzki, A. Okhotin, K. Salomaa, and S. Yu, Eds., vol. 3317 of *LNCS*, Springer, pp. 178–189.

[44] Jirásková, G. State complexity of some operations on regular languages. In *Descriptional Complexity of Formal Systems: Fifth International Workshop* (2003), E. Csuhaj-Varjú, C. Kintala, D. Wotschke, and G. Vaszil, Eds., pp. 114–125.

[45] Jullien, P. Sur un théorème d'extension dans la théorie des mots. *CR Acad. Sc. Paris (Série A) 266* (1968), 851–854.

[46] Jürgensen, H., and Konstantinidis, S. Codes. pp. 511–600. In [84], volume I.

[47] Jürgensen, H., Salomaa, K., and Yu, S. Transducers and the decidability of independence in free monoids. *Theor. Comp. Sci. 134* (1994), 107–117.

[48] Kadrie, A., Dare, V., Thomas, D., and Subramanian, K. Algebraic properties of the shuffle over ω-trajectories. *Inf. Proc. Letters 80*, 3 (2001), 139–144.

[49] Kari, L. Generalized derivatives. *Fund. Inf. 18* (1993), 27–39.

[50] Kari, L. On language equations with invertible operations. *Theor. Comp. Sci. 132* (1994), 129–150.

[51] Kari, L., Konstantinidis, S., and Sosík, P. On properties of bond-free DNA languages. Tech. Rep. 609, Computer Science Department, University of Western Ontario, 2003.

[52] Kari, L., Konstantinidis, S., and Sosík, P. Substitutions on trajectories. In *Theory is Forever* (2004), J. Karhumäki, H. Maurer, G. Paun, and G. Rozenberg, Eds., vol. 3113 of *LNCS*, Springer, pp. 145–158.

[53] Kari, L., Konstantinidis, S., and Sosík, P. Substitutions, trajectories and noisy channels. In *Implementation and Application of Automata* (2004), M. Domaratzki, A. Okhotin, K. Salomaa, and S. Yu, Eds., vol. 3317 of *LNCS*, Springer, pp. 202–212.

[54] Kari, L., Konstantinidis, S., and Sosík, P. On properties of bond-free DNA languages. *Theor. Comp. Sci. 334* (2005), 131–159.

[55] Kari, L., and Sosík, P. Language deletions on trajectories. Tech. Rep. 606, Computer Science Department, University of Western Ontario, 2003.

[56] Kari, L., and Sosík, P. Aspects of shuffle and deletion on trajectories. *Theor. Comp. Sci. 332*, 1–3 (2005), 47–61.

[57] Kari, L., and Thierrin, G. k-catenation and applications: k-prefix codes. *J. Inf. Opt. Sci. 16*, 2 (1995), 263–276.

[58] Kari, L., and Thierrin, G. Contextual insertions/deletions and computability. *Inf. and Comp. 131* (1996), 47–61.

[59] Kari, L., and Thierrin, G. Maximal and minimal solutions to language equations. *J. Comp. Sys. Sci. 53* (1996), 487–496.

[60] Kosaraju, S. Correction to "Regularity Preserving Functions". *ACM SIGACT News 6*, 3 (1974), 22.

[61] Kosaraju, S. Regularity preserving functions. *ACM SIGACT News 6*, 2 (1974), 16–17.

[62] Kosaraju, S. Context-free preserving functions. *Math. Sys. Theor. 9*, 3 (1975).

[63] Kozen, D. On regularity-preserving functions. Tech. Rep. TR95-1559, Department of Computer Science, Cornell University, 1995.

[64] Kunc, M. Simple language equations. *Bull. Eur. Assoc. Theor. Comp. Sci. 85* (2005), 81–102.

[65] Lam, N. Finite maximal infix codes. *Semigroup Forum 61* (2000), 346–356.

[66] Latteux, M., and Roos, Y. Synchronized shuffle and regular languages. In *Jewels are Forever: Contributions on Theoretical Computer Science in Honor of Arto Salomaa* (1999), J. Karhumäki, H. Maurer, G. Păun, and G. Rozenberg, Eds., Springer, pp. 35–44.

[67] Leiss, E. *Language Equations*. Monographs in Computer Science. Springer, 1999.

[68] Long, D. On nilpotency of the syntactic monoid of a language. In *Words, Languages and Combinatorics II* (1992), M. Ito and H. Jürgensen, Eds., World Scientific, pp. 279–293.

[69] Long, D. On two infinite hierarchies of prefix codes. In *Proceedings of the Conference on Ordered Structures and Algebra of Computer Languages* (1993), K. Shum and P. Yuen, Eds., World Scientific, pp. 81–90.

[70] Long, D. k-bifix codes. *Rivista di Matematica Pura ed Applicata 15* (1994), 33–55.

[71] Long, D. *Study of Coding Theory and its Application to Cryptography*. PhD thesis, City University of Hong Kong, 2002.

[72] Long, D., Jia, W., Ma, J., and Zhou, D. k-p-infix codes and semaphore codes. *Disc. Appl. Math. 109* (2001), 237–252.

[73] Long, D., Ma, J., and Zhou, D. Structure of 3-infix-outfix maximal codes. *Theor. Comp. Sci. 188* (1997), 231–240.

[74] Lothaire, M. *Combinatorics on Words*. Addison-Wesley, 1983.

[75] Martin-Vide, C., Mateescu, A., Rozenberg, G., and Salomaa, A. Contexts on trajectories. *Intl. J. Comp. Math. 73*, 1 (1999), 15–36.

[76] Mateescu, A. CD grammar systems and trajectories. *Acta. Cyb. 13*, 2 (1997), 141–157.

[77] Mateescu, A. Splicing on routes: a framework of DNA computation. In *Unconventional Models of Computation* (1998), C. Calude, J. Casti, and M. Dinneen, Eds., Springer, pp. 273–285.

[78] Mateescu, A. Words on trajectories. *Bull. Eur. Assoc. Theor. Comp. Sci. 65* (1998), 118–135.

[79] Mateescu, A., Rozenberg, G., and Salomaa, A. Shuffle on trajectories: Syntactic constraints. *Theor. Comp. Sci. 197* (1998), 1–56.

[80] Okhotin, A., and Salomaa, K. Contextual grammars with uniform sets of trajectories. *Fund. Inf. 64* (2005), 341–351.

[81] Pin, J.-E., and Sakarovitch, J. Some operations and transductions that preserve rationality. In *6th GI Conference* (1983), A. Cremers and H.-P. Kriegel, Eds., vol. 145 of *Lecture Notes in Computer Science*, Springer, pp. 277–288.

[82] Prescott, D., Ehrenfeucht, A., and Rozenberg, G. Template-guided recombination for ies elimination and unscrambling of genes in stichotrichous ciliates. *J. Theoret. Biol. 222* (2003), 323–330.

[83] Păun, G., and Salomaa, A. Thin and slender languages. *Disc. Appl. Math. 61* (1995), 257–270.

[84] Rozenberg, G., and Salomaa, A., Eds. *Handbook of Formal Languages*. Springer, 1997.

[85] Salomaa, A., and Yu, S. On the decomposition of finite languages. In *Developments in Language Theory* (1999), G. Rozenberg and W. Thomas, Eds., pp. 22–31.

[86] Seiferas, J., and McNaughton, R. Regularity-preserving relations. *Theor. Comp. Sci. 2* (1976), 147–154.

[87] Shallit, J. Numeration systems, linear recurrences, and regular sets. *Inf. and Comp. 113*, 2 (1994), 331–347.

[88] Shyr, H. *Free Monoids and Languages*. Hon Min Book Company, Taichung, Taiwan, 2001.

[89] Shyr, H., and Thierrin, G. Hypercodes. *Inf. and Cont. 24*, 1 (1974), 45–54.

[90] Stearns, R., and Hartmanis, J. Regularity preserving modifications of regular expressions. *Inf. and Cont. 6*, 1 (1963), 55–69.

[91] SZILARD, A., YU, S., ZHANG, K., AND SHALLIT, J. Characterizing regular languages with polynomial densities. In *Mathematical Foundations of Computer Science 1992* (1992), I. Havel and V. Koubek, Eds., vol. 629 of *Lecture Notes in Computer Science*, Springer, pp. 494–503.

[92] THIERRIN, G. Convex languages. In *Automata, Languages and Programming, Colloquium, Paris, France* (1972), M. Nivat, Ed., pp. 481–492.

[93] VAN LEEUWEN, J. Effective constructions in well-partially ordered free monoids. *Disc. Math. 21* (1978), 237–252.

[94] YU, S., ZHUANG, Q., AND SALOMAA, K. The state complexities of some basic operations on regular languages. *Theor. Comp. Sci. 125* (1994), 315–328.

[95] YU, S. S. *d*-Minimal languages. *Disc. Appl. Math. 89* (1998), 243–262.

[96] ZHANG, G.-Q. Automata, boolean matrices, and ultimate periodicity. *Inf. and Comp. 152* (1999), 138–154.

[97] ZHANG, L., AND SHEN, Z. Completion of recognizable bifix codes. *Theor. Comp. Sci. 145* (1995), 345–355.