Chapter 3

Related Work

3.1 Introduction

In this chapter, we review the literature relevant to this thesis. Our focus is on word operations, such as shuffle, insertion, and quotient, which are specific instances of the formalisms we present in this thesis. We focus primarily on research which is either of theoretical interest, or relates directly to the topics we investigate later in the thesis.

3.2 Shuffle

Shuffle is one of the most studied operations on formal languages which is not among the defining operations of regular expressions. Ginsburg and Spanier introduce a definition of shuffle in 1965 [53] in their study of generalized sequential machines. This is the first reference to shuffle as an operation on languages we have been able to find. The natural application of shuffle as a model for interleaving processes yielded much research into shuffle and related operations. In an early paper on shuffle, Ogden *et al.* show that there exist DCFLs L_1, L_2 such that $L_1 \sqcup\!\sqcup L_2$ is NP-complete [156]. Hausler and Zeiger [63] give an interesting representation theorem for r.e. languages using the homomorphic image of the intersection of a regular language and the shuffle of two fixed Dyck languages.

We now consider three specific areas of research on arbitrary shuffle: iterated shuffle, shuffle decompositions and grammar formalisms involving shuffle.

3.2.1 Iteration

The iteration of shuffle has received much attention in the literature over the last thirty years. This operation is defined much in the same way as Kleene closure: given a language L its shuffle closure is defined as

$$(\sqcup\!\sqcup)^*(L) = \bigcup_{i \geq 0} (\sqcup\!\sqcup)^i(L),$$

where $(\sqcup\!\sqcup)^0(L) = \{\epsilon\}$, $(\sqcup\!\sqcup)^{i+1}(L) = (\sqcup\!\sqcup)^i(L) \sqcup\!\sqcup L$ for all $i \geq 0$. Several notations are used in the literature for denoting $(\sqcup\!\sqcup)^*(L)$, including $L^{\otimes}$ and $L^{\dagger}$.

Much of the interest in shuffle closure comes from the theory of concurrency and formal software engineering research communities. For example, Shaw, in describing the shuffle closure operation in the context of *flow expressions*, notes that shuffle closure is a “concurrent analogue of [the Kleene closure operation]”, which “is useful where there may be a variable number of interleaves of some flow [of control], for example in describing systems in which processes or resources may be dynamically created and destroyed. [182, p. 243]”. Riddle also performed early research into software engineering using the shuffle closure operation [170]. While the shuffle closure operation is fundamental to this research, various authors (including both Shaw and Riddle) also incorporate synchronization methods for research into software engineering. More recently, Igarashi and Kobayashi [71] cite shuffle expressions as a valid manner in specifying trace sets for use in their formal analysis of resource usage.

Other research into iterated shuffle has proceeded from a purely theoretical standpoint. Warmuth and Haussler [198] show the following elegant result:

Theorem 3.2.1 *Let $\Sigma = \{a, b, c\}$. Given words $u, v \in \Sigma^*$, it is* NP*-complete to determine whether $u \in (\sqcup\!\sqcup)^*(v)$.*

Imreh *et al.* [74] have written on the shuffle closure of commutative regular languages. In particular, they give two characterizations of when the shuffle closure of a commutative regular language is again regular.

3.2.2 Decomposition

The shuffle decomposition problem has received much attention recently. For shuffle on trajectories, the problem was introduced by Mateescu *et al.* [147], who asked, given a language L, is it possible to write $L = L_1 \sqcup\!\sqcup_T L_2$ for some L_1, L_2, T, where the complexity of L_1, L_2, T are "somehow smaller [147, p. 38]" than the complexity of L (e.g., each are situated lower in the Chomsky hierarchy than L). They called such a simpler expression for L a *parallelization* of L, and noted that some languages, such as the non-context free languages $L = \{ww : w \in \Sigma^*\}$ and $L = \{a^n b^{n^2} : n \geq 0\}$ do not have parallelizations into context-free languages.

Câmpeanu *et al.* [21] have studied the problem of deciding whether a regular language R has a parallelization $R = L_1 \sqcup\!\sqcup L_2$, i.e., the case when $T = (0+1)^*$. If such a parallelization exists, and $L_1, L_2 \neq \{\epsilon\}$, such an expression is called a (non-trivial) *shuffle decomposition*. Despite much effort, Câmpeanu *et al.* [21] were not able to resolve whether it is decidable, given a regular language R, whether R has a non-trivial shuffle decomposition. For certain subclasses of regular languages, Câmpeanu *et al.* were able to positively decide whether a language from that subclass has a non-trivial shuffle decomposition.

Ito [75] has also examined the shuffle decomposition problem for regular languages. Let $\mathcal{I}(n, \Sigma)$ be the class of all regular languages over Σ which are accepted by some DFA with at most n states. The main result of Ito [75] is the following:

Theorem 3.2.2 *Given a regular language $R \subseteq \Sigma^*$ and $n \in \mathbb{N}$, it is decidable whether there exist L_1, L_2 with $L_1 \in \mathcal{I}(n, \Sigma)$ and $L_2 \neq \{\epsilon\}$ such that $R = L_1 \sqcup\!\sqcup L_2$.*

The general problem of determining whether a regular language has a non-trivial shuffle decomposition is still open. We will examine the shuffle decomposition problem with respect to a set of trajectories T (i.e., deciding whether there exists L_1, L_2 such that $R = L_1 \sqcup\!\sqcup_T L_2$) in Chapter 7.

Iwama [84] has considered shuffle decomposition in a different sense. Say that languages $(L_1, \ldots, L_n)$ are *uniquely shuffle-decomposable* if each word in $z \in L_1 \sqcup\!\sqcup L_2 \sqcup\!\sqcup \cdots \sqcup\!\sqcup L_n$ can be represented uniquely as $z \in x_1 \sqcup\!\sqcup x_2 \sqcup\!\sqcup \cdots \sqcup\!\sqcup x_n$ with $x_i \in L_i$ for $1 \leq i \leq n$. Given regular languages $(L_1, \ldots, L_n)$, Iwama gives an algorithm to decide whether they are uniquely shuffle-decomposable.

3.2.3 Grammar Formalisms

In the theory of concurrency and software engineering, several models have been proposed which adjoin grammars and regular expressions with shuffle and iterated shuffle.

Several papers have considered the class of languages defined by regular expressions adjoined with shuffle and iterated shuffle. This class of languages, under various names, has been extensively studied, and we can only give a list of the work done so far, including that of Gisher [55], Araki *et al.* [8], Araki and Tokura [7], Jędrezejowicz [87, 88, 89, 90, 91, 92], Janzten [86], Jędrzejowicz and Szipietowski [93], and many others.

Guo *et al.* [56] have introduced *synchronization expressions*, which are regular expressions augmented with a restricted form of shuffle. Synchronization expressions were developed as a model for specifying the synchronization which occurs between processes in a parallel system. The notion of synchronization expressions has been further examined by Salomaa and Yu [177, 178] and Clerbout *et al.* [26, 27, 172].

The concept of shuffle-star height (analogous to the usual (Kleene-) star height) has been implicitly studied by Gisher [55] and subsequently by Jędrezejowicz [88, 89, 90], where it was first shown that there exist languages of shuffle-star height n for all $n \geq 0$, over an alphabet of size $3n$ [89]. Jędrezejowicz [90] later extended this to show that there exist languages of shuffle-star height n for all $n \geq 0$ over an alphabet of size seven. Jędrezejowicz leaves open the problem of whether the alphabet size seven is optimal, as well as the problem of characterizing all morphisms which preserve shuffle-star height [90, Rem. 5.2].

Araki and Tokura [7] investigate decision problems for regular expressions augmented with shuffle and shuffle-closure, and show, e.g., that the membership and emptiness problems for these expressions are decidable, while their equivalence and containment problems are undecidable. Further decidability problems are studied by Jędrezojowicz [91].

Shoudai [183] describes a P-complete language using shuffle expressions.

3.3 Insertion and Deletion Operations

We now consider results on insertion and deletion operations. The insertion operations we consider are those modelled by shuffle on trajectories, and thus have special relevance to the work in this thesis. We do not survey research on insertion operations which are not modelled by shuffle on trajectories, e.g., the work of Kari [107] on controlled insertion and deletion. The deletion operations we will survey are primarily those which can be modelled by deletion on trajectories, which we introduce in Chapter 5.

3.3.1 Insertion Operations

Besides shuffle and concatenation, the (sequential) insertion operation is perhaps the most natural operation which inserts all of the symbols of one word into another. It is defined as follows:

$$u \leftarrow v = \{u_1 v u_2 \ : \ u_1 u_2 = u\}.$$

We noted in Section 2.7.1 that insertion is a particular case of shuffle on trajectories. Kari has studied the properties of insertion [104, 106], including the solutions of language equations involving insertion. We generalize these results in Chapter 7.

The bi-catenation operation is defined as follows: $u \odot v = \{uv, vu\}$. The bi-catenation operation was defined by Shyr and Yu [187], and further studied by Hsiao *et al.* as a particular case of their general study of binary word operations [69]. Shyr and Yu are motivated by considering bi-catenation as a restriction of shuffle, and related code-theoretic properties.

Kari and Thierrin [114, 115] have defined the operation of k-insertion as follows: given $k \geq 0$, the k-insertion of $u, v \in \Sigma^*$ is defined as

$$u \leftarrow^k v = \{u_1 v u_2 \ : \ u = u_1 u_2, |u_2| \leq k\}.$$

We note that k-insertion can be modelled by shuffle on trajectories, and also that

$$u \leftarrow v = \bigcup_{k \geq 0} u \leftarrow^k v.$$

The k-insertion operation is motivated by Kari and Thierrin as follows:

> Even though insertion generalizes catenation, catenation cannot be obtained as a particular case of it, as we cannot force the insertion to take place at the end of the word. The k-insertion provides the control needed to overcome this drawback. The k-insertion is thus more nondeterministic than catenation, but more restrictive than insertion. [115, p. 479]

Kari and Thierrin [114] study the k-insertion (and corresponding k-deletion) closure of a language. They also define the notion of k-prefix codes [114], which are a particular case of T-codes introduced in Chapter 6. However, we note that k-prefix codes are one of the few cases of research into codes where a novel definition is based primarily on a new language operation, rather than a new binary relation on words.

Berard [16] has introduced both the *literal* and *initial literal shuffle* operations. The motivation is modelling concurrent processes; literal shuffle models synchronized transmission where "each transmitter emits, in turn, one elementary signal [16, p. 51]". Both literal and initial literal shuffle are particular cases of shuffle on trajectories, and are given by $T = (0^* + 1^*)(01)^*(0^* + 1^*)$ and $T = (01)^*(0^* + 1^*)$, respectively. Literal shuffle has been further studied by Tanaka [191] on the closure of the class of prefix codes under literal shuffle, and by Ito and Tanaka [81] who consider the density of initial literal shuffles. Moriya and Yamasaki [154] have studied literal shuffle on ω-words.

3.3.2 Deletion Operations

Many deletion operations which are specific instances of the deletion along trajectories model we suggest in Chapter 5 have been considered in the literature. This shows the usefulness of the deletion along trajectories model.

The most studied deletion operations are the left- and right-quotient operations. The first formal study of quotient appears to be by Ginsburg and Spanier [52], who show three fundamental results on right-quotient: that the right-quotient of a CFL by a regular language (or of a regular language by a CFL) is a CFL, that CF is not closed under quotient, and given two CFLs L_1, L_2, it is undecidable whether L_1/L_2 is a CFL. Ginsburg and Spanier attribute the notion of quotient to the "SHARE Theory of Information Handling Committee [52, p.487]".

Latteux *et al.* [130] show that a restricted class of CFLs, called the *one-counter languages*, are closed under quotient, and that every recursively enumerable language can be expressed as the quotient of two LCFLs.

Another well-studied deletion operation is known as *scattered deletion*. Given two words $x, y \in \Sigma^*$, their scattered deletion, denoted $x \rightsquigarrow y$, is given by

$$x \rightsquigarrow y = \left\{ \prod_{i=1}^{n+1} x_i \ : \ x = (\prod_{i=1}^{n} x_i y_i) x_{n+1}, \ y = \prod_{i=1}^{n} y_i \text{ with } x_i, y_j \in \Sigma^* \right\}.$$

We extend $\rightsquigarrow$ to languages as expected. The scattered deletion operation, a natural operation on words, has a long history in the literature. For instance, the scattered deletion operation is an implicit operation in the theory of flow expressions (see, e.g., Shaw [182]).

Kari (as Sântean [179]) appears to be the first author to have formally studied the scattered deletion operation (under the name *literal subtraction*) and established several closure properties. This investigation is continued by Kari in a subsequent paper [105].

Also investigated by Kari [105] are several other deletion operations, some of which are modelled by our framework (e.g., sequential deletion), and others which are not (e.g., controlled deletion, parallel deletions and deletion with permuted components). Closure properties of each of these operations are investigated.

The sequential deletion operation is given by $x \rightarrow y = \{x_1x_2 \ : \ x_1yx_2 = x\}$. Kari *et al.* [111] explore results on the cardinality of $w \rightarrow L$, for $w \in \Sigma^*$ and $L \subseteq \Sigma^*$, as well as the decidability of the following problem: given a finite set F, do there exist $w \in \Sigma^*$ and $L \subseteq \Sigma^*$ such that $F = w \rightarrow L$?

Language equations involving deletion have been studied by Kari [106]. Recently, Kari and Sosík have continued the investigation of language equations involving scattered deletion, quotient and sequential deletion [113].

Meduna [153] has introduced an interesting deletion operation, called *middle quotient*, defined as follows:

$$L_1|L_2 = \{w \in \Sigma^* \ : \ \exists v \in L_2 \text{ such that } vwv \in L_1\}.$$

The main motivation for introducing this operation is that for any recursively enumerable language L, there exist linear CFLs L_1, L_2 such that $L = L_1|L_2$ [153].

A popular topic in the theory of formal languages is proportional removals. Given a binary relation $r \subseteq \mathbb{N}^2$, the proportional removal of a language $L \subseteq \Sigma^*$ with respect to r is the language

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

Proportional removals have been studied by Stearns and Hartmanis [189], Amar and Putzolu [4, 5] Seiferas and McNaughton [180], Kosaraju [120, 121, 122], Kozen [123], Zhang [205], the author [35], and others. We study proportional removals extensively in Chapter 5.

Berstel *et al.* [17] consider filtering, which is a deletion operation specified by a sequence of natural numbers $s \subseteq \mathbb{N}$. We will see that filtering is a specific case of deletion along trajectories. Necessary and sufficient conditions on a sequence of natural numbers preserving regularity are given by Berstel *et al.* [17].

3.3.3 Interaction

Kari [102] has studied conditions on which the operations of insertion and deletion are reversible and deterministic. In particular, given the inverse operations (intuitively, but also in a sense we will define in Chapter 5) of (sequential) insertion and deletion, Kari examines under what conditions on words u, v the language $(u \leftarrow v) \rightarrow v$ consists of only one word.

3.3.4 Iteration

Iterated insertion and deletion operations have been studied by Ito *et al.* [78, 79], and Kari and Thierrin [117]. The iterated insertion operations considered are sequential insertion, shuffle and k-insertion; the corresponding iterated deletion operations are also considered. In each case, the authors consider the *residual* of a language L under the studied operation, and show its relation to the closure of L under the corresponding insertion operation. We generalize these notions for shuffle and deletion along trajectories in Chapter 8.

Ito and Silva [80] have examined closure properties of iterated scattered and sequential deletion. Two open problems proposed by Ito and Silva have been solved by the author and Okhotin [42].

Ito *et al.* [82] have examined shuffle-closed languages, strongly shuffle-closed languages and extended shuffle bases. Characterizations of (strongly) shuffle-closed commutative regular languages are obtained. The notion of extended bases has been developed in the more general setting of binary word operations by Hsiao *et al.* [69].

Kari and Thierrin have generalized the notion of primitivity from Kleene closure to iterated shuffle and insertion [118]. In a broader setting, Hsiao *et al.* [69] have considered iteration and primitivity of arbitrary word operations. However, the setting is so general that obtaining results often requires many assumptions, and results such as closure properties and decidability cannot be obtained.

An interesting application of results on iteration of insertion and deletion operations was noted by Parkes and Thomas [161, 162]. In particular, the word problem for the syntactic monoid of a regular language R can be expressed as the intersection of the insertion- and deletion-closure of R, which were introduced by Ito *et al.* [78]. Similar observations were made by Tully [194], but phrased in more group-theoretic terms. Ramesh Kumar and Rajan [169] have further explored the concepts introduced by Tully.

3.3.5 Decomposition and Related Language Equations

The problem of decomposition of languages for insertion operations has not been widely studied, except for the case of concatenation. Given a regular language R, the problem of determining whether there exist L_1, L_2 such that $R = L_1L_2$ has been considered by Conway [28], Kari [106], and Kari and Thierrin [117]. This problem is decidable. Choffrut and Karhumäki [25] and Polák [167] have considered more general systems of equations and inequalities (see also Baader and Küsters [11] and Baader and Narendran [13], who reduce solving similar systems of equations to solving a single language equation). The equations considered by Choffrut and Karhumäki and Polák include the decomposition equation $R = X_1X_2$ studied previously by Conway, Kari and Kari and Thierrin, but also include equations of the form $R = r(X_1, \ldots, X_n)$, where R is a regular language and $r(X_1, \ldots, X_n)$ is a regular expression over the variables $X_1, \ldots, X_n$.

Given a language R, we say that it is prime if $R = L_1L_2$ implies that $\{L_1, L_2\} = \{\{\epsilon\}, R\}$. Salomaa and Yu [176] show that the problem of deciding whether a regular language is prime is decidable; see also Mateescu *et al.* [151]. Wood [199] has given conditions on R which ensure that a decomposition $R = L_1L_2$ is unique.

3.4 Shuffle on Trajectories

As already mentioned, shuffle on trajectories was defined by Mateescu *et al.* [147]. Harju *et al.* [61] consider the syntactic monoids recognizing a language constructed from regular languages with shuffle on trajectories. We examine the complementary question for deletion along trajectories in Section 5.3.1. We now describe other areas of research related to shuffle on trajectories.

3.4.1 Infinite Words

While we do not deal with infinite words in this thesis, the concept of shuffle on trajectories for infinite words has received attention in the literature. Mateescu *et al.* [147] introduced the notion of shuffle on trajectories for infinite words along with shuffle on trajectories for finite words, and examined similar algebraic properties for infinite trajectories as for finite trajectories. Trajectories for infinite words are called *ω-trajectories*. Kadrie *et al.* [101] have defined a binary relation defined on Σ^ω and briefly examined its properties (we consider the analog for finite words in Chapter 6).

3.4.2 Fairness

Defining a fair operation, that is, one which allows both input languages to have a corresponding letter be "shuffled in" during some reasonable time frame, has been the subject of research related to shuffle on trajectories.

Mateescu *et al.* [147] use the concept of fairness as an example of the usefulness of the model of shuffle on trajectories. They define explicit sets of trajectories and ω-trajectories which have the desired fairness properties. Mateescu *et al.* [152] have extended this to study fairness of multiple languages, which requires defining an extended shuffle on trajectories operation to operation on n languages instead of two. Mateescu and Mateescu [145] have examined the fair and associative trajectories on ω-words.

3.4.3 Related Concepts

The notion of shuffle on trajectories has been used in other interesting settings, including grammars, combinatorics and timed automata. We survey these now.

Grammar Formalisms

Martin-Vide *et al.* [142] introduce the notion of contextual grammars on trajectories. These are an extension of the notion of a contextual grammar by the addition of a set of trajectories.

In particular a *contextual grammar with contexts shuffled on trajectories* (abbreviated CST) is a four-tuple $G = (\Sigma, B, C, T)$ where Σ is an alphabet, B, C are finite languages over Σ, called the *base* and *contexts*, respectively, and $T = (T_c)_{c \in C}$ is a family of trajectories indexed by elements of C, i.e., for each $c \in C$, $T_c \subseteq \{0, 1\}^*$.

The generation of words in G is accomplished as follows: let $x, y \in \Sigma^*$. Then we use the notation $x \Rightarrow_G y$ to denote the fact that there exists $c \in C$ such that $y \in x \sqcup\!\sqcup_{T_c} c$. Let $\Rightarrow_G^*$ be the reflexive and transitive closure of $\Rightarrow_G$. Then the language generated by $G = (\Sigma, B, C, T)$ is denoted $L(G)$ and is given by

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

Martin-Vide *et al.* give the following example: let $G = (\Sigma, B, C, T)$ be given by $\Sigma = \{a, b\}$, $B = \{\epsilon\}$, $C = \{aa, bb\}$ and $T = (T_{aa}, T_{bb})$, where $T_{aa} = T_{bb} = \{01^n01^n \ : \ n \geq 0\}$. Then $L(G) = \{ww \ : \ w \in \{a, b\}^*\}$.

Martin-Vide *et al.* investigate the relationship between CST and other contextual grammar classes. They also examine the relationship between the complexity of the members of T as languages and the generative capacity of G.

Mateescu has also extended the notion of co-operating distributed grammars (CD grammars) to encompass the notion of trajectories [143]. 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 $G_i = (V, \Sigma, S, P_i)$, as defined in Section 2.3, 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 language generated by Γ, denoted $L(\Gamma)$, is the set of all words generated by Γ. The usual notion of a CD grammar corresponds to $T = 0^*1^*$. Other more complicated notions of acceptance are also considered. 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 $T \subseteq \{0, \ldots, n-1\}^*$.

Timed Automata

Krishnan [124] has utilized the notion of trajectories in the context of discrete event systems and timed automata. The concept of a trajectory is extended to the concept of a scheduler for real-time events.

Combinatorics

The notion of shuffle on trajectories has been employed in an interesting combinatorial setting. In particular, Vajnovski [195] has constructed a Gray code for the so-called Motzkin words; the use of shuffle on trajectories in the construction is essential. We do not describe Gray codes or Motzkin words here, the reader may consult [195] for definitions. Baril and Vajnovski [15] also define a Gray code for derangements (permutations with no fixed points), again using shuffle on trajectories in a combinatorial setting.

Vajnovski has also used the concept of shuffle on trajectories as a combinatorial constructor for *multiset permutations* [196] (given $n_0, n_1, n_2, \ldots, n_k \geq 0$, a multiset permutation is a sequence integers in which i appears n_i times for all $0 \leq i \leq k$). A combinatorial constructor enables one to construct complex combinatorial objects (in this case, multiset permutations) out of simpler objects, which is a common theme in combinatorial research. The construction of Vajnovski allows Gray code generation of multiset permutations by a so-called *loopless* method [196], by using shuffle on trajectories.

3.4.4 Splicing on Routes

The notion of shuffle on trajectories was extended by Mateescu [144] to encompass certain splicing operations. This extension is called *splicing on routes*. We give the formal definition of splicing on routes in Section 5.7. Splicing on routes is a proper extension of shuffle on trajectories, and also encompasses several unary operations. We discuss the unary operations modelled by splicing on routes in Section 5.7. Bel-Enguix *et al.* use the concept of splicing on routes to model dialog in natural language [12].

3.4.5 Concurrent Work

Independent to this thesis, the concept of deletion on trajectories has been introduced by Kari and Sosík [112]. The authors develop the same framework, and investigate similar closure properties and decidability of solutions to language equations in one variable. Algebraic properties not studied in this thesis are also considered by Kari and Sosík. Unlike the case of shuffle on trajectories, these algebraic characterizations for deletion along trajectories are satisfied only by trivial deletion operations. For example, a deletion operation $\diamond$ modelled by deletion along trajectories is commutative if and only if $L_1 \diamond L_2 \subseteq \{\epsilon\}$ for all languages L_1, L_2 [112].

Kari and Sosík [112] also introduce the notion of substitution and right-difference on trajectories. This concept is similar to shuffle and deletion along trajectories, but involves substitution of words rather than interleaving of words. The reader is referred to Kari and Sosík for details. The notion of substitution and right-difference on trajectories is further investigated and applied to the modelling of noisy channels by Kari *et al.* [110].

The use of shuffle and deletion along trajectories has been employed by Kari *et al.* [108] to investigate properties of bonding in DNA strands. The formalism defined by Kari *et al.* is called *bond-free properties*. There are similarities between bond-free properties and the notion of T-codes developed in Chapter 6. We discuss these similarities in greater detail in Chapter 6. Kari *et al.* [109] have extended this work on bond-free properties, with particular emphasis on DNA strands satisfy constraints based on the Hamming distance.

Deletion on trajectories has also been used as a tool to characterize when commutative languages are regular by the author and others [41]. We do not examine this application in this thesis.

Work on decidability of language equations involving shuffle on trajectories has been continued by the author and Salomaa [45]. In particular, it is shown that there exists a fixed linear context-free set of trajectories T such that the following problem is undecidable: “given regular languages R_1, R_2, R_3, does $R_1 \sqcup\!\sqcup_T R_2 = R_3$ hold?” Similar results are given for language equations of the form $R_1 \sqcup_T X = R_3$ where R_1, R_3 are regular and X is unknown.