Splicing on Routes versus Shuffle and Deletion along Trajectories

Technical Report 2003-471

Michael Domaratzki*
School of Computing, Queen's University,
Kingston, Ontario, Canada K7L 3N6.
email: domaratz@cs.queensu.ca

1 Introduction

Splicing on routes (see Section 2 for definitions) was introduced by Mateescu [7] to model generalizations of the crossover splicing operation (see, e.g., Kececioglu and Gusfield [6]). Crossover splicing simulates the manner in which two DNA strands may be spliced together at multiple locations to form several new strands, see Mateescu for a discussion [7].

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 [7]). Splicing on routes is also a generalization of the shuffle on trajectories operation, introduced by Mateescu *et al.* [8], which is itself a generalization of the shuffle operation. Shuffle on trajectories again restricts the way in which words may be shuffled together by imposing a set of restrictions in the form of a language T of trajectories.

Recently, the author has introduced the notion of deletion along trajectories [1]. This concept unifies several deletion operations, many of which

*Research supported by an NSERC PGS-B graduate scholarship.

have been studied extensively by Kari and others [2, 3, 4, 5]. Deletion along trajectories serves to provide an inverse (in the sense of Kari [3]) to shuffle on trajectories, which allows us to generalize certain language equations considered by Kari [3], see [1] for details.

In this note, we consider the simulation of splicing on a route by shuffle and deletion along trajectories. We show that there exist three fixed weak codings π_1, π_2, π_3 such that for all routes t, we can simulate the splicing on t of two languages L_1, L_2 by a fixed combination of the shuffle and deletion of the same languages L_1, L_2 along the routes $\pi_1(t), \pi_2(t), \pi_3(t)$. As a corollary, it is shown that every unary operation defined by splicing on routes can also be performed by a deletion along trajectories.

2 Definitions

For additional background in formal languages theory, please see Yu [9]. Let Σ be a finite set of symbols, called *letters*. Then Σ^* is the set of all finite sequences of letters from Σ, which are called *words*. The empty word ϵ is the empty sequence of letters. The *length* of a word $w = w_1 w_2 \cdots w_n \in \Sigma^*$, where $w_i \in \Sigma$, is n, and is denoted $|w|$. Note that ϵ is the unique word of length 0. A *language* L is any subset of Σ^*.

We denote by $\mathbb{N}$ the set of natural numbers: $\mathbb{N} = \{0, 1, 2, \dots\}$. Given alphabets Σ, Δ, a *morphism* is a function $h : \Sigma^* \to \Delta^*$ satisfying $h(xy) = h(x)h(y)$ for all $x, y \in \Sigma^*$. A morphism $h : \Sigma^* \to \Delta^*$ is said to be a *weak coding* if $h(a) \in \Delta + \epsilon$ for all $a \in \Sigma$.

We recall the definition of shuffle on trajectories, originally given by Mateescu *et al.* [8]. Shuffle on trajectories is defined by first defining the shuffle of two words x and y over an alphabet Σ on a trajectory t, which is simply a word in $\{0, 1\}^*$. We denote the shuffle of x and y along 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.

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

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

We now define *deletion along trajectories*, which models deletion operations controlled by a set of trajectories [1]. 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 $x \neq \epsilon$, $x \rightsquigarrow_{\epsilon} y = \epsilon$. Further, $\epsilon \rightsquigarrow_t y = \epsilon$ if $t = y = \epsilon$. Otherwise, $\epsilon \rightsquigarrow_t y = \emptyset$.

Example 2.1 *Let $x = abcabc$, $y = bac$ and $t = (id)^3$. Then we have that $x \rightsquigarrow_t y = acb$. If $t = i^2 d^3 i$ then $x \rightsquigarrow_t y = \emptyset$.*

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

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

Note that $\rightsquigarrow_T$ is neither an associative nor a commutative operation on languages, in general.

We define the concept of splicing on routes, and note the difference between deletion on trajectories from splicing on routes, which allows discarding symbols 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 symbol from the appropriate word, and $\overline{0}, \overline{1}$ means discard that symbol 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

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

Finally, we set $\epsilon \bowtie_t \epsilon = \epsilon$ if $t = \epsilon$ and $\emptyset$ otherwise. We extend $\bowtie_t$ to sets of trajectories and languages as expected.

3 Simulation of Splicing on Routes

We now demonstrate that splicing on routes can be simulated by a combination of shuffle on trajectories and deletion along trajectories.

Theorem 3.1 *There exists 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*

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

Proof. We first define the weak codings: let $\pi_1, \pi_2 : \{0, \overline{0}, 1, \overline{1}\}^* \to \{i, d\}^*$ and $\pi_3 : \{0, \overline{0}, 1, \overline{1}\} \to \{0, 1\}^*$ be given by

$$\begin{array}{cccccccccccc} \pi_1(0) & = & i; & \pi_1(\overline{0}) & = & d; & \pi_1(1) & = & \epsilon; & \pi_1(\overline{1}) & = & \epsilon. \\ \pi_2(0) & = & \epsilon; & \pi_2(\overline{0}) & = & \epsilon; & \pi_2(1) & = & i; & \pi_2(\overline{1}) & = & d; \\ \pi_3(0) & = & 0; & \pi_3(\overline{0}) & = & \epsilon; & \pi_3(1) & = & 1; & \pi_3(\overline{1}) & = & \epsilon. \end{array}$$

We first show the left-to-right inclusion. Let $z \in x \bowtie_t y$. The result is by induction on $|t|$. If $|t| = 0$, then $x = y = z = \epsilon$. Thus, we can easily verify that $z \in (\epsilon \rightsquigarrow_\epsilon \epsilon) \sqcup\!\sqcup_\epsilon (\epsilon \rightsquigarrow_\epsilon \epsilon)$.

Let $|t| > 0$. Then $t = ct$ for $c \in \{0, \overline{0}, 1, \overline{1}\}$. We prove only the case where $c = 0$ and $c = \overline{0}$. The other two cases are similar and are left to the reader.

(a) $c = 0$. Then $x = ax'$ and $z \in a(x' \bowtie_{t'} y)$ for some $x' \in \Sigma^*$. Thus, $z = az'$ for some $z' \in (x' \bowtie_{t'} y)$. By induction, $z' \in (x' \rightsquigarrow_{\pi_1(t')} \Sigma^*) \sqcup\!\sqcup_{\pi_3(t')} (y \rightsquigarrow_{\pi_2(t')} \Sigma^*)$. Let $u \in x' \rightsquigarrow_{\pi_1(t')} \Sigma^*$ and $v \in y \rightsquigarrow_{\pi_2(t')} \Sigma^*$ be such that $z' \in u \sqcup\!\sqcup_{\pi_3(t')} v$.

Note that $\pi_1(t) = i\pi_1(t')$. Thus by definition of $\rightsquigarrow_T$, $au \in x \rightsquigarrow_{\pi_1(t)} \Sigma^*$. Similarly, as $\pi_2(t) = \pi_2(t')$, $v \in y \rightsquigarrow_{\pi_2(t)} \Sigma^*$. Finally $\pi_3(t) = 0\pi_3(t')$. Thus, $au \sqcup\!\sqcup_{\pi_3(t)} v = a(u \sqcup\!\sqcup_{\pi_3(t')} v) \ni az' = z$. Thus, the result holds for $c = 0$.

(b) $c = \overline{0}$. Then $x = ax'$ and $z \in (x' \bowtie_{t'} y)$ for some $x' \in \Sigma^*$. Thus, by induction $z \in (x' \rightsquigarrow_{\pi_1(t')} \Sigma^*) \sqcup\!\sqcup_{\pi_3(t')} (y \rightsquigarrow_{\pi_2(t')} \Sigma^*)$. Let $u \in x' \rightsquigarrow_{\pi_1(t')} \Sigma^*$ and $v \in y \rightsquigarrow_{\pi_2(t')} \Sigma^*$ be such that $z \in u \sqcup\!\sqcup_{\pi_3(t')} v$.

Note in this case that $\pi_1(t) = d\pi_1(t')$. Thus, $u \in x \rightsquigarrow_{\pi_1(t)} \Sigma^*$. Similarly, as $\pi_2(t) = \pi_2(t')$, $v \in y \rightsquigarrow_{\pi_2(t)} \Sigma^*$. Finally, $\pi_3(t) = \pi_3(t')$. Thus, $u \sqcup\!\sqcup_{\pi_3(t)} v = u \sqcup\!\sqcup_{\pi_3(t')} v \ni z$. Thus, the result holds for $c = \overline{0}$.

We now prove the reverse inclusion. Let $z \in (x \rightsquigarrow_{\pi_1(t)} \Sigma^*) \sqcup\!\sqcup_{\pi_3(t)} (y \rightsquigarrow_{\pi_2(t)} \Sigma^*)$. We show the result by induction on t. For $|t| = 0$, $t = \epsilon$. Thus $\pi_1(t) = \pi_2(t) = \pi_3(t) = \epsilon$. By definition of $\sqcup\!\sqcup_t$, $L_1 \sqcup\!\sqcup_\epsilon L_2$ is non-empty iff $\epsilon \in L_1 \cap L_2$, which implies $z = \epsilon$. Thus, $\epsilon \in (x \rightsquigarrow_\epsilon \Sigma^*)$, and similarly for y in place of x. By definition of $\rightsquigarrow_t$, this implies that $x = y = \epsilon$. Thus, $z \in x \bowtie_t y$, by definition. The inclusion is proven for $|t| = 0$.

Let $|t| > 0$. Thus, there is some $c \in \{0, \overline{0}, 1, \overline{1}\}$, and $t' \in \{0, \overline{0}, 1, \overline{1}\}^*$ such that $t = ct'$. We distinguish between four cases, for each choice of c in $\{0, \overline{0}, 1, \overline{1}\}$, however, we only prove the cases $c = 0$ and $c = \overline{0}$. The other two cases are very similar, and are left to the reader.

(a) $c = 0$. Note that $\pi_1(t) = i\pi_1(t')$, $\pi_2(t) = \pi_2(t')$ and $\pi_3(t) = 0\pi_3(t')$. Let $u, v \in \Sigma^*$ be words such that $u \in x \rightsquigarrow_{\pi_1(t)} \Sigma^*$, $v \in y \rightsquigarrow_{\pi_2(t)} \Sigma^*$ and $z \in u \sqcup\!\sqcup_{\pi_3(t)} v$.

As $\pi_3(t) = 0\pi_3(t')$, we have, by definition of $\sqcup\!\sqcup_t$, that $u = au'$, $z = az'$ and $z' \in u' \sqcup\!\sqcup_{\pi_3(t')} v$ for some $a \in \Sigma$ and $u', z' \in \Sigma^*$. Now, as $au' \in x \rightsquigarrow_{i\pi_1(t')} \Sigma^*$, there exists $x' \in \Sigma^*$ such that $x = ax'$ and $u' \in x' \rightsquigarrow_{\pi_1(t')} \Sigma^*$. Also, note that $v \in y \rightsquigarrow_{\pi_2(t')} \Sigma^*$. Thus, combining these yields that

$$z' \in (x' \rightsquigarrow_{\pi_1(t')} \Sigma^*) \sqcup\!\sqcup_{\pi_3(t')} (y \rightsquigarrow_{\pi_2(t')} \Sigma^*).$$

By induction, $z' \in x' \bowtie_{t'} y$. Thus,

$$z = az \in a(x' \bowtie_{t'} y) = (ax' \bowtie_{0t'} y) = (x \bowtie_t y).$$

Thus, the inclusion is proven.

(b) $c = \overline{0}$. Then $\pi_1(t) = d\pi_1(t')$, $\pi_2(t) = \pi_2(t')$ and $\pi_3(t) = \pi_3(t')$. Let $u, v \in \Sigma^*$ be such that $u \in x \rightsquigarrow_{\pi_1(t)} \Sigma^*$, $v \in y \rightsquigarrow_{\pi_2(t)} \Sigma^*$ and $z \in u \sqcup\!\sqcup_{\pi_3(t)} v$.

As $u \in x \rightsquigarrow_{\pi_1(t)} \Sigma^*$, let $u_0 \in \Sigma^*$ be such that $u \in x \rightsquigarrow_{\pi_1(t)} u_0$. As $\pi_1(t) = d\pi_1(t')$, there are some $b \in \Sigma$, $x', u_0' \in \Sigma^*$ such that $x = bx'$, $u_0 = bu_0'$ and $u \in x' \rightsquigarrow_{\pi_1(t')} u_0'$. Thus, $u \in x' \rightsquigarrow_{\pi_1(t')} \Sigma^*$. Note that $v \in y \rightsquigarrow_{\pi_2(t')} \Sigma^*$. Thus, $z \in u \sqcup\!\sqcup_{\pi_3(t')} v \subseteq (x' \rightsquigarrow_{\pi_1(t')} \Sigma^*) \sqcup\!\sqcup_{\pi_3(t')} (y \rightsquigarrow_{\pi_2(t')} \Sigma^*)$. By induction, $z \in x' \bowtie_{t'} y$. Thus, we can see that $(bx' \bowtie_t y) = x' \bowtie_{t'} y \ni z$. This proves the inclusion.

The result is now proven. ■

Corollary 3.2 *There exist weak codings π_1, π_2, π_3 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 $T = \{\overline{0}01\overline{1}, 0\overline{01}1\}$, 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, which is easily established:

Corollary 3.3 *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 [7, pp. 4–5] for a discussion of unary operations defined by splicing on routes.

4 Conclusion

We have demonstrated how to simulate splicing on a route by a fixed combination of shuffle and deletion along trajectories.

References

[1] Domaratzki, M. Deletion along trajectories. Tech. Rep. 2003–464, School of Computing, Queen's University at Kingston, 2003.

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

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

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

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

[6] Kececioglu, J., and Gusfield, D. Reconstructing a history of recombinations from a set of sequences. In *Proc. 5th ACM-SIAM SODA 1994* (1994), pp. 471–480.

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

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

[9] Yu, S. Regular languages. In *Handbook of Formal Languages, Vol. I*, G. Rozenberg and A. Salomaa, Eds. Springer-Verlag, 1997, pp. 41–110.