Derivatives and Partial Derivatives for Regular Shuffle Expressions

Martin Sulzmann[a], Peter Thiemann[b]

[a]*Faculty of Computer Science and Business Information Systems, Karlsruhe University of Applied Sciences Moltkestraße 30, 76133 Karlsruhe, Germany*
[b]*Faculty of Engineering, University of Freiburg, Georges-Köhler-Allee 079, 79110 Freiburg, Germany*

Abstract

There is a rich variety of shuffling operations ranging from asynchronous interleaving to various forms of synchronizations. We introduce a general shuffling operation which subsumes earlier forms of shuffling. We further extend the notion of a Brzozowski derivative and a Antimirov partial derivative to the general shuffling operation and thus to many earlier forms of shuffling. This extension enables the direct construction of automata from regular expressions involving shuffles that appear in specifications of concurrent systems.

Keywords: automata and logic, shuffle expressions, derivatives, partial derivatives

1. Introduction

We consider an extension of regular expressions with a binary shuffle operation which bears similarity to shuffling two decks of cards. Like the extension with negation and complement, the language described by an expression extended with shuffling remains regular. That is, any expression making use of shuffling can be expressed in terms of basic operations such as choice, concatenation and Kleene star. However, the use of shuffling yields a much more succinct representation of problems that occur in modeling of concurrent systems [1, 2].

Email addresses: `martin.sulzmann@hs-karlsruhe.de` (Martin Sulzmann), `thiemann@acm.org` (Peter Thiemann)

Our goal is to extend Brzozowski's derivatives [3] and Antimirov's partial derivatives [4] to regular expressions with shuffle operations. Briefly, the derivative of a regular expression r w.r.t. some symbol x is a regular expression r' such that its language $\mathcal{L}(r')$ is $x\backslash\mathcal{L}(r)$, the left quotient of $\mathcal{L}(r)$ by the symbol x. Brzozowski gave a simple structural definition to compute derivatives and established two important properties: Each expression can be represented as a sum of its derivatives (*representation*) and the set of iterated derivatives is finite when considered modulo idempotence, commutativity and associativity of alternatives (*finiteness*). Thus, Brzozowski derivatives yield a direct construction of a deterministic finite automaton (DFA) from a regular expression.

Partial derivatives are a (non-deterministic) generalization of derivatives. They may be thought of as representing the states of a nondeterministic finite automaton (NFA). Similar to Brzozowski derivatives, partial derivatives also support representation and finiteness results.

In summary, thanks to representation and finiteness, derivatives and partial derivatives both support the elegant and efficient construction of automata-based word recognition algorithms and are also useful in the development of related algorithms for equality and containment among regular expressions [5, 6]. Hence, it is desirable to extend the notion of a derivative and a partial derivative to shuffle expressions.

Various flavors of shuffle operations exist in the literature that differ in the notion of synchronization that they support [7, 1, 8, 9, 10]. We focus on four common forms of shuffling which we refer to as (plain) shuffling, weak synchronized shuffling, strong synchronized shuffling, and synchronous composition. For each of these forms we develop the notions of a derivative and a partial derivative and establish representation and finiteness results.

Studying each form in separation is tedious and leads to repetition of results and proofs. In some cases, it is even difficult to provide a simple structural definition of a (partial) derivative. Therefore, our approach is as follows. We first introduce a general shuffle operation which subsumes the above four variants of shuffling. For this general shuffle operation we define derivatives and partial derivatives and establish the results indicated. Thus, derivatives and partial derivatives for each specific instance of shuffling fall out almost for free.

Contributions and outline. After introducing our notation in Section 2 and reviewing existing variants of shuffling in Section 3, we claim the following contributions:

- We introduce a general shuffle operation which subsumes previous forms of shuffling (Section 4).
- We extend the notion of Brzozowski's derivatives and Antimirov's partial derivatives to the general shuffle operation and prove representation and finiteness (Sections 5 and 6).
- Based on the general shuffle operation, we provide systematic methods to obtain (partial) derivatives for specific variants of shuffling. We further observe that for forms such as weak synchronized shuffling a direct, simple structural (partial) derivative definition is not possible. This justifies the need for a general shuffle operation (Section 7).

An earlier version of this work appeared in conference proceedings [11]. The present work includes further examples and proof details and adds the complete treatment of partial derivatives. Further related work is discussed in Section 8. We conclude in Section 9.

2. Preliminaries

Let Σ be a fixed alphabet (i.e., a finite set of symbols). We usually denote symbols by x, y and z. The set Σ^* denotes the set of finite words over Σ. We write Γ, Δ to denote subsets (sub-alphabets) of Σ. We write ϵ to denote the empty word and $v \cdot w$ to denote the concatenation of two words v and w.

We generally use $L, L_1, L_2 \subseteq \Sigma^*$ for languages over Σ and employ standard notation for operations on languages: $L_1 \cdot L_2 = \{v \cdot w \mid v \in L_1, w \in L_2\}$, $L^0 = \{\epsilon\}$, $L^{i+1} = L \cdot L^i$, and $L^* = \bigcup_{i=0}^{\infty} L^i$. We write $L_2 \backslash L_1$ to denote the left quotient of L_1 with L_2 where $L_2 \backslash L_1 = \{w \mid \exists v \in L_2 . v \cdot w \in L_1\}$. We often write just v in place of the singleton language $\{v\}$ in arguments for operations on languages, for example, $x \backslash L$ is a shorthand for $\{x\} \backslash L$.

We write $\alpha(w)$ to denote the set of symbols which appear in a word. The inductive definition is as follows: (1) $\alpha(\epsilon) = \emptyset$, (2) $\alpha(x \cdot w) = \alpha(w) \cup \{x\}$. The extension to languages is as follows: $\alpha(L) = \bigcup_{w \in L} \alpha(w)$.

We write $\Pi_\Gamma(w)$ to denote the projection of a word w onto a sub-alphabet Γ. The inductive definition is as follows:

$$\Pi_\Gamma(\epsilon) \;=\; \epsilon \qquad\qquad \Pi_\Gamma(x \cdot w) \;=\; \begin{cases} x \cdot \Pi_\Gamma(w) & x \in \Gamma \\ \Pi_\Gamma(w) & x \notin \Gamma \end{cases}$$

If $\odot : \Sigma^* \times \Sigma^* \to \wp(\Sigma^*)$ is an operation mapping two words to a set of words, then we lift this operation to an operator $\odot : \wp(\Sigma^*) \times \wp(\Sigma^*) \to \wp(\Sigma^*)$ on languages as follows:

$$L_1 \odot L_2 = \{u \in \Sigma^* \mid \exists v \in L_1, \exists w \in L_2, u \in v \odot w\}$$

We use this extension to lift shuffling operations from words to languages.

3. Shuffling Operations

Definition 1 (Shuffling). *The* shuffle operator $\|$ :: $\Sigma^* \times \Sigma^* \to \wp(\Sigma^*)$ *is defined inductively as follows.*

$$\begin{array}{lcl} \epsilon \| w & = & \{w\} \\ w \| \epsilon & = & \{w\} \\ x \cdot v \| y \cdot w & = & \{x \cdot u \mid u \in v \| y \cdot w\} \cup \{y \cdot u \mid u \in x \cdot v \| w\} \end{array}$$

For example, we find that $x \cdot y \| z = \{x \cdot y \cdot z, x \cdot z \cdot y, z \cdot x \cdot y\}$.

While the shuffle operator represents the asynchronous interleaving of two words $v, w \in \Sigma^*$, there are also shuffle operators that include some synchronization. The strongly synchronized shuffle of two words w.r.t. some sub-alphabet Γ imposes the restriction that the traces must synchronize on all symbols in Γ. All symbols not appearing in Γ are shuffled. In its definition, we write $(p) \Rightarrow X$ for: if p then X else $\emptyset$.

Definition 2 (Strongly Synchronized Shuffling). *The* strongly synchronized shuffling operator *w.r.t.* $\Gamma \subseteq \Sigma$*,* $|||_\Gamma$ :: $\Sigma^* \times \Sigma^* \to \wp(\Sigma^*)$*, is defined inductively as follows.*

$$\begin{array}{lcll} \epsilon |||_\Gamma w & = & (\Gamma \cap \alpha(w) = \emptyset) \Rightarrow \{w\} & \textit{(S1)} \\ w |||_\Gamma \epsilon & = & (\Gamma \cap \alpha(w) = \emptyset) \Rightarrow \{w\} & \textit{(S2)} \\ x \cdot v |||_\Gamma y \cdot w & = & (x = y \wedge x \in \Gamma) \Rightarrow \{x \cdot u \mid u \in v |||_\Gamma w\} & \textit{(S3)} \\ & & \cup (x \notin \Gamma) \Rightarrow \{x \cdot u \mid u \in v |||_\Gamma y \cdot w\} & \textit{(S4)} \\ & & \cup (y \notin \Gamma) \Rightarrow \{y \cdot u \mid u \in x \cdot v |||_\Gamma w\} & \textit{(S5)} \end{array}$$

The base cases (S1) and (S2) impose the condition (via $\Gamma \cap \alpha(w) = \emptyset$) that none of the symbols in w shall be synchronized. If the condition is violated we obtain the empty set. For example, $\epsilon |||_{\{x\}} y \cdot z = \{y \cdot z\}$, but $\epsilon |||_{\{x\}} x \cdot y \cdot z = \emptyset$.

In the inductive step, a symbol in Γ appearing on both sides forces synchronization (S3). If the leading symbol on either side does not appear in Γ, then it can be shuffled arbitrarily. See cases (S4) and (S5). These three cases ensure progress until one side is reduced to the empty string. For example, we find that $x \cdot y|||_{\{x\}} x \cdot z = \{x \cdot y \cdot z, x \cdot z \cdot y\}$. On the other hand, $x \cdot y|||_{\{x,y\}} x \cdot z = \emptyset$.

Shuffling and strongly synchronized shuffling correspond to the *arbitrary* synchronized shuffling and strongly synchronized shuffling operations by Beek and coworkers [8]. Our inductive definitions simplify the proofs.

Beek and coworkers [8] also introduce a *weak* synchronized shuffling operation. In its definition, we write $L \cdot x$ as a shorthand for $\{w \cdot x \mid w \in L\}$ and $x \cdot L$ as a shorthand for $\{x \cdot w \mid w \in L\}$

Definition 3 (Weak Synchronized Shuffling). *Let $v, w \subseteq \Sigma^*$ and $\Gamma \subseteq \Sigma$. Then, we define*

$$\begin{array}{ll} v| \sim |_{\Gamma} w = \{u \mid & \exists n \geq 0, x_i \in \Gamma, v_i \in \Gamma, w_i \in \Gamma. \\ & v = v_1 \cdot x_1 ... x_n \cdot v_{n+1} \wedge w = w_1 \cdot x_1 ... x_n \cdot w_{n+1} \wedge \\ & u \in (v_1 \| w_1) \cdot x_1 ... x_n \cdot (v_{n+1} \| w_{n+1}) \\ & \alpha(v_i) \cap \alpha(w_i) \cap \Gamma = \emptyset\} \end{array}$$

The weak synchronized shuffle of two words v and w synchronizes only on those symbols in Γ that occur in both v and w. For example, $x \cdot y| \sim |_{\{x,y\}} x \cdot z = \{x \cdot y \cdot z, x \cdot z \cdot y\}$ because $y \notin \alpha(x \cdot z)$ whereas $x \cdot y|||_{\{x,y\}} x \cdot z = \emptyset$.

Another variant of synchronous shuffling is called *synchronous composition* [7, 1]. The difference to strongly synchronized shuffling is that synchronization occurs on symbols common to both operands. Thus, synchronous composition can be defined by projecting onto the symbols of the operands.

Definition 4 (Synchronous Composition). *The synchronous composition operator $|||$ is defined by:*

$$L_1|||L_2 = \{w \in (\alpha(L_1) \cup \alpha(L_2))^* \mid \Pi_{\alpha(L_1)}(w) \in L_1 \wedge \Pi_{\alpha(L_2)}(w) \in L_2\}$$

For example, $x \cdot y|||x \cdot z$ equals $\{x \cdot y \cdot z, x \cdot z \cdot y\}$.

It turns out that the strong synchronized shuffling operation $|||_{\Gamma}$ subsumes synchronous composition due to the customizable set Γ. In Section 4, we show an even stronger result: All of the shuffling variants we have seen can be expressed in terms of a general synchronous shuffling operation.

4. General Synchronous Shuffling

The general synchronous shuffling operation is parameterized by a set of synchronizing symbols, Γ, and two additional sets P_1 and P_2 that keep track of 'out of sync' symbols from Γ.

Definition 5 (General Synchronous Shuffling). *Let $\Gamma, P_1, P_2 \subseteq \Sigma$. The general synchronous shuffling operator ${}^{P_1}||_{\Gamma}^{P_2} :: \Sigma^* \times \Sigma^* \to \wp(\Sigma^*)$ is defined inductively as follows.*

$$
\begin{array}{lcll}
\epsilon^{P_1}||_{\Gamma}^{P_2} w & = & ((\alpha(w) \cap \Gamma = \emptyset) \vee (P_1 \cap (P_2 \cup \alpha(w)) = \emptyset)) \Rightarrow \{w\} & \textit{(G1)} \\
w^{P_1}||_{\Gamma}^{P_2} \epsilon & = & ((\alpha(w) \cap \Gamma = \emptyset) \vee (P_1 \cup \alpha(w)) \cap P_2 = \emptyset)) \Rightarrow \{w\} & \textit{(G2)} \\
x \cdot v^{P_1}||_{\Gamma}^{P_2} y \cdot w & = & (x \notin \Gamma) \Rightarrow \{x \cdot u \mid u \in v^{P_1}||_{\Gamma}^{P_2} y \cdot w\} & \textit{(G3)} \\
 & & \cup (y \notin \Gamma) \Rightarrow \{y \cdot u \mid u \in x \cdot v^{P_1}||_{\Gamma}^{P_2} w\} & \textit{(G4)} \\
 & & \cup (x = y \wedge x \in \Gamma \wedge P_1 \cap P_2 = \emptyset) \Rightarrow & \\
 & & \quad \{x \cdot u \mid u \in v^{\emptyset}||_{\Gamma}^{\emptyset} w\} & \textit{(G5)} \\
 & & \cup (x = y \wedge x \in \Gamma \wedge P_1 \cap P_2 \neq \emptyset) \Rightarrow & \\
 & & \quad \{x \cdot u \mid u \in v^{P_1}||_{\Gamma}^{P_2} w\} & \textit{(G6)} \\
 & & \cup (x \in \Gamma \wedge (P_1 \cup x) \cap P_2 = \emptyset) \Rightarrow & \\
 & & \quad \{x \cdot u \mid u \in v^{P_1 \cup x}||_{\Gamma}^{P_2} y \cdot w\} & \textit{(G7)} \\
 & & \cup (y \in \Gamma \wedge P_1 \cap (P_2 \cup y) = \emptyset) \Rightarrow & \\
 & & \quad \{y \cdot u \mid u \in x \cdot v^{P_1}||_{\Gamma}^{P_2 \cup y} w\} & \textit{(G8)}
\end{array}
$$

The definition of general synchronous shuffling is significantly more involved compared to the earlier definitions. Cases (G1-8) are necessary to encode the earlier shuffle operations from Section 3. The exact purpose of the individual cases will become clear shortly.

In our first result we observe that ${}^{\Sigma}||_{\Gamma}^{\Sigma}$ exactly corresponds to strongly synchronized shuffling ($|||_{\Gamma}$).

Theorem 1. *For any $L_1, L_2 \subseteq \Sigma^*$ and $\Gamma \subseteq \Sigma$: $L_1|||_{\Gamma}L_2 = L_1{}^{\Sigma}||_{\Gamma}^{\Sigma}L_2$.*

Proof. We choose a 'maximal' assignment for P_1 and P_2 by setting both to Σ. [1] Then, definition of ${}^{P_1}||_{\Gamma}^{P_2}$ reduces to $|||_{\Gamma}$.

In more detail, we observe that property $P_1 = \Sigma \wedge P_2 = \Sigma$ (SP) is an invariant. For cases (G3-4) and (G6) the invariant property clearly holds.

[1] Alternatively, we could set P_1 and P_2 to $\emptyset$ which yields the same result.

For cases (G5), (G7-8) the preconditions are violated. Hence, for ${}^{\Sigma}||_{\Gamma}^{\Sigma}$ only cases (G1-4) and (G6) will ever apply.

Under the given assumptions, we can relate the cases in Definition 2 and Definition 5 as follows. Cases (G1-2) correspond to cases (S1-2). Cases (G3-4) correspond to cases (S4-5). Case (G6) corresponds to case (S3). Due to the invariant property (SP) cases (G5) and (G7-8) never apply.

Hence, ${}^{\Sigma}||_{\Gamma}^{\Sigma}$ and $|||_{\Gamma}$ yield the same result. $\square$

Via similar reasoning we can show that for $\Gamma = \emptyset \wedge P_1 = \emptyset \wedge P_2 = \emptyset$ general synchronized shuffling boils down to (arbitrary) shuffling.

Theorem 2. *For any $L_1, L_2 \subseteq \Sigma^*$:* $L_1 || L_2 = L_1{}^{\emptyset}||_{\emptyset}^{\emptyset} L_2$.

An immediate consequence of Theorem 1 (set Σ and Γ to $\emptyset$) and Theorem 2 is that shuffling can also be expressed in terms of strong synchronized shuffling.

Corollary 1. *For any $L_1, L_2 \subseteq \Sigma^*$:* $L_1 || L_2 = L_1 |||_{\emptyset} L_2$.

Our next result establishes a connection to weak synchronized shuffling [8].

Theorem 3. *For any $L_1, L_2 \subseteq \Sigma^*, \Gamma \subseteq \Sigma$:* $L_1 | \sim |_{\Gamma} L_2 = L_1{}^{\emptyset}||_{\Gamma}^{\emptyset} L_2$.

Proof. Property $P_1, P_2 \subseteq \Gamma \wedge P_1 \cap P_2 = \emptyset$ (WP) is an invariant of ${}^{P_1}||_{\Gamma}^{P_2}$. For cases (G3-4) the invariant property clearly holds. More interesting are cases (G7-8) where P_1, resp., P_2 is extended. Under the precondition property (WP) remains invariant. Case (G6) never applies. Case (G5) clearly maintains the invariant.

Recall that for strong synchronized shuffling the roles of (G5) and (G6) are switched. See proof of Theorem 1. This shows that while cases (G5) and (G6) look rather similar both are indeed necessary.

As we can see, under the invariant condition (WP), the purpose of P_1, P_2 is to keep track of "out of sync" symbols from Γ. See cases (G7-8). Case (G5) synchronizes on $x \in \Gamma$. Hence, P_1 and P_2 are (re)set to $\emptyset$.

Thus, we can show (via some inductive argument) that ${}^{\emptyset}||_{\Gamma}^{\emptyset}$ under the (WP) invariant corresponds to weak synchronized shuffling as defined in Definition 3. $\square$

It remains to show that synchronous composition is subsumed (i.e. can be expressed) by general synchronized shuffling. First, we verify that synchronous composition is subsumed by strongly synchronous shuffling via the following identity $L_1|||L_2 = L_1|||_{\alpha(L_1)\cap\alpha(L_2)}L_2$.

To establish the direction $L_1|||L_2 \supseteq L_1|||_{\alpha(L_1)\cap\alpha(L_2)}L_2$ we verify that the projection of a strongly synchronizable word w.r.t. $\alpha(L_1)\cap\alpha(L_2)$ yields words in the respective languages L_1 and L_2 as stated by the following lemma.

Lemma 1. *Let $u, v, w \in \Sigma^*$, $L_1, L_2 \subseteq \Sigma^*$, $\Gamma \subseteq \Sigma$ such that $u \in v|||_\Gamma w$ where $v \in L_1$, $w \in L_2$ and $\Gamma \subseteq \alpha(L_1)\cap\alpha(L_2)$. Then, we find that (1) $\Pi_{\alpha(L_1)}(u) = v$ and (2) $\Pi_{\alpha(L_2)}(u) = w$.*

Proof. The proof proceeds by induction over u.

Case $u = \epsilon$**:** By definition of $|||_\Gamma$ we must have $\epsilon \in L_1, L_2$. Thus, (1) and (2) follow immediately.

Case $u = x \cdot u'$**:** We distinguish among the following subcases.

Subcase $x \in \Gamma$**:** From the assumptions and definition of $|||_\Gamma$, $v = x \cdot v'$, $w = x \cdot w'$, and $xu' \in x(v'|||_\Gamma w')$, where $x \cdot v' \in L_1$ and $x \cdot w' \in L_2$.

Hence, $u' \in v'|||_\Gamma w'$ where $v' \in x\backslash L_1$ and $w' \in x\backslash L_2$.

By induction we find that $\Pi_{\alpha(x\backslash L_1)}(u') = v'$ and $\Pi_{\alpha(x\backslash L_2)}(u') = w'$.

Thus, $\Pi_{\alpha(L_1)}(x \cdot u') = x \cdot \Pi_{\alpha(L_1)}(u')$ by definition. Clearly, $\Pi_{\alpha(L_1)}(u') = \Pi_{\alpha(x\backslash L_1)}(u')$. Hence, $\Pi_{\alpha(L_1)}(x \cdot u') = x \cdot v'$ and thus we have established (1). Via a similar reasoning we can establish (2).

Subcase $x \notin \Gamma$**:** There are two further subcases to consider. Either (a) $x \in \alpha(L_1)$ and $x \notin \alpha(L_2)$, or (b) $x \notin \alpha(L_1)$ and $x \in \alpha(L_2)$. We start with subcase (a).

From the assumption we conclude that $v = x \cdot v'$ where $x \cdot u' \in x \cdot v'|||_\Gamma w$.

Hence, $u' \in v'|||_\Gamma w$ where $v' \in x\backslash L_1$.

By induction we find that $\Pi_{\alpha(x\backslash L_1)}(u') = v'$ and $\Pi_{\alpha(L_2)}(u') = w$.

Because $x \notin \alpha(L_2)$, we have $\Pi_{\alpha(L_1)}(x \cdot u') = x \cdot v'$ and $\Pi_{\alpha(L_2)}(x \cdot u') = w$.

Hence, we have established (1) and (2).

Subcase (b) can be verified similarly. $\square$

Direction $L_1|||L_2 \supseteq L_1|||_{\alpha(L_1)\cap\alpha(L_2)}L_2$ can be verified similarly. We show that if the projection of a word onto $\alpha(L_1)$ and $\alpha(L_2)$ yields words in the respective language, then the word must be strongly synchronizable w.r.t. $\alpha(L_1)\cap\alpha(L_2)$.

Lemma 2. *Let $u, v, w \in \Sigma^*$, $L_1, L_2 \subseteq \Sigma^*$ such that $w \in (\alpha(L_1) \cup \alpha(L_2))^*$, $\Pi_{\alpha(L_1)}(w) = u \in L_1$ and $\Pi_{\alpha(L_1)}(w) = v \in L_2$. Then, we find that $w \in u|||_{\alpha(L_1)\cap\alpha(L_2)}v$.*

Proof. The proof proceeds by induction over w.

Case $w = \epsilon$: Straightforward.

Case $w = x \cdot w'$: From $x \cdot w' \in (\alpha(L_1) \cup \alpha(L_2))^*$ we conclude that $w' \in (\alpha(x\backslash L_1) \cup \alpha(x\backslash L_2))^*$.

We continue by distinguishing among the following subcases.

Subcase $x \in \alpha(L_1) \cap \alpha(L_2)$:

$$\begin{array}{rcll} \Pi_{\alpha(L_1)}(x \cdot w') & = & x \cdot \Pi_{\alpha(L_1)}(w') & \text{by definition} \\ & = & x \cdot \Pi_{\alpha(x\backslash L_1)}(w') & \text{by } w' \in (\alpha(x\backslash L_1) \cup \alpha(x\backslash L_2))^* \\ & = & x \cdot u' & \text{where } u' \in x\backslash L_1 \end{array}$$

Similarly, $\Pi_{\alpha(x\backslash L_2)}(w') = v'$ where $v' \in x\backslash L_2$.

By induction $w' \in u'|||_{\alpha(x\backslash L_1)\cap\alpha(x\backslash L_2)}v'$.

It follows that $w' \in u'|||_{\alpha(L_1)\cap\alpha(L_2)}v'$. If $x \in \alpha(x\backslash L_1) \cap \alpha(x\backslash L_2)$ then there is no harm adding x. Otherwise, there is no harm either as $u' \in x\backslash L_1$ and $v' \in x\backslash L_2$.

By definition of $|||_{\alpha(L_1)\cap\alpha(L_2)}$ we find that $x \cdot w' \in x \cdot u'|||_{\alpha(L_1)\cap\alpha(L_2)}x \cdot v'$ and we are done.

Subcase $x \in \alpha(L_1)$, $x \notin \alpha(L_2)$: Using the same reasoning as in the first subcase we find that $\Pi_{\alpha(x\backslash L_1)}(w') = u' \in x\backslash L_1$.

Because of $x \notin \alpha(L_2)$, we find that $\Pi_{\alpha(L_2)}(x \cdot w') = \Pi_{\alpha(L_2)}(w') = v \in L_2$ and $w' \in (\alpha(x\backslash L_1) \cup \alpha(L_2))^*$.

By induction $w' \in u'|||_{\alpha(x\backslash L_1)\cup\alpha(L_2)}v$. It follows that $w' \in u'|||_{\alpha(L_1)\cup\alpha(L_2)}v$ because $x \notin \alpha(L_2)$.

By definition of $|||_{\alpha(L_1)\cap\alpha(L_2)}$ we find that $x \cdot w' \in x \cdot u'|||_{\alpha(L_1)\cap\alpha(L_2)}v$ and we are done.

Subcase $x \in \alpha(L_2)$, $x \notin \alpha(L_1)$: Similar to the above.

Subcase $x \notin \alpha(L_1)$, $x \notin \alpha(L_2)$: Does not apply. □

Theorem 4. *For any $L_1, L_2 \subseteq \Sigma^*$ we find that $L_1|||L_2 = L_1|||_{\alpha(L_1)\cap\alpha(L_2)}L_2$.*

Proof. Result follows immediately from Lemmas 1 and 2. □

An immediate consequence of Theorem 1 and Theorem 4 is the following result. Synchronous composition is subsumed by general synchronous shuffling.

Corollary 2. *For any $L_1, L_2 \subseteq \Sigma^*$ we find that $L_1|||L_2 = L_1{}^{\Sigma}||^{\Sigma}_{\alpha(L_1)\cap\alpha(L_2)} L_2$.*

In summary, all four shuffling variants from Section 3 can be expressed in terms of the general shuffle operation via the following identities. For each identity, we highlight the relevant cases from Definition 5 to establish the identity.

- $L_1 \| L_2 = L_1{}^{\Sigma}||^{\Sigma}_{\emptyset} L_2$ Cases (G1-4)
- $L_1|||L_2 = L_1{}^{\Sigma}||^{\Sigma}_{\alpha(L_1)\cap\alpha(L_2)} L_2$ Cases (G1-2), (G3-4), (G6)
- $L_1|||_{\Gamma} L_2 = L_1{}^{\Sigma}||^{\Sigma}_{\Gamma} L_2$ Cases (G1-2), (G3-4), (G6)
- $L_1| \sim |_{\Gamma} L_2 = L_1{}^{\emptyset}||^{\emptyset}_{\Gamma} L_2$ Cases (G1-2), (G3-4), (G5), (G7-8)

5. Derivatives for General Synchronous Shuffling

Brzozowski derivatives [3] are a useful tool to translate regular expressions into finite automata and to obtain decision procedures for equivalence and containment for regular expressions. We show that Brzozowski's results and their applications can be extended to regular expressions that contain shuffle operators. Based on the results of the previous section, we restrict our attention to regular expressions extended with the general synchronous shuffle operator.

Definition 6. *The set R_Σ of* regular shuffle expressions *is defined inductively by $\phi \in R_\Sigma$, $\epsilon \in R_\Sigma$, $\Sigma \subseteq R_\Sigma$, and for all $r, s \in R_\Sigma$ and $\Gamma, P_1, P_2 \subseteq \Sigma$ we have that $r + s$, $r \cdot s$, r^*, $r^{P_1}||^{P_2}_{\Gamma} s \in R_\Sigma$.*

Definition 7. *The language $\mathcal{L}(\_) : R_\Sigma \to \Sigma^*$ denoted by a regular shuffle expression is defined inductively as follows. $\mathcal{L}(\phi) = \emptyset$. $\mathcal{L}(\epsilon) = \{\epsilon\}$. $\mathcal{L}(x) = \{x\}$. $\mathcal{L}(r + s) = \mathcal{L}(r) \cup \mathcal{L}(s)$. $\mathcal{L}(r \cdot s) = \mathcal{L}(r) \cdot \mathcal{L}(s)$. $\mathcal{L}(r^*) = \mathcal{L}(r)^*$. $\mathcal{L}(r^{P_1}||^{P_2}_{\Gamma} s) = \mathcal{L}(r)^{P_1}||^{P_2}_{\Gamma}\mathcal{L}(s)$.*

An expression r is *nullable* if $\epsilon \in \mathcal{L}(r)$. The following function $n(\_)$ detects nullable regular shuffle expressions.

Definition 8. *We define $n(\_) : R_\Sigma \to Bool$ inductively as follows. $n(\phi) = false$. $n(\epsilon) = true$. $n(x) = false$. $n(r+s) = n(r) \vee n(s)$. $n(r \cdot s) = n(r) \wedge n(s)$. $n(r^*) = true$. $n(r^{P_1}||^{P_2}_{\Gamma} s) = n(r) \wedge n(s)$.*

Lemma 3. *For all $r \in R_\Sigma$ we have that $\epsilon \in \mathcal{L}(r)$ iff $n(r) = true$.*

PROOF. The proof proceeds by induction over r. We only consider the shuffle case as the remaining cases are standard. $\epsilon \in \mathcal{L}(r^{P_1}||_\Gamma^{P_2} s)$ iff (by definition) $\epsilon \in v^{P_1}||_\Gamma^{P_2} w$ for some $v \in \mathcal{L}(r)$ and $w \in \mathcal{L}(s)$. By definition of ${}^{P_1}||_\Gamma^{P_2}$ it must be that $v = \epsilon$ and $w = \epsilon$. By induction, this is equivalent to $n(r) \wedge n(s)$. $\square$

The derivative of an expression r w.r.t. some symbol x, written $d_x(r)$ yields a new expression for the left quotient of $\mathcal{L}(r)$ by x. In its definition, we write $(p) \Rightarrow r$ for: if p then r else ϕ.

Definition 9. *The derivative of $r \in R_\Sigma$ w.r.t. $x \in \Sigma$, written $d_x(r)$, is computed inductively as follows.*

$$
\begin{array}{lcll}
d_x(\phi) & = & \phi & \textit{(D1)} \\
d_x(\epsilon) & = & \phi & \textit{(D2)} \\
d_y(x) & = & (x = y) \Rightarrow \epsilon & \textit{(D3)} \\
d_x(r + s) & = & d_x(r) + d_x(s) & \textit{(D4)} \\
d_x(r \cdot s) & = & (d_x(r)) \cdot s + (n(r)) \Rightarrow d_x(s) & \textit{(D5)} \\
d_x(r^*) & = & (d_x(r)) \cdot r^* & \textit{(D6)} \\
d_x(r^{P_1}||_\Gamma^{P_2} s) & = & (x \not\in \Gamma) \Rightarrow ((d_x(r)^{P_1}||_\Gamma^{P_2} s) + (r^{P_1}||_\Gamma^{P_2} d_x(s))) & \textit{(D7)} \\
 & & +(x \in \Gamma \wedge P_1 \cap P_2 = \emptyset) \Rightarrow (d_x(r)^{\emptyset}||_\Gamma^{\emptyset} d_x(s)) & \textit{(D8)} \\
 & & +(x \in \Gamma \wedge P_1 \cap P_2 \neq \emptyset) \Rightarrow (d_x(r)^{P_1}||_\Gamma^{P_2} d_x(s)) & \textit{(D9)} \\
 & & +(x \in \Gamma \wedge (P_1 \cup x) \cap P_2 = \emptyset) \Rightarrow (d_x(r)^{P_1 \cup x}||_\Gamma^{P_2} s) & \textit{(D10)} \\
 & & +(x \in \Gamma \wedge P_1 \cap (P_2 \cup x) = \emptyset) \Rightarrow (r^{P_1}||_\Gamma^{P_2 \cup x} d_x(s)) & \textit{(D11)}
\end{array}
$$

The definition extends to words and sets of words. We define $d_\epsilon(r) = r$ and $d_{xw}(r) = d_w(d_x(r))$. For $L \subseteq \Sigma^*$ we define $d_L(r) = \{d_w(r) \mid w \in L\}$.

We refer to the special case $d_{\Sigma^*}(r)$ as the set of descendants *of r. A descendant is either the expression itself, a derivative of the expression, or the derivative of a descendant.*

The first six cases (D1-6) correspond to Brzozowski's original definition [3]. As a minor difference we may concatenate with ϕ when building the derivative of a concatenated expression whose first component is not nullable. The new subcases (D7-11) for the general shuffle closely correspond to the subcases of Definition 5. For example, compare (D7) and (G3-4), (D8) and (G5), (D9) and (G6), (D10) and (G7), and lastly (D11) and (G8).

An easy induction shows that the derivative of a shuffle expression is again a shuffle expression.

Theorem 5 (Closure). *For $r \in R_\Sigma$ and $x \in \Sigma$ we have that $d_x(r) \in R_\Sigma$.*

Brzozowski proved that the derivative of a regular expression denotes a left quotient. This result extends to shuffle expressions. In our proof we will make use of the following helper statements which correspond to the cases (D7-D11) in the definition of derivatives. See Definition 9.

Lemma 4 (Left Quotient Case D7a). *Let $r', r, s \in R_\Sigma$, $\Gamma, P_1, P_2 \subseteq \Sigma$ and $x \notin \Gamma$ such that $\mathcal{L}(r') \subseteq x \backslash \mathcal{L}(r)$. Then, $\mathcal{L}(r'^{P_1}||^{P_2}_\Gamma s) \subseteq x \backslash \mathcal{L}(r^{P_1}||^{P_2}_\Gamma s)$.*

Proof. Suppose $w \in \mathcal{L}(r'^{P_1}||^{P_2}_\Gamma s)$. By definition $w \in u^{P_1}||^{P_2}_\Gamma v$ for some $u \in \mathcal{L}(r')$ and $v \in \mathcal{L}(s)$. If $x \notin \Gamma$ we find $x \cdot w \in x \cdot u^{P_1}||^{P_2}_\Gamma v$. See case (G3) in Definition 5. If $u \in \mathcal{L}(r')$ then by assumption $x \cdot u \in \mathcal{L}(r)$. Hence, $x \ \cdot w \in \mathcal{L}(r^{P_1}||^{P_2}_\Gamma s)$ and we are done. □

Lemma 5 (Left Quotient Case D7b). *Let $s', r, s \in R_\Sigma$, $\Gamma, P_1, P_2 \subseteq \Sigma$ and $x \notin \Gamma$ such that $\mathcal{L}(s') \subseteq x \backslash \mathcal{L}(s)$. Then, $\mathcal{L}(r^{P_1}||^{P_2}_\Gamma s') \subseteq x \backslash \mathcal{L}(r^{P_1}||^{P_2}_\Gamma s)$.*

Proof. Similar to the proof of Lemma 4. □

.

Lemma 6 (Left Quotient Case D8). *Let $r', s', r, s \in R_\Sigma$, $\Gamma, P_1, P_2 \subseteq \Sigma$ and $x \in \Gamma$ such that $P_1 \cap P_2 = \emptyset$, $\mathcal{L}(r') \subseteq x \backslash \mathcal{L}(r)$ and $\mathcal{L}(s') \subseteq x \backslash \mathcal{L}(s)$. Then, $\mathcal{L}(r'^{P_1}||^{P_2}_\Gamma s') \subseteq x \backslash \mathcal{L}(r^{P_1}||^{P_2}_\Gamma s)$.*

Proof. Suppose $w \in \mathcal{L}(r'^{P_1}||^{P_2}_\Gamma s')$. By definition $w \in u^{P_1}||^{P_2}_\Gamma v$ for some $u \in \mathcal{L}(r')$ and $v \in \mathcal{L}(s')$. If $P_1 \cap P_2 = \emptyset$ we find that $x \cdot w \in x \cdot u^{\emptyset}||^{\emptyset}_\Gamma x \cdot v$. See case (G5) in Definition 5. If $u \in \mathcal{L}(r')$ and $v \in \mathcal{L}(s')$ then by assumption $x \cdot u \in \mathcal{L}(r)$ and $x \cdot v \in \mathcal{L}(s)$. Hence, $x \cdot w \in \mathcal{L}(r^{\emptyset}||^{\emptyset}_\Gamma s)$. By inspection of the definition of general synchronous shuffling we find that $\mathcal{L}(r^{\emptyset}||^{\emptyset}_\Gamma s) = \mathcal{L}(r^{P_1}||^{P_2}_\Gamma s)$ for $P_1 \cap P_2 = \emptyset$ and thus we are done. □

Lemma 7 (Left Quotient Case D9). *Let $r', s', r, s \in R_\Sigma$, $\Gamma, P_1, P_2 \subseteq \Sigma$ and $x \in \Gamma$ such that $P_1 \cap P_2 \neq \emptyset$, $\mathcal{L}(r') \subseteq x \backslash \mathcal{L}(r)$ and $\mathcal{L}(s') \subseteq x \backslash \mathcal{L}(s)$. Then, $\mathcal{L}(r'^{P_1}||^{P_2}_\Gamma s') \subseteq x \backslash \mathcal{L}(r^{P_1}||^{P_2}_\Gamma s)$.*

Proof. Similar to the proof of Lemma 6. □

Lemma 8 (Left Quotient Case D10). *Let $r', r, s \in R_\Sigma$, $\Gamma, P_1, P_2 \subseteq \Sigma$ and $x \in \Gamma$ such that $(P_1 \cup x) \cap P_2 = \emptyset$ and $\mathcal{L}(r') \subseteq x \backslash \mathcal{L}(r)$. Then, $\mathcal{L}({r'}^{P_1 \cup x} ||_\Gamma^{P_2} s) \subseteq x \backslash \mathcal{L}(r^{P_1} ||_\Gamma^{P_2} s)$.*

PROOF. Suppose $w \in \mathcal{L}({r'}^{P_1 \cup x} ||_\Gamma^{P_2} s)$. By definition $w \in u^{P_1 \cup x} ||_\Gamma^{P_2} v$ for some $u \in \mathcal{L}(r')$ and $v \in \mathcal{L}(s)$. If $x \in \Gamma, (P_1 \cup x) \cap P_2 = \emptyset$ we find $x \cdot w \in x \cdot u^{P_1} ||_\Gamma^{P_2} v$. See case (G7) in Definition 5. If $u \in \mathcal{L}(r')$ then by assumption $x \cdot u \in \mathcal{L}(r)$. Hence, $x \ \cdot w \in \mathcal{L}(r^{P_1} ||_\Gamma^{P_2} s)$ and we are done. □

Lemma 9 (Left Quotient Case D11). *Let $s', r, s \in R_\Sigma$, $\Gamma, P_1, P_2 \subseteq \Sigma$ and $x \in \Gamma$ such that $P_1 \cap (P_2 \cup x) = \emptyset$ and $\mathcal{L}(s') \subseteq x \backslash \mathcal{L}(s)$. Then, $\mathcal{L}(r^{P_1} ||_\Gamma^{P_2 \cup x} s') \subseteq x \backslash \mathcal{L}(r^{P_1} ||_\Gamma^{P_2} s)$.*

PROOF. Similar to the proof of Lemma 8. □

Theorem 6 (Left Quotients). *For any $r \in R_\Sigma$ and $x \in \Sigma$ we have that $\mathcal{L}(d_x(r)) = x \backslash \mathcal{L}(r)$.*

PROOF. It suffices to consider the new case of general synchronous shuffling. We consider the direction $\mathcal{L}(d_x(r^{P_1} ||_\Gamma^{P_2} s)) \subseteq x \backslash \mathcal{L}(r^{P_1} ||_\Gamma^{P_2} s) x = \{w \mid x \cdot w \in \mathcal{L}(r^{P_1} ||_\Gamma^{P_2} s)\}$.

Suppose $u \in \mathcal{L}(d_x(r^{P_1} ||_\Gamma^{P_2} s))$. We will verify that $x \cdot u \in \mathcal{L}(r^{P_1} ||_\Gamma^{P_2} s)$. We proceed by distinguishing among the following cases.

Case $x \not\in \Gamma$: By definition of the derivative operation, we find that either (D7a) $u \in \mathcal{L}(d_x(r)^{P_1} ||_\Gamma^{P_2} s)$ or (D7b) $u \in \mathcal{L}(r^{P_1} ||_\Gamma^{P_2} d_x(s))$. By induction $\mathcal{L}(d_x(r)) \subseteq x \backslash r$ and $\mathcal{L}(d_x(s)) \subseteq x \backslash s$. Via Lemmas 4 and 5 we find that $x \cdot u \in \mathcal{L}(r^{P_1} ||_\Gamma^{P_2} s)$ and we are done.

Case $x \in \Gamma$: By definition of the derivative operation (D8) $u \in \mathcal{L}(d_x(r)^{\emptyset} ||_\Gamma^{\emptyset} d_x(s))$ where $P_1 \cap P_2 = \emptyset$, or (D9) $u \in \mathcal{L}(d_x(r)^{P_1} ||_\Gamma^{P_2} d_x(s))$ where $P_1 \cap P_2 \neq \emptyset$, or (D10) $u \in \mathcal{L}(d_x(r)^{P_1 \cup x} ||_\Gamma^{P_2} s)$ where $(P_1 \cup x) \cap P_2 = \emptyset$, or (D11) $u \in \mathcal{L}(r^{P_1} ||_\Gamma^{P_2 \cup x} d_x(s))$ where $P_1 \cap (P_2 \cup x) = \emptyset$. Via similar reasoning as above and application of the appropriate Lemmas 6, 7, 8 and 9 we find that $x \cdot u \in \mathcal{L}(r^{P_1} ||_\Gamma^{P_2} s)$ which concludes the proof.

The other direction $\mathcal{L}(d_x(r^{P_1} ||_\Gamma^{P_2} s)) \supseteq \{x \cdot w \mid w \in \mathcal{L}(r^{P_1} ||_\Gamma^{P_2} s)\}$ follows via similar reasoning. □

Based on the above result, we obtain a simple algorithm for membership testing. Given a word w and expression r, we exhaustively apply the derivative operation and on the final expression we apply the nullable test.

Lemma 10. *For all $r \in R_\Sigma$ and $w \in \Sigma^*$, $w \in \mathcal{L}(r)$ iff $n(d_w(r))$.*

For example, consider $x \cdot y^{\emptyset} ||^{\emptyset}_{\{x,y\}} x \cdot z$ for which $x \cdot y \cdot z \in \mathcal{L}(x \cdot y^{\emptyset} ||^{\emptyset}_{\{x,y\}} x \cdot z)$. We build the derivative w.r.t symbols x, y and z and then test if the resulting expression is nullable.

In a first step, we find that

$$d_x(x \cdot y^{\emptyset} ||^{\emptyset}_{\{x,y\}} x \cdot z) = \underbrace{y^{\emptyset} ||^{\emptyset}_{\{x,y\}} z}_{\text{(D8)}} + \underbrace{y^{\{x\}} ||^{\emptyset}_{\{x,y\}} x \cdot z}_{\text{(D10)}} + \underbrace{x \cdot y^{\emptyset} ||^{\{x\}}_{\{x,y\}} z}_{\text{(D11)}}$$

where subterms are connected to the cases in Definition 9. For brevity, some simplifications such as $\epsilon \cdot y = y$ are applied. The second and third subterm both lead to ϕ. Hence, we continue with $y^{\emptyset} ||^{\emptyset}_{\{x,y\}} z$ and find that

$$d_x(x \cdot y^{\emptyset} ||^{\emptyset}_{\{x,y\}} x \cdot z) = \underbrace{y^{\emptyset} ||^{\emptyset}_{\{x,y\}} z}_{\text{(D8)}} + \underbrace{\cancel{y^{\{x\}} ||^{\emptyset}_{\{x,y\}} x \cdot z}}_{\text{(D10)}} + \underbrace{\cancel{x \cdot y^{\emptyset} ||^{\{x\}}_{\{x,y\}} z}}_{\text{(D11)}}$$

We strike through subterms which lead to ϕ. The next steps are as follows

$$d_y(y^{\emptyset} ||^{\emptyset}_{\{x,y\}} z) = \underbrace{\cancel{\phi^{\emptyset} ||^{\emptyset}_{\{x,y\}} \phi}}_{\text{(D8)}} + \underbrace{\epsilon^{\{y\}} ||^{\emptyset}_{\{x,y\}} z}_{\text{(D10)}} + \underbrace{\cancel{y^{\emptyset} ||^{\{y\}}_{\{x,y\}} \phi}}_{\text{(D11)}}$$

$$d_z(\epsilon^{\{y\}} ||^{\emptyset}_{\{x,y\}} z) = \underbrace{\cancel{\phi^{\{y\}} ||^{\emptyset}_{\{x,y\}} \phi}}_{\text{(D8)}} + \underbrace{\cancel{\phi^{\{y,z\}} ||^{\emptyset}_{\{x,y\}}} z}_{\text{(D10)}} + \underbrace{\epsilon^{\{y\}} ||^{\{z\}}_{\{x,y\}} \epsilon}_{\text{(D11)}}$$

In conclusion, we find that $d_z(d_y(d_x(x \cdot y^{\emptyset} ||^{\emptyset}_{\{x,y\}} x \cdot z))) = ... + \epsilon^{\{y\}} ||^{\{z\}}_{\{x,y\}} \epsilon$ where expression $\epsilon^{\{y\}} ||^{\{z\}}_{\{x,y\}} \epsilon$ is nullable. Hence, $x \cdot y \cdot z \in \mathcal{L}(x \cdot y| \sim |_{\{x,y\}} x \cdot z)$.

In general, it seems wasteful to repeatedly generate derivatives just for the sake of testing a specific word. A more efficient method is to construct a DFA via which we can then test many words. Brzozowski recognized that there is an elegant DFA construction method based on derivatives. Expressions are treated as states. For each expression r and symbol x, there is a transition to its derivative $d_x(r)$.

For this construction to work we must establish that (1) the transitions implied by the derivatives *represent* all cases, and (2) the set of states remains *finite*. This is what we will consider next.

First, we establish (1) by verifying that each shuffle expression can be represented as a sum of its derivatives, extending another result of Brzozowski.

Theorem 7 (Representation). *For all $r \in R_\Sigma$,*
$\mathcal{L}(r) = \mathcal{L}((n(r)) \Rightarrow \epsilon) \cup \bigcup_{x \in \Sigma} x \cdot \mathcal{L}(d_x(r))$.

Proof. Follows immediately from Lemma 3 and Theorem 6. $\square$

States are descendants of expression r. Hence, we must verify that the set $d_{\Sigma^*}(r)$ is finite. In general, this may not be the case as shown by the following example

$$\begin{array}{lcl} d_x(x^*) & = & \epsilon \cdot x^* \\ d_x(\epsilon \cdot x^*) & = & \phi \cdot x^* + \epsilon \cdot x^* \\ d_x(\phi \cdot x^* + \epsilon \cdot x^*) & = & (\phi \cdot x^* + \epsilon \cdot x^*) + (\phi \cdot x^* + \epsilon \cdot x^*) \\ ... & & \end{array}$$

To guarantee finiteness, we consider expressions modulo *similarity*.

Definition 10 (Similarity). *Two expressions $r, s \in R_\Sigma$ are* similar*, written $r \approx s$, if one can be transformed into the other by applying the following identities:*

(I1) $r + s = s + r$ *(I2)* $r + (s + t) = (r + s) + t$ *(I3)* $r + r = r$

For $S \subseteq R_\Sigma$ we write $S/_\approx$ to denote the set of equivalence classes of all similar expressions in S.

To verify (dis)similarity among descendants we normalize expressions via identities (I1-3) as follows. (1) Alternatives are kept in a list, see (I2), and sorted according to the number of occurrences of symbols, see (I1). (2) Any duplicates in the list are removed, see (I3).

For example, consider the above regular expression $(\phi \cdot x^* + \epsilon \cdot x^*) + (\phi \cdot x^* + \epsilon \cdot x^*)$. In an intermediate step, we find the form $\phi \cdot x^* + \phi \cdot x^* + \epsilon \cdot x^* + \epsilon \cdot x^*$. Parentheses are omitted due to normalization via identities (I1) and (I2). Finally, duplicates are removed via (I3) which leads to $\phi \cdot x^* + \epsilon \cdot x^*$.

This normalization step must be applied to all subexpressions. For example, consider expression $x^* \cdot x$ where repeated application of the derivative operation yield the expression $(... + \epsilon \cdot x^* + ... + \epsilon \cdot x^*) \cdot x + ...$

It is easy to see that via this normalization step we reach a canonical normal form for each expression. Hence, to verify that two expressions are (dis)similar we build their canonical normal forms and then check if both forms are syntactically equivalent.

Thus, the set $d_{\Sigma^*}(r)/_\approx$ is obtained by generating dissimilar descendants, starting with $\{r\}$, where descendants are normalized. That this generation step reaches a fixpoint is guaranteed by the following result.

Theorem 8 (Finiteness). *For $r \in R_\Sigma$ the set $d_{\Sigma^*}(r)/_\approx$ is finite.*

Proof. It suffices to consider the new case of shuffle expressions. Our argumentation is similar to the case of concatenation in Brzozowski's work [3], in particular the proofs of Theorems 4.3(a) and 5.2. It suffices to consider the new form $r\,{}^{P_1}||^{P_2}_{\Gamma}\,s$ which we abbreviate by t.

By inspection of the definition of $d_x(\_)$ on ${}^{P_1}||^{P_2}_{\Gamma}$ and application of identity (I2) (associativity) we find that all descendants of t can be represented as a sum of expressions which are either of the shape ϕ or $r'\,{}^{P'_1}||^{P'_2}_{\Gamma}\,s'$ where r' is a descendant of r and s' is a descendant of s, but P'_1 and P'_2 are arbitrary subsets of Σ.

Thus, we can apply a similar argument as in Brzozowski's original proof to approximate the number of descendants of $r\,{}^{P_1}||^{P_2}_{\Gamma}\,s$ by the number of descendants of r and s.

Suppose $\sharp D_r$ denotes the number of all descendants of r and n is the number of elements in Σ. Then, we can approximate $\sharp D_t$ as follows:

$$\sharp D_t \leq 2^{\sharp D_r * \sharp D_s * 2^n * 2^n}$$

The exponent counts the number of different factors of the form $r'\,{}^{P_1}||^{P_2}_{\Gamma}\,s'$ and a sum corresponds to a subset of these factors. The factor $2^n * 2^n$ arises because P_1, P_2 range over subsets of Σ where $n = |\Sigma|$. The factor $\sharp D_s * \sharp D_r$ arises from the variation of r' and s' over the descendants of r and s, respectively.

As $\sharp D_r$ and $\sharp D_s$ are finite, by the inductive hypothesis, we obtain a very large, but finite bound on $\sharp D_t$. $\square$

In summary, we can state the following definition and theorem.

Definition 11 (Derivative-based DFA Construction). *For $r \in R_\Sigma$ we define $\mathcal{D}(r) = (Q, \Sigma, q_0, \delta, F)$ where $Q = d_{\Sigma^*}(r)/_\approx$, $q_0 = r$, for each $q \in Q$ and $x \in \Sigma$ we define $\delta(q, x) = d_x(q)$, and $F = \{q \in Q \mid n(q)\}$.*

Theorem 9. *For any $r \in R_\Sigma$ we have that $\mathcal{L}(r) = \mathcal{L}(\mathcal{D}(r))$.*

6. Partial Derivatives for General Synchronous Shuffling

In this section, we define partial derivatives for regular shuffle expressions and establish their basic properties in analogy to Antimirov's work [4]. Essentially, we prove the representation theorem and finiteness of the set of iterated partial derivatives of regular shuffle expressions. Thus, it is possible to construct nondeterministic finite automata directly from regular shuffle expressions.

In this section, the guard expression $(b) \Rightarrow X$, where b is a boolean and X is a set, is defined by $(\text{true}) \Rightarrow X = X$ and $(\text{false}) \Rightarrow X = \{\}$.

Definition 12. *The smart concatenation constructor $\odot : R_\Sigma \times R_\Sigma \to R_\Sigma$ is defined as follows for $r_1, r_2 \in R_\Sigma$.*

$$r_1 \odot r_2 = \begin{cases} \phi & r_1 = \phi \vee r_2 = \phi \\ r_2 & r_1 = \epsilon \\ r_1 & r_2 = \epsilon \\ r_1 \cdot r_2 & \textit{otherwise} \end{cases}$$

It is lifted to sets of expressions by omitting empty expressions, that is, for $R \subseteq R_\Sigma$ and $r \in R_\Sigma$ define

$$R \odot r_2 = \{r \mid r = r_1 \odot r_2, r_1 \in R, r \neq \phi\}$$

Definition 13. *The set of partial derivatives of $r \in R_\Sigma$ w.r.t. $x \in \Sigma$, written $pd_x(r) \subseteq R_\Sigma$, are defined inductively in Figure 1.*

The definition extends to words inductively by $pd_\epsilon(r) = \{r\}$ and $pd_{wx}(r) = \bigcup\{pd_x(r') \mid r' \in pd_w(r)\}$. It further extends to languages $L \subseteq \Sigma^*$ by $pd_L(r) = \bigcup\{pd_w(r) \mid w \in L\}$.

We refer to the special case $pd_{\Sigma^*}(r)$ as the set of descendants *of r. A descendant is either the expression itself, a derivative of the expression, or the derivative of a descendant.*

For a set of regular expressions $R \subseteq R_\Sigma$, we define $\mathcal{L}(R) = \bigcup_{r \in R} \mathcal{L}(r)$.

Theorem 10 (Left Quotients). *For all $r \in R_\Sigma$ and $x \in \Sigma$,*

$$\mathcal{L}(pd_x(r)) = x \backslash \mathcal{L}(r).$$

$$
\begin{array}{lcll}
pd_x(\phi) & = & \{\} & \text{(PD1)} \\
pd_x(\epsilon) & = & \{\} & \text{(PD2)} \\
pd_x(x) & = & \{\epsilon\} & \text{(PD3a)} \\
pd_x(y) & = & \{\} & \text{(PD3b)} \\
pd_x(r+s) & = & pd_x(r) \cup pd_x(s) & \text{(PD4)} \\
pd_x(r \cdot s) & = & pd_x(r) \odot s \cup (n(r)) \Rightarrow pd_x(s) & \text{(PD5)} \\
pd_x(r^*) & = & pd_x(r) \odot r^* & \text{(PD6)} \\
pd_x(r^{P_1}||_\Gamma^{P_2} s) & = & (x \notin \Gamma) \Rightarrow (\{r'^{P_1}||_\Gamma^{P_2} s \mid r' \in pd_x(r)\} \cup & \\
 & & \qquad\qquad \{r^{P_1}||_\Gamma^{P_2} s' \mid s' \in pd_x(s)\}) & \text{(PD7)} \\
 & & \cup (x \in \Gamma \wedge P_1 \cap P_2 = \emptyset) \Rightarrow & \\
 & & \quad \{r'^{\emptyset}||_\Gamma^{\emptyset} s' \mid r' \in pd_x(r), s' \in pd_x(s)\} & \text{(PD8)} \\
 & & \cup (x \in \Gamma \wedge P_1 \cap P_2 \neq \emptyset) \Rightarrow & \\
 & & \quad \{r'^{P_1}||_\Gamma^{P_2} s' \mid r' \in pd_x(r), s' \in pd_x(s)\} & \text{(PD9)} \\
 & & \cup (x \in \Gamma \wedge (P_1 \cup x) \cap P_2 = \emptyset) \Rightarrow & \\
 & & \quad \{r'^{P_1 \cup x}||_\Gamma^{P_2} s \mid r' \in pd_x(r)\} & \text{(PD10)} \\
 & & \cup (x \in \Gamma \wedge P_1 \cap (P_2 \cup x) = \emptyset) \Rightarrow & \\
 & & \quad \{r^{P_1}||_\Gamma^{P_2 \cup x} s' \mid s' \in pd_x(s)\} & \text{(PD11)}
\end{array}
$$

Figure 1: Partial derivatives of regular shuffle expressions. We assume $y \in \Sigma$ with $x \neq y$.

PROOF. Consider $\mathcal{L}(pd_x(r)) = \bigcup_{r' \in pd_x(r)} \mathcal{L}(r')$.

Let $r' \in pd_x(r)$ and show that $\mathcal{L}(r') \subseteq x \backslash \mathcal{L}(r)$ by induction on r. Thus, we obtain $\mathcal{L}(pd_x(r)) \subseteq x \backslash \mathcal{L}(r)$.

The cases for ϕ, ϵ, x, and $y \neq x$ are immediate.

Case $r + s$**.** Immediate by inductive hypothesis.

Case $r \cdot s$**.** If $r' \in pd_x(r) \odot s$, then $r' = r'' \odot s$, for some $r'' \in pd_x(r)$. Induction yields $\mathcal{L}(r'') \subseteq x \backslash \mathcal{L}(r)$ so that $\mathcal{L}(r'' \odot s) = \mathcal{L}(r'') \cdot \mathcal{L}(s) \subseteq x \backslash \mathcal{L}(r) \cdot \mathcal{L}(s) = x \backslash \mathcal{L}(r \cdot s)$.

If $n(r)$ and $r' \in pd_x(s)$, then induction yields $\mathcal{L}(r') \subseteq x \backslash \mathcal{L}(s) \subseteq x \backslash \mathcal{L}(r \cdot s)$ because $n(r)$.

Case r^***.** If $r' \in pd_x(r) \odot r^*$, then $r' = r'' \odot r^*$, for some $r'' \in pd_x(r)$. Induction yields $\mathcal{L}(r'') \subseteq x \backslash \mathcal{L}(r)$ so that $\mathcal{L}(r'' \odot r^*) = \mathcal{L}(r'') \cdot \mathcal{L}(r*) \subseteq x \backslash \mathcal{L}(r) \cdot \mathcal{L}(r^*) = x \backslash \mathcal{L}(r \cdot r^*)$.

Case $r^{P_1}||_\Gamma^{P_2} s$**.** We distinguish among the following subcases.

Subcase $x \notin \Gamma$**:** Assume $r' = r''^{P_1}||_\Gamma^{P_2} s$ where $r'' \in pd_x(r)$. By induction $\mathcal{L}(r'') \subseteq x \backslash \mathcal{L}(r)$. Via Lemma 4 we obtain $\mathcal{L}(r''^{P_1}||_\Gamma^{P_2} s) \subseteq x \backslash \mathcal{L}(r^{P_1}||_\Gamma^{P_2} s)$ and we are done.

Otherwise $r' = r^{P_1}||_{\Gamma}^{P_2} s''$ where $s'' \in pd_x(s'')$. Via a similar argument we can show that $\mathcal{L}(r^{P_1}||_{\Gamma}^{P_2} s'') \subseteq x\backslash\mathcal{L}(r^{P_1}||_{\Gamma}^{P_2} s)$ which concludes this subcase.

Subcase $x \in \Gamma$ and $P_1 \cap P_2 = \emptyset$: We find $r' = r''^{\emptyset}||_{\Gamma}^{\emptyset} s$ where $r'' \in pd_x(r)$. By induction $\mathcal{L}(r'') \subseteq x\backslash\mathcal{L}(r)$. Via Lemma 6 we obtain $\mathcal{L}(r''^{\emptyset}||_{\Gamma}^{\emptyset} s) \subseteq x\backslash\mathcal{L}(r^{\emptyset}||_{\Gamma}^{\emptyset} s)$. As observed earlier, for $P_1 \cap P_2 = \emptyset$ we have that $\mathcal{L}(r^{\emptyset}||_{\Gamma}^{\emptyset} s) = \mathcal{L}(r^{P_1}||_{\Gamma}^{P_2} s)$ and we are done.

Subcase $x \in \Gamma$ and $P_1 \cap P_2 \neq \emptyset$: Can be verified similarly to the above.

Subcase $x \in \Gamma$ and $(P_1 \cup x) \cap P_2 = \emptyset$: We find $r' = r''^{P_1 \cup x}||_{\Gamma}^{P_2} s$ where $r'' \in pd_x(r)$. By induction $\mathcal{L}(r'') \subseteq x\backslash\mathcal{L}(r)$ and Lemma 8 we can establish the desired result.

Subcase $x \in \Gamma$ and $P_1 \cap (P_2 \cup x) = \emptyset$: Can be verified similarly to the above.

The proof for the other direction $x\backslash\mathcal{L}(r) \subseteq \mathcal{L}(pd_x(r))$ is similar. A noteworthy point is that for this direction we require distributivity of alternatives over concatenation and shuffling:

$$\mathcal{L}((r_1 + r_2) \cdot r_3) = \mathcal{L}(r_1 \cdot r_3 + r_2 \cdot r_3)$$

$$\mathcal{L}((r_1 + r_2)^{P_1}||_{\Gamma}^{P_2} r_3) = \mathcal{L}({r_1}^{P_1}||_{\Gamma}^{P_2} r_3 + {r_2}^{P_1}||_{\Gamma}^{P_2} r_3)$$

The first equivalence obviously holds. The second equivalence holds by easy inspection of the definition of general synchronous shuffling.

The proof of $x\backslash\mathcal{L}(r) \subseteq \mathcal{L}(pd_x(r))$ proceeds by induction where we encounter intermediate forms such as "$pd_x(r)^{P_1}||_{\Gamma}^{P_2} s$". Recall $\mathcal{L}(pd_x(r)) = \bigcup_{r' \in pd_x(r)} \mathcal{L}(r')$. Thanks to the above distributivity laws such forms can be transformed into $+_{r' \in pd_x(r)} r'^{P_1}||_{\Gamma}^{P_2} s$ on which we can then apply similar reasoning steps as above. □

Lemma 11. *For all $r \in R_\Sigma$ and $w \in \Sigma^*$, $\mathcal{L}(pd_w(r)) = w\backslash\mathcal{L}(r)$.*

Proof. The proof is by induction on w.

Case ϵ. Immediate.

Case $x \cdot w$.

$$\begin{array}{rl}
\mathcal{L}(pd_{x\cdot w}(r)) & = \bigcup\{\mathcal{L}(pd_x(r')) \mid r' \in pd_w(r)\} \\
& \{\text{by Lemma 10}\} \\
& = \bigcup\{x\backslash\mathcal{L}(r') \mid r' \in pd_w(r)\} \\
& = x\backslash\mathcal{L}(\bigcup\{r' \mid r' \in pd_w(r)\}) \\
& = x\backslash\mathcal{L}(pd_w(r)) \\
& \{\text{by induction}\} \\
& = x\backslash(w\backslash\mathcal{L}(r)) \\
& = x \cdot w\backslash\mathcal{L}(r)
\end{array}$$

Theorem 11 (Representation). *For $r \in R_\Sigma$, $\mathcal{L}(r) = ((n(r)) \Rightarrow \{\epsilon\}) \cup \bigcup_{x\in\Sigma} x \cdot \mathcal{L}(\sum pd_x(r))$.*

Proof. Straightforward using Theorem 10.

We prove finiteness following the method suggested by Broda and coworkers [12]. We first define a function that computes a finite superset of $pd_{\Sigma^+}(r)$. Then we show that the image of this function is closed under taking the partial derivative, which proves the claim.

Definition 14. *The iterated partial derivatives of $r \in R_\Sigma$, written $pd^+(r)$, are defined inductively as follows.*

$$\begin{array}{lcl}
pd^+(\phi) & = & \{\} \\
pd^+(\epsilon) & = & \{\} \\
pd^+(x) & = & \{\epsilon\} \\
pd^+(r+s) & = & pd^+(r) \cup pd^+(s) \\
pd^+(r\cdot s) & = & pd^+(r) \odot s \cup pd^+(s) \\
pd^+(r^*) & = & pd^+(r) \odot r^* \\
pd^+(r^{P_1}||^{P_2}_{\Gamma} s) & = & \{r'^{R}||^{S}_{\Gamma} s' \mid r' \in \{r\} \cup pd^+(r), s' \in \{s\} \cup pd^+(s), R, S \subseteq \Gamma\}
\end{array}$$

Lemma 12. *For each r, $pd^+(r)$ is finite.*

Proof. Straightforward induction on r.

Lemma 13. *For all r, s, and $x \in \Sigma$, $pd_x(r \odot s) = pd_x(r \cdot s)$.*

PROOF. Consider the cases in the definition of $r \odot s$.

Case $r = \phi$ **or** $s = \phi$, then $pd_x(r \odot s) = pd_x(\phi) = \{\}$.
$pd_x(\phi \cdot s) = pd_x(\phi) \odot s \cup (n(\phi)) \Rightarrow pd_x(s) = \{\}$.
$pd_x(r \cdot \phi) = pd_x(r) \odot \phi \cup (n(r)) \Rightarrow pd_x(\phi) = \{\}$.
Case $r = \epsilon$, then $pd_x(r \odot s) = pd_x(s)$.
$pd_x(\epsilon \cdot s) = pd_x(\epsilon) \odot s \cup (n(\epsilon)) \Rightarrow pd_x(s) = pd_x(s)$
Case $s = \epsilon$, then $pd_x(r \odot s) = pd_x(r)$.
$pd_x(r \cdot \epsilon) = pd_x(r) \odot \epsilon \cup (n(r)) \Rightarrow pd_x(\epsilon) = pd_x(r)$.
Case $\{s, r\} \cap \{\phi, \epsilon\} = \emptyset$ is obvious.

Lemma 14. *For all* $r' \in \{r\} \cup pd^+(r)$ *and* $x \in \Sigma$, $pd_x(r') \subseteq pd^+(r)$.

PROOF. By induction on r.

Case ϕ**,** ϵ**,** y**.** Immediate.
Case $r + s$**.** Immediate by induction.
Case $r \cdot s$**.** First consider $r \cdot s$.

$$\begin{aligned}
pd_x(r \cdot s) &= pd_x(r) \odot s \cup (n(r)) \Rightarrow pd_x(s) \\
&\subseteq pd_x(r) \odot s \cup pd_x(s) \\
&\{\text{inductive hypothesis for } r \text{ and } s\} \\
&\subseteq pd^+(r) \odot s \cup pd^+(s) \\
&= pd^+(r \cdot s)
\end{aligned}$$

Now consider $r' \in pd^+(r \cdot s) = pd^+(r) \odot s \cup pd^+(s)$.
If $r' \in pd^+(s)$, then $pd_x(r') \subseteq pd^+(s) \subseteq pd^+(r \cdot s)$ by induction.
If $r' \in pd^+(r) \odot s$, then $\exists r'' \in pd^+(r)$ such that $r' = r'' \odot s$. Then

$$\begin{aligned}
pd_x(r') &= pd_x(r'' \odot s) \\
&\{\text{by Lemma 13}\} \\
&= pd_x(r'' \cdot s) \\
&= pd_x(r'') \odot s \cup (n(r'')) \Rightarrow pd_x(s) \\
&\{\text{by induction}\} \\
&\subseteq pd^+(r) \odot s \cup pd^+(s) \\
&= pd^+(r \cdot s)
\end{aligned}$$

Case r^*. First consider r^*.

$$\begin{aligned} pd_x(r^*) &= pd_x(r) \odot r^* \\ &\{\text{by induction}\} \\ &\subseteq pd^+(r) \odot r^* \\ &= pd^+(r^*) \end{aligned}$$

Now consider $r' \in pd^+(r^*) = pd^+(r) \odot r^*$. Thus, there exists $r'' \in pd^+(r)$ such that $r' = r'' \odot r^*$.

$$\begin{aligned} pd_x(r') &= pd_x(r'' \odot r^*) \\ &\{\text{by Lemma 13}\} \\ &= pd_x(r'' \cdot r^*) \\ &= pd_x(r'') \odot r^* \\ &\{\text{by induction}\} \\ &\subseteq pd^+(r) \odot r^* \\ &= pd^+(r^*) \end{aligned}$$

Case $r^{P_1}||^{P_2}_{\Gamma}s$. First consider the whole expression and show that $P := pd_x(r^{P_1}||^{P_2}_{\Gamma}s) \subseteq pd^+(r^{P_1}||^{P_2}_{\Gamma}s)$.

Subcase (PD7): Let $P_7 := (x \notin \Gamma) \Rightarrow \{r'^{P_1}||^{P_2}_{\Gamma}s \mid r' \in pd_x(r)\} \cup \{r^{P_1}||^{P_2}_{\Gamma}s' \mid s' \in pd_x(s)\}$. By induction, $pd_x(r) \subseteq pd^+(r)$ and $pd_x(s) \subseteq pd^+(s)$. Furthermore, $P_1, P_2 \subseteq \Gamma$, so that $P_7 \subseteq pd^+(r^{P_1}||^{P_2}_{\Gamma}s)$.

Subcase (PD8): Let $P_8 := (x \in \Gamma \wedge P_1 \cap P_2 = \emptyset) \Rightarrow \{r'^{\emptyset}||^{\emptyset}_{\Gamma}s' \mid r' \in pd_x(r), s' \in pd_x(s)\}$. By induction $pd_x(r) \subseteq pd^+(r)$ and $pd_x(s) \subseteq pd^+(s)$. Furthermore, $\emptyset \subseteq \Gamma$, so that $P_8 \subseteq pd^+(r^{P_1}||^{P_2}_{\Gamma}s)$.

Subcase (PD9): Let $P_9 := (x \in \Gamma \wedge P_1 \cap P_2 \neq \emptyset) \Rightarrow \{r'^{P_1}||^{P_2}_{\Gamma}s' \mid r' \in pd_x(r), s' \in pd_x(s)\}$. By induction, $pd_x(r) \subseteq pd^+(r)$ and $pd_x(s) \subseteq pd^+(s)$. Furthermore, $P_1, P_2 \subseteq \Gamma$, so that $P_9 \subseteq pd^+(r^{P_1}||^{P_2}_{\Gamma}s)$.

Subcase (PD10): Let $P_{10} := (x \in \Gamma \wedge (P_1 \cup x) \cap P_2 = \emptyset) \Rightarrow \{r'^{P_1 \cup x}||^{P_2}_{\Gamma}s \mid r' \in pd_x(r)\}$. By induction $pd_x(r) \subseteq pd^+(r)$. Furthermore, $P_1 \cup x \subseteq \Gamma$ and $P_2 \subseteq \Gamma$, so that $P_{10} \subseteq pd^+(r^{P_1}||^{P_2}_{\Gamma}s)$.

Subcase (PD11): Let $P_{11} := (x \in \Gamma \wedge P_1 \cap (P_2 \cup x) = \emptyset) \Rightarrow \{r^{P_1}||^{P_2 \cup x}_{\Gamma}s' \mid s' \in pd_x(s)\}$. By induction $pd_x(s) \subseteq pd^+(s)$. Furthermore, $P_1 \subseteq \Gamma$ and $P_2 \cup x \subseteq \Gamma$, so that $P_{11} \subseteq pd^+(r^{P_1}||^{P_2}_{\Gamma}s)$.

Concluding: By definition, $P = P_7 \cup P_8 \cup P_9 \cup P_{10} \cup P_{11}$ and from the subcases, we can conclude that $P \subseteq pd^+(r^{P_1}||^{P_2}_{\Gamma}s)$.

Next consider some $r_0 \in pd^+(r^{P_1}||^{P_2}_{\Gamma} s)$ and show that $pd_x(r_0) \subseteq pd^+(r^{P_1}||^{P_2}_{\Gamma} s)$. That is, there exists $r' \in \{r\} \cup pd^+(r)$, $s' \in \{s\} \cup pd^+(s)$, and $R, S \subseteq \Gamma$ such that $r_0 = r'^R||^S_{\Gamma} s'$. Now consider $pd_x(r'^R||^S_{\Gamma} s')$.

Subcase (PD7): Let $P_7 := (x \notin \Gamma) \Rightarrow \{r''^{P_1}||^{P_2}_{\Gamma} s' \mid r'' \in pd_x(r')\} \cup \{r'^{P_1}||^{P_2}_{\Gamma} s'' \mid s'' \in pd_x(s')\}$. By induction $pd_x(r') \subseteq pd^+(r)$ and $pd_x(s') \subseteq pd^+(s)$. Furthermore $P_1, P_2 \subseteq \Gamma$. Hence, $P_7 \subseteq pd^+(r^{P_1}||^{P_2}_{\Gamma} s)$.

The remaining subcases are analogous to the above subcases. As above, we conclude that $pd_x(r_0) \subseteq pd^+(r^{P_1}||^{P_2}_{\Gamma} s)$.

Lemma 15. *For all r, $pd_{\Sigma^+}(r) \subseteq pd^+(r)$.*

Proof. Show that for each $w \in \Sigma^+$, $pd_w(r) \subseteq pd^+(r)$, by induction on w.

If $w = x$, a singleton word, then $pd_x(r) \subseteq pd^+(r)$ by Lemma 14.

If $w = vx$, for some $v \in \Sigma^+$, then induction yields that $pd_v(r) \subseteq pd^+(r)$.

Now $pd_{vx}(r) = \bigcup\{pd_x(r') \mid r' \in pd_v(r)\} \subseteq \bigcup\{pd_x(r') \mid r' \in pd^+(r)\} \subseteq pd^+(r)$ by Lemma 14.

Corollary 3. *For all r, $pd_{\Sigma^+}(r)$ is finite.*

Proof. Immediate by finiteness of $pd^+(r)$ (Lemma 12) and Lemma 15.

Definition 15 (Derivative-based NFA Construction). *For $r \in R_{\Sigma}$ we define the nondeterministic finite automaton $\mathcal{N}(r) = (Q, \Sigma, q_0, \delta, F)$ where $Q = pd_{\Sigma^*}(r)$, $q_0 = r$, for each $q \in Q$ and $x \in \Sigma$ we define $\delta(q, x) = pd_x(q)$, and $F = \{q \in Q \mid n(q)\}$.*

Theorem 12. *For all $r \in R_{\Sigma}$, we have that $\mathcal{L}(r) = \mathcal{L}(\mathcal{N}(r))$.*

7. Discussion

(Partial) Derivatives for free via Transformation. Based on the results from Sections 3, 5 and 6, we obtain for free derivatives and partial derivatives for interleaving, weakly and strongly synchronized shuffling as well as synchronous composition. Our method is as follows:

1. Transform each shuffle expression into its general synchronous shuffle representation as specified in Section 4.
2. Apply the (partial) derivative constructions from Sections 5 and 6 to the resulting expression.

For interleaving ($||$), weakly ($| \sim |_\Gamma$) and strongly synchronized shuffling ($|||_\Gamma$) the transformation step is purely syntactic. The results from Section 4 imply the following identities:

$$r||s = r^\Sigma||^\Sigma_\emptyset s \quad r|||_\Gamma s = r^\Sigma||^\Sigma_\Gamma s \quad r| \sim |_\Gamma s = r^\emptyset||^\emptyset_\Gamma s$$

These identities are exhaustively applied from left to right. It is easy to see that this process eventually terminates and we thus obtain an expression only composed of the general synchronous shuffling operation.

For synchronous composition ($|||$) the identity $r|||s = r^\Sigma||^\Sigma_{\alpha(r)\cap\alpha(s)} s$ applies. For this case the transformation step is more involved as we must compute the alphabet of expressions (resp. the alphabet of the underlying languages). We define $\alpha(r) = \alpha(\mathcal{L}(r))$.

For plain regular expressions, we can easily compute the alphabet by observing the structure of expressions. For example, $\alpha(r^*) = \alpha(r)$. $\alpha(x) = \{x\}$. $\alpha(\phi) = \{\}$. $\alpha(\epsilon) = \{\}$. $\alpha(r + s) = \alpha(r) \cup \alpha(s)$. $\alpha(r \cdot s) = \alpha(r) \cup \alpha(s)$ if $\mathcal{L}(r), \mathcal{L}(s) \neq \{\}$. Otherwise, $\alpha(r \cdot s) = \{\}$. The test $\mathcal{L}(r) \neq \{\}$ can again be defined by observing the expression structure. We omit the details.

In the presence of shuffle expressions such as synchronous composition, it is not obvious how to appropriately extend the above structural definition. For example, consider $x \cdot x \cdot y|||x \cdot y$. We find that $\alpha(x \cdot x \cdot y) = \{x, y\}$ and $\alpha(x \cdot y)$ but $\alpha(x \cdot x \cdot y|||x \cdot y) = \{\}$ due to the fact that $x \cdot x \cdot y|||x \cdot y$ equals ϕ.

Hence, to compute the alphabet of some $r \in R_\Sigma$ we first convert r into a DFA M using the derivative-based automata construction. To compute the alphabet of M we use a variant of the standard emptiness check algorithm for DFAs. First, we compute all reachable paths from any of the final states to the initial state. To avoid infinite loops we are careful not to visit a transition twice on a path. Then, we obtain the alphabet of M by collecting the set of all symbols on all transitions along these paths.

Thus, the transformation of expressions composed of synchronous composition proceeds as follows. In the to be transformed expression, we pick any subexpression $r_1|||r_2$ where $r_1, r_2 \in R_\Sigma$. If there is none we are done. Otherwise, we must find r_1 and r_2 which have already been transformed, resp., do not contain any shuffling operator. Alphabets $\alpha(r_1)$ and $\alpha(r_2)$ are computed as described and subexpression $r_1|||r_2$ is replaced by ${r_1}^\Sigma||^\Sigma_{\alpha(r_1)\cap\alpha(r_2)} r_2$. This process repeats until all synchronous composition operations have been replaced.

Specialization of (partial) derivative method. Instead of first transforming each shuffle variant into its general form and then applying the general (partial) derivative method, we might want to specialize the general (partial) derivative method. The key to obtain specialized (partial) derivative methods is to show that the general (partial) derivative methods are closed for certain classes of expressions. As we will shortly see, this only applies to shuffling and strongly synchronized shuffling but not to weak synchronous shuffling and synchronous composition.

We will write $R_{\Sigma}^{|\cdot|}$ to denote the subset of regular expressions restricted to shuffle expressions composed of $|\cdot|$ where $|\cdot|$ stands for any of the four shuffling forms we have seen so far.

Theorem 13 (Closure for Shuffling). *For any $r \in R_{\Sigma}^{\|}$ and $x \in \Sigma$ we have that (1) $d_x(r) \in R_{\Sigma}^{\|}$ and (2) for each $r' \in pd_x(r)$ we find that $r' \in R_{\Sigma}^{\|}$.*

Proof. Recall that $\|$ can be expressed as ${}^{\emptyset}||_{\emptyset}^{\emptyset}$. By case analysis of Definitions 9 and 13. Cases (D7) and (PD7) apply only. $\square$

Theorem 14 (Closure for Strongly Synchronized Shuffling). *For any $r \in R_{\Sigma}^{|||_\Gamma}$ and $x \in \Sigma$ we have that (1) $d_x(r) \in R_{\Sigma}^{|||_\Gamma}$ and (2) for each $r' \in pd_x(r)$ we find that $r' \in R_{\Sigma}^{|||_\Gamma}$.*

Proof. Recall that $|||_\Gamma$ can be expressed as ${}^{\Gamma}||_{\Gamma}^{\Gamma}$. Again by case analysis. This time only cases (D7) and (D9) in case of derivatives and cases (PD7) and (PD9) in case of partial derivatives apply. $\square$

We obtain the following specialized functions for computing derivatives

$$\begin{array}{lcl} d_x(r\|s) & = & d_x(r)\|s + r\|d_x(r) \\ d_x(r|||_\Gamma s) & = & (x \in \Gamma) \Rightarrow d_x(r)|||_\Gamma d_x(s) \\ & & +(x \notin \Gamma) \Rightarrow d_x(r)|||_\Gamma s + r|||_\Gamma d_x(s) \end{array}$$

and partial derivatives

$$\begin{array}{lcl} pd_x(r\|s) & = & \{r'\|s \mid r' \in pd_x(r)\} \cup \{r\|s' \mid s' \in pd_x(r)\} \\ pd_x(r|||_\Gamma s) & = & (x \in \Gamma) \Rightarrow \{r'|||_\Gamma s' \mid r' \in pd_x(r)\ s' \in pd_x(s)\} \\ & & \cup(x \notin \Gamma) \Rightarrow \{r'|||_\Gamma s \mid r' \in pd_x(r)\} \cup \{r|||_\Gamma s' \mid s' \in pd_x(r)\} \end{array}$$

This result does not extend to weak synchronous shuffling and synchronous composition, as they are not closed under derivative. For example, consider weak synchronous shuffling $| \sim |_\Gamma$ which is expressed by ${}^{\emptyset}||^{\emptyset}_{\Gamma}$. For the expression $x \cdot z^{\emptyset}||^{\emptyset}_{\{x\}}y$, we find that

$$d_x(x \cdot z^{\emptyset}||^{\emptyset}_{\{x\}}y) = z^{\{x\}}||^{\emptyset}_{\{x\}}y + x \cdot z^{\emptyset}||^{\{x\}}_{\{x\}}\phi$$

The expression on the right-hand side is *not* part of $R_\Sigma^{|\sim|_\Gamma}$ due to subexpressions of the form $z^{\{x\}}||^{\emptyset}_{\{x\}}y$. Hence, the closure property does no longer hold. A similar observation applies to synchronous composition.

The above implies that simple, direct (partial) derivative methods for weak synchronous shuffling and synchronous composition do not seem to exist. For example, consider the following attempt to define a direct derivative method for synchronous composition

$$\begin{array}{rcl} d_x(r|||s) & = & (x \in \alpha(r) \cap \alpha(s)) \Rightarrow d_x(r)|||d_x(s) \\ & & +(x \notin \alpha(r) \cap \alpha(s)) \Rightarrow d_x(r)|||s + r|||d_x(s) \end{array}$$

The above is similar to the specialized definition for strong synchronized shuffling. Expressions generated are clearly in $R_\Sigma^{|||}$. However, the definition is flawed as shown by the following example.

Consider $(x \cdot y \cdot z)|||(x \cdot y + z)$ and repeatedly compute the derivative w.r.t. the symbols x, y and z. First, we compute $d_x((x \cdot y \cdot z)|||(x \cdot y + z))$ $= (y \cdot z)|||y$ where for brevity we apply simplifications such as $\epsilon \cdot r = r$ etc. Next, we build $d_y((y \cdot z)|||y) = z|||\epsilon$ followed by $d_z(z|||\epsilon) = \epsilon|||\epsilon = \epsilon$. This derivation (wrongly) suggests that $x \cdot y \cdot z \in \mathcal{L}((x \cdot y \cdot z)|||(x \cdot y + z))$ because $d_{x \cdot y \cdot z}((x \cdot y \cdot z)|||(x \cdot y + z))$ is nullable.

The mistake in our calculation lies in the third derivative construction step $d_z(z|||\epsilon) = \epsilon|||\epsilon$. Based on the expression $z|||\epsilon$, the symbol z does not need to be synchronized because z only appears in the left sub-expression. However, this is reasoning step is wrong! Symbol z is part of the alphabets of the left and right sub-expressions in $(x \cdot y \cdot z)|||(x \cdot y + z)$. Hence, both sub-expressions (left and right) need to be synchronized. In terms of the derivative operation, we should have calculated $d_z(z)|||d_z(\epsilon)$ instead.

The problem is that this information about symbol z is only apparent in the initial expression and the subsequently computed expressions using the synchronous composition operator have lost this information. Hence, the above direct derivative method for synchronous composition is flawed. A similar observation applies to weak synchronous shuffling. Hence, to appropriately define derivatives for weak synchronous shuffling derivatives it is strictly necessary to enrich the expression language with general synchronous shuffling.

8. Related Work

To the best of our knowledge, there is almost no prior work which studies the notion of Brzozowski derivatives and Antimirov's partial derivatives in connection with shuffling operators. We are only aware of Lodaya and coworkers [13] whose work seems to imply a definition of derivatives for strongly synchronized shuffling [8]. Broda and coworkers [12] define partial derivatives for plain shuffling. Further work in the area studies the construction of automata for plain shuffling commonly referred to as asynchronous interleaving [9, 10]. In contrast, we provide detailed definitions how to obtain derivatives and partial derivatives including formal results for various shuffling operations [7, 1, 8]. Our results imply algorithms for constructing automata as shown in this paper. We believe they extend to checking equality and containment of regular expressions with shuffles in analogy to algorithms by Grabmayer [6] and Antimirov [5].

9. Conclusion

Thanks to a general form of synchronous shuffling introduced in this paper we can extend the notion of a Brzozowski derivative and a Antimirov partial derivative to various forms of shuffling operators which appear in the literature [8, 7, 1]. Our construction enables the application of algorithms based on derivatives for shuffle expressions such as automata-based word recognition algorithms [14] and equality/containment checking [5, 6].

References

[1] V. K. Garg, M. T. Ragunath, Concurrent regular expressions and their relationship to petri nets, Theor. Comput. Sci. 96 (2) (1992) 285–304.

[2] P. D. Stotts, W. Pugh, Parallel finite automata for modeling concurrent software systems, J. Syst. Softw. 27 (1) (1994) 27–43.

[3] J. A. Brzozowski, Derivatives of regular expressions, J. ACM 11 (4) (1964) 481–494.

[4] V. M. Antimirov, Partial derivatives of regular expressions and finite automaton constructions, Theoretical Computer Science 155 (2) (1996) 291–319.

[5] V. M. Antimirov, Rewriting regular inequalities, in: Proc. of FCT'95, Vol. 965 of LNCS, Springer-Verlag, 1995, pp. 116–125.

[6] C. Grabmayer, Using proofs by coinduction to find "traditional" proofs, in: Proc. of CALCO'05, Springer-Verlag, 2005, pp. 175–193.

[7] R. de Simone, Langages infinitaires et produit de mixage, Theor. Comput. Sci. 31 (1984) 83–100.

[8] M. H. ter Beek, C. Martín-Vide, V. Mitrana, Synchronized shuffles, Theor. Comput. Sci. 341 (1-3) (2005) 263–275.

[9] W. Gelade, Succinctness of regular expressions with interleaving, intersection and counting, Theor. Comput. Sci. 411 (31-33) (2010) 2987–2998.

[10] A. Kumar, A. K. Verma, A novel algorithm for the conversion of parallel regular expressions to non-deterministic finite automata, Applied Mathematics & Information Sciences 8 (2014) 95–105.

[11] M. Sulzmann, P. Thiemann, Derivatives for regular shuffle expressions, in: Proc. of LATA'15, Vol. 8977 of LNCS, Springer, 2015, pp. 275–286.

[12] S. Broda, A. Machiavelo, N. Moreira, R. Reis, Partial derivative automaton for regular expressions with shuffle, To appear in Proceedings of the 17th Int. Workshop on Descriptional Complexity of Formal Systems (DCFS15) (2015).

[13] K. Lodaya, M. Mukund, R. Phawade, Kleene theorems for product systems, in: Proc. of DCFS'11, Vol. 6808 of LNCS, Springer, 2011, pp. 235–247.

[14] S. Owens, J. Reppy, A. Turon, Regular-expression derivatives reexamined, Journal of Functional Programming 19 (2) (2009) 173–190.