ARTICLE IN PRESS
YINCO:4316

Information and Computation ••• (••••) •••–•••

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

On the complexity and decidability of some problems involving shuffle

Joey Eremondi [a,2], Oscar H. Ibarra [b,1], Ian McQuillan [c,*,2]

[a] *Department of Information and Computing Sciences, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands*
[b] *Department of Computer Science, University of California, Santa Barbara, CA 93106, USA*
[c] *Department of Computer Science, University of Saskatchewan, Saskatoon, SK S7N 5A9, Canada*

ARTICLE INFO

Article history:
Received 1 December 2015
Available online xxxx

Keywords:
Automata and logic
Shuffle
Counter machines
Pushdown machines
Reversal-bounds
Determinism
Commutativity
Strings

ABSTRACT

The complexity and decidability of various decision problems involving the shuffle operation (denoted by $\sqcup\!\sqcup$) are studied. The following three problems are all shown to be NP-complete: given a nondeterministic finite automaton (NFA) M, and two words u and v, is $L(M) \not\subseteq u \sqcup\!\sqcup v$, is $u \sqcup\!\sqcup v \not\subseteq L(M)$, and is $L(M) \neq u \sqcup\!\sqcup v$? It is also shown that there is a polynomial-time algorithm to determine, for NFAs M_1, M_2, and a deterministic pushdown automaton M_3, whether $L(M_1) \sqcup\!\sqcup L(M_2) \subseteq L(M_3)$. The same is true when M_1, M_2, M_3 are one-way nondeterministic l-reversal-bounded k-counter machines, with M_3 being deterministic. Other decidability and complexity results are presented for testing whether given languages L_1, L_2, and R from various languages families satisfy $L_1 \sqcup\!\sqcup L_2 \subseteq R$, and $R \subseteq L_1 \sqcup\!\sqcup L_2$. Several closure results on shuffle are also shown.

1. Introduction

The shuffle operator models the natural interleaving between strings. It was introduced by Ginsburg and Spanier [1], where it was shown that context-free languages are closed under shuffle with regular languages, but not context-free languages. It has since been applied in a number of areas such as concurrency [2], coding theory [3], verification [4], database schema [5], and biocomputing [3,6], and has also received considerable study in the area of formal languages. However, there remains a number of open questions, such as the long-standing problem as to whether it is decidable, given a regular language R to tell if R has a non-trivial decomposition; that is, $R = L_1 \sqcup\!\sqcup L_2$, for some L_1, L_2 that are not the language consisting of only the empty word [7].

This paper addresses several complexity-theoretic and decidability questions involving shuffle. In the past, similar questions have been studied by Ogden, Riddle, and Round [2], who showed that there exists deterministic context-free languages L_1, L_2 where $L_1 \sqcup\!\sqcup L_2$ is NP-complete. More recently, L. Kari studied problems involving solutions to language equations of the form $R = L_1 \sqcup\!\sqcup L_2$, where some of R, L_1, L_2 are given, and the goal is to determine a procedure, or determine that none exists, to solve for the variable(s) [8]. Also, there has been similar decidability problems investigated involving shuffle on trajectories [9], where the patterns of interleaving are restricting according to another language $T \subseteq \{0,1\}^*$ (a zero indicates

* Corresponding author.
E-mail addresses: j.s.eremondi@students.uu.nl (J. Eremondi), ibarra@cs.ucsb.edu (O.H. Ibarra), mcquillan@cs.usask.ca (I. McQuillan).
1 Supported, in part, by NSF Grant CCF-1117708.
2 Supported, in part, by Natural Sciences and Engineering Research Council of Canada Grant 327486-2010.

http://dx.doi.org/10.1016/j.ic.2017.09.002
0890-5401/

Please cite this article in press as: J. Eremondi et al., On the complexity and decidability of some problems involving shuffle, Inf. Comput. (2017), http://dx.doi.org/10.1016/j.ic.2017.09.002

that a letter from the first operand will be chosen next, and a one indicates a letter from the second operand is chosen). L. Kari and Sosík show that it is decidable, given L_1, L_2, R as regular languages with a regular trajectory set T, whether $R = L_1 \sqcup\!\sqcup_T L_2$ (the shuffle of L_1 and L_2 with trajectory set T). Furthermore, if L_1 is allowed to be context-free, then the problem becomes undecidable as long as, for every $n \in \mathbb{N}$, there is some word of T with more than n 0's (with a symmetric result if there is a context-free language on the right). This implies that it is undecidable whether $L_1 \sqcup\!\sqcup L_2 = R$, where R and one of L_1, L_2 are regular, and the other is context-free. In [10], it is demonstrated that given two linear context-free languages, it is not semi-decidable whether their shuffle is linear context-free, and given two deterministic context-free languages, it is not semi-decidable whether their shuffle is deterministic context-free. Complexity questions involving so-called *shuffle languages*, which are augmented from regular expressions by shuffle and iterated shuffle, have also been studied [11]. It has also been determined that it is NP-hard to determine if a given string is the shuffle of two identical strings (independently in [12] and [13]).

Recently, there have been several papers involving the shuffle of two words. It was shown that the shuffle of two words with at least two letters has a unique decomposition into the shuffle of words [14]. In fact, the shuffle of two words, each with at least two letters, has a unique decomposition over arbitrary sets of words [15]. Also, a polynomial-time algorithm has been developed that, given a deterministic finite automaton (DFA) M and two words u, v, can test if $u \sqcup\!\sqcup v \subseteq L(M)$ [16]. In the same work, an algorithm was presented that takes a DFA M as input and outputs a "candidate solution" u, v; this means, if $L(M)$ has a decomposition into the shuffle of two words, u and v must be those two unique words. But the algorithm cannot guarantee that $L(M)$ has a decomposition. This algorithm runs in $O(|u| + |v|)$ time, which is often far less than the size of the input DFA, as DFAs accepting the shuffle of two words can be exponentially larger than the words [17]. It has also been shown [18] that the following problem is NP-complete: given a DFA M and two words u, v, is it true that $L(M) \not\subseteq u \sqcup\!\sqcup v$?

In this paper, problems are investigated involving three given languages R, L_1, L_2, and the goal is to determine decidability and complexity of testing if $R \not\subseteq L_1 \sqcup\!\sqcup L_2$, $L_1 \sqcup\!\sqcup L_2 \not\subseteq R$, and $L_1 \sqcup\!\sqcup L_2 \neq R$, depending on the language families of L_1, L_2 and R. In Section 3, it is demonstrated that the following three problems are NP-complete: to determine, given an NFA M and two words u, v whether $u \sqcup\!\sqcup v \not\subseteq L(M)$ is true, $L(M) \not\subseteq u \sqcup\!\sqcup v$ is true, and $u \sqcup\!\sqcup v \neq L(M)$ is true. Then, the DFA algorithm from [16] that can output a "candidate solution" is extended to an algorithm on NFAs that operates in polynomial time, and outputs two words u, v such that if the NFA is decomposable into the shuffle of words, then $u \sqcup\!\sqcup v$ is the unique solution. And in Section 4, decidability and the complexity of testing if $L_1 \sqcup\!\sqcup L_2 \subseteq R$ is investigated involving more general language families. In particular, it is shown that it is decidable in polynomial time, given NFAs M_1, M_2 and a deterministic pushdown automaton M_3, whether $L(M_1) \sqcup\!\sqcup L(M_2) \subseteq L(M_3)$. The same is true given M_1, M_2 that are one-way nondeterministic l-reversal-bounded k-counter machines, and M_3 is a one-way deterministic l-reversal-bounded k-counter machine. However, if M_3 is a nondeterministic 1-counter machine that makes only one reversal on the counter, and M_1 and M_2 are fixed DFAs accepting a^* and b^* respectively, then the question is undecidable. Also, if we have fixed languages $L_1 = (a + b)^*$ and $L_2 = \{\lambda\}$, and M_3 is an NFA, then testing whether $L_1 \sqcup\!\sqcup L_2 \not\subseteq L(M_3)$ is PSPACE-complete. Also, testing whether $a^* \sqcup\!\sqcup \{\lambda\} \not\subseteq L$ is NP-complete for L accepted by an NFA. For finite languages L_1, L_2, and L_3 accepted by an NPDA, it is NP-complete to determine if $L_1 \sqcup\!\sqcup L_2 \not\subseteq L_3$. Results on unary languages are also provided. In Section 5, testing $R \subseteq L_1 \sqcup\!\sqcup L_2$ is addressed. This is already undecidable if R and L_1 are deterministic pushdown automata. However, it is decidable if L_1, L_2 are any commutative, semilinear languages, and R is a context-free language (even if augmented by reversal-bounded counters). Then, in Section 6, several other decision problems, and some closure properties of shuffle are investigated.

2. Preliminaries

We assume an introductory background in formal language theory and automata [19], as well as computational complexity [20]. We assume knowledge of pushdown automata, finite automata, and Turing machines, and we use notation from [19]. Let $\Sigma = \{a_1, \ldots, a_m\}$ be a finite alphabet. Then Σ^* (Σ^+) is the set of all (non-empty) words over Σ. A language over Σ is any $L \subseteq \Sigma^*$. Given a language $L \subseteq \Sigma^*$, the complement of L, $\overline{L} = \Sigma^* - L$. The length of a word $w \in \Sigma^*$ is $|w|$, and for $a \in \Sigma$, $|w|_a$ is the number of a's in w.

Let $\mathbb{N}$ be the positive integers, and $\mathbb{N}_0$ be the non-negative integers. For $n \in \mathbb{N}_0$, then define $\pi(n)$ to be 0 if $n = 0$, and 1 otherwise.

Next, we formally define reversal-bounded counter machines [21]. A *one-way k-counter machine* is a tuple $M = (k, Q, \Sigma, \lhd, \delta, q_0, F)$, where $Q, \Sigma, \lhd, q_0, F$ are respectively, the finite set of states, input alphabet, right input end-marker (not in Σ), the initial state, and the set of final states. The transition function δ is a relation from $Q \times (\Sigma \cup \{\lhd\}) \times \{0, 1\}^k$ into $Q \times \{\mathrm{S}, \mathrm{R}\} \times \{-1, 0, +1\}^k$, such that if $\delta(q, a, c_1, \ldots, c_k)$ contains $(p, d, d_1, \ldots, d_k)$ and $c_i = 0$ for some i, then $d_i \geq 0$ (this is to prevent negative values in any counter). The symbols S and R give the direction of the input tape head, being either *stay* or *right* respectively. Furthermore, M is deterministic if δ is a partial function. A configuration of M is a tuple $(q, w, c_1, \ldots, c_k)$ indicating that M is in state q with w (in Σ^* or $\Sigma^*\lhd$) as the remaining input, and $c_1, \ldots, c_k \in \mathbb{N}_0$ are the contents of the counters. The derivation relation $\vdash_M$ is defined by, $(q, aw, c_1, \ldots, c_k) \vdash_M (p, w', c_1 + d_1, \ldots, c_k + d_k)$, if $(p, d, d_1, \ldots, d_k) \in \delta(q, a, \pi(c_1), \ldots, \pi(c_k))$ where $d = \mathrm{S}$ implies $w' = aw$, and $d = \mathrm{R}$ implies $w' = w$. Then $\vdash_M^*$ is the reflexive, transitive closure of $\vdash_M$. A word $w \in \Sigma^*$ is accepted by M if $(q_0, w\lhd, 0, \ldots, 0) \vdash_M^* (q, \lhd, c_1, \ldots, c_k)$, for some $q \in F, c_1, \ldots, c_k \in \mathbb{N}_0$. The language accepted by M, $L(M)$, is the set of all words accepted by M. Essentially, a k-counter machine is a k-pushdown