INFORMATIQUE THÉORIQUE ET APPLICATIONS

A. SZEPIETOWSKI

There is no complete axiom system for shuffle expressions

Informatique théorique et applications, tome 33, n° 3 (1999), p. 271-277

<http://www.numdam.org/item?id=ITA_1999__33_3_271_0>

Theoretical Informatics and Applications
Theoret. Informatics Appl. **33** (1999) 271–277

THERE IS NO COMPLETE AXIOM SYSTEM FOR SHUFFLE EXPRESSIONS

A. SZEPIETOWSKI[1]

Abstract. In this paper we show that neither the set of all valid equations between shuffle expressions nor the set of schemas of valid equations is recursively enumerable. Thus, neither of the sets can be recursively generated by any axiom system.

AMS Subject Classification. 68Q45.

1. INTRODUCTION

Suppose that we have two shuffle expressions E and F. They are constructed from some symbols $\Sigma_r = \{\sigma_1, \ldots \sigma_r\}$ by regular operations, shuffle and shuffle closure operators. We can consider two problems. First, we can ask if these two expressions generate the same shuffle language over the alphabet Σ_r (we treat every symbol σ_i as a single letter). Second, we can consider symbols appearing in the expressions as variables and ask whether the equation $E = F$ is true for all instantiations of its variables by formal languages.

Thus we can consider two sets. First, the set of valid equations between shuffle expressions, where the symbols in the expressions represent single letters, and the two expressions describe the same shuffle language. Second, the set of schemas or identities of the form $E = F$, where symbols represent variables, and the identity is valid if it is true for all instantiations of its variables by formal languages.

It is a well known fact that for regular expressions these two sets are equivalent, because if two regular expressions generate the same language, then they are equal under all instantiations of its variables by formal languages. This is not true for expressions containing the shuffle operator. For example, if a , b stand for single letters, the expressions $a \odot b$ and $ab + ba$ describe the same language (namely, $\{ab, ba\}$). However, the identity $a \odot b = ab + ba$ is not true under some instantiation (indeed, instantiate a by $\{c\}$ and b by $\{de\}$).

[1] Mathematical Institute, University of Gdańsk, ul. Wita Stwosza 57, 80-952 Gdańsk, Poland; e-mail: matszp@halina.univ.gda.pl

The natural question is whether there are some ways of characterizing the set of valid equations between shuffle expressions or the set of schemas of valid equations.

Salomaa [8] presented a consistent and complete axiom system for the set of equations between regular expressions. His system consists of a recursive set of axioms (equations) and a finite set of computable rules. A proof of an equation $X = Y$ is a finite sequence of equations such that each of them is either an axiom or may be derived by the rules from equations occurring earlier in the sequence; and the equation $X = Y$ is the last equation in the sequence. Consistency of the system means that there are proofs only for valid equations and completeness means that every valid equation has its proof. In an axiom system like this the set of proofs is recursive and the set of proven equations is recursively enumerable. Another axiom system for equations between regular expressions was presented by Krob in [6]. Kimura [5] proposed an axiom system for shuffle expressions and asked whether his system is consistent and complete.

Schemas of valid equations between shuffle expressions were discussed in [1, 2], and [7]. Meyer and Rabinovitch [7] proved that the set of schemas constructed from variables by the regular operations and the shuffle operator (without shuffle closure) is decidable. More precisely, they showed that following problem is decidable: given are expressions E and F constructed from variables by the regular operations and the shuffle operator. Is the identity $E = F$ true for all instantiation of its variables by formal languages. Blum and Ésik [1] proved that variety **Lang** generated by all language structures $(P_\Sigma, \cdot, \odot, +, O, 1)$ is not finitely axiomatizable (where P_Σ consists of all subset of Σ^*, $\cdot$, $\odot$, and $+$ are concatenation, shuffle, and sum operators, 0 is the empty set, and 1 is the singleton set containing the empty word).

In this paper we show that neither the set of all valid equations between shuffle expressions nor the set of schemas of valid equations is recursively enumerable. Thus, neither of the sets can be recursively generated by any axiom system.

In Section 3 we show that the set of all valid equations between shuffle expressions is not recursively enumerable. We shall show this by proving that the set of all pairs of shuffle expressions which are not equivalent (*i.e.* they do not generate the same language) is recursively enumerable. Thus the set of all pairs which are equivalent cannot be recursively enumerable, because otherwise it would have been recursive contrary to the fact proved by Iwama [4] that the universe problem for shuffle expressions (the problem whether a shuffle expression generates the whole universe Σ^*) is undecidable.

In Section 4 we show that the set of schemas of valid equations between shuffle expressions is not recursively enumerable. First we show that the set is not recursive, and then that the set of schemas which are not valid under some substitution is recursively enumerable.

2. SHUFFLE EXPRESSIONS

Let Σ be any fixed alphabet and λ the empty word. By $u \cdot v$ we denote the concatenation of two words u and v. We shall also use the notation $\prod_{i\in I}\sigma_i$, to denote the concatenation $\prod_{i\in I}\sigma_i = \sigma_{i_1}\sigma_{i_2}\ldots\sigma_{i_s}$, where $I = \{i_1, i_2 \ldots i_s\}$ and $i_k < i_{k+1}$ for every $1 \leq k \leq s-1$.

The shuffle operation $\odot$ is defined inductively as follows:

- $u \odot \lambda = \lambda \odot u = \{u\}$, for $u \in \Sigma^*$ and
- $au \odot bv = a(u \odot bv) \cup b(au \odot v)$, for $u, v \in \Sigma^*$ and $a, b \in \Sigma$.

Note that $u \odot v$ consist of all words $z \in \Sigma^*$ which can be decomposed into $z = w_1 \cdot w_2 \cdot \cdots \cdot w_r$ with $w_i \in \Sigma^*$, $u = \prod_{i\in I} w_i$ and $v = \prod_{i\notin I} w_i$, for some subset $I \subset \{1, 2, \ldots, r\}$.

For any languages $L_1, L_2 \subset \Sigma^*$ the shuffle $L_1 \odot L_2$ is defined as

$$L_1 \odot L_2 = \bigcup_{u\in L_1, v\in L_2} u \odot v.$$

For any language L, the shuffle closure operator is defined by:

$$L^{\otimes} = \bigcup_{i=0}^{\infty} L^{\odot i}, \quad \text{where} \quad L^{\odot 0} = \{\lambda\} \quad \text{and} \quad L^{\odot i} = L^{\odot i-1} \odot L.$$

Definition 1. Each $a \in \Sigma$, λ and $\emptyset$ are shuffle expressions. If S_1, S_2 are shuffle expressions, then $(S_1 \cdot S_2)$, ${S_1}^*$, $(S_1 + S_2)$, $(S_1 \odot S_2)$ and ${S_1}^{\otimes}$ are shuffle expressions, and nothing else is a shuffle expression.

The language $L(S)$ generated by a shuffle expression S is defined as follows. $L(a) = \{a\}$, $L(\lambda) = \{\lambda\}$, $L(\emptyset) = \emptyset$. If $L(S_1) = L_1$ and $L(S_2) = L_2$, then $L((S_1 \cdot S_2)) = L_1 \cdot L_2$, $L((S_1 + S_2)) = L_1 \cup L_2$, $L({S_1}^*) = L_1^*$, $L((S_1 \odot S_2)) = L_1 \odot L_2$, and $L({S_1}^{\otimes}) = L_1^{\otimes}$.

3. EQUATIONS BETWEEN SHUFFLE EXPRESSIONS

Consider the alphabet $\Sigma_r = \{\sigma_1, \ldots \sigma_r\}$. We denote by EQ_r the set of all valid equations between shuffle expressions over the alphabet Σ_r. Thus, the equation $X = Y$ belongs to EQ_r if and only if X and Y are shuffle expressions over Σ_r generating the same language, *i.e.* $L(X) = L(Y)$. It is obvious that

$$EQ_1 \subsetneq EQ_2 \subsetneq \ldots.$$

The union of all sets EQ_r is denoted by EQ_ω.

Let $INEQ_r$ denote the set of all inequalities between shuffle expressions over Σ_r. Thus $X \neq Y$ belongs to $INEQ_r$ if and only if X and Y are shuffle expressions over Σ_r and they generate different languages, *i.e.* $L(X) \neq L(Y)$. First we prove.

Lemma 2. *For every r, the set $INEQ_r$ is recursively enumerable.*

Proof. In order to prove the lemma we shall present a Turing machine M_1 which accepts the set $INEQ_r$. For an input of the form $X \neq Y$, the machine M_1 checks one by one, for every word $w \in \Sigma_r^*$, whether $w \in L(X)$ and $w \in L(Y)$. This is possible, because the word problem for shuffle expressions is decidable, *i.e.* there exists a Turing machine M' which for every word $w \in \Sigma_r^*$ and shuffle expression X decides whether $w \in L(X)$. The machine M' constructs an LBA (linear bounded automaton) A_X, such that $L(X) = L(A_X)$ and then checks if A_X accepts w.

The Turing machine M_1 stops and accepts if it finds a word w such that $w \in L(X)$ and $w \notin L(Y)$ or $w \notin L(X)$ and $w \in L(Y)$. □

Now we shall show that EQ_2 is not recursively enumerable. We shall use an argument similar to the one used by Post in his theorem (see [3], Th. 8.3). Suppose, for a contradiction that EQ_2 is recursively enumerable and that there exists a Turing machine M_2 accepting EQ_2. Now we can construct a Turing machine M which recognizes if an equation $X = Y$ belongs to EQ_2, or not. The machine M simulates in parallel two machines: M_2 on the input $X = Y$ and M_1 (described in the proof of Lem. 2) on the input $X \neq Y$, and answers "Yes" if M_2 accepts, and "No" if M_1 accepts. On the other hand Iwama [4] proved that the universe problem for shuffle expressions (even over binary alphabet) is undecidable. In other words, it is undecidable whether an expression over Σ_2 generates Σ_2^*. Hence the set EQ_2 is not recursive, a contradiction. Thus, we have proven the following theorem.

Theorem 3. *The set EQ_2, of all valid equations between shuffle expressions over Σ_2, is not recursively enumerable.*

Corollary 4. *The set EQ_ω and EQ_r, for every $r \geq 2$, are not recursively enumerable.*

Proof. Suppose that there exists a Turing machine M_3 accepting EQ_ω or EQ_r, for some $r > 2$. We can construct a Turing machine M_4 which accepts EQ_2. For an input of the form $X = Y$, M_4 first checks if X and Y are shuffle expressions over Σ_2 and then simulates M_3. □

Note that the set EQ_1 is decidable, because shuffle expressions over one letter alphabet are equivalent to regular expressions (shuffle operator is then equivalent to concatenation and the shuffle closure to the Kleene star).

From Theorem 3 it follows that there is no consistent and complete axiom system generating all valid equations between shuffle expressions in a sense proposed by Salomaa [8] for regular expressions.

4. Schemas of valid equations

We denote by VS_r the set of valid schemas between shuffle expressions. More specifically, VS_r consists of equations of the form $E = F$, where E and F are shuffle expressions over the alphabet $\Sigma_r = \{\sigma_1, \ldots \sigma_r\}$ such that always a valid

equation results whenever each letter of Σ_r appearing in E or F is substituted by some language. The union of all sets VS_r is denoted by VS_ω.

First we show that the set VS_2 is not decidable (recursive). Suppose that $\Sigma = \{a, b\}$. In this chapter we shall rather use the notation $E(a, b)$ instead of E to denote a shuffle expression over symbols $\{a, b\}$. Then $L(E(a, b))$ will denote the language over $\{a, b\}$ generated by the expression $E(a, b)$. By $E(L_a, L_b)$ we shall denote the language obtained by substitution of a and b by languages L_a and L_b. We shall also use this notation if $E(a, b)$ is a single word $w(a, b)$. For example, for $w(a, b) = aaba$, $w(L_a, L_b) = L_a \cdot L_a \cdot L_b \cdot L_a$.

Theorem 5. *The set VS_2 is not recursive.*

Proof. We shall reduce the universe problem for shuffle expressions over $\Sigma = \{a, b\}$ to VS_2. The theorem will then follow from the fact that the universe problem is not recursive [4].

Let $E(a, b)$ be any shuffle expression over $\{a, b\}$. We shall show that $L(E(a, b)) = (a+b)^*$ if and only if the inclusion $(a+b)^* \subset E(a, b)$ or equation $(a+b)^* + E(a, b) = E(a, b)$ is valid for all instantiations of a and b by formal languages L_a and L_b.

If $L(E(a, b)) \neq (a + b)^*$ then the scheme $(a + b)^* + E(a, b) = E(a, b)$ is not valid under substitution $L_a = \{a\}$ and $L_b = \{b\}$.

It is obvious that every word $z \in (L_a + L_b)^*$ belongs to $w(L_a, L_b)$ for some $w(a, b) \in (a + b)^*$. Thus, it is enough to show that if a word $w(a, b)$ belongs to $L(E(a, b))$ then the language $w(L_a, L_b)$ is contained in $E(L_a, L_b)$. We shall prove this by induction on the structure of $E(a, b)$.

The claim is obvious for $E(a, b) = \emptyset, \lambda, a$ or b. Consider now the case when $E(a, b) = G(a, b) \odot H(a, b)$ and suppose that $w(a, b) = \sigma_1\sigma_2 \ldots \sigma_r$, where each $\sigma_i \in \{a, b\}$. Then there is a subset of the set of indexes $I \subset \{1, 2, \ldots, r\}$ such that $u(a, b) = \prod_{i \in I} \sigma_i \in G(a, b)$ and $v(a, b) = \prod_{i \notin I} \sigma_i \in H(a, b)$.

Now $w(L_a, L_b) = L_1 \cdot L_2 \cdot \cdots \cdot L_r$ with $L_i = L_a$, if $\sigma_i = a$, and $L_i = L_b$, if $\sigma_i = b$, and every word $z \in w(L_a, L_b)$ can be decomposed into $z = w_1 \cdot w_2 \cdot \cdots \cdot w_r$ with $w_i \in L_i$, and if we take $x = \prod_{i \in I} w_i$ and $y = \prod_{i \notin I} w_i$, then we have $z \in x \odot y$, $x \in u(L_a, L_b)$ and $y \in v(L_a, L_b)$. By induction, $u(L_a, L_b) \subset G(L_a, L_b)$ and $v(L_a, L_b) \subset H(L_a, L_b)$. Thus, $z \in G(L_a, L_b) \odot H(L_a, L_b) = E(L_a, L_b)$.

If $E(a, b)$ is of the form $E(a, b) = (G(a, b))^{\otimes}$ and $w(a, b) \in E(a, b)$ then $w(a, b) \in (G(a, b))^{\odot i}$ for some i and by the fact just proved above $w(L_a, L_b) \subset (G(L_a, L_b))^{\odot i} \subset (G(L_a, L_b))^{\otimes}$.

The cases when $E = G + H$, $E = G \cdot H$, or $E = G^*$ can be proved similarly. □

In much the same way as in the proof of Corollary 4. one can show the following.

Corollary 6. *The set VS_ω and VS_r, for every $r \geq 2$, are not recursive.*

Now we shall show that the set VS_2 of valid schemas with two variables is not recursively enumerable. By a similar argument as in Section 3, it is enough to prove that the set of nonvalid schemas is recursively enumerable. The set of nonvalid schemas with two variables, denoted by NVS_2, consists of all inequalities $E \neq F$ where E and F are shuffle expressions over Σ_2 which are not equal under

some substitution of a and b by languages L_a and L_b. But first we prove the following:

Lemma 7. *If a scheme $E(a,b) = F(a,b)$ is equal under every instantiation of a, b by finite languages, then it is also equal under every instantiation by any formal languages.*

Proof. Suppose that there exist two languages L_a and L_b such that $E(L_a, L_b) \neq F(L_a, L_b)$. We shall show that there exist finite languages $X_a \subset L_a$ and $X_b \subset L_b$ such that $E(X_a, X_b) \neq F(X_a, X_b)$. Without loss of generality we can assume that there is a word $z \in E(L_a, L_b)$ and $z \notin F(L_a, L_b)$. One can easily show, by induction on the structure of the expression $E(a,b)$, that if a word $z \in E(L_a, L_b)$ then there exist finite languages $X_a \subset L_a$ and $X_b \subset L_b$ such that $z \in E(X_a, X_b)$. We omit details of the proof. We only prove for $E = G \odot H$ and $E = G^{\otimes}$. In the first case there exist two words x and y such that $z \in x \odot y$, $x \in G(L_a, L_b)$ and $y \in H(L_a, L_b)$. By induction, there exist finite sets X_a, X_b, Y_a, and Y_b such that $x \in G(X_a, X_b)$ and $y \in H(Y_a, Y_b)$. And we have $z \in E(X_a \cup Y_a, X_b \cup Y_b)$. If $E = G^{\otimes}$ and $z \in E(L_a, L_b)$ then there exist i such that $z \in (G(L_a, L_b))^{\odot i}$ and using the fact just proved there exist finite sets X_a and X_b, such that $z \in (G(X_a, X_b))^{\odot i} \subset (G(X_a, X_b))^{\otimes}$.

Note that $z \notin F(X_a, X_b)$ because $z \notin F(L_a, L_b)$ and $F(X_a, X_b) \subset F(L_a, L_b)$. □

Theorem 8. *The set of schemes VS_2 is not recursively enumerable.*

Proof. We show that the set NVS_2 of nonvalid schemas is accepted by some Turing machine M. The machine M on input $E \neq F$ looks for a pair of finite languages L_a and L_b (represented here as regular expressions) for which the two shuffle expressions $E(L_a, L_b)$ and $F(L_a, L_b)$ are not equal. To find that two shuffle expressions $E(L_a, L_b)$ and $F(L_a, L_b)$ are not equal machine M uses the Turing machine M_1, described in the proof of Lemma 2, which accepts the set of inequalities between shuffle expressions. Note that M_1 does not stop on inputs that are not accepted. For this reason the machine M works in stages. In the first stage M simulates the first step of M_1 on the first pair. In the i-th stage M simulates first i steps of M_1 on the first i pairs of finite languages. M stops if it finds that $E(L_a, L_b) \neq F(L_a, L_b)$ for some pair L_a, L_b.

By Lemma 7, if the expressions E and F are not equal under some instantiation of a and b by languages L_a, L_b, then they are not equal under some instantiation by some finite languages, and the machine M accepts the input $E \neq F$. If E and F are equal under all instantiations, then M does not accept the input $E \neq F$. □

References

[1] S.L. Bloom and Z. Ésik, Axiomatizing shuffle and concatenation in languages. *Inform. and Comput.* **139** (1997) 62–91.

[2] S.L. Bloom and Z. Ésik, Shuffle binoids. *Theor. Informatics Appl.* **32** (1998) 175–198.

[3] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley (1979).

[4] K. Iwama, The universe problem for unrestricted flow languages. *Acta Inform.* **19** (1983) 85–96.
[5] T. Kimura, *An algebraic systems for process structuring and interprocess communication*, Proc. 8 Annual Symposium on Theory of Computing (1976) 92–100.
[6] D. Krob, Complete systems of B-rational identities. *Theoret. Comput. Sci.* **89** (1991) 207–343.
[7] A.R. Meyer and A. Rabinovich, *A solution of an interleaving decision problem by a partial order techniques*, Proc. Workshop on Partial-order Methods in Verification, July 1996, Princeton NJ, Ed. G. Holzmann, D. Peled, V. Pratt, AMS-DIMACS Series in Discrete Math.
[8] A. Salomaa, *Theory of Automata*, Pergamon Press, Oxford (1969).

Communicated by Ch. Choffrut.
Received January, 1999. Accepted March, 1999.