Theoretical Computer Science 48 (1986) 329-337
North-Holland

NOTE

ON THE DECIDABILITY OF SOME PROBLEMS ABOUT RATIONAL SUBSETS OF FREE PARTIALLY COMMUTATIVE MONOIDS

Alan GIBBONS and Wojciech RYTTER *
Department of Computer Science, University of Warwick, Coventry CV4 7AL, U.K.

Communicated by D. Perrin
Received August 1986
Revised November 1986

Abstract. Let $I = A \cup B$ be a partially commutative alphabet such that two letters commute iff one of them belongs to A and the other one belongs to B. Let $M = A^* \times B^*$ denote the free partially commutative monoid generated by I. We consider the following six problems for rational (given by regular expressions) subsets X, Y of M:

(Q1): $X \cap Y = \emptyset$?
(Q2): $X \subseteq Y$?
(Q3): $X = Y$?
(Q4): $X = M$?
(Q5): $M - X$ finite?
(Q6): X is recognizable?

It is known (see (Berstel, 1979)) that all these problems are undecidable if Card $A > 1$ and Card $B > 1$, and they are decidable if Card A = Card $B = 1$ (Card U denotes the cardinality of U).

It was conjectured (see (Choffrut, 1986, p. 79)) that these problems are decidable in the remaining cases, where Card $A = 1$ and Card $B > 1$. In this paper we show that if Card $A = 1$ and Card $B > 1$, then the problem (Q1) is decidable, and problems (Q2)-(Q6) are undecidable. Our paper is an application of results concerning reversal-bounded, nondeterministic, multicounter machines and nondeterministic, general sequential machines.

1. Introduction

Languages over partially commutative alphabets are generalizations of classical formal languages. A partially commutative alphabet (called also a concurrent alphabet) is a pair (I, C), where I is a finite set of symbols and C is a symmetric irreflexive relation on I. The symbols of I can represent processes (see [7, 8]) and the relation C then represents which of these processes can be executed independently (C is also called the concurrency relation).

* On leave from Institute of Informatics, Warsaw University, Poland.

Two strings v and w are said to be equivalent (with respect to C) if v can be obtained from w by several applications of the operation of commuting certain two adjacent symbols a, b such that $(a, b) \in C$. We write in this case $v \approx_C w$ (we shall omit later the subscript C). In this paper we consider only one relation, namely $C = A \times B$, where $B = \{a, b\}$, $A = \{1\}$, and $I = A \cup B$ is a partition of the alphabet I. For example, in this case, $a1b11aa \approx 111abaa$.

The free partially commutative monoid (fpcm, for short) over I is the set M of equivalence classes of the relation $\approx_C$. These equivalence classes were called traces in [2, 7, 8, 9] and subsets of an fpcm M were called trace languages in [2]. Classical formal languages are languages over alphabets in which no symbols commute. Any classical language L over the alphabet I has a corresponding trace language, by taking all traces containing at least one element of L. In this sense rational subsets of M (rational trace languages) correspond to classical regular languages L. In problems (Q1)-(Q6) the sets X, Y are given by regular expressions describing some classical regular languages $X1$, $Y1$. It is technically simpler to deal with sets of strings instead of sets X, Y of equivalence classes of strings. Hence, instead of considering subsets of fpcm's (trace languages) we consider in this paper their classical language versions. This will not affect complexity of the problems (Q1)-(Q6), but it will help considerably to apply some automata theoretic results related to classical formal languages. To this end we introduce the operation CL. Let L be a classical language over the alphabet I, by CL(L) we denote the set

$$\{w : w \approx v \text{ for some } v \in L\},$$

CL is called the closure operation:

The notion of regular flat languages was introduced in [9]. The language L is called a regular flat language (rfl, for short) iff $L = \mathrm{CL}(L1)$ for some regular language $L1$. The class of rfl's is not very regular. It was proved in [9] that this class is an anti-AFL if the concurrency relation is not fixed (because this class is closed under none of the following six operations: union, concatenation, Kleene's closure *, homomorphism, inverse homomorphism, and intersection with regular sets). Notice that an rfl X can be a nonregular language, for example $\mathrm{CL}((a1)^*)$ is the set of all strings containing the same number of a's and 1's provided the symbols a and 1 commute. The problem of recognizability of the rational subset of an fpcm corresponds to the problem of regularity of an rfl (see [3, p. 67]). We can redefine the notion of recognizability of a subset of M as follows: the set L of traces is a recognizable set iff its corresponding language (the union of all equivalence classes belonging to L) is regular.

Problems (Q1)-(Q6) now correspond to problems for rfl's. We can replace X, Y by CL($X1$), CL($Y1$), respectively, where $X1$, $Y1$ are classical regular languages. These problems can now be reformulated as follows:

(Q1): $\mathrm{CL}(X1) \cap \mathrm{CL}(Y1) = \emptyset$?
(Q2): $\mathrm{CL}(X1) \subseteq \mathrm{CL}(Y1)$?
(Q3): $\mathrm{CL}(X1) = \mathrm{CL}(Y1)$?

(Q4): $\mathrm{CL}(X1) = I^*$?
(Q5): $I^* - \mathrm{CL}(X1)$ finite?
(Q6): $\mathrm{CL}(X1)$ regular?

In what follows we refer to questions (Q1)-(Q6) in this format.

2. Automata-theoretic characterizations of regular flat languages

We fix $I = \{a, b, 1\}$, the symbol 1 commutes with a, b. Our first characterization of rfl's over the alphabet I is in terms of nondeterministic reversal-bounded counter machines (nrbm's, for short). An nrbm A is a device with finite-state control, a two-way read-only head which reads symbols from the input tape delimited by endmarkers and one counter capable of storing any integer. Initially, A is in some specified state and the counter is set to zero. One step of the machine consists of moving the input head and changing the counter by $-1, 0$ or 1. The input head cannot travel beyond the endmarkers. The input is accepted if A can reach one of the specified accepting states. The number of reversals of the input head and the number of reversals of the counter is bounded by a constant. A reversal of the input head refers to a change of its direction, whilst a reversal of the counter refers to changing from an increasing mode to a decreasing mode. For the purposes of this paper we restrict ourselves to machines in which the input head goes from the left to the right, next goes back to the left endmarker and scans the text again and (this time) stops at the right endmarker. The counter makes only one reversal. We refer the reader to [5] for the formal (and more general) definition of nrbm's. Let $\mathrm{Lan}(A)$ denote the language accepted by an nrbm A.

Lemma 1. *If L is a regular language over I, then we can effectively construct an nrbm A' such that* $\mathrm{CL}(L) = \mathrm{Lan}(A)$.

Proof. We describe informally how the automaton A' works on the input text w. Let A be a deterministic finite automaton accepting the language L. Let w be the input string. The machine A' guesses (letter by letter) the string $v \in L$ such that $w \approx v$, if there is such a string. The strings w and v differ only in the order of occurrences of symbols 1, they are the same if 1's are disregarded. In the first pass from left endmarker to the right endmarker, A' ignores 1's occurring in w (does not change the current state).

In phase 1 A' simulates the automaton A on the guessed input string v. A' guesses one letter of v in one move (on-line); if a guessed letter is 1, then A' increments its counter by one (operation P), otherwise it checks whether this letter matches the next letter in w which is different from 1.

Let us consider the example of A presented in Fig. 1. Assume that $q0$ is the initial state and that $q2$ is the accepting state of A. Let $w = ab11ab11$. In the first phase the only relevant part of w to A' is $w' = abab$. The automaton A' guesses the string

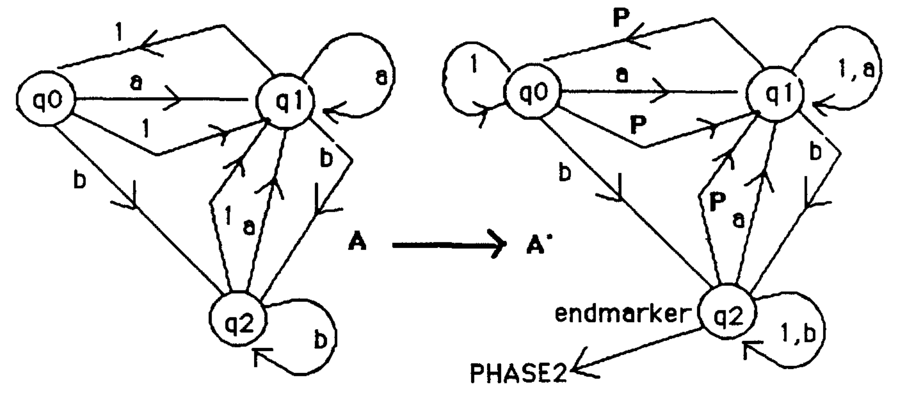

Fig. 1. An example of a finite automaton A and the diagram of the first phase of the nrbm A' corresponding to A. $P \approx$ (counter := counter + 1). Observe that A is here deterministic and A' is not.

$v = a1ba111b$, which it does in the following way. Starting in state $q0$, A' reads a and goes to $q1$, it nondeterministically chooses the arrow corresponding to the operation P, and goes to state $q0$ without scanning the input string. Then it reads b and goes to $q2$. Now it reads two symbols 1 and ignores them by remaining in the same state $q2$. Next it reads the symbol a and goes to $q1$. Then A' decides to make three operations P, after that, it terminates in state $q0$. It reads the symbol b, goes to $q2$ and then ignores the last two symbols of w which are 11. The endmarker has been reached. The automaton is in state $q2$ which is an accepting state of A. Then A' proceeds to the second phase. At this moment the value of the counter is 4 because A' has decided four times to use arrows corresponding to the operation P.

In this way in the first phase A' verifies whether $h(w) = h(v)$, where h is a homomorphism erasing 1's. If the guessed string v is not accepted by A, then A' rejects. Otherwise, A' passes to phase 2. It moves its input head to the left endmarker and in the next sweep from left to right verifies whether the number of 1's in w is equal to the value of the counter (by decreasing the counter by one whenever the next 1 is encountered, and checking at the right endmarker whether the counter is zero). If this is so, then A' accepts. In this way A' accepts w iff there exists a string $v \in L$ such that $w \approx v$.

Let us continue with our example. A' starts phase 2. It goes to the left position of w and in one left-to-right sweep verifies whether the number of 1's in w equals 4. Four times the counter is decreased by one. Finally, A' accepts because the final value of the counter is zero.

Clearly A' is an nrbm. This completes the proof. □

The next useful device is a nondeterministic, generalized sequential machine (ngsm, for short). Ngsm is a generalization of Mealy's sequential machines, which are finite automata with output. The automaton in one move, depending on the current state and scanned input symbol, changes its state and outputs a symbol. Mealy's machines are deterministic and output one symbol per one input symbol. The ngsm works in a similar way, however, now the machine can choose in each step one action from a specified set of alternative actions. The action consists of changing the state and printing an output string. The machine can now produce many symbols per one input symbol. Another difference is that an ngsm has a specified set of accepting states. Ngsm A determines a relation $R(A)$, which we call the input-output relation described by A. A pair (v, w) is an element of $R(A)$ iff v is a string of input symbols and w is one of the possible resulting output strings. In other words, A starting in the initial state after reading the input string v can (nondeterministically) produce the output string w and simultaneously end in an accepting state. We refer the reader to [6] for a more formal description of ngsm's and their input-output relations. The equivalence problem for ngsm's is the problem of determining, for each two given ngsm's, whether they define the same input-output relations.

This problem is solvable for deterministic generalized sequential machines and also for nondeterministic machines which produce one output symbol per one input symbol. However, the problem is undecidable for general ngsm's, even if the input alphabet is two-element and the output alphabet is unary, see [6]. This will be our main tool for proving undecidability of problems (Q2)–(Q6). First we establish a correspondence between ngsm's and rfl's. Assume that the input alphabet of all considered ngsm's is $\{a, b\}$ and the output alphabet is $\{1\}$. The relation $\approx$ is induced by commutativity between these alphabets. For each ngsm A we define the language

$$H(A)=\{x\in I^*: x\approx vw \text{ for some } (v, w)\in R(A)\}.$$

Observe the following fact.

Fact. *Ngsm's A and B are input-output equivalent iff $H(A)=H(B)$.*

The next lemma shows that each $H(A)$ is an rfl. It is based on a relation between transducers and corresponding languages. Our transducers are ngsm's. We refer the reader to [1] for related results about transducers.

Lemma 2. *For every ngsm A we can effectively find a regular expression describing a regular language L such that $H(A)=\mathrm{CL}(L)$.*

Proof. Let A be an ngsm with the set of states Q. Consider the possible output actions performed by A if it reads the input symbol a and goes from state $s1$ to state $s2$. The machine can nondeterministically print one of the output strings out_1, $\mathrm{out}_2, \ldots, \mathrm{out}_k$. We denote this set of possible outputs, which correspond to the transition $s1 \to^{a} s2$, by $\mathrm{OUT}(s1, s2, a)$. Define the substitution h from pairs of states of A into sets of strings. Each such string is the concatenation of some input symbol a followed by one possible corresponding output string.

$$h((s1, s2))=\{ax: x\in \mathrm{OUT}(s1, s2, a),\ a \text{ is an input symbol}\}.$$

Let $L1$ be the set of all sequences of edges in the diagram of A leading from the initial state to an accepting state (each edge is of the form $(s1, s2)$ and it is treated as one symbol). Clearly, $L1$ is a regular language and can be described by a regular expression. The diagram of A can be treated as a finite automaton (recognizer) whose input alphabet is the set of edges. The required language is $L = h(L1)$. This language is a regular language because each language $h(s1, s2)$ is a finite set (hence, regular) and the class of regular languages is closed under the operation of substitution. Each string from L can be partitioned into two disjoint subsequences forming strings u and w, where u is a string over the input alphabet and w is a string over output alphabet; w is a possible output of A corresponding to the input string u, and each letter of w commutes with each letter of u. Clearly, $H(A) = \mathrm{CL}(L)$. This completes the proof. □

Example. Let A be the ngsm presented in Fig. 2. The label a/x means that A reads a and outputs the string x when going to a specified state.

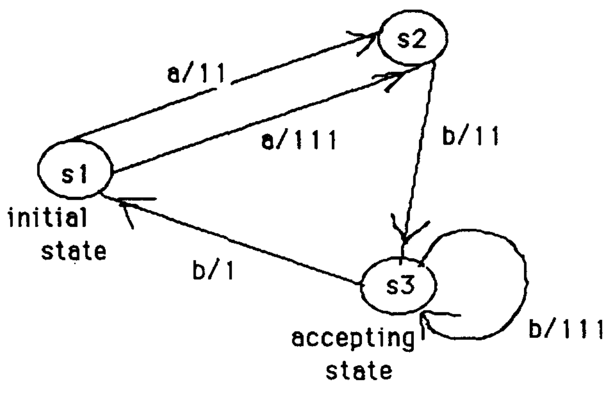

Fig. 2.

Observe that, in the state $s1$, A can read a and go to $s2$ producing either 11 or 111 as the output. Hence, $\mathrm{OUT}(s1, s2, a) = \{11, 111\}$. We have $h((s1, s2)) = \{a11, a111\}$, $h((s2, s3)) = \{b11\}$, $h((s3, s3)) = \{b111\}$, and $h((s3, s1)) = \{b1\}$. In this case

$$L1 = (s1, s2)(s2, s3)((s3, s3)^*(s3, s1)(s1, s2)(s2, s3))^*(s3, s3)^*$$

and

$$L = (a11 \cup a111)^* b11((b111)^* b1\{a11 \cup a111\} b11)^*(b111)^*.$$

Observe that in this case the language $\mathrm{CL}(L) = H(A)$ is not regular.

The history of a computation of a Turing machine on a given input can be represented by a sequence of configurations and encoded as a string over the alphabet $\{a, b\}$. We refer the reader to [6] for details. The proof of undecidability for the equivalence problem for ngsm's with unary output alphabet (see [6]) involves the reduction of the halting problem of a Turing machine to the equivalence problem for ngsm's. We can assume that the Turing machine loops when it is in an accepting state and instead of the halting problem, the existence of an accepting computation can be considered. Let Acc(T) be the set of all strings over the alphabet $\{a, b\}$ encoding the histories of accepting computations of T with an initially blank tape. Notice that Acc(T) is now empty or infinite. The following lemma was implicitly proved in [6] as a side effect of proving the undecidability of the equivalence problem for ngsm's with unary output alphabet. See the proof of [6, Theorem 1].

Lemma 3. *Let T be a single-tape Turing machine with an initially blank tape. We can effectively construct an ngsm A with input alphabet $\{a, b\}$ and output alphabet $\{1\}$ such that, for each input string x of length n, $(x, 1^{2n})$ is not an element of $R(A)$ iff $x \in \mathrm{Acc}(T)$.*

Remark. A characterization of rfl's in terms of two-way multihead pushdown automata was given in [9]. Every rfl is accepted by a deterministic multihead pushdown automaton and every context-free flat language is accepted by a nondeterministic multihead pushdown automaton. In both cases the number of heads depends on the relation C.

3. Applications of automata-theoretic characterizations

We are making use of results concerning nrbm's and ngsm's; however, observe that, essentially, we are using the power of nondeterminism. In the case of nrbm's nondeterminism is used to demonstrate the existence of algorithms for some complicated problems, and in the case of ngsm's nondeterminism is used to show the undecidability of (seemingly) simple problems.

The commutative alphabet is fixed and is $I = \{a, b, 1\}$, where 1 commutes with a and b.

Theorem 1. *Problem* (Q1) *is decidable.*

Proof. Let $X1$, $Y1$ be two regular languages and $X = \mathrm{CL}(X1)$, $Y = \mathrm{CL}(Y1)$. It follows from Lemma 1 that nrbm's $A1, A2$ can be effectively found such that $\mathrm{Lan}(A1) = X$, $\mathrm{Lan}(A2) = Y$. The disjointness of X and Y is now equivalent to the disjointness problem for $A1, A2$. This problem has been proved to be decidable for nrbm's (see [5, Theorem 3.1]). This completes the proof. □

Theorem 2. *Problems* (Q2)–(Q6) *are undecidable.*

Proof. Assume that the Turing machines considered below loop in the accepting state. Now, consider the following problem:

(Q0): for a given single-tape Turing machine T with an initially black tape, decide whether Acc(T) is empty.

(Observe that Acc(T) is empty iff it is finite.) We reduce (Q0) to each of the problems (Q2)-(Q6). We construct regular languages

$$Z1 = ((a11 \cup b11)^*1)^* \quad \text{and} \quad Z2 = ((a11 \cup b11)^*(a \cup b)(\varepsilon \cup 1))^*$$

(ε denotes here the empty word). Observe that $Z1$, $Z2$ contain each string over the alphabet $\{a, b\}$ as a subsequence. The number of 1's in every string in $Z1$ is bigger than twice the number of other symbols. The number of 1's in every string in $Z2$ is smaller than twice the number of occurrences of a and b. CL($Z1 \cup Z2$) contains every string such that the number of 1's is not equal to twice the number of other symbols.

For a given Turing machine T we construct the ngsm A corresponding to T in Lemma 3. Let L be a regular language such that CL(L) = $H(A)$. Such a language L can be effectively found according to Lemma 2. Take

$$Z = \text{CL}(Z1 \cup Z2 \cup L).$$

It follows from Lemma 3 that Z has the following property: for each string x of length n over the alphabet $\{a, b\}$, $x1^k \in I^* - Z$ iff $x \in \text{Acc}(T)$ and $k = 2n$.

Hence, Acc(T) is empty iff $I^* = Z$ and so problem (Q0) is effectively reduced to problem (Q4). This proves the undecidability of (Q4) and of problems (Q2), (Q3) (since (Q4) can be reduced to each of them). Undecidability of (Q5) follows from the fact that Acc(T) is finite iff it is empty. The undecidability of (Q6) follows the following observations: iff Acc(T) is empty, then $Z = I^*$ is a regular language, otherwise $(I^* - Z)$ is an infinite set of strings, in each of them the number of 1's is exactly twice the number of other symbols. However, it can be easily proved that such a set is not a regular language, which implies also nonregularity of Z. Hence, Z is regular iff $Z = I^*$ (which is equivalent to Acc(T) = $\emptyset$). It follows that (Q6) is undecidable. This completes the proof. □

References

[1] J. Berstel, *Transductions and Context-free Languages* (Teubner, Stuttgart, 1979).

[2] A. Bertoni, G. Mauri and N. Sabadini, Equivalence and membership problem for regular trace languages, in: *ICALP '82*, Lecture Notes in Computer Science **140** (Springer, Berlin, 1982) 61-71.

[3] C. Choffrut, Free partially commutative monoids, Tech. Rept., Laboratoire Informatique Théorique et Programmation 86.20, March 1986.

[4] M. Chrobak and W. Rytter, The unique decipherability problem with partially commutative alphabet, in: *Proc. Math. Found. of Comput. Science*, Lecture Notes in Computer Science **233** (Springer, Berlin, 1986) 256-263.

[5] O.H. Ibarra, Reversal-bounded multicounter machines and their decision problems, *J. ACM* **25**(1) (1978) 106-133.

[6] O.H. Ibarra, The unsolvability of the equivalence problem for e-free ngsm's with unary input (output) alphabet and applications, *SIAM J. Comput.* 7(4) (1978) 524-532.

[7] A. Mazurkiewicz, Concurrent program schemes and their interpretations, DAIMIPB 78, Aarhus University, 1977.

[8] A. Mazurkiewicz, Traces, histories, graphs: instances of processes monoid, Lecture Notes in Computer Science **176** (Springer, Berlin, 1984) 115-133.
[9] W. Rytter, Some properties of trace languages, *Fund. Inform.* **VII**(1) (1984) 107-127.
[10] M. Szijarto, A classification and closure properties of languages for describing concurrent systems behaviours, *Fund. Inform.* **IV**(3) (1981) 531-550.
[11] A. Tarlecki, Notes on the implementability of formal languages by concurrent systems, ICS PAS Repts. 481 Warsaw, 1982.