Research report 79

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

by

Alan Gibbons and Wojciech Rytter*

(RR79)

Abstract

Let I=A+B be a partially commutative alphabet such that two letters commute if one of them belongs to A and the other one belongs to B. Let M=A* 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 was proved by Choffrut (see [2]) 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 [2], 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.

Department of Computer Science
University of Warwick
Coventry CV4 7AL, England

* on leave from Institute of Informatics, Warsaw University, Poland

August 1986

On the decidability of some problems about rational subsets of free partially commutative monoids

Alan Gibbons and Wojciech Rytter *
Dept. of Computer Science, University of Warwick
Coventry CV4 7AL , U.K.

Abstract.
Let I=A+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 was proved by Choffrut (see [2]) 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 [2],page 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 relation on I. The symbols of I can represent processes (see [6,7]) 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

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) ε 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+B is a partition of the alphabet I. For example in this case a1b11aa≈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 [1,6,7,8] and subsets of a fpcm M were called trace languages in [1]. 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 (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≈v for some v ε L}.

CL is called the closure operation.

The notion of regular flat languages was introduced in [8]. The language L is called a regular flat language (rfl, for short) iff L=CL(L1) for some regular language L1. The class of rfl's is not very regular. It was proved in [8] 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 a rfl X can be a nonregular language, for example $CL((a1)^*)$ 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 a fpcm corresponds to the problem of regularity of a rfl (see [2], page 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: $CL(X1) \cap CL(Y1) = \emptyset$?
Q2: $CL(X1) \subseteq CL(Y1)$?
Q3: $CL(X1) = CL(Y1)$?
Q4: $CL(X1) = I^*$?
Q5: $I^* - CL(X1)$ finite ?
Q6: $CL(X1)$ 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). A 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 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 [4] for the formal (and more general) definition of nrbm's. Let Lan(A) denote the language accepted by a nrbm A.

Lemma 1.
If L is a regular language over I then we can effectively construct a nrbm A such that CL(L)=Lan(A).

Proof.
We describe informally how the automaton A works on the input text w. Let B be a deterministic finite automaton accepting the language L. Let $w \in CL(L)$. The machine A guesses (letter by letter) the string $v \in L$ such that $w \approx v$. The strings w, v differ only in the order of occurences 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 occuring in w, instead A simulates the automaton B on the guessed input string v. A guesses one letter of v in one move (on-line) , if this letter is 1 then A increments its

counter by one, otherwise it checks whether this letter matches the next letter in w which is different from 1. In this way A verifies whether h(w)=h(v), where h is a homomorphism erasing 1's. If the guessed string v is not accepted by B then A rejects. Otherwise A 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 $w \approx v$. Clearly A is a 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 produce now many symbols per one input symbol. Another difference is that a 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 produce (nondeterministically) the output string w and simultaneously end in an accepting state. We refer the reader to [5] 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 [5]. 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 conidered ngsm's is {a,b} and 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 a rfl.

Lemma 2

For every ngsm A we can effectively find a regular expression describing a regular language L such that H(A)=CL(L).

Proof.

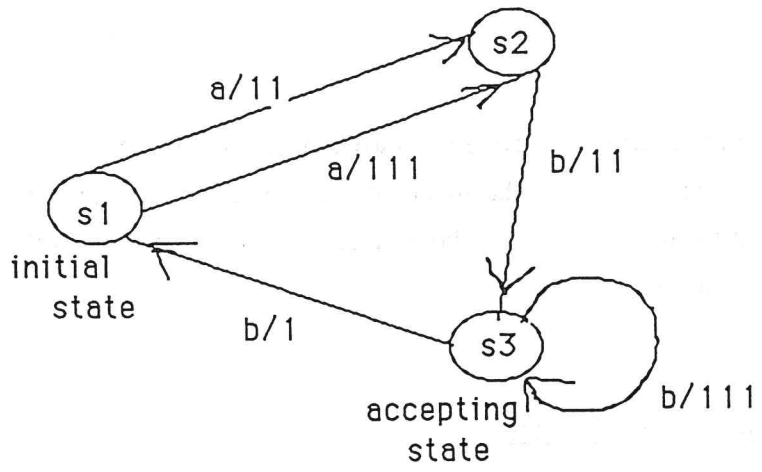
Define the substitution h from pairs of states of A into sets of strings.Each such string is the concatenation of some a_i followed by one possible corresponding output string.

h((s1,s2))= { $a_i out_i$: ngsm A ,being in state s1 and reading the input symbol a_i, can go in one step to state s1 producing the output string out_i}.

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 aregular expression. The diagram of A can be treated as a a finite automaton (recognizer) whose input alphabet is the set of edges. The required language is L=h(L1). This completes the proof.

Example

Let A be the ngsm presented on the figure below. The label a/x means that A reads a and outputs the string x when going to a specified state.



Observe that in the state s1 A can read a and go to s2 producing either 11 or 111 as the output. 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

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

Observe that in this case the language 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 [5] for details. The proof of undecidability for the equivalence problem for ngsm's with unary output alphabet (see [5]) 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 [5] as a side effect of proving the undecidability of the equivalence problem for ngsm's with unary output alphabet. See the proof of Theorem 1 in [5].

Lemma 3

Let T be a single-tape Turing machine with an initially blank tape. We can effectively construct a 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 $x \in Acc(T)$.

Remark

A characterization of rfl's in terms of two-way multihead pushdown automata was given in [8]. 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=CL(X1), Y=CL(Y1). It follows from Lemma 1 that nrbm's A1, A2 can be effectively found such that Lan(A1)=X, Lan(A2)=Y. The disjointness of X, 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 considered below machines loop in the accepting state.
Now consider the following problem:

Q0: for a given single-tape Turing machine T with an initially blank 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

$Z1 = ((a1 1 \cup b1 1)^* 1)^*$ and

$Z2 = ((a1 1 \cup b1 1)^* (a \cup b)(\varepsilon \cup 1))^*$.

(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 occurences of a and b. CL(Z1 Z2) contains every string such that the number of 1's in not equal 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
$Z = CL(Z1 \cup Z2 \cup L)$.
It follows from Lemma 3 that Z has the following property:

for each string x of length n over the alphabet {a,b}

$x1^k \varepsilon I^* - Z$ iff $x \varepsilon Acc(T)$ 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 from 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 bigger than 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] A.Bertoni,G.Mauri,N.Sabadini. Equivalence and membership problem for regular trace languages. in ICALP 82, Lect.Notes in Comp.Science 140,(1982),pp.61-71

[2] C.Choffrut. Free partially commutative monoids. Techn.Report, Laboratoire Informatique Theorique et Programmation 86.20, March (1986)

[3] M.Chrobak,W.Rytter. The unique decipherability problem with partially commutative alphabet. to appear in the proceedings of Math.Found.of Comp.Science, Lect.Notes in Comp.Science (1986)

[4] O.H.Ibarra. Reversal-bounded multicounter machines and their decision problems. Journal of ACM 25:1 (1978) pp.116-133

[5] O.H.Ibarra. The unsolvability of the equivalence problem for e-free ngsm's with unary input (output) alphabet and applications. SIAM Journal on Computing 7:4 (1978) pp.524-532

[6] A.Mazurkiewicz.Concurrent program schemes and their interpretations,DAIMI PB 78,AArhus University (1977)

[7] A.Mazurkiewicz. Traces, histories, graphs: instances of processes monoid. Lecture Notes in Comp.Science 176

[8] W.Rytter. Some properties of trace languages. Fundamenta Informaticae VII.1 (1984) pp.117-127

[9] M.Szijarto. A classification and closure properties of languages for describing concurrent systems behaviours. Fundamenta Informaticae IV.3 (1981) pp.531-550

[10] A.Tarlecki. Notes on the implementability of formal languages by concurrent systems. ICS PAS Reports 481 (1982) Warsaw