数理解析研究所講究録
第 848 巻 1993 年 177-187

The algorithms for deciding some properties of finite convergent string rewriting systems

Guo Mingchao, Li Lian
Lanzhou University, Gansu, China

Abstract. The following problems are decidable in $O(mn^2)$ time for convergent system R on alphabet A, where $m = |A|$, n is the length of R, *i.e.*, the sum of all length of words appeared in R.

(1) Is the monoid presented by R finite ?

(2) How many elements in the monoid presented by R if it is finite ?

and a sofyware is given to decide these properties and some other properties for such monoid.

1. String rewriting systems

Let A be an alphabet, R a subset of $A^* \times A^*$ which is called the set pf string rewriting rules. Elements in R has the the form of (x, y). Let $>$ be an order which satisfies that if $x > y$, then for arbitrary strings $w, z \in R$, $wxz > wyz$. A string rewriting system RS is a double (A, R) where A is an alphabet and R is a set of rewriting rules. An oriented RS is triple $(A, R, >)$, for each rule (x, y) in R, we have $x > y$ and denote as $x \to y$.

Some reduction relation on A^* is defined as follows:

(a) $w_1 \to w_2$ iff there exists $z_1, z_2, (x, y) \in R$, such that $w_1 = z_1xz_2$, $w_2 = z_1yz_2$.

(b) $\to^*$ is the reflexive transitive closure of $\to$.

(c) $\leftrightarrow^*$ is the symmetric closure of $\to^*$.

A word w is called irreducible if there is no word z such that $w \to z$.

An ORS is Noetherian if there is no infinite chain $w_1 \to w_2 \to \cdots \to w_n \to \cdots$.

An ORS is confluent iff for arbitrary words $w_1, w_2, w_1 \leftrightarrow^* w_2$, there exists a ward z such that $w_1 \to^* z$, $w_2 \to^* z$.

An ORS is convergent iff it is Noetherian and confluent.

G is a generating relation of a monoid on alphabet A. Let ρ be the minimum congruence containing G, then

$$M \cong A^*/\rho.$$

$\rho = \leftrightarrow^*$ when G is reviewed as a set of rules, so we call $A^*/\leftrightarrow^*$ is the monoid defined by ORS. When ORS is convergent, the following properties are decidable [2].

(1) Is the monoid presented by R finite ?

(2) How many elements are there the monoid presented by R if it is finite ?

(3) How is the multiplication table of the monoid presented by R created ?

(4) Is it a trivial monoid ?

(5) Is it a group ?

(6) Is it commutative ?

(7) Is it a free monoid ?

Whether an ORS is convergent is decidable [2]. If the ORS is not convergent, we can use the Knuth-Bendix convergent procedure to get an equivalent one in most case (But there indeed exists some ORS that has no convergent system). In the following discussion, we always suppose that ORS is convergent. About the properties of (4), (5), (6) and (7), F. Otto has given some algorithms in polynomial time [2, 4]. Now we turn our attention to the properties of (1), (2) and (3).

2. The decision of Properties (1), (2) and (3).

The number of equivalent classes is the number of elements in monoid when the ORS is convergent. And every class has exactly one irreducible element. The set of all irreducible words $IRR(R)$ is a regular language which can be accepted by a finite state automaton. The cardinal of $IRR(R)$ is equal to get the cardinal of monoid defined by the rewriting system. So we can count the cardinal to get the cardinal of the monoid. Some details are as follows.

Denote
$$\begin{aligned} \text{suffix}(x) &:= \{v \mid x = uv, u, v \in A^*\} \\ \text{dom}(R) &:= \{x \mid (x, y) \in R\} \\ \text{prefix}(R) &:= \{x \mid l = xy, y \in A^+, l \in \text{dom}(R)\}. \end{aligned}$$

We construct the automaton FSA recognizing $IRR(R)$. The states of FSA are those words which are proper prefixs of the left-side of the rules in R where e is the start state. All the states are final states. We denote the state set as F. The transitive function δ is constructed by following way.

$$\delta(w, a) = \begin{cases} \text{undefined} & \text{suffix}(wa) \cap \text{dom}(R) \neq \emptyset \\ s & s \in L = \{x \mid x \in \text{suffix}(wa) \cap \text{prefix}(R)\}, \\ & \quad \text{and } y \in L - \{s\}, |s| > |y| \end{cases}$$

Lemma 1: $IRR(R) = L(FSA)$ [5].

Theorem 2: Let $FSA(Q, A, \delta, e, F)$ be the automaton on alphabet A constructed as above, Then for the state $w \in F$ and $a \in A$, to determine the next state $\delta(w, a)$ needs $O(n)$ time, where $n = |R|$.

Proof: The key to determine the next state $\delta(w, a)$ is to find the suffix of wa such that wa is exactly a prefix of the left-side of a rule in R.

Suppose the word u is the left-side of a rule, $S(u) = \{$ The longest word in prefix of u and suffix of wa $\}$; obviously, $S(u) \neq \emptyset$.

If u itself is a suffix of wa, then $\delta(w, a)$ is undefined, otherwise for arbitrary u in the left-side of the rule, the problem to find the longest word $S(u)$ is a substring recognizing problem [1]. It takes $O(|u|)$ time. When u runs all the left of the rule in R, it needs $O(n)$ time.

Theorem 3: FSA as above can be constructed in $O(mn^2)$ time, where $m = |A|$, $n = |R|$.

Proof: The automaton has been constructed if we have the transitive function. For every state w and $a \in A$, the next state $\delta(w, a)$ can be determined in $O(n)$ steps. The number of states is less than n, so for all states and some $a \in A$, to determine the next states needs $O(n^2)$ time. When a runs through A, it takes $O(mn^2)$.

Theorem 4: The monoid presented by R is infinite iff the automaton FSA recognizing $IRR(R)$ has a circle.

Proof: If the monoid is infinite, then there is an element $w \in IRR(R)$, $|w| > n$, where n is the number of the states of FSA. For FSA has only n states, so w must pass some state twice at least, *i.e.*, FSA contains a circle. Conversely, if FSA has a circle, there exist states q, r and a word $w = xyz$, which satisfies

$$\delta(e, x) = q, \quad \delta(q, y) = q, \quad \delta(q, z) = r.$$

Then FSA accepts all the words like xy^kz. The monoid presented by R is infinite.

Algorithm 5:
INPUT: A finite automaton recognizing $IRR(R)$ on alphabet A.

begin $s_1 := \{e\}$;
  for $i := 1$ **to** n **do**
    begin $S_2 := \emptyset$;
      $S_2 := \bigcup_{p \in S_1, a \in A} \delta(p, a)$;
      $S_1 := S_2$;
    end;
  If $S_2 = \emptyset$ **then OUTPUT** : The monoid is finite
      else OUTPUT : The monoid is infinite
end.

Theorem 6: The algorithm above needs $O(mn)$ time to determine the existence of circle, where $m = |R|$ and n is the number of states in automaton FSA.

Proof: clearly.

If the monoid presented by R is finite, then FSA recognizing $IRR(R)$ has no circle, denote the number of words accepted by state q as $N(q)$, obviously, $N(e) = 1$.

$R(q) = \{p \mid \exists a \in A : \delta(p,a) = q\}$.

$T(q,p) = |\{a \in A \mid \delta(p,a) = q\}|$.

Theorem 7: Let $FSA(Q, A, \delta, e, F)$ be a automaton recognizing $IRR(R)$ which has no circle. Then it satisfies

(a) For each q

$$N(q) = \begin{cases} 1 & q = e \\ \sum\limits_{p \in R(q)} N(p)T(p,q) & q \neq e \end{cases}$$

(b) $|IRR(R)| = \sum\limits_{q \in F} N(q)$

Proof: (a). denote $L(q)$ as the path length from start state to the state q. Now let us induce on $L(q)$.

$L(q) = 0$, then q is start state. It is obviously that $N(q) = 1$.

If $L(q) < t$, the result is true. Let $L(q) = t$, every path from e to q must pass uniquely a state p in $R(q)$, by the induction, the number of word from e to p is $N(p)$. So the number of words passing p from e to q is $N(p)T(p,q)$, *i.e.*, $N(q) = \sum\limits_{p \in R(q)} N(p)T(p,q)$

(b) is clear.

Now we give the algorithm for calculating the order of the monoid.

Algorithm 8:
INPUT : Automaton $FSA(Q, A, \delta, e, F)$ accepting $IRR(R)$.

begin $U := \{e\}$;
  $N\{e\} := 1$;
  $S := \{q \mid \exists p \in U, a \in A : \delta(p,a) = q\}$;
  While $S \neq \emptyset$ **do**
    begin $T := \emptyset$;
      for each $q \in S$ **do**
        if $R(q) \subseteq U$ **then**
          begin $U := U \cup \{q\}$;
            $N(q) := \sum\limits_{p \in R(q)} N(p)T(p,q)$;

$T := T \cup \{q\}$;

end;

$S := T$;

end;

end.

OUTPUT : *order* $:= \sum_{q \in F} N(q)$.

Theorem 9: The algorithm 8 takes $O(mn^2)$ time where $m = |A|$, n is the number of states of the automaton.

For constructing the multiplication table from a monoid, we can calculate the irreducible words of monoid at first. Denote $W(p) := \{x | \delta(e,x) := p\}$. The algorithm is similar with algorithm 8 and given as follows:

Algorithm 10:
INPUT : A automaton $FSA(Q, A, \delta, e, F)$ recognizing $IRR(R)$,
where $IRR(R)$ is finite.

begin $W(e) := \{1\}$;
$U := \{e\}$;
$S := \{q \,|\, \exists p \in U, a \in A : \delta(p,a) = q\}$;
While $S \neq \emptyset$ **do**
begin $T := \emptyset$;
for each $p \in S$ **do**
if $R(p) \subseteq U$ **then**
begin $U := U \cup \{p\}$;
$W(p) := \bigcup_{p \in R(p), a \in A} \{xa | x \in W(q), \delta(q,a) = p\}$;
$T := T \cup \{p\}$;
end;
$S := T$;
end;
end.

OUTPUT : $IRR(R) := \bigcup_{p \in F} W(p)$.

Theorem 11: If monoid presented by R is finite, we can calculating $IRR(R)$ in polynomial $O(mn^2t)$ time, where $m = |a|$, n is the number of states of FSA, $t = |IRR(R)|$.

If the order of elements is t and the length of longest element is n, to calculate the irreducible words for all production of pairwise elements needs $O(n)$ time. For there is a linear time algorithm to get the irreducible word from a given word with length $2n$ [2]. So we can create the multiplication table in $O(nt^2)$ time.

If the order of the monoid to large, or it is infinite, to construct yhe whole multiplication table is impossible. But we can create a finite block of the table when the concrete elements is given. The following algorithm give finite elements which lengths are less than K in a monoid.

Algorithm 12:
INPUT : $FSA(Q, A, \delta, e, F)$ recognizing $IRR(R)$, an integer $k > 0$.

begin$S := \{e\}$;
    $W := \{1\}$;
    While $(S \neq \emptyset)$ and $(loop < k)$ do
    begin $T := \emptyset$;
        $loop := loop + 1$;
        For each $(x, q) \in S$ **do**
        begin $M_{(x,q)} := \{(xa, p) \mid \delta(q, a) = p, a \in A\}$
            If $M_{(x,q)} \neq \emptyset$ **then**
            begin $W := W \cup \{y \mid (y, p) \in M_{(x,q)}\}$;
                $T := T \cup M_{(x,q)}$;
            end;
        $S := T$;
    end;

OUTPUT : **If** $S = \emptyset$ **then** $IRR(R) := W$
else W is a irreducible words set whose element's length is less than k.

When $|R| = 1$, then algorithm 12 takes $O(k)$ time. When $|A| > 1$, then it takes $O(\frac{m^{k+1}-1}{m-1})$ time.

3. Some examples about monoid presented by rewriting systems

Based on the discussion above and [2, 4]. A software is designed and works well. It can get a convergent system from a generated relation and decide the properties of 1 to 7. The following are examples running on the software.

1.

Generated relation :

$R = \{aaaab = ba \quad bbb = 1 \quad babb = aaaa \quad aaaba = b \quad baa = ab \quad aaaaaa = babab$
$bbab = aa \quad ababa = bb \quad aabb = bba \quad aabab = baba\}$

Lexicographical order

Convergent system :

$R = \{aabab = baba \quad aabb = bba \quad ababa = bb \quad bbab = aa \quad aaaaaa = babab$
$baa = ab \quad aaaba = b \quad babb = aaaa \quad bbb = 1 \quad aaaab = ba\}$

Properties :

trivial:	NO
finite:	YES
order:	21
commutative:	NO
free monoid:	NO
group:	YES

A finite multiplication table as following:

=======Semigroup Multiplication Table========

	aaab	abba	abab	aaaaa
b	abab	aaaaa	babab	aaba
ba	bba	aa	aaa	aaab
bb	babab	aaba	aaab	abba
aa	aba	abab	bb	1
ab	baba	babab	1	b
aba	abba	aaa	aaaa	ba
abb	1	b	ba	bab
aaa	aaba	baba	abb	a
aab	bb	1	a	ab
bba	a	ab	aba	abab
bab	aaa	aaab	b	bb
baba	aaaaa	aba	aab	bba
aaba	bab	aaaa	aaaaa	aba
aaaa	b	bb	bba	aa
aaab	abb	a	aa	aab
abba	aa	aab	aaba	baba
abab	aaaa	ba	ab	abb
aaaaa	ab	abb	abba	aaa
babab	aab	bba	bab	aaaa

2.

Generated relation :

$R = \{abb = aa \quad abababababab = b \quad ab = ba\}$

Lexicographical order

Convergent system :

$R = \{ba = ab \quad aaaaaaaa = b \quad abb = aa \quad bbb = ab\}$

Properties:

trivial:	NO
finite:	YES
order:	19
commutative:	YES
free monoid:	NO
group:	NO

A finite multiplication table as following:

=======Semigroup Multiplication Table=======

	1	b	a	ab
1	1	b	a	ab
b	b	bb	ab	aa
a	a	ab	aa	aab
ab	ab	aa	aab	aaa
aa	aa	aab	aaa	aaab
bb	bb	ab	aa	aab
aab	aab	aaa	aaab	aaaa
aaa	aaa	aaab	aaaa	aaaab
aaab	aaab	aaaa	aaaab	aaaaa
aaaa	aaaa	aaaab	aaaaa	aaaaab
aaaab	aaaab	aaaaa	aaaaab	aaaaaab
aaaaa	aaaaa	aaaaab	aaaaaa	aaaaaaa
aaaaab	aaaaab	aaaaaa	aaaaaab	aaaaaaab
aaaaaa	aaaaaa	aaaaaab	aaaaaaa	aaaaaaaa
aaaaaab	aaaaaab	aaaaaaa	aaaaaaab	aaaaaaaab
aaaaaaa	aaaaaaa	aaaaaaab	aaaaaaaa	aaaaaaaaa
aaaaaaab	aaaaaaab	aaaaaaaa	aaaaaaaab	b
aaaaaaaa	aaaaaaaa	aaaaaaaab	b	bb
aaaaaaaab	aaaaaaaab	b	bb	ab

3.

Generated relation :

$R = \{baba = abab \quad cbacbab = bcbacba \quad cbcb = bcbc \quad ca = ac \quad aa = 1 \quad bb = 1 \quad cc = 1\}$

Lexicographical order

Convergent system :

R is convergent.

Properties:

trivial:	NO
finite:	NO
commutative:	NO
free monoid:	NO
group:	YES

A finite multiplication table as following:

=======Semigroup Multiplication Table========

	bab	bcb	abc	aba
acbab	ac	acbacb	cbac	cb
acbabc	acbabcbab	acbacbc	cbabcbc	cbabcba
acbacb	abcbc	acbab	abcbacbac	abcbacb
abacbc	ababcbacb	ababc	abacbacbc	abacbacba
abacba	abcba	abacbabcb	aba	abc
ababcb	babcb	aba	ababcbabc	babcbab
abcbac	acbacba	abcbabcbc	acb	acbac
abcbab	abc	abcbacb	abacbac	abacb
bcbabc	bcbabcbab	bcbacbc	bacbabcbc	bacbabcba
bcbacb	cbc	bcbab	cbacbac	cbacb
babcbc	bacbacb	bac	babcbacbc	babcbacba
babcba	ababcba	babcbabcb	bab	ababc
bacbac	babcbacba	bacbabcbc	babcb	babcbac
bacbab	bac	bacbacb	bcbac	bcb
cbacbc	cbabcbacb	cbabc	cbacbacbc	cbacbacba
cbacba	bcbac	bcbacbacb	cba	cbc
cbabcb	acbabcb	cba	cbabcbabc	acbabcbab
cbabcbc	cbacbacb	cbac	cbabcbacbc	cbabcbacba
cbabcba	acbabcba	bcbabcbabcb	cbab	acbabc

4.

Generated relation :

$R = \{cc = 1 \quad bb = 1 \quad aa = 1 \quad ca = ac \quad bab = aba \quad cbacbacbcb = bcbacbacbc \quad cbcbc = bcbcb \quad cbcbac = bcbcba \quad cbacbacbacba = bcabacbacbacb\}$

Lexicographical order

Convergent system :

R is convergent.

Properties :

trivial:	NO
finite:	YES
order:	120
commutative:	NO
free monoid:	NO
group:	YES

A finite multiplication table as following:

=======Semigroup Multiplication Table========

	cb	cbc	cba	bac
bacbcba	abacbacba	abacbacbac	abacbacb	bacbacbc
bacbacb	bacbacbcb	bcbacbcb	bacbacbcba	bacb
bcbacbc	bcbac	bcba	bcbc	bacbacbcba
bcbacba	bcbacbacb	bcbacbacbc	bcbacbacba	cbcb
abcbcba	acbacba	acbacbac	acbacb	abcbacbc
abcbacb	abcbacbcb	abacbacbcb	abcbacbcba	abcb
abacbac	abcba	abcbac	abcb	abacbacbac
abacbcb	bacbc	bacb	bacbac	abacba
acbacbac	acbcba	abcbcba	acbcb	acbacbacbac
acbacbcb	cbacbc	cbacb	cbacbac	acbacba
abacbcba	bacbacba	bacbacbac	bacbacb	abacbacbc
abacbacb	abacbacbcb	abcbacbcb	abacbacbcba	abacb
abcbacbc	abcbac	abcba	abcbc	abacbacbcba
abcbacba	abcbacbacb	abcbacbacbc	abcbacbacba	acbcb
bcbacbac	bcbcba	cbcba	bcbcb	bcbacbacbac
bcbacbcb	bacbacbc	bacbacb	bacbacbac	bcbacba
bacbacbc	bacbac	bacba	bacbc	bcbacbcba

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The design and analysis of computer algorithms.* Addison Wesley, 1974.

[2] B. Benninghofen, S. Kemmerich, and M. Richter. *Systems of Reductions.* Spring-Verlag, 1987.

[3] J. E. Hopcroft, and J. D. Ullman. *Introduction to automate theory, language and computation.* Addison Wesley, 1979.

[4] Paliath Narendran, and Friedrich Otto. Elements of finite order for finite weight-reducing and confluent thue systems. *Acta Informatica*, **25**:573–591, 1988.

[5] Wang Shuiting. Construction of the multiplication table of the semigroup and its complex degree. *J. of Lanzhou University*, **28**:38–42, 1992.