INFORMATION AND COMPUTATION **130**, 61–70 (1996)
ARTICLE NO. 0082

Computing the Rabin Index of a Regular Language of Infinite Words

Thomas Wilke*

Christian-Albrechts-Universität zu Kiel, Institüt für Informatik und Praktische Mathematik, 24098 Kiel, Germany
E-mail: tw@informatik.uni-kiel.de

and

Haiseung Yoo

Universität Passau, Lehrstuhl für Programmiersysteme, 94030 Passau, Germany
E-mail: yoo@fmi.uni-passau.de

The Rabin index of a regular language of infinite words is the minimum number of accepting pairs used in any deterministic Rabin automaton recognizing this language. We show that the Rabin index of a language given by a Muller automaton with n states and m accepting sets is computable in time $O(m^2nc)$ where c is the cardinality of the alphabet.

1. INTRODUCTION

The Rabin acceptance condition was introduced in (Rabin, 1969) for automata on infinite binary trees, but has since also been considered for automata on infinite words. From McNaughton's theorem (McNaughton, 1966; see also Safra, 1988), it follows that every regular ω-language is recognized by a deterministic finite automaton with Rabin acceptance. Wagner showed in (Wagner, 1979; see also Wagner, 1977) that the number of accepting pairs required to accept a regular ω-language defines a strict infinite hierarchy. In Wagner's paper and in (Barua, 1992), it was explained that this hierarchy has a meaning from a topological point of view.

For a given regular ω-language, the number of accepting pairs that are required to recognize this language is often called its *Rabin index*. Krishnan *et al.* (1995) showed that computing the Rabin index of languages given by deterministic Rabin (or Streett) automata is NP-complete. The situation is different with Muller automata. The methods developed by Krishnan *et al.* can be used to design an

* While producing the final manuscript, the first author was supported by a DIMACS postdoctoral fellowships funded by NSF Award 91-19999 and the New Jersey Commission on Science and Technology.

0890-5401/96 $18.00

algorithm for computing the Rabin index in polynomial time, provided the languages are given by deterministic Muller automata. We present an efficient algorithm that runs in time $\mathcal{O}(m^2nc)$ where m is the number of accepting sets, n the number of states, and c the cardinality of the alphabet. The upper bound that would be obtained using the methods of (Krishnan *et al.*, 1995) is $\mathcal{O}(m^2n^2c)$.

For background on ω-automata, see (Thomas, 1990).

2. RABIN INDEX AND ALTERNATING CHAINS

Throughout this paper, all Muller automata are assumed to be deterministic and reduced, that is, all states are reachable from the initial state.

Let $\mathfrak{A} = (Q, q_0, \delta, \mathscr{F})$ be a Muller automaton over an alphabet A. The *transition graph* G of $\mathfrak{A}$ is the directed graph (Q, E) with $E = \{(q, q') \mid \exists (a \in A)\, \delta(q, a) = q'\}$.

Remark 1. $|E| \leqslant |Q|\,|A|$.

In particular, a graph search on G can be done in time $\mathcal{O}(|Q|\,|A|)$.

A *loop* in $\mathfrak{A}$ is a strongly connected subset of G. It is called *positive* if it belongs to $\mathscr{F}$ and *negative* if it does not. The set of all loops of $\mathfrak{A}$ is denoted by $\mathscr{C}$, the set of all positive loops by $\mathscr{P}$, and the set of all negative loops by $\mathscr{N}$; so $\mathscr{C}$ is the disjoint union of $\mathscr{P}$ and $\mathscr{N}$. (Observe that $\mathscr{P} \subseteq \mathscr{F}$ and that $\mathscr{P} \subset \mathscr{F}$ if and only if $\mathscr{F}$ contains sets that are not strongly connected in G. The automata $\mathfrak{A}$ and $(Q, q_0, \delta, \mathscr{P})$, however, are equivalent.) The set of maximal loops (w. r. t. set inclusion) is denoted by $\mathscr{M}$. (Note that an element of $\mathscr{M}$ is what is also known as a strongly connected component.)

An *alternating chain of length n* is a sequence $C_1, ..., C_n$ of loops such that $C_i \subset C_{i+1}$ and $C_i \in \mathscr{P}$ iff $C_{i+1} \notin \mathscr{P}$ for $i = 1, ..., n-1$. A *positive alternating chain* starts with a positive loop, i.e., $C_1 \in \mathscr{P}$. The length of a longest positive alternating chain in $\mathfrak{A}$ is denoted by $m^+(\mathfrak{A})$. By convention, $m^+(\mathfrak{A}) = 0$ if $\mathscr{P}$ is empty.

Wagner proved in (Wagner, 1977; see also Wagner, 1979), the following about $m^+(\mathfrak{A})$ and its connection to the Rabin index of the language recognized by $\mathfrak{A}$.

THEOREM 1 (Wagner). (Invariance Property) *If $\mathfrak{A}$ and $\mathfrak{B}$ are equivalent Muller automata over the same alphabet, then*

$$m^+(\mathfrak{A}) = m^+(\mathfrak{B}).$$

(Rabin Index). *The Rabin index of a regular ω-language recognized by a Muller automata $\mathfrak{A}$ is given by the expression*

$$\lfloor (m^+(\mathfrak{A}) + 1)/2 \rfloor,$$

where $\lfloor x \rfloor$ denotes the largest integer not greater than x.

3. THE ALGORITHM

In view of Theorem 1, the Rabin index of a regular ω-language defined by a Muller automaton $\mathfrak{A}$ is obtained from $m^+(\mathfrak{A})$ by two simple arithmetic operations.

By definition, $m^+(\mathfrak{A})$ is the maximum of the function $c: \mathscr{P} \to \mathbf{N}$ that assigns to every positive loop P the length of a longest alternating chain in $\mathfrak{A}$ starting with P (provided $\mathscr{P}$ is not empty; in this case, $m^+(\mathfrak{A}) = 0$). The function c, however, is determined by the simple rules stated in the following lemma.

LEMMA 1. 1. *If P is maximal in $\mathscr{P}$, then*

$$c(P) = \begin{cases} 2, & \text{if } P \notin \mathscr{M}, \\ 1, & \text{otherwise.} \end{cases}$$

2. *Let P be a non-maximal element of $\mathscr{P}$, and define k and $\mathscr{P}'$ as follows*:

$$k = \max\{c(P') \mid P' \in \mathscr{P} \wedge P \subset P'\},$$

$$\mathscr{P}' = \{P' \in \mathscr{P} \mid P \subset P' \wedge c(P') = k\}.$$

Then

$$c(P) = \begin{cases} k+2, & \text{if there are } N \in \mathscr{N} \text{ and } P' \in \mathscr{P}' \text{ such that } P \subset N \subset P', \\ k, & \text{otherwise.} \end{cases}$$

Proof. The first part of the lemma is trivial.

For the proof of the second part, we first show $c(P) \in \{k, k+2\}$. For this, we first argue that $k \leqslant c(P) \leqslant k+2$.

If $C_1 \subset C_2 \subset \cdots \subset C_l$ is an alternating chain of length l starting with an arbitrary positive loop P' containing P, then $P \subset C_2 \subset \cdots \subset C_l$ is an alternating chain of length l starting with P, and thus $c(P) \geqslant \max\{c(P') \mid P \subset P'\} = k$. On the other hand, if $C_1 \subset C_2 \subset \cdots \subset C_l$ is an alternating chain of length $l \geqslant 3$ starting with P, then $C_3 \subset C_4 \subset \cdots \subset C_l$ is an alternating chain of length $l-2$ starting with C_3, and thus $c(P) \leqslant c(C_3) + 2 \leqslant \max\{c(P') \mid P \subset P'\} + 2 = k+2$.

We now argue that $c(P) = k+1$ is impossible. Let P' be a maximal element of $\mathscr{P}'$, and let M be the strongly connected component P and P' are contained in. (Observe that $P' = M$ iff $c(P') = 1$.) Furthermore, let $C_1 \subset C_2 \subset \cdots \subset C_{c(P)}$ and $C'_1 \subset C'_2 \subset \cdots \subset C'_{c(P')}$ be maximum-length alternating chains starting with P and P', respectively. Then $C_1 \subset C_2 \subset \cdots \subset C_{c(P)-1} \subset M$ and $C'_1 \subset C'_2 \subset \cdots \subset C'_{c(P')-1} \subset M$ are also maximum-length alternating chains starting with P (or M if $c(P) = 1$) and P', respectively. Therefore, $c(P)$ and $c(P')$ are either both even or both odd; in particular, $c(P) \neq c(P') + 1 = k+1$.

To complete the proof, it is now enough to show that there exist $N \in \mathscr{N}$ and $P' \in \mathscr{P}'$ such that $P \subset N \subset P'$ if and only if $c(P) = k+2$.

For the implication from left to right, assume there exist such N and P'. Then $c(P) \geqslant k+2$, as we can extend an alternating chain of length k starting with P' to an alternating chain of length $k+2$ starting with P by prefixing it with P and N. Thus, by the above consideration, $c(P) = k+2$. For the other implication, assume $c(P) = k+2$, and let $C_1 \subset C_2 \subset \cdots \subset C_{k+2}$ be a maximum-length alternating chain

starting with P. Then there exist $N \in \mathcal{N}$ and $P' \in \mathcal{P}'$ such that $P \subset N \subset P'$, namely, C_2 and C_3. ∎

The previous lemma motivates and proves the correctness of the following algorithm computing the Rabin index.

ALGORITHM RABININDEX.

1. Compute G, $\mathcal{P}$, and $\mathcal{M}$ as defined above.
2. If $\mathcal{P} = \emptyset$, output 0.
3. For every $P \in \mathcal{P}$ in non-increasing order:
   (a) Let $\mathcal{P}' = \{P' \in \mathcal{P} \mid P \subset P'\}$.
   (b) If $\mathcal{P}' = \emptyset$ then:
      i. If $P \in \mathcal{M}$, let $c(P) = 1$.
      ii. Else let $c(P) = 2$.
   (c) Else:
      i. Let $k = \max\{c(P') \mid P' \in \mathcal{P}'\}$.
      ii. Let $\mathcal{P}'' = \{P' \in \mathcal{P}' \mid c(P') = k\}$.
      iii. Let $c(P) = k$.
      iv. For every $P' \in \mathcal{P}''$ in non-decreasing order,[1] if there is $N \in \mathcal{N}$ such that $P \subset N \subset P'$, let $c(P) = k + 2$ and exit this for loop.
4. Let $m^+(\mathfrak{A})$ be the maximum value of c.
5. Output $\lfloor (m^+(\mathfrak{A}) + 1)/2 \rfloor$.

The crucial part is the test in 3.(c).iv, namely, whether there exists a negative loop N such that $P \subset N \subset P'$. In a straightforward implementation of this test, one would compute the set $\mathcal{N}$ of all negative loops, which, in the worst case, could take exponential time. We will describe an efficient implementation, based on the assumption that the sets P' are visited in non-decreasing order in 3.(c).iv, just as suggested in the above description of the algorithm. Before, however, we analyze what this assumption implies.

If we visit the P' in non-decreasing order, every time we perform the test in 3.(c).iv we know that there is no N' and no P_0 such that $P \subset N \subset P_0 \subset P'$, because otherwise we had exited the for loop earlier. Also, since $c(P')$ is maximal among all loops greater than P, there is no P_0 and no N such that $P \subset P_0 \subset N \subset P'$. It is convenient to state this with different notation.

Given loops C_* and C^*, we use $B(C_*, C^*)$ for $\{C \in C \mid C_* \subset C \subset C^*\}$, $B^+(C_*, C^*)$ for $B(C_*, C^*) \cap P$, and $B^-(C_*, C^*)$ for $B(C_*, C^*) \cap N$.

Using the new notation, the above says that every time the crucial test in 3.(c).iv is performed the sets $B^+(P, P')$ and $B^-(P, P')$ are elementwise incomparable, i.e., no element of $B^+(P, P')$ is a subset of an element of $B^-(P, P')$, and no element of $B^-(P, P')$ is a subset of an element of $B^+(P, P')$.

We now describe an efficient refinement of the test in 3.(c).iv, the procedure INBETWEEN. Given loops C_* and C^* with $C_* \subset C^*$, INBETWEEN returns true if

[1] For the correctness of this algorithm, it is not necessary to visit the elements of $\mathcal{P}''$ in non-decreasing order. The refinement that is discussed below, however, builds on this assumption.

$B^-(C_*, C^*)$ is non-empty and false otherwise, provided the sets $B^+(C_*, C^*)$ and $B^-(C_*, C^*)$ are elementwise incomparable. We use $\mathrm{Adj}(q)$ to denote the set of nodes *adjacent* to a node q, i.e., $\mathrm{Adj}(q) = \{q' \mid (q, q') \in E\}$. Similarly, $\mathrm{Adj}(C) = \{q' \mid \exists (q \in C)\ (q, q') \in E\}$ for a set $C \subseteq Q$.

PROCEDURE INBETWEEN.

1. Restrict G to C^*.
2. Find a shortest path $c_1, p_1, ..., p_r, c_2$ with $c_1, c_2 \in C_*, p_i \notin C_*$ for every i, and $r > 0$ (by, e.g., a modified breadth-first search). Let $C = C_* \cup \{p_1, ..., p_r\}$.
3. If $C = C^*$, return false, and if $C \notin \mathscr{P}$, return true.
4. If $\mathrm{Adj}(C_*) \backslash C \neq \emptyset$, let q_1 be an element of this set difference. Otherwise, let u be minimal such that $\mathrm{Adj}(p_{u-1}) \backslash C \neq \emptyset$, and let q_1 be an element of $\mathrm{Adj}(p_{u-1}) \backslash C$.
5. Construct a depth-first search tree t for G restricted to $C^* \backslash C$ starting from q_1. If t branches, return false, else $t = q_1, ..., q_k$ for appropriate q_i and an appropriate k.
6. (a) If $\mathrm{Adj}(q_k) \cap C = \emptyset$, return false.
   (b) If $\mathrm{Adj}(q_k) \cap C_* \neq \emptyset$, let $C' = C_* \cup \{p_1, ..., p_{u-1}, q_1, ..., q_k\}$.
   (c) Otherwise, choose l maximal with $p_{l+1} \in \mathrm{Adj}(q_k)$, and let $C' = C_* \cup \{p_1, ..., p_{u-1}, q_1, ..., q_k, p_{l+1}, ..., p_r\}$.
7. If $C' \notin \mathscr{P}$, return true else return false.

The existence of a path as required in step 2 and a node q_1 as required in step 4 are guaranteed by the assumption that C_* is a proper subset of C^* and that C^* is a loop, i. e., strongly connected.

It is clear that the procedure terminates and returns true only if the set $B^-(C_*, C^*)$ is non-empty. The other direction will be proven at the end of the next section.

Let's analyze the running time of RABININDEX and consider crucial implementation details. The set $\mathscr{P}$ can be obtained from $\mathscr{F}$ by deleting all sets that are not strongly connected. So the construction of $\mathscr{P}$ can be carried out in time $\mathcal{O}(|\mathscr{F}|\,|Q|\,|A|)$. By comparing the elements of $\mathscr{P}$ with each other, we can construct the inclusion relation for $\mathscr{P}$ and partition $\mathscr{P}$ into maximal and non-maximal elements in time $\mathcal{O}(|\mathscr{F}|^2\,|Q|)$. Non-increasing and non-decreasing orderings for $\mathscr{P}$ can easily be obtained by sorting its elements according to their cardinality. So RABININDEX can be implemented in time $\mathcal{O}(|\mathscr{F}|\,|Q|\,|A| + |\mathscr{F}|^2\,|Q| + |\mathscr{F}|^2 h)$, where h is an upper bound on the running time of INBETWEEN.

The running time of INBETWEEN is determined by the two graph searchs (DFS and BFS), each of whose running time is bounded by $\mathcal{O}(|Q||A|)$, and the time that is needed to check whether a given set belongs to $\mathscr{P}$. The latter can be done in time $\mathcal{O}(|Q|)$ once a binary search tree for $\mathscr{P}$ has been computed. This can be accomplished in time $\mathcal{O}(|\mathscr{F}|\,|Q|)$ and should be done as a preprocessing outside the procedure itself. Putting all together, we have:

THEOREM 2. *The algorithm* RABININDEX *with the procedure* INBETWEEN *called in step* 3.(c).iv *computes the Rabin index of a regular ω-language L in time*

$\mathcal{O}(|\mathcal{F}|^2 |Q| |A|)$ *where* L *is supposed to be represented by a deterministic Muller automaton* $\mathfrak{A} = (Q, q_0, \delta, \mathcal{F})$ *over an alphabet* A.

4. GRAPH-THEORETIC ANALYSIS AND CORRECTNESS PROOF

Throughout this section, let $\mathfrak{A}$ be a Muller automaton and G, $\mathscr{C}$, $\mathscr{P}$, and $\mathscr{N}$ as before. We fix loops C_* and C^* and assume that $B^+(C_*, C^*)$ and $B^-(C_*, C^*)$ are elementwise incomparable and non-empty. For convenience, we shorten $B(C_*, C^*)$, $B^+(C_*, C^*)$, and $B^-(C_*, C^*)$ to B, B^+, and B^-, respectively.

LEMMA 2 (Complementation Property). *If* $P \in B^+$ *and* $N \in B^-$, *then* $P \cup N = C^*$.

Proof. Write C for $P \cup N$ and, for contradiction, assume $C \subset C^*$. Then either $C \in B^+$ or $C \in B^-$. In the first case, we would have $N \subset C$, i.e., N and C would be comparable, contradicting the general assumption, and in the second case, P and C would be comparable, again contradicting the general assumption. ▌

LEMMA 3. *Every minimal element of* B^+ *or* B^- *is a minimal element of* B.

Proof. For contradiction, assume first that P was a minimal element of B^+ but not of B. A smaller element C of B would have to be negative, i.e., an element of B^-, and P and C would then be comparable, contradicting the general assumption. A symmetric argument shows that the claim is also true for B^+. ▌

Since we assume that B^+ and B^- are non-empty, both sets have minimal elements, say, P and N. By the complementation property, we know $P \cup N = C^*$. Since P and N are minimal elements in B (Lemma 3), there exist simple paths c_1, $p_1, ..., p_r$, c_2 and c_3, $n_1, ..., n_s$, c_4 such that

- $P = C_* \cup \{p_1, ..., p_r\}$ and $N = C_* \cup \{n_1, ..., n_s\}$,
- $c_1, ..., c_4 \in C_*$, $p_i, n_j \notin C_*$ for every i and j, respectively.

LEMMA 4. 1. *There exists an* i *with* $1 \leqslant i \leqslant \min(r, s)$ *such that* $p_i \notin N$, $n_i \notin P$, *and* $p_j = n_j$ *for all* j *with* $1 \leqslant j < i$.

2. *There exists an* i *with* $0 \leqslant i < \min(r, s)$ *such that* $p_{r-i} \notin N$, $n_{s-i} \notin P$, *and* $p_{r-j} = n_{s-j}$ *for all* j *with* $0 \leqslant j < i - 1$.

Proof. By symmetry, it is enough to prove only the first part of the lemma.

By the general assumption, P and N are incomparable. Therefore, there is an i with $1 \leqslant i \leqslant \min(r, s)$ such that $p_i \notin N$ or $n_i \notin P$, and $p_j = n_j$ for all j with $1 \leqslant j < i$. W. l. o. g., assume $p_i \notin N$. We show $n_i \notin P$. For contradiction, assume $n_i \in P$. Then $n_i = p_j$ for some j with either $j < i$ or $j > i$. If we had $j > i$, then $C \cup \{p_1, p_2, ..., p_{i-1}, p_j, ..., p_r\}$ would be a loop smaller than P, contradicting Lemma 3. The case $j < i$ is also impossible, as otherwise $C \cup \{p_1, p_2, ..., p_j, p_{i+1}, ..., p_r\}$ would be a loop smaller than P, also contradicting Lemma 3. ▌

Let u be the number i from the first part of the previous lemma, and set $v = r - i$ and $w = s - i$ with i being the number from the second part of the previous lemma.

LEMMA 5 (Edge Classification). *Let e be an edge in the restriction of G to C^*.*

- *If $e=(q, p_i)$ or $e=(q, n_i)$ for some $q \in C_*$, then $i=1$, and e is called an* out edge.
- *If $e=(p_i, q)$ or $e=(n_j, q)$ for some $q \in C_*$, then $i=r$ or $j=s$, respectively, and e is called an* in edge.
- *If $e=(p_i, p_j)$ or $e=(n_i, n_j)$, then*

$$j=i+1, \text{ and } e \text{ is called a path edge, or}$$

$$j \leqslant i, \text{ and } e \text{ is called a back edge.}$$

- *If $e=(p_i, n_j)$ for some i and j with $u \leqslant i \leqslant v$ and $u \leqslant j \leqslant w$, then $i=v$ or $j=u$.*

If $e=(n_j, p_i)$ for some i and j with $u \leqslant i \leqslant v$ and $u \leqslant j \leqslant w$, then $i=u$ or $j=w$.

In both cases, e is called a cross edge.

- *If $e=(q, q')$ for some $q, q' \in C_*$, then e is called a C_*-edge.*

At least one of the above cases applies to e.

The situation is illustrated in Fig. 1. (*Notice that when $u>1$, we can assume $c_1=c_3$, and when $v<s$, we can assume $c_2=c_4$.*)

Proof. Since $C^*=P \cup N$ by the complementation property, at least one of the listed cases applies to every edge e in the restriction of G to C^*.

There are no edges (q, p_i) with $q \in C_*$ and $i>1$, for otherwise $C_* \cup \{p_i, ..., p_r\}$ would be a smaller element of B than P, contradicting the minimality of P (see Lemma 3). A symmetric argument rules out edges (q, n_i) with $q \in C_*$ and $i>1$. Similarly, if $e=(p_i, q)$ or $e=(n_j, q)$ for some $q \in C_*$, then $i=r$ or $j=s$, respectively.

There are no forward edges, that is, edges of the form (p_i, p_j) or (n_i, n_j) with $j-i>1$. For if we had $(p_i, p_j) \in E$ with $j-i>1$, then $C_* \cup \{p_1, ..., p_i, p_j, ..., p_r\}$ would be an element of B smaller than P, contradicting the minimality of P (see Lemma 3). A symmetric argument applies in the other case, where we have an edge (n_i, n_j) with $j-i>1$.

We are left with edges (p_i, n_j) or (n_j, p_i) with $u \leqslant i \leqslant v$ and $u \leqslant j \leqslant w$. So assume we are given an edge of the first type. Consider the loop C defined by $C=C_* \cup \{p_1, ..., p_i, n_j, ..., n_s\}$. We proceed by case distinction on whether C is negative or positive.

First case, C is positive. Then $C \cup N=C^*$ by the complementation property, in particular, p_v has to be an element of $C \cup N$. Since p_v is not an element of N, p_v must be among $p_1, ..., p_i$, hence $i=v$.

Second case, C is negative. Then $C \cup P=C^*$ again by the complementation property, and, similar to the previous case, n_u has to be an element of C. Since n_u does not belong to P, we conclude $j=u$.

The other situation, where we have an edge (n_j, p_i) with $u \leqslant i \leqslant v$ and $u \leqslant j \leqslant w$, is symmetric. ∎

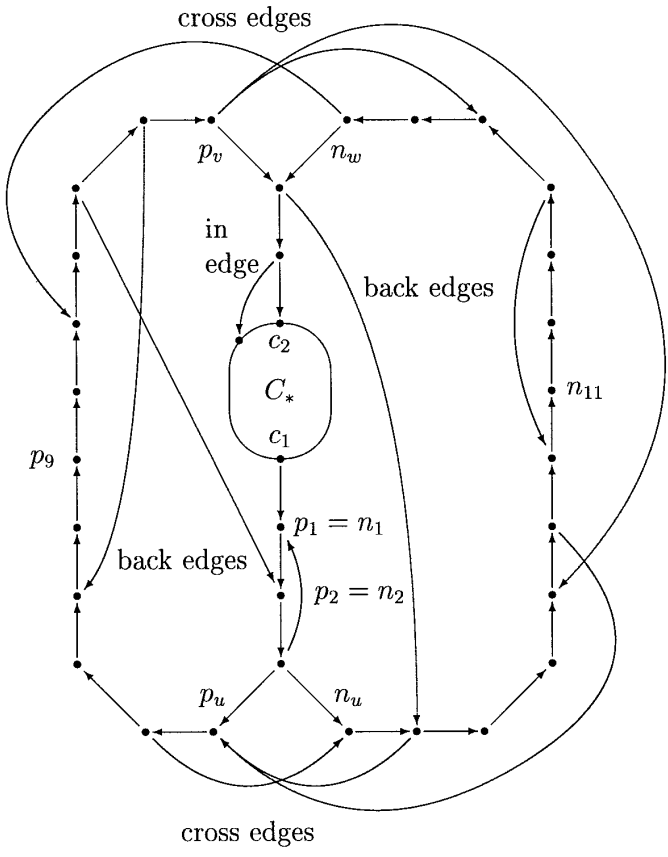

FIG. 1. A typical example for the situation analyzed in Section 4.

COROLLARY 1. *The loops P and N are unique, that is, there exist no minimal elements of B^+ and B^- other than P and N, respectively.*

Proof. From the previous lemma, it follows immediately that if $q_1, q_2, ..., q_l$ is a path through G with $q_1 \in \text{Adj}(C_*)$, $\{q_1, ..., q_l\} \subseteq C^* \backslash C_*$, and $q_l = p_i$ for some i, then $\{p_1, ..., p_i\} \subseteq \{q_1, ..., q_l\}$ or $\{n_1, ..., n_w\} \subseteq \{q_1, ..., q_l\}$. Similarly, if $q_l = n_i$ for some i, then $\{n_1, ..., n_i\} \subseteq \{q_1, ..., q_l\}$ or $\{p_1, ..., p_v\} \subseteq \{q_1, ..., q_l\}$. Hence every element of B contains P or N, and thus P and N are unique. ∎

COROLLARY 2. *The sets $\{p_u, ..., p_v\}$ and $\{n_u, ..., n_w\}$ are disjoint.*

Proof. Suppose we had $p_i = n_j$ with $u \leqslant i \leqslant v$ and $v \leqslant j \leqslant w$. Since $p_u \notin N$, $p_v \notin N$, $n_u \notin P$, and $n_w \notin P$, this could only be the case if we had $u < i < v$ and $u < j < w$. Then, however, (p_{i-1}, p_i), which is the same as (p_{i-1}, n_j), would have to be a cross edge, which implies $i - 1 = v$ or $p_{i-1} = n_w$; the first case is impossible, as we assume $i \leqslant v$, and the second case is impossible, since $n_w \notin P$. ∎

Remaining Part of the Correctness Proof for INBETWEEN. We have to show that if there is a negative loop N such that $C_* \subset N \subset C^*$, then INBETWEEN will report

that. Since in step 2 a shortest path is chosen, the loop C is a minimal loop. So c_1, $p_1, ..., p_r, c_2$ can be thought of as the path fixed at the beginning of this section, provided C is positive, what we want to assume (because otherwise true will be returned in step 3 anyway). We identify C and P. Choose N and $c_3, n_1, ..., n_s, c_4$ just as at the beginning of this section, so that Lemma 5 applies.

In step 4 we will choose q_1 to be n_u, as all out edges point to p_1 or n_1 and we do not have forward edges.

Notice that $\{n_u, ..., n_w\}$ is the set of nodes reachable from q_1 $(=n_u)$ in G restricted to $C^*\backslash C$ (Corollary 2). Since every edge which is not a back edge belongs to the path $n_u, ..., n_w$, the DFS will construct $n_u, ..., n_w$ as its DFS tree. In particular, we will have $n_w = q_k$.

In step 6, we will therefore not encounter case (a). Since we have no forward edges and all in edges start from n_s, we will choose C' to be N either in step 6.(b), namely, if $w = s$, or in step 6.(c), by first choosing l to be v and then C' to be N. Thus the procedure will return true in step 7. ▌

5. REMARKS

The procedure InBetween described in the conference version of this paper, (Wilke and Yoo, 1995), is based on Corollary 1, which was proven in quite a different way there.

In (Wilke and Yoo, 1995) it was also indicated how one can compute the "topological complexity," i.e., the Wadge and the Lifschitz degree of a regular ω-language L in time $O(|\mathscr{F}|^2 |Q||A| + k \log k)$, where L is supposed to be represented by a deterministic Muller automaton $\mathfrak{A} = (Q, q_0, \delta, \mathscr{F})$ over an alphabet A and k is the number of strongly connected components of the transition graph of $\mathfrak{A}$. Of course, the implementation of InBetween suggested here can be used instead of the implementation described in (Wilke and Yoo, 1995).

The dual version of Wagner's theorem says that the Streett index of a regular ω-language given by a deterministic Muller automaton is determined by the length of a longest *negative* alternating chain. Based on this, our algorithm can easily be modified to compute the Streett index within the same time bound.

Received January 10, 1996; final manuscript received July 19, 1996

REFERENCES

Barua, R. (1992), The Hausdorff–Kuratowski hierarchy of ω-regular languages and a hierarchy of Muller automata, *Theoret. Comput. Sci.* **96**, 345–360.

Krishnan, S. C., Puri, A., and Brayton, R. K. (1995), Structural complexity of ω-automata, *in* "Proceedings, STACS 95: 12th Annual Symposium on Theoretical Aspects of Computer Science, Munich, Germany" (E. W. Mayr and C. Puech, Eds.), Lecture Notes in Computer Science, Vol. 900, pp. 143–156, Springer-Verlag, Berlin/Heidelberg/New York.

McNaughton, R. (1966), Testing and generating infinite sequences by a finite automaton, *Information and Control (5)* **9**, 521–530.

Rabin, M. O. (1969), Decidability of second-order theories and finite automata on infinite trees, *Trans. Amer. Math. Soc.* **141**, 1–35.

Safra, S. (1988), On the complexity of ω-automata, *in* "Proceedings, 29th Annual Symposium on Foundations of Computer Science, White Plains, New York," pp. 319–327.

Thomas, W. (1990), Automata on infinite objects, *in* "Handbook of Theoretical Computer Science" (Jan van Leeuwen, Ed.), Vol. B, pp. 134–191, Elsevier, Amsterdam.

Wagner, K. W. (1977), Eine topologische Charakterisierung einiger Klassen regulärer Folgenmengen, *Elektron. Informationsverarb. Kybernet. (9)* **13**, 473–487.

Wagner, K. W. (1979), On ω-regular sets, *Inform. and Control (2)* **43**, 123–177.

Wilke, Th., and Yoo, H. (1995), Computing the Wadge degree, the Lifschitz degree, and the Rabin index of a regular language of infinite words in polynomial time, *in* "Proceedings, Tapsoft '95: Theory and Practice of Software Development, Aarhus, Denmark" (P. D. Mosses *et al.*, Eds.), Lecture Notes in Computer Science, Vol. 915, pp. 288–302, Springer-Verlag, Berlin/Heidelberg/New York.

Yoo, H. (1994), "Ein effizienter Algorithmus zur Bestimmung des Rabin-Index in Muller-Automaten," diploma thesis, Inst. f. Inform. u. Prakt. Math, Christian-Albrechts-Universität zu Kiel, Kiel, Germany.