DFAs with a Bounded Activity Level

Marius Konitzer and Hans Ulrich Simon

Fakultät für Mathematik, Ruhr-Universität Bochum, D-44780 Bochum
{`marius.konitzer,hans.simon`}`@rub.de`

Abstract. Lookahead DFAs are used during parsing for sake of resolving conflicts (as described in more detail in the introduction). The parsing of an input string w may require many DFA-explorations starting from different letter positions. This raises the question how many of these explorations can be active at the same time. If there is a bound on this number depending on the given DFA M only (i.e., the bound is valid for all input strings w), we say that M has a *bounded activity level*. The main results in this paper are as follows. We define an easy-to-check property of DFAs named *prefix-cyclicity* and show that precisely the non prefix-cyclic DFAs have a bounded activity level. Moreover, the largest possible number ℓ_M of mutually overlapping explorations of a given non prefix-cyclic DFA M with $t+1$ states, the so-called *maximum activity level of* M, is bounded from above by $2^t - 1$, and this bound is tight. We show furthermore that the maximum activity levels of equivalent DFAs coincide so as to form an invariant of the underlying regular language, which leads us to a characterization of prefix-cyclicity in terms of the Nerode relation. We finally establish some complexity results. For instance, the problem of computing ℓ_M for a given non prefix-cyclic DFA M is shown to be PSPACE-complete.

Key words: parsing, lookahead DFA, computational complexity

1 Introduction

LR-regular (LRR) parsing [12] is one of the few parsing techniques utilizing unbounded lookahead. LRR languages properly include the deterministic context-free languages [7]. LRR parsers allow for a large amount of interesting grammars with practical relevance (such as the original version of the Java language [6]), which cannot be handled by any LR(k) parser. The parsers generated with the algorithm from [12] clearly have linear runtime, although they are a little cumbersome. The algorithm is rather of theoretical interest as membership in the class of LR-regular grammars is undecidable and as some implementation details remain unclear. Practical LRR parsing techniques such as [1] and [4] basically work like the well-known LR(k) shift-reduce parsers [7], yet use regular lookaheads of arbitrary length instead of the normal fixed length ones. Starting with an inconsistent LR(0) automaton, practical LRR parser generation techniques set out to build disjoint prefix-free regular envelopes for each inconsistent LR(0) state. This aims at separating the state's conflicting suffix languages from each other. These regular envelopes are typically built as prefix-free deterministic finite automata (DFA), so called *lookahead DFAs*, which are used for lookahead exploration during parsing whenever necessary. Different lookahead explorations operating on a common substring of the input string may overlap each other. As explained in the abstract (and formally defined in section 2), this leads to the notion of the maximum activity level ℓ_M associated with a given DFA M (set to ∞ in the unbounded case).

If the number of mutually overlapping explorations on strings of length n is bounded from above by $B \leq n$, the whole parser has time bound $O(Bn)$ on inputs of length n (as illustrated in Fig. 1). If the parser employs prefix-cyclic DFAs, this leads to the time bound $O(n^2)$.[2] If, however, only non prefix-cyclic lookahead DFAs are employed during parsing, then B does not depend on n (but still depends on the sizes of the lookahead DFAs).[3] As for of a fixed LR-regular grammar with a fixed collection of non prefix-cyclic lookahead DFAs, one may think of B as a (possibly large) constant. But once we think in terms of practical LRR parser generators, the dependence on the sizes of the employed lookahead DFAs becomes an issue.

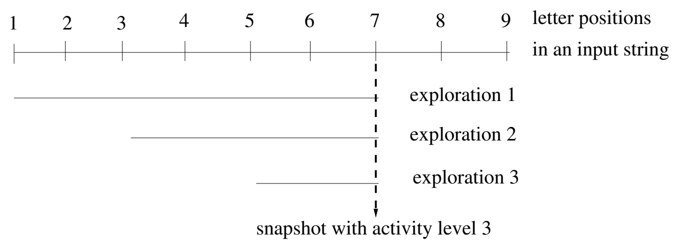

Fig. 1. The parser and all of the $B = 3$ DFA-explorations require up to n computational steps, respectively.

The notion of prefix-cyclicity (among other related notions) was introduced and exploited in [8]. However, the run time analysis in [8] treats the parameter B as a constant whenever it does not depend on n. In this paper, we take care of the dependence of B on the sizes of the employed lookahead DFAs and study the dependence of ℓ_M on the number t of M's non-initial states. We extend the work in [8] in various directions. Section 3 presents the (tight) upper bound $2^t - 1$ on ℓ_M. Section 4 casts ℓ_M as an invariant of the underlying language $L(M)$. Section 5 characterizes prefix-cyclicity in terms of the Nerode-relation. Section 6 is devoted to complexity issues. Specifically, it is shown that the computation of ℓ_M for a given non prefix-cyclic DFA M is PSPACE-complete.

[2] An LRR-grammar leading indeed to quadratic run time for parsing with (unbounded) lookahead DFAs is found in [10].

[3] See [8] for several grammar constructs leading to non prefix-cyclic lookahead DFAs (e.g. HTML forms [4] and Ada calls [2, 10]).

2 Definitions and Notations

Let M be a Deterministic Finite Automaton (DFA) given by its finite set of states, Q, its input alphabet, Σ, its partially defined transition function $\delta : Q \times \Sigma \to Q$, and its initial state $q_0 \in Q$. (For the time being, we do not need to distinguish between accepting and non-accepting states.) If M reads symbol a in state q and $\delta(q, a)$ is undefined, then we may think of M as terminating its computation. As usual, the mapping δ can be extended to a partially defined mapping $\delta^* : Q \times \Sigma^* \to Q$:

$$\delta^*(q, \varepsilon) = q$$

$$\delta^*(q, aw) = \begin{cases} \text{undefined} & \text{if } \delta(q, a) \text{ is undefined} \\ \delta^*(\delta(q, a), w) & \text{otherwise} \end{cases}$$

Here, ε denotes the empty string, $q \in Q$, $a \in \Sigma$, and $w \in \Sigma^*$. If not undefined, then $\delta^*(q, w)$ is the state reached by a computation of M that was started in state q and has processed all letters of w. We say that w is *fully processed by* M if $\delta^*(q_0, w)$ is not undefined. Suppose that $w = a_1 \cdots a_n \in \Sigma^n$. Then, for all $1 \le i < j \le n$, $w_{i,j}$ denotes the substring $a_i \cdots a_{j-1}$. We say that M *has activity level* ℓ *at position* j *of input* w if there exist $1 \le i_1 < \ldots < i_\ell < j$ such that, for all $l = 1, \ldots, \ell$, $w_{i_l,j}$ is fully processed by M. We say that M has an *unbounded activity level* if for any $\ell \ge 1$ there is a string w and a letter position j such that M has activity level ℓ at position j of input w. We define $\ell_M = \infty$ if M has an unbounded activity level, and as the highest possible activity level otherwise. Note that ℓ_M represents the largest possible number of mutually overlapping explorations when M is used as a lookahed DFA as described in Section 1.

3 DFAs with a Bounded Activity Level

Section 3.1 characterizes DFAs with an unbounded activity level: exactly the "prefix-cyclic" DFAs M are the ones with $\ell_M = \infty$. In Section 3.2, it is shown that $\ell_M \le 2^t - 1$ for any non prefix-cyclic DFA with $t+1$ states. It is furthermore shown that there exists a non prefix-cyclic DFA M with $t + 1$ states and $\ell_M = 2^t - 1$ (so that the general upper bound is tight).

3.1 Characterization of DFAs with an Unbounded Activity Level

A DFA $M = (Q, \Sigma, \delta, q_0)$ with a partially defined transition function δ is said to be *prefix-cyclic* if it satisfies the following condition:

$$\exists q \in Q, \exists w \in \Sigma^+ : \delta^*(q_0, w) = q = \delta^*(q, w) \tag{1}$$

DFAs with an unbounded activity level can be characterized as follows:

Theorem 1 ([8]). *A DFA M is prefix-cyclic iff it has an unbounded activity level.*

Proof. If M is prefix-cyclic as witnessed by $q \in Q$ and $w \in \Sigma^+$, then the strings $(w^\ell)_{\ell \geq 1}$ and the letter positions $i_l = 1 + (l-1)|w|$ for $l = 1, \ldots, \ell$ witness that M has an unbounded activity level.
Suppose now that M has an unbounded activity level. Let $\ell \geq 1$ be a sufficiently large number whose precise definition is given below. Pick a string w and letter positions $1 \leq i_1 < \ldots < i_\ell < j$ which witness that $\ell_M \geq \ell$. Let K_ℓ denote the complete graph with ℓ nodes. Consider the edge-coloring of K_ℓ where each edge $\{l, l'\}$ such that $l < l'$ is colored $\delta^*(q_0, w_{i_l, i_{l'}})$. Note that this coloring uses $t := |Q|$ colors. Let $r(3,t)$ denote the smallest number of nodes of a complete graph such that any t-coloring of its edges leads to at least one monochromatic triangle.[4] It is well-known [5, 3, 11] that

$$2^t < r(3,t) < 1 + \frac{e - e^{-1} + 3}{2} \cdot t! < 3t! \enspace .$$

Let now $\ell := r(3,t) < 3t!$. Then, with the coloring defined above (as for any t-coloring), K_ℓ has at least one monochromatic triangle. By construction of the coloring, this means that there exist $1 \leq l < l' < l'' < j$ such that $\delta^*(q_0, w_{i_l, i_{l'}}) = \delta^*(q_0, w_{i_l, i_{l''}}) = \delta^*(q_0, w_{i_{l'}, i_{l''}})$. Setting $q := \delta^*(q_0, w_{i_l, i_{l'}})$, we obtain

$$\delta^*(q, w_{i_{l'}, i_{l''}}) = \delta^*(q_0, w_{i_l, i_{l''}}) = \delta^*(q_0, w_{i_{l'}, i_{l''}}) = q$$

so that (1) holds with $w_{i_{l'}, i_{l''}}$ in the role of w. It follows that M is prefix-cyclic. □

We obtain the following

Corollary 2 ([8]). *Suppose that the DFA $M = (Q, \Sigma, \delta, q_0)$ is not prefix-cyclic and has $t+1$ states. Then $\ell_M < r(3,t)$.*

Proof. There can be no string $w \in \Sigma^+$ such that $\delta^*(q_0, w) = q_0$ because, otherwise, Condition (1) would be satisfied with q_0 in the role of q. Assume for sake of contradiction that $\ell_M \geq r(3,t)$. An inspection of the second part of the proof of Theorem 1 shows that this leads to a t-coloring (the color q_0 is not used!) of the complete graph with $r(3,t)$ nodes so that there is a monochromatic triangle and, consequently, M would be prefix-cyclic (in contradiction to the assumption made in Corollary 2). □

3.2 Tight Bounds on the Activity Level (Arbitrary Alphabet)

Suppose $M = (Q, \Sigma, \delta, q_0)$ is not prefix-cyclic. Let $t = |Q| - 1$ denote the number of non-initial states. According to Corollary 2, $\ell_M < r(3,t)$. Typically, bounds obtained from Ramsey theory are far from being tight. We will however show in this section that the upper bound $r(3,t)$ on ℓ_M is not so far from the truth. We begin our considerations with another upper bound on ℓ_M.

Theorem 3. *For any non prefix-cyclic DFA M with $t+1$ states: $\ell_M \leq 2^t - 1$.*

[4] In Ramsey Theory, $r(3,t)$ is known as the "triangular Ramsey Number with t colors".

Proof. Let $\ell = \ell_M$. Pick a string w and letter positions $1 \le i_1 < \ldots < i_\ell < j$ such that, for all $l = 1, \ldots, \ell$, the substrings $w_{i_l,j}$ are fully processed. For convenience, set $i_{\ell+1} = j$. For $l' = 1, \ldots, \ell$, the "l'-snapshot" is defined as the set

$$Q_{l'} := \{\delta^*(q_0, w_{i_l, i_{l'+1}}) :\ l = 1, \ldots, l'\} \subseteq Q \setminus \{q_0\} \ .$$

In other words: if we consider the l' computational processes created by starting M in positions $i_1, \ldots, i_{l'}$, then $Q_{l'}$ records the set of states of these processes when they have reached position $i_{l'+1}$. Note that $Q_{l'} \ne \emptyset$ for all $l' = 1, \ldots, \ell$ so that there can be at most $2^t - 1$ distinct snapshots. All what remains to do is showing that they actually are distinct. Suppose for sake of contradiction that $Q_{l'} = Q_{l''}$ for some $1 \le l' < l'' \le \ell$. It follows that we can push the activity level beyond any given bound m simply by replacing the substring $u = w_{i_{l'}, i_{l''}}$ of w by u^m. As an unbounded activity level would imply that M is prefix-cyclic, we arrived at a contradiction. It follows that the snapshots are distinct and, therefore, $\ell \le 2^t - 1$. □

The following result shows that the bound in Theorem 3 is tight:

Theorem 4. *There exists a non prefix-cyclic DFA M with $t + 1$ states and alphabet size t such that $\ell_M \ge 2^t - 1$.*

Proof. Let $M = (Q, \Sigma, \delta, q_0)$ be given by $Q = \{q_0, q_1, \ldots, q_t\}$, $\Sigma = \{a_1, \ldots, a_t\}$, and

$$\delta(q_i, a_j) = \begin{cases} q_j & \text{if } i < j \\ q_i & \text{if } i > j \\ \text{undefined} & \text{if } i = j \end{cases} \ . \tag{2}$$

The following statements obviously hold for any $q_k \in Q$ and any $w \in \Sigma^+$:

$$\delta^*(q_k, w) = q_k \Leftrightarrow k \ge 2 \wedge w \in \{a_1, \ldots, a_{k-1}\}^+$$
$$\delta^*(q_0, w) = q_k \Rightarrow \text{letter } a_k \text{ occurs in } w$$

It follows that M is not prefix-cyclic because Condition (1) cannot be satisfied. With the following inductively defined strings $w(1), \ldots, w(t)$, we will be able to push the activity level up to $2^t - 1$:

$$w(1) = a_1 \ \text{ and } \ w(k) = w(k-1) a_k w(k-1)$$

The first members of this sequence evolve as follows:

$$w(1) = a_1 \ , \ w(2) = a_1 a_2 a_1 \ , \ w(3) = a_1 a_2 a_1 a_3 a_1 a_2 a_1 \ , \ \ldots$$

Clearly $|w(t)| = 2^t - 1$. We claim that all $2^t - 1$ suffixes of $w(t)$ are fully processed by M (which would readily imply that $\ell_M \ge 2^t - 1$). The claim is obtained from the following observations:

1. The snapshot[5] after reading $w(k)$ contains state q_i with multiplicity 2^{i-1} for $i = 1, \ldots, k$ (and no other states).

[5] Here, the snapshot is considered as a multiset so as to take multiplicities of states into account.

2. The snapshot after reading $w(k)a_{k+1}$ contains the state q_{k+1} with multiplicity 2^k (and no other states).

The second statement immediately follows from the first one. The first statement for $w(k)$ immediately follows inductively from the second statement for $w(k-1)a_k$. □

4 Activity Level of DFAs with a Binary Alphabet

We argue in Section 4.1 that the maximum activity level ℓ_M of a DFA M can be associated with the language $L(M) = L$ generated by M (and can therefore be written ℓ_L). In Section 4.2, we define a mapping $L \mapsto L_{bin}$ that transforms a prefix-closed language over an arbitrary alphabet into a corresponding prefix-closed language over a binary alphabet. It is analyzed how ℓ_L and $\ell_{L_{bin}}$ are related. In Section 4.3, we show that there exists a non prefix-cyclic DFA M over a binary alphabet that has $1 + 3t\lceil \log t \rceil$ states and satisfies $\ell_M \geq 2^t - 1$. A comparison with Theorem 4 shows that the restriction of having a binary input alphabet does not reduce the largest possible activity-level dramatically.

4.1 Activity Level as an Invariant of the Underlying Language

A DFA $M = (Q, \Sigma, \delta, q_0, F)$ with a partially defined transition function δ is called *prefix-closed* if $F = Q$, i.e., all states of M are accepting. Note that, in this case, the language $L(M)$ coincides with the set of input strings which are fully processed by M. A language L is called *prefix-closed* if $w \in L$ implies that every prefix of w belongs to L too. In other words, any extension of a string $w \notin L$ does not belong to L either. Obviously the following holds:

– If M is a prefix-closed DFA, then $L(M)$ is a prefix-closed regular language.
– Any prefix-closed regular language can be be recognized by a prefix-closed DFA.

We define the *maximum activity level of a language* L as follows:

$$\ell_L = \sup\{\ell : \ (\exists w_1, \ldots, w_\ell \in \Sigma^+, \forall l = 1, \ldots, \ell : w_l \cdots w_\ell \in L)\}$$

For ease of later reference, we say that $w_1, \ldots, w_\ell$ *are witnesses for* $\ell_M \geq \ell$ if $w_l \cdots w_\ell \in L$ for $l = 1, \ldots, \ell$.

Let M be a prefix-closed DFA. It is then evident from the definition of ℓ_M and ℓ_L that $\ell_M = \ell_{L(M)}$. Thus, $\ell_M = \ell_{M'}$ for any DFA M' such that $L(M) = L(M')$.

4.2 From an Arbitrary to a Binary Alphabet

For a language L, let $\mathrm{Pref}(L)$ denote the language of all prefixes of strings from L. So L is prefix-closed iff $\mathrm{Pref}(L) = L$.

Let $L \subseteq \Sigma^*$ be a prefix-closed language over the alphabet $\Sigma = \{a_0, \ldots, a_{K-1}\}$, and let $k = \lceil \log K \rceil$ so that every letter a_j can be encoded as a binary string

$\mathrm{bin}(j)$ of length precisely k (with leading zeros if necessary). Let R denote the homomorphism from Σ^* to $\{0,1\}^*$ that is induced by $a_j \mapsto \mathrm{bin}(j)$. Then $R(L) = \{R(w) : \ w \in L\}$ is the image of L under mapping R. Note that the length of any string in $R(\Sigma^*)$ is a multiple of k. The language $L_{bin} = \mathrm{Pref}(R(L))$ is called the *binary version* of L in what follows. Note that L_{bin} is prefix-closed by construction.

Lemma 5. *With these notations, the following holds for any prefix-closed language $L \subseteq \Sigma^*$:*

1. *If $w \in L$ then $R(w) \in L_{bin}$.*
2. *If $x \in L_{bin}$ and $|x|$ is a multiple of k, then there exists a string $v \in L$ such that $x = R(v)$.*

Proof. The first statement is obvious from $w \in L \Leftrightarrow R(w) \in R(L)$ and $R(L) \subseteq \mathrm{Pref}(R(L)) = L_{bin}$. As for the second statement, $x \in L_{bin} = \mathrm{Pref}(R(L))$ implies that there exists a suffix $y \in \{0,1\}^*$ such that $xy \in R(L)$. $xy \in R(L)$ implies that $|xy|$ is a multiple of k and, since $|x|$ is a multiple of k by assumption, $|y|$ is a multiple of k too. The definition of $R(L)$ now implies that there exist strings $v, w \in \Sigma^*$ such that $x = R(v)$, $y = R(w)$ and $vw \in L$. Since L is prefix-closed by assumption, it follows that $v \in L$. □

Theorem 6. *With the above notations, the following holds for any prefix-closed language L:*

1. *$\ell_L = \infty$ iff $\ell_{L_{bin}} = \infty$.*
2. *If $\ell_L < \infty$, then $\ell_L \le \ell_{L_{bin}} \le k \cdot \ell_L + 1$.*

Proof. It suffices to show that, for all $\ell \ge 1$,

$$\ell_L \ge \ell \Rightarrow \ell_{L_{bin}} \ge \ell \ \text{ and } \ell_{L_{bin}} \ge k\ell + 1 \Rightarrow \ell_L \ge \ell \ .$$

Let $w_1, \ldots, w_\ell \in \Sigma^+$ be witnesses for $\ell_L \ge \ell$. It readily follows that the strings $R(w_1), \ldots, R(w_\ell) \in \{0,1\}^+$ are witnesses for $\ell_{L_{bin}} \ge \ell$.
Let now $x_1, \ldots, x_{\ell'} \in \{0,1\}^+$ be witnesses for $\ell_{L_{bin}} \ge \ell'$, i.e.,

$$\forall l = 1, \ldots, \ell' : y_l := x_l \cdots x_{\ell'} \in L_{bin} \ .$$

Let us introduce for the moment the following additional assumption:

$$\forall l = 1, \ldots, \ell' : |y_l| \text{ is a multiple of } k \ , \tag{3}$$

which is equivalent to saying that, for $l = 1, \ldots, \ell'$, $|x_l|$ is a multiple of k. Then the second statement in Lemma 5 (and the fact that R is a homomorphism) let us conclude that there exist strings $w_1, \ldots, w_{\ell'} \in \Sigma^+$ with the following properties:

- For all $l = 1, \ldots, \ell'$, $R(w_l) = x_l$.
- $w_1, \ldots, w_{\ell'}$ are witnesses for $\ell_L \ge \ell' = \ell_{L_{bin}}$.

Of course our assumption of $|y_l|$ being a multiple of k is not justified. However, if $\ell' = k\ell + 1$, we can argue as follows. We put y_l in a bucket with number $|y_l| \bmod k \in \{0, \ldots, k-1\}$. By the pigeon-hole principle, there must exist a number $\kappa \in \{0, \ldots, k-1\}$ such that the bucket with number κ contains $\ell + 1$ suffixes of $x_1 \cdots x_{\ell'}$, say $y_{l_1}, \ldots, y_{l_\ell}, y_{l_{\ell+1}}$ when ordered according to decreasing length. Note that the shortest suffix, $y_{l_{\ell+1}}$, is a common suffix of all the other ones. Let us erase the suffix $y_{l_{\ell+1}}$ from $y_{l_1}, \ldots, y_{l_\ell}$, respectively, and obtain the new sequence $y'_{l_1}, \ldots, y'_{l_\ell}$. Note that $y'_{l_1}, \ldots, y'_{l_\ell}$ still belong to L_{bin} because L_{bin} is prefix-closed. From $|y_l| \bmod k = \kappa$ for $l = 1, \ldots, \ell + 1$, we can conclude that $|y'_l| \bmod k = 0$ for $l = 1, \ldots, \ell$. Thus, assumption (3) on which our previous analysis was based is now satisfied, indeed, with ℓ in the role of ℓ'. It follows that $\ell_L \geq \ell$ provided that $\ell_{L_{bin}} \geq k\ell + 1$. □

4.3 A Lower Bound on the Activity Level (Binary Alphabet)

The following lower bound, valid for a DFA with a binary input alphabet, should be compared with the (only slightly superior) lower bound from Theorem 4 (which however makes use of a DFA with a very large input alphabet).

Theorem 7. *There exists a non prefix-cyclic DFA with a binary input alphabet,* $1 + 3t\lceil \log t \rceil$ *states and an activity level of at least* $2^t - 1$.

Proof. Let $M = (Q, \Sigma, \delta, q_0)$ be the DFA from the proof of Theorem 4 except for the following technical modification: we use alphabet $\Sigma = \{a_0, \ldots, a_{t-1}\}$ instead of $\{a_1, \ldots, a_t\}$, and state set $Q = \{q_{-1}, q_0, \ldots, q_{t-1}\}$ instead of $\{q_0, q_1, \ldots, q_t\}$. Now q_{-1} is the initial state. As before, δ is given by (2).[6] We know from Theorem 4 that $\ell_M \geq 2^t - 1$. Let $L = L(M)$. We will design a DFA $M' = (Q', \{0,1\}, \delta', q'_0)$ for L_{bin}. According to Theorem 6, $\ell_{M'} \geq \ell_M \geq 2^t - 1$. It suffices therefore to make sure that $|Q'| = 1 + 3t\lceil \log t \rceil$ states are sufficient for the design of M'. Let $k = \lceil \log t \rceil$. Q' is now chosen as the union of $\{q'_0\}$ with the following set:

$$\{(b, \kappa, s) : \; b \in \{\text{bin}(i) : i = 0, 1, \ldots, t-1\}, \kappa \in \{0, 1, \ldots, k-1\}, s \in \{<, >, ?\}\}$$

The intuition behind this definition is as follows:

- The computation of M on input $w \in \Sigma^*$ is simulated by running the computation of M' on input $R(w)$ where R is the homomorphism induced by $a_j \mapsto \text{bin}(j)$ from Section 4.2.
- When M is in state q_i, then M' keeps $b = \text{bin}(i) \in \{0,1\}^k$ in its finite control. The parameter κ indicates how many bits of $\text{bin}(j)$ are processed by M' already when M is currently processing symbol a_j. The flag s is set to "<" (resp. ">") if M' already knows that $i < j$ (resp. $i > j$). Before a successful comparison of i and j, the flag s is set to "?".

[6] Reason for the modification: we will map the underlying language L to L_{bin}, and we won't leave the bit pattern bin(0) unused.

It is not hard to see that the transition function δ' of M' can be defined such that M' simulates a transition $\delta(q_i, a_j)$ of DFA M. The main point is that the comparison of the binary encodings of two numbers i and j can easily be done bitwise (where bin(i) is kept in the finite control and bin(j) is processed from left to right on the input tape). The details of this simulation, omitted here due to space constraints, are found in the full version of the paper. □

5 Unbounded Activity Level and Nerode Relation

We briefly remind the reader that the Nerode relation induced by a language $L \subseteq \Sigma^*$, denoted $\overset{L}{\equiv}$, is a right-congruent equivalence relation on Σ^* that has finitely many equivalence classes iff L is regular. The equivalence classes can then be viewed as the states of the so-called Nerode DFA M_L for L.

Theorem 8. *Let L be a prefix-closed regular language. Then, $\ell_L = \infty$ iff there exists a non-empty string $w \in L$ such that $w \overset{L}{\equiv} w^2$.*

The proof of this theorem, omitted here due to space constraints, is based on the fact that the condition $w \overset{L}{\equiv} w^2$ is equivalent to the condition $q := \delta^*(q_0, w) = \delta^*(q, w)$ for the transition function δ of the (prefix-free version of the) Nerode DFA M_L.

6 Some Complexity Issues

The following theorem is immediate from Theorem 1 and from the definition of "prefix-cyclic" in (1):

Theorem 9 ([8]). *It can be decided within $O(t^2)$ steps whether a DFA M of size t has an unbounded activity level.*

Proof. Let q_0 denote the initial state of M, and let $M' = M$. Compute the product automaton of M and M' and check whether its transition graph contains a non-trivial path from (q_0, q) to (q, q) for some state q. □

Suppose now that M is not prefix-cyclic so that it has a bounded activity level. It turns out that the computation of the maximum activity level, ℓ_M, is a hard problem:

Theorem 10. *Given a non prefix-cyclic DFA M with $t + 1$ states and given a threshold T, the problem to decide whether $\ell_M > T$ is PSPACE-complete.*

Proof. Membership in PSPACE can be seen as follows. Guess a string $w = a_1 a_2 \ldots a_n$ letter by letter, start a computation of M on each letter of w so that up to i computations could potentially be active after the first i letters $a_1, \ldots, a_i$ have been processed. For each non-initial node q, keep track of the number $j(q)$ of active computations which are in state q after having processed

$a_1, \ldots, a_i$ (where the variable $j(q)$ must be updated whenever a new letter is processed). Furthermore keep track of $J := \sum_q j(q)$. As soon as $J > T$ accept. Since, at any time, only t numbers in the range from 0 to $T + 1$ are stored, this is a space-efficient non-deterministic procedure for the given decision problem. Because of Savich's theorem, it can be turned into a space-efficient deterministic procedure.
In order to show PSPACE-hardness, we present a polynomial reduction from "Finite Automata Intersection (FAI)" to our problem. FAI, which is known to be PSPACE-complete [9], is the following problem: given $T \geq 2$ and a list $M_1, \ldots, M_T$ of DFAs with the same input alphabet Σ and with one accepting state per DFA, does there exist an input string $w \in \Sigma^*$ that is accepted by every DFA in the list? In the sequel, the initial state of M_j is denoted q_0^j and its (unique) accepting state is denoted q_+^j. We may consider the state sets $Q_1, \ldots, Q_T$ of $M_1, \ldots, M_T$, respectively, as pairwise disjoint. We plan to build a non prefix-cyclic DFA M from $M_1, \ldots, M_T$ such that $\ell_M > T$ iff there exists a string w that is accepted by $M_1, \ldots, M_T$. To this end, let $\vdash, \dashv \notin \Sigma$ be new symbols, and let q_0, q_+ be new states. M with input alphabet $\Sigma \cup \{\vdash, \dashv\}$, state set $\{q_0, q_+\} \cup \bigcup_{i=1}^T Q_i$ and initial state q_0, has precisely the following transitions:

- M inherits all transitions from $M_1, \ldots, M_T$.
- When reading $\vdash$ in state q_0, M moves to state q_0^1.
- When reading $\vdash$ in state q_0^j for some $j < T$, M moves to state q_0^{j+1}.
- When reading $\dashv$ in state q_0 or in state q_+^j for some $j \in \{1, \ldots, T\}$, M moves to state q_+.

Suppose that there is a string w which is accepted by all of $M_1, \ldots, M_T$. Consider the string $\vdash^T w \dashv$ and assume that we start a computation of M on each of the T occurrences of $\vdash$. Note that the computation started on the j-th occurrence of $\vdash$ will be in state q_0^{T+1-j} when reaching the first letter of w. Thus we run a computation on w for each of the given T DFAs. After having processed w, the T active computations are in states $q_+^1, \ldots, q_+^T$, respectively. When processing the final letter of $\dashv$, we again start a new computation of M, which leads to a state transition from q_0 to q_+. In addition, we have the state transitions from q_+^j to q_+ for $j = 1, \ldots, T$. Thus, we have now $T + 1 > T$ active computations running simultaneously (though all of them will be finished at the very next step).
As for the reverse direction, we have to show that an activity level exceeding T can be reached only if there exists a word over the alphabet Σ that is accepted by all of $M_1, \ldots, M_T$. The main technical observations are as follows:

- In state q_0 only the symbols $\vdash$ and $\dashv$ are processed. Thus, a new computation can be successfully started on these two letters only.
- Any occurrence of symbol $\vdash$ will terminate all computations which are not in a state from $\{q_0, q_0^1, \ldots, q_0^{T-1}\}$.
- In state q_+ no symbol is processed.
- Symbol $\dashv$ is processed only when M is in state q_0 or q_+^j for some $j \in \{1, \ldots, T\}$. Thereafter, all still active computations are in state q_+ (so that these computations are terminated in the next step).

From these observations, it easily follows that we have at most T active computations (one for each of the DFAs $M_1, \dots, M_T$) as long as $\dashv$ is not processed. If $\dashv$ is processed, the number of active computations can be at most $T+1$ (implying that M is not prefix-cyclic). Moreover the case of $T+1$ active computations can occur only on strings which contain the symbols $\vdash$ and $\dashv$ and which have the property that the word between the last occurrence of $\vdash$ and the first occurrence of $\dashv$ is accepted by all of $M_1, \dots, M_T$. $\square$

The procedure for the determination of ℓ_M that was suggested in the proof of Theorem 10 shows membership in PSPACE but is not a realistic one. We briefly sketch a procedure that is realistic at least for DFAs with a small number of states. It is based on the notion of a "snapshot" (similar to the notion that was used in the proof of Theorem 3). Let $M = (Q, \Sigma, \delta, q_0)$ be the given non prefix-cyclic DFA. A set $S \subseteq Q \setminus \{q_0\}$ is called a *snapshot* if $S = \emptyset$ or if there exists a string $w = a_1 \dots a_n$ and letter positions $1 \le i_1 < \dots i_\ell \le n$ such that $S = \{\delta^*(q_0, w_{i_l,n+1}) : \ l = 1, \dots, \ell\}$ and such that all strings $w_{i_l,n+1}$ are fully processed. We build a directed snapshot graph $G = (V, E)$ as follows. V is defined as the set of snapshots. V and E are computed iteratively as follows:

1. Initially set $V := \{\emptyset\}$ and $E := \emptyset$.
2. For each $S \in V$ and for each $a \in \Sigma$ such that $\delta(q, a)$ is defined for all $q \in S$, add $S' := \{\delta(q, a) : \ q \in S\}$ and $S'' := S' \cup \{\delta(q_0, a)\}$ to V. Moreover, add the edges (S, S') and (S, S'') to E, where edge (S, S'') is declared "special".

The second step is applied to every node only once. Intuitively, an edge represents a possible next a-transition of a collection of currently active DFA-explorations (provided that no exploration terminates when processing a), where a special edge reflects the option to start a new computation on a. An example is shown in Fig. 2.

It is easy (and similar to the proof of Theorem 3) to show the following:

– The strongly connected components of $G = (V, E)$ do not contain special edges (because, otherwise, $\ell_M = \infty$).
– If we assign length 1 to special edges and length 0 to all remaining ones, then ℓ_M coincides with the total length of a longest path in G.

In order to compute the total length of the longest path efficiently, an auxiliary directed acyclic "super-graph" G' is computed as follows:

1. The "super-nodes" in G' are the strongly connected components of G.
2. An edge e' is drawn from a strongly connected component K_1 to another strongly connected component K_2 iff E contains an edge e leading from a node in K_1 to a node in K_2. The edge e' is declared "special" iff the underlying edge $e \in E$ can be chosen as a special one.

Since strongly connected components do not contain special edges, the total length of the longest path in G equals the total length of the longest path in G'. But the latter quantity is easy to compute provided that the nodes of G' are processed in topological order. If everything is implemented properly then the run-time is linear in the size of snapshot graph G. Note, however, that this size can be exponential in the size t of the given DFA M.

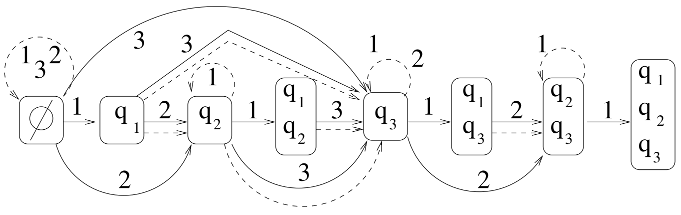

Fig. 2. The snapshot graph induced by the DFA from the proof of Theorem 4 for $t = 3$: an edge is labeled i if it represents an a_i-transition. Special edges are solid, the remaining ones are dashed.

Acknowledgements

We would like to thank four anonymous referees for their comments and suggestions. We furthermore thank Eberhard Bertsch for communicating the problem statement to us and for many helpful suggestions.

References

1. Bermudez, M.E., Schimpf, K.M.: Practical arbitrary lookahead LR parsing. Journal of Computer and System Sciences 41(2), 230–250 (1990)
2. Boullier, P.: Contribution à la construction automatique d'analyseurs lexicographiques et syntaxiques. Ph.D. thesis, Université d'Orléans (1984)
3. Chung, F.R.K., Grinstead, C.M.: A survey of bounds for classical Ramsey numbers. Journal of Graph Theory 7(1), 25–37 (1983)
4. Farré, J., Gálvez, J.F.: A bounded graph-connect construction for LR-regular parsers. In: Proceedings of the 10th International Conference on Compiler Construction. pp. 244–258 (2001)
5. Fredricksen, H.: Schur numbers and the Ramsey numbers N(3,3,...,3;2). Journal of Combinatorial Theory, Series A 27(3), 376–377 (1979)
6. Gosling, J., Joy, B., Steele, G.: The Java$^{\text{TM}}$ Language Specification. Addison-Wesley (1996)
7. Knuth, D.E.: On the translation of languages from left to right. Information and Control 8(6), 607–639 (1965)
8. Konitzer, M.: Laufzeitanalyse und Optimierung von Parsern für LR-reguläre Grammatiken. Ph.D. thesis, Ruhr-University Bochum (2013)
9. Kozen, D.: Lower bounds for natural proof systems. In: Proceedings of the 18th Symposium on Foundations of Computer Science. pp. 254–266 (1977)
10. Schmitz, S.: Approximating Context-Free Grammars for Parsing and Verification. Ph.D. thesis, Université de Nice-Sophia Antipolis (2007)
11. Wan, H.: Upper bounds for Ramsey numbers R(3,3,...,3) and Schur numbers. Journal of Graph Theory 26(3), 119–122 (1997)
12. Čulik, K., Cohen, R.: LR-regular grammars - an extension of LR(k) grammars. Journal of Computer and System Sciences 7(1), 66–96 (1973)