Regular Languages Viewed from a Graph-Theoretic Perspective

Marius Konitzer and Hans Ulrich Simon

Horst-Görtz Institute for IT Security and Faculty of Mathematics
Ruhr-Universität Bochum, Germany

Abstract

In this paper[1], we consider the graph $G(L|w)$, resp. the directed graph $\vec{G}(L|w)$, associated with an arbitrary language $L \subseteq \Sigma^*$ and an arbitrary string $w \in \Sigma^*$. The clique number of L is then defined as the supremum of the clique numbers of the graphs $G(L|w)$ where w ranges over all strings in Σ^*. The maximum in- or outdegree of L is defined analogously. We characterize regular languages with an infinite clique number and determine tight upper bounds in the finite case. We obtain analogous results for the maximum indegree and the maximum outdegree of a regular language. As an application, we consider the problem of determining the maximum activity level of a prefix-closed regular language — a parameter that is related to the computational complexity of parsing techniques utilizing unbounded regular lookahead. Finally, we determine the computational complexity of various problems arising from our graph-theoretic approach.

Keywords: regular language, lookahead DFA, clique number, maximum degree, computational complexity, query learning, minimum adequate teacher

1. Introduction

In Section 1.1, we describe the connection between LR-regular parsing and the maximum activity level[2] of the lookahead DFA utilized during parsing. In Section 1.2, we describe the more general approach taken in this paper:

[1] an extended version of (Konitzer and Simon, 2014)

[2] which was the central notion in the conference version of this paper

we look at languages from a broader graph-theoretic perspective and relate structural properties of regular languages to properties of the (di-)graphs induced by them.

1.1. LR-regular Parsing and Lookahead DFAs

LR-regular (LRR) parsing (Čulik and Cohen, 1973) is one of the few parsing techniques utilizing unbounded lookahead. LRR languages properly include the deterministic context-free languages (Knuth, 1965). LRR parsers allow for a large number of interesting grammars with practical relevance (such as the original version of the Java language (Gosling et al., 1996), which cannot be handled by any LR(k) parser). The parsers generated with the algorithm from (Čulik and Cohen, 1973) 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 (Bermudez and Schimpf, 1990) and (Farré and Gálvez, 2001) basically work like the well known LR(k) shift-reduce parsers (Knuth, 1965), 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 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 deterministic finite automata (DFAs), so called *lookahead DFAs*, which are used for lookahead exploration during parsing whenever necessary. Different explorations of the same lookahead DFA, operating on a common substring of the input string, may overlap each other. As formally defined in Section 2, this leads to the notion of the maximum activity level of a prefix-closed regular language.

If the number of mutually overlapping explorations on strings of length n is bounded from above by some $B \leq n$, the whole parser has time bound $O(Bn)$ on inputs of length n (as illustrated in Fig. 1). It turns out that either B is a constant (in the case of a bounded activity level) or B may grow linearly with the length n of an input string (in the case of an unbounded activity level). In the latter case, parsing may take $O(n^2)$ steps.[3] In the

[3] An LRR-grammar leading to quadratic run time for parsing with (unbounded) lookahead DFAs is found in (Schmitz, 2007).

former case, the time bound is linear in n.[4] But note that B may still depend on the size of the employed lookahead DFA, which would become an issue once we think in terms practical LRR parser generators.

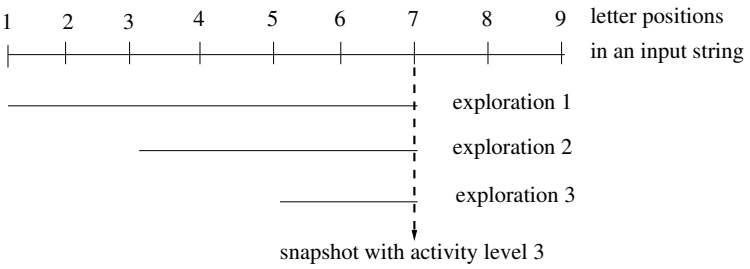

Figure 1: $B = 3$ mutually overlapping lookahead explorations.

1.2. A Graph-theoretic View on Languages

We consider in this paper arbitrary regular languages and several graph-theoretic parameters. By associating a family of (di-)graphs with a language L, we will be in the position to formally define the clique number and the maximum in- or outdegree of a given language L. Each of these numbers is defined as a supremum over a family of (di-)graphs and is therefore possibly infinite. Our main results are as follows:

- We characterize the regular languages with an infinite clique number and derive a tight upper bound on the clique number in the finite case. See Section 3.
- We obtain analogous results for the maximum in- and the maximum outdegree of a regular language. See Section 4.
- We show that it can be efficiently decided whether a regular language, given as a DFA, has a finite clique number (resp. a finite maximum indegree or a finite maximum outdegree). Moreover the maximum outdegree of L (given as a DFA) can be efficiently determined. This algorithm can also be used for the computation of the maximum indegree

[4]See (Konitzer, 2013) for several grammar constructs leading to lookahead DFAs with a bounded activity level (e.g. HTML forms (Farré and Gálvez, 2001) and Ada calls (Boullier, 1984; Schmitz, 2007)).

of a regular language L, but then it runs in time polynomial in the sizes of the minimum DFAs for L *and* the reverse of L (which can be exponential in the size of the minimum DFA for L). See Section 5.1.

- The problem of deciding whether the maximum clique number of a regular language L (given as a DFA) is above a given threshold is shown to be PSPACE-complete (even if L is assumed to be prefix-closed). The corresponding problem for the maximum indegree is PSPACE-complete too. The corresponding problem for the maximum outdegree is PSPACE-complete if L is given as an NFA. See Section 5.2.

As explained in Section 2, the maximum activity level of a prefix-closed regular language L coincides with the maximum indegree of L (with the obvious implications on the computational complexity of computing the maximum activity level).

2. Definitions, Notations, Facts and Easy Observations

In Section 2.1, we call to mind some basic notions and facts in the theory of finite automata. Readers being familiar with the theory of finite automata may skip this section without much loss of continuity. In Section 2.2, we look at regular languages from a graph-theoretic perspective thereby presenting some central notions along with some easy observations and some useful facts.

2.1. Regular Languages and Finite Automata

Let M be a deterministic finite automaton (DFA) given by its finite set of states, Q, its input alphabet, Σ, its transition function, $\delta : Q \times \Sigma \to Q$, its initial state, $q_0 \in Q$, and its set of final (accepting) states, $F \subseteq Q$. The states from $Q \setminus F$ are said to be *non-final*. A *trap state* is a state q from Q such that $\delta(q, a) = q$ for all $a \in \Sigma$ (so that a computation that reaches the state q cannot escape from there). As usual, Σ^* denotes the set of strings over Σ, including the empty string ε, and $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$. The mapping δ can be extended to a mapping $\delta^* : Q \times \Sigma^* \to Q$:

$$\delta^*(q, \varepsilon) = q \quad \text{and} \quad \delta^*(q, aw) = \delta^*(\delta(q, a), w) \ .$$

Here, $q \in Q$, $a \in \Sigma$, and $w \in \Sigma^*$. $\delta^*(q, w)$ is the state reached by a computation of M that was started in state q and has processed all letters of w. A state $q \in Q$ is said to be *superfluous* if at least one of the following conditions is satisfied:

- q is not reachable from the start state, i.e., there does not exist a string $w \in \Sigma^*$ such that $\delta^*(q_0, w) = q$.
- q is not productive, i.e., there does not exist a string $w \in \Sigma^*$ such that $\delta^*(q, w) \in F$ although q is not a non-final trap state.

Every DFA can be simplified so as to avoid superfluous states.

The *transition diagram* of a DFA $M = (Q, \Sigma, \delta, q_0, F)$ is a directed graph $D_M = (Q, E)$ with node- and edge-labels and with $q_0 \in Q$ as a distinguished start node. Nodes from F are labeled as "final (accepting)" while the remaining nodes are labeled "non-final". For every transition of the form $\delta(q, a) = q'$, E contains an arc (q, q') that is labeled "a". Note that, for all $q, q' \in Q$ and $w \in \Sigma^*$, $\delta^*(q, w) = q'$ iff the unique path $P_M(q, w)$ that starts from q and whose edges are labeled by the letters of w ends in q'. Let

$$\begin{aligned} L(M) &= \{w \in \Sigma^* : \ \delta^*(q_0, w) \in F\} \\ &= \{w \in \Sigma^* : \ P_M(q_0, w) \text{ ends in a node from } F\} \ . \end{aligned}$$

A language L is said to be *recognized by* M if $L = L(M)$. The class of languages that can be recognized by some DFA is precisely the class of regular languages. Suppose that L is a regular language. The *Nerode relation* is the following equivalence relation on $\Sigma^* \times \Sigma^*$:

$$u \equiv_L v \Leftrightarrow (\forall w \in \Sigma^* : uw \in L \Leftrightarrow vw \in L) \ .$$

The equivalence class containing the string $u \in \Sigma^*$ is denoted $[u]_L$. The following facts are well known. L is regular if and only if the Nerode relation has only finitely many equivalence classes. For all strings $u, v, x \in \Sigma^*$, $u \equiv_L v$ implies that $ux \equiv_L vx$. Moreover, $u \equiv_L v$ and $u \in L$ implies that $v \in L$. These observations make sure that the following automaton $M_L = (Q_L, \Sigma, \delta_L, q_0, F_L)$, called the *Nerode automaton* for L, is a well defined DFA which recognizes L:

- $Q_L = \{[u]_L : u \in \Sigma^*\}$, $q_0 = [\varepsilon]_L$ and $F_L = \{[u]_L : u \in L\}$.
- For all $u \in \Sigma^*$ and $a \in \Sigma$, $\delta_L([u]_L, a) = [ua]_L$.

It is well known that M_L is a DFA with the smallest number of states among all DFAs that recognize L and, with this property, M_L is unique up to renaming of states. For this reason, the Nerode automaton for L is sometimes called the *minimum DFA* for L.

A non-deterministic finite automaton (NFA) M is given by its finite set of states, Q, its input alphabet, Σ, its transition function $\delta : Q \times \Sigma \to 2^Q$, its set of initial states, Q_0, and its set of final (accepting) states, F. Here 2^Q denotes the powerset of Q. The mapping δ can be extended to a mapping $\delta^* : Q \times \Sigma^* \to 2^Q$:

$$\delta^*(q, \varepsilon) = \{q\} \quad \text{and} \quad \delta^*(q, aw) = \bigcup_{q' \in \delta(q,a)} \delta^*(q', w) \ .$$

The language that is recognized by M is then given by

$$L(M) = \{w \in \Sigma^* : \ (\exists q \in Q_0 : \ \delta^*(q, w) \cap F \neq \emptyset)\} \ .$$

The definition of a transition diagram of an NFA M is similar to the definition of a transition diagram of a DFA. Here nodes are labeled as either initial or non-initial and as either final or non-final. We draw an arc labeled a from q to q' iff $q' \in \delta(q, a)$.

Two finite automata are said to be *equivalent* if they recognize the same language. Every NFA $M = (Q, \Sigma, \delta, Q_0, F)$ can be transformed into an equivalent DFA $M' = (Q', \Sigma, \delta', q_0', F')$ by setting $Q' = 2^Q$ (the powerset of Q), $q_0' = Q_0$, $F' = \{P \subseteq Q : P \cap F \neq \emptyset\}$ and, for each $P \subseteq Q$ and each $a \in \Sigma$, by setting $\delta'(P, a) = \cup_{q \in P} \delta(q, a)$. An algorithm that performs this transformation can be made more efficient by generating only those states from $Q' = 2^Q$ that can be reached from the initial state $q_0' = Q_0$. In what follows, we refer to this (more efficient) procedure simply as the "powerset construction".

Given the DFAs $M_i = (Q_i, \Sigma, \delta_i, q_{0,i}, F_i)$ for $i = 1, 2$, the DFA $M = (Q, \Sigma, \delta, q_0, F)$ such that $Q = Q_1 \times Q_2$, $q_0 = (q_{0,1}, q_{0,2})$, and $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$ is called the *product (automaton)* of M_1 and M_2. Note that $L(M) = L(M_1) \cap L(M_2)$ if we set $F = F_1 \times F_2$, and $L(M) = L(M_1) \cup L(M_2)$ if we set $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$.

A language $L \subseteq \Sigma^*$ is called *prefix-free* (resp. *suffix-free*) if, for each $w \in L$, no proper prefix (resp. no proper suffix) of w belongs to L. Let $\text{Pref}(L) = \{u \in \Sigma^* : (\exists w \in L : \ u \text{ is prefix of } w)\}$ and $\text{Suff}(L) = \{u \in \Sigma^* : (\exists w \in L : \ u \text{ is suffix of } w)\}$ L is called *prefix-closed* (resp. *suffix-closed*) if $\text{Pref}(L) = L$ (resp. $\text{Suff}(L) = L$). A DFA $M = (Q, \Sigma, \delta, q_0, F)$ such that $F = Q \setminus \{q_-\}$ for a non-final trap state $q_- \in Q$ is said to be *prefix-closed*. It is obvious that $L(M)$ is a prefix-closed language if M is a prefix-closed DFA. Moreover, each prefix-closed regular language can be recognized by some prefix-closed DFA. For instance, the Nerode automaton for a prefix-closed regular language is prefix-closed.

For a word $w = a_1 \dots a_n \in \Sigma^n$, we set $w^R = a_n \dots a_1$. For a language $L \subseteq \Sigma^*$, we set $L^R = \{w^R : \ w \in L\}$. Suppose that $M = (Q, \Sigma, \delta, Q_0, F)$ is an NFA that recognizes L. Then the NFA $M^R = (Q, \Sigma, \delta^R, F, Q_0)$ such that

$$\delta^R(q', a) = \{q \in Q : \ q' \in \delta(q, a)\} \tag{1}$$

recognizes L^R. Note that the final (resp. initial) states of M become the initial (resp. final) states of M^R. Equation (1) means that the arcs in the transition diagram of M^R are obtained from the arcs in the transition diagram of M by reversing their orientations, respectively. We will refer to the whole procedure of computing M^R from M as "reversing the transition diagram" of M.

As for a DFA M (which can be considered as a special case of an NFA in the obvious fashion), we briefly note the following

Lemma 2.1. *Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA, M^R the corresponding NFA as defined above, and M^R_{det} the DFA obtained from M^R by the powerset construction. Then, M^R_{det} has at most $2^{|Q|-1}$ final states. Moreover, if M is prefix-closed with a non-final trap state q_-, then the state set of M^R can be set to $Q \setminus \{q_-\}$ and M^R_{det} has at most $2^{|Q|-2}$ final states.*

Proof The final states of M^R_{det} are given by the subsets of Q which contain q_0 (except for those which are not reachable from the initial state of M^R_{det}). Moreover, if M is prefix-closed, then q_- is not reachable from the initial states of M^R so that M^R has the state set $Q \setminus \{q_-\}$. In this case, the final states of M^R_{det} are given by the subsets of $Q \setminus \{q_-\}$ which contain q_0 (except for those which are not reachable from the initial state of M^R_{det}). □

The following observation of Brzozowski forms the basis of his algorithm for DFA minimization:

Lemma 2.2 (Brzozowski (1962)). *Let M be a DFA all of whose states are reachable from the start state and let $L = L(M)$. Then the powerset construction[5] applied to the NFA M^R yields a minimum DFA for L^R.*

This clearly implies that, given a DFA M, the minimum DFA for $L(M)^R$ can be computed in time polynomial in the sizes of M and the minimum DFA

[5]Recall our convention that the powerset construction generates reachable states only.

for L^R. As already exploited by Brzozowski (1962), another application of the powerset construction, now to the reverse of the minimum DFA for L^R, yields the minimum DFA for $L(M)$.

2.2. A Graph-theoretic Perspective

Let $L \subseteq \Sigma^*$ be a language over the alphabet Σ. Let $w = a_1 \cdots a_n \in \Sigma^n$. Then, for all $1 \leq i < j \leq n+1$, $w_{i,j}$ denotes the substring $a_i \cdots a_{j-1}$. The graph $G(L|w) = (V, E)$ is defined by setting $V = \{1, \ldots, n+1\}$ and

$$E = \{\{i, j\} : \ 1 \leq i < j \leq n+1 \text{ and } w_{i,j} \in L\} \ .$$

We briefly note that $V = \{1\}$ and $E = \emptyset$ for $w = \varepsilon$.

Let $G = (V, E)$ be a graph. As usual, a set $C \subseteq V$ whose nodes are pairwise adjacent in G is called a *clique* in G. The *clique number* of G, denoted $\omega(G)$, is the cardinality of the largest clique in G. In the sequel, the clique number of $G(L|w)$ is simply denoted as $\omega(L|w)$. We define the *clique number of a language* $L \subseteq \Sigma^*$ as follows:

$$\omega(L) = \sup_{w \in \Sigma^+} \omega(L|w) \ .$$

We say that L has a *finite clique number* if $\omega(L) < \infty$.

The directed counterpart of the graph $G(L|w)$ for a language L and a string $w = a_1 \cdots a_n \in \Sigma^n$ is the digraph $\vec{G}(L|w) = (V, A)$ given by $V = \{1, \ldots, n+1\}$ and

$$A = \{(i, j) : \ 1 \leq i < j \leq n+1 \text{ and } w_{i,j} \in L\} \ .$$

Let $\vec{G} = (V, A)$ be a digraph. Recall that the *outdegree* (resp. *indegree*) of a node v in G, denoted $d^+_G(v)$ (resp. $d^-_G(v)$) is the number of arcs from A leaving v (resp. entering v). The maximum outdegree and the maximum indegree in $\vec{G}$ are then given by

$$d^+_{max}(\vec{G}) = \max_{v \in V} d^+_G(v) \ \text{ and } \ d^-_{max}(\vec{G}) = \max_{v \in V} d^-_G(v) \ ,$$

respectively. In the sequel, the maximum outdegree of $\vec{G}(L|w)$ is simply denoted as $d^+_{max}(L|w)$. The analogous remark applies to the maximum indegree of $\vec{G}(L|w)$. The *maximum outdegree* and the *maximum indegree of a language* $L \subseteq \Sigma^*$ are then given by

$$d^+_{max}(L) = \sup_{w \in \Sigma^+} d^+_{max}(L|w) \ \text{ and } \ d^-_{max}(L) = \sup_{w \in \Sigma^+} d^-_{max}(L|w) \ ,$$

respectively. We say that L has a *finite maximum outdegree* (resp. a *finite maximum indegree*) if $d^+_{max}(L) < \infty$ (resp. $d^-_{max}(L) < \infty$).

If $L = \emptyset$ or $L = \{\varepsilon\}$, then the graphs $G(L|w)$ resp. $\vec{G}(L|w)$ contain isolated vertices only. All other languages L have a clique number of at least 2 and a maximum outdegree (resp. maximum indegree) of at least 1 as it will become evident from the following

Example 2.3. *Let $L \subseteq \Sigma^*$ be a language containing at least one non-empty string. Let w be a shortest non-empty string in L, say $|w| = n \geq 1$. Setting $V = \{1, \ldots, n, n+1\}$, it follows that $G(L|w) = (V, \{\{1, n+1\}\})$ and $\vec{G}(L|w) = (V, \{(1, n+1)\})$. It follows that $\omega(L) \geq \omega(L|w) = 2$, $d^+_{max}(L) \geq d^+_{max}(L|w) = 1$ and $d^-_{max}(L) \geq d^-_{max}(L|w) = 1$.*

For a graph G with nodes numbered from 1 to n, we denote by $\vec{G}$ the digraph resulting from G by orienting each edge in direction to the larger node number. Clearly, $d^+_{max}(\vec{G}) \geq \omega(G) - 1$ and $d^-_{max}(\vec{G}) \geq \omega(G) - 1$. Since this holds specifically for the graph $G(L|w)$ and the corresponding digraph $\vec{G}(L|w)$, we get:

Observation 1: Let $L \subseteq \Sigma^*$ be a language. Then, $d^+_{max}(L) \geq \omega(L) - 1$ and $d^-_{max}(L) \geq \omega(L) - 1$.

We write $G_1 \simeq G_2$ to indicate that a graph G_1 is isomorphic to another graph G_2. The same notation is used for digraphs. If $\vec{G}$ is a digraph, then we denote by $\vec{G}^R$ the digraph obtained from $\vec{G}$ by reversing the orientation of each edge. With these notations (and with the notations w^R and L^R from Section 2.1), the following is fairly obvious:

Observation 2: $G(L|w) \simeq G(L^R|w^R)$ and $\vec{G}(L|w) \simeq \vec{G}(L^R|w^R)^R$.

Clearly isomorphic graphs have the same clique number, $d^+_{max}(\vec{G}) = d^-_{max}(\vec{G}^R)$ and $d^-_{max}(\vec{G}) = d^+_{max}(\vec{G}^R)$. Since this particularly applies to the graphs $G(L|w)$ and $\vec{G}(L|w)$, we get:

Corollary 2.4. *Let $L \subseteq \Sigma^*$ be a language. Then, $\omega(L) = \omega(L^R)$, $d^+_{max}(L) = d^-_{max}(L^R)$ and $d^-_{max}(L) = d^+_{max}(L^R)$.*

The following result (whose easy proof is included for the sake of completeness) will be used several times in this paper:

Lemma 2.5. *For every language $L \subseteq \Sigma^*$, the following holds:*

$$\omega(L) \geq \ell \;\Leftrightarrow\; \exists u_1, \ldots, u_{\ell-1} \in \Sigma^+, \forall 1 \leq i \leq j \leq \ell - 1 : u_i \ldots u_j \in L. \quad (2)$$

$$d^+_{max}(L) \geq \ell \;\Leftrightarrow\; \exists u_1, \ldots, u_\ell \in \Sigma^+, \forall 1 \leq i \leq \ell : \; u_1 \ldots u_i \in L \; . \quad (3)$$

$$d^-_{max}(L) \geq \ell \;\Leftrightarrow\; \exists u_1, \ldots, u_\ell \in \Sigma^+, \forall 1 \leq i \leq \ell : \; u_i \ldots u_\ell \in L \; . \quad (4)$$

Proof We present the proof for the clique number. (The proof for d^+_{max} resp. d^-_{max} is quite similar.)
Clearly, if $\omega(L) \geq \ell$, there must be a "witness" $w \in \Sigma^n$ such that $\omega(L|w) \geq \ell$. It is easy to see that a minimal string w with this property (so that no proper substring u of w satisfies $\omega(L|u) \geq \ell$) satisfies $\omega(G|w) = \ell$ and each clique of size ℓ in $G(L|w)$ contains the nodes 1 and $n+1$. Thus, C is of the form $C = \{i_1, i_2, \ldots, i_{\ell-1}, i_\ell\}$ for $1 = i_1 < i_2 < \ldots < i_{\ell-1} < i_\ell = n+1$. Then the strings $u_j = w_{i_j, i_{j+1}}$ for $j = 1, \ldots, \ell - 1$ satisfy condition (2).
Suppose conversely that condition (2) is valid. Let $w = u_1 \ldots u_{\ell-1}$ and $i_j = 1 + |u_1| + \ldots + |u_{j-1}|$ for $j = 1, \ldots, \ell$. It follows that $\{i_1, \ldots, i_\ell\}$ is a clique in $G(L|w)$ so that $\omega(L) \geq \omega(L|w) \geq \ell$. □

The following result characterizes languages with maximum outdegree (resp. maximum indegree) 1. Moreover, it relates the maximum clique number of a prefix-closed (resp. suffix-closed) language to its maximum indegree (resp. maximum outdegree).[6]

Corollary 2.6. 1. *L is prefix-free iff $d^+_{max}(L) = 1$.*
2. *L is suffix-free iff $d^-_{max}(L) = 1$.*
3. *If L is prefix-closed, then $\omega(L) = d^-_{max}(L) + 1$.*
4. *If L is suffix-closed, then $\omega(L) = d^+_{max}(L) + 1$.*

Proof We start with the first statement. A language $L \subseteq \Sigma^*$ is not prefix-free iff there exist two strings $u_1, u_2 \in \Sigma^+$ such that $u_1, u_1 u_2 \in L$. According to Lemma 2.5, this is equivalent to $d^+_{max}(L) \geq 2$.
The second statement is proved analogously.
We proceed with the proof of the third statement. According to Observation 1 in Section 2, $d^-_{max}(L) \geq \omega(L) - 1$. It suffices therefore to show that $\omega(L) \geq d^-_{max}(L) + 1$. Let $\ell = d^-_{max}(L) + 1$. Choose $u_1, \ldots, u_\ell \in \Sigma^+$ according to

[6]The latter relation makes use the convention $\infty = \infty + 1$.

condition (4) from Lemma 2.5. Since L is assumed as prefix-closed, it follows that condition (2) from Lemma 2.5 (with $\ell + 1$ in the role of ℓ) is valid too. Another application of Lemma 2.5 yields that $\omega(L) \geq \ell + 1$.
The fourth statement is proved analogously. □

At the end of this section, we come back to lookahead DFAs that are utilized for resolving parsing conflicts. Recall from the introduction and from Figure 1 that, given a lookahead DFA M, we are only interested in the question of which substrings of an input string are fully processed by M. Clearly the language of all strings that are fully processed by M is prefix-closed. Thus it should be clear that the following definition (with $L(M)$ in the role of L) captures how many computations of M can be active at the same time if M starts a computation at several positions of the same input string:

Definition 2.7. *The* maximum activity level *of a prefix-closed regular language L is denoted as $\ell(L)$ and defined as follows:*

$$\ell(L) = \sup\{\ell : \ (\exists u_1, \ldots, u_\ell \in \Sigma^+, \forall i = 1, \ldots, \ell : u_i \cdots u_\ell \in L)\} \ .$$

An inspection of Lemma 2.5 shows that $\ell(L) = d^-_{max}(L)$. Moreover, since Definition 2.7 is concerned with prefix-closed regular languages only, we also get $\ell(L) = \omega(L) - 1$ according to the third statement in Corollary 2.6. Thus, the study of $\omega(L)$ and $d^-_{max}(L)$ for arbitrary regular languages collapses to the study of $\ell(L)$ whenever we specialize our results to regular languages that are prefix-closed.

3. The Clique Number of a Regular Language

The following result characterizes the regular languages with an infinite clique number:

Theorem 3.1. *Let $L \subseteq \Sigma^*$ be a regular language. Then the following statements are equivalent:*

1. $\omega(L) = \infty$.
2. *For every DFA $M = (Q, \Sigma, \delta, q_0, F)$ such that $L(M) = L$, the following holds:*
$$\exists q \in F, \exists w \in \Sigma^+ : q = \delta^*(q_0, w) = \delta^*(q, w) \tag{5}$$

3. $\exists w \in L \setminus \{\varepsilon\} : w \equiv_L w^2$.
4. $\exists w \in L \setminus \{\varepsilon\}, \forall i \geq 1 : w^i \in L$.

Proof We first prove that the first statement implies the second one. Suppose that L has an infinite clique number. Let $\ell \geq 1$ be a sufficiently large number whose precise definition is given below. Choose strings $u_1, \ldots, u_{\ell-1} \in \Sigma^+$ according to condition (2) from Lemma 2.5. For the sake of brevity, let $w_{i,j} = u_i \ldots u_{j-1}$ for all $1 \leq i < j \leq \ell$. Let K_ℓ denote the complete graph with ℓ nodes. Consider the edge-coloring of K_ℓ where each edge $\{i, j\}$ such that $i < j$ is colored by $\delta^*(q_0, w_{i,j})$. Note that this coloring uses $|F|$ colors. Let $t = |F|$ and 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.[7] It is well known (Fredricksen, 1979; Chung and Grinstead, 1983; Wan, 1997) that

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

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 $1 \leq i < j < k \leq \ell - 1$ such that $\delta^*(q_0, w_{i,j}) = \delta^*(q_0, w_{i,k}) = \delta^*(q_0, w_{j,k})$. Setting $q := \delta^*(q_0, w_{i,j})$, we obtain

$$\delta^*(q, w_{j,k}) = \delta^*(q_0, w_{i,k}) = \delta^*(q_0, w_{j,k}) = q$$

so that the second statement holds with $w_{j,k}$ in the role of w.
The second statement implies the third one because we may choose M especially as the Nerode automaton for L. Then M satisfies $\delta^*(q_0, x) = [x]_L$ for all $x \in \Sigma^*$. From $q = \delta^*(q_0, w) = \delta^*(q, w)$, we may therefore conclude that

$$[w]_L = \delta^*(q_0, w) = \delta^*(q, w) = \delta^*(q_0, w^2) = [w^2]_L \ .$$

We show now that the third statement implies the fourth one. The condition $w \equiv_L w^2$ implies that $w^2 = ww \equiv_L w^2 w = w^3$ so that w, w^2, w^3 are Nerode-equivalent. Iteration of this argument yields that $w \equiv_L w^i$ for all $i \geq 1$. Since $w \in L$, it follows that $w^i \in L$ for all $i \geq 1$.
We finally show that the fourth statement implies the first one. Suppose that there exists some $w \in L \setminus \{\varepsilon\}$ such that, for all $i \geq 1$, $w^i \in L$. Then, for each $\ell \geq 1$, $\omega(L) \geq \omega(L|w^{\ell-1}) \geq \ell$. Thus, $\omega(L) = \infty$. □

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

For the ease of later reference, we note the following. Condition (5) is satisfied with $q = q_0$ if and only if the following holds:

$$q_0 \in F \wedge (\exists w \in \Sigma^+ : \delta^*(q_0, w) = q_0) \ . \tag{6}$$

This condition can clearly be checked in linear time.

The proof of Theorem 3.1 implies that either $\omega(L) = \infty$ or $\omega(L) < r(3, t)$ since any clique of size $r(3, t)$ or larger allows us to conclude that the second statement in Theorem 3.1 is valid, which implies that $\omega(L) = \infty$. Typically, bounds obtained from Ramsey theory are far from being tight. However, the upper bound $r(3, t)$ on $\omega(L)$ is not so far from the following bound (which later is shown to be tight):

Theorem 3.2. *For every regular language $L \subseteq \Sigma^*$ with Nerode automaton $M = (Q, \Sigma, \delta, q_0, F)$, the following holds. If $\omega(L) < \infty$, then $\omega(L) \leq 2^{|F \setminus \{q_0\}|}$.*

Proof Let $\ell = \omega(L)$. Pick a number $n \geq 1$ and a string $w \in \Sigma^n$ such that $\omega(L) = \omega(L|w)$. Thus, there are letter positions $1 \leq k_1 < \ldots < k_\ell \leq n + 1$ such that, for all $1 \leq i < j \leq \ell$, it holds that $w_{k_i, k_j} \in L$. For $l' = 1, \ldots, \ell - 1$, the "l'-snapshot" is defined as the set

$$Q_{l'} := \{\delta^*(q_0, w_{k_l, k_{l'+1}}) : \ l = 1, \ldots, l'\} \subseteq F \ .$$

In other words: if we consider the l' computational processes created by starting M in positions $k_1, \ldots, k_{l'}$, then $Q_{l'}$ records the set of states of these processes when they have reached position $k_{l'+1}$. It is easy to see that every snapshot is a subset of $F \setminus \{q_0\}$ because the existence of a non-empty string $x \in L$ with $\delta^*(q_0, x) = q_0$ would imply that the maximum clique number of L is infinite. Since $Q_{l'} \neq \emptyset$ for all $l' = 1, \ldots, \ell - 1$, there can be at most $2^{|F \setminus \{q_0\}|} - 1$ distinct snapshots. All that remains to do is to show that the $\ell - 1$ snapshots actually are distinct. Suppose for the sake of contradiction that $Q_{l'} = Q_{l''}$ for some $1 \leq l' < l'' \leq \ell$. It follows that we can push the clique number beyond any given bound m simply by replacing the substring $u = w_{k_{l'}, k_{l''}}$ of w by u^m. We arrived at a contradiction to the assumption $\omega(L) < \infty$. It follows that the $\ell - 1$ snapshots are distinct so that $\ell - 1 \leq 2^{|F \setminus \{q_0\}|} - 1$ and $\omega(L) = \ell \leq 2^{|F \setminus \{q_0\}|}$.

□

4. The Maximum Degree of a Regular Language

We proceed with the characterization of regular languages with an infinite maximum outdegree:

Theorem 4.1. *Let $L \subseteq \Sigma^*$ be a regular language. Then the following statements are equivalent:*

1. $d^+_{max}(L) = \infty$.
2. *For every DFA $M = (Q, \Sigma, \delta, q_0, F)$ such that $L(M) = L$, the following holds:*

$$\exists q \in F, \exists v \in \Sigma^*, \exists w \in \Sigma^+ : q = \delta^*(q_0, v) = \delta^*(q, w) \ . \tag{7}$$

   If M does not contain superfluous states, then (7) simplifies to

$$\exists q \in F, \exists w \in \Sigma^+ : q = \delta^*(q, w) \ . \tag{8}$$

3. $\exists v \in L, \exists w \in \Sigma^+ : v \equiv_L vw$.
4. $\exists v, \exists w \in \Sigma^+, \forall i \geq 0 : vw^i \in L$.

Proof We prove that the first statement implies the second one. Suppose that L has an infinite maximum outdegree. Let $\ell = 1 + |F|$. Choose $u_1, \ldots, u_\ell \in \Sigma^+$ according to condition (3) from Lemma 2.5. Thus, for all $i = 1, \ldots, \ell$, $\delta^*(q_0, u_1 \ldots u_i) \in F$. According to the pigeonhole principle, there must exist $1 \leq i < j \leq \ell$ such that $\delta^*(q_0, u_1 \ldots u_i) = \delta^*(q_0, u_1 \ldots u_j)$. Setting $v = u_1 \ldots u_i$, $w = u_{i+1} \ldots u_j$ and $q = \delta^*(q_0, v)$, we obtain (7). If M does not contain superfluous states, then every state can be reached from q_0 and (7) clearly simplifies to (8).
The second statement, when specialized to the Nerode automaton for L, is easily seen to imply the third one: from (7), we may conclude that $v \in L$ and

$$[v]_L = \delta^*(q_0, v) = \delta^*(q, w) = \delta^*(q_0, vw) = [vw]_L \ .$$

We show now that the third statement implies the fourth one. The condition $v \equiv_L vw$ implies that $vw \equiv_L vww = vw^2$ so that v, vw, vw^2 are Nerode-equivalent. Iteration of this argument yields that $vw \equiv_L vw^i$ for all $i \geq 0$. Since $v \in L$, it follows that $vw^i \in L$ for all $i \geq 0$.
We finally show that the fourth statement implies the first one. Suppose that there exist some $v \in L$ and some $w \in \Sigma^+$ such that, for all $i \geq 0$, $vw^i \in L$. Then, for each $\ell \geq 1$, $d^+_{max}(L) \geq d^+_{max}(L|vw^{\ell-1}) \geq \ell$. Thus, $d^+_{max}(L) = \infty$. $\square$

According to Corollary 2.4, $d^-_{max}(L) = d^+_{max}(L^R)$. In combination with Theorem 4.1 we get:

Theorem 4.2. *Let $L \subseteq \Sigma^*$ be a regular language. Then the following statements are equivalent:*

1. $d^-_{max}(L) = \infty$.
2. *For every DFA $M = (Q, \Sigma, \delta, q_0, F)$ such that $L(M) = L^R$, the condition (7) from the second statement in Theorem 4.1 is valid. If M does not contain superfluous states, then condition (7) simplifies to condition (8).*
3. $\exists v \in L^R, \exists w \in \Sigma^+ : v \equiv_{L^R} vw$.
4. $\exists v \in L^R, \exists w \in \Sigma^+, \forall i \geq 0 : vw^i \in L^R$.
5. $\exists v \in L, \exists w \in \Sigma^+, \forall i \geq 0 : w^i v \in L$.
6. *For every DFA $M = (Q, \Sigma, \delta, q_0, F)$ such that $L(M) = L$, the following holds:*

$$\exists q \in Q, \exists v \in \Sigma^*, \exists w \in \Sigma^+ : q = \delta^*(q_0, w) = \delta^*(q, w) \wedge \delta^*(q, v) \in F \ . \tag{9}$$

If M has no superfluous states, then (9) simplifies to

$$\exists q \in Q, \exists w \in \Sigma^+ : q = \delta^*(q_0, w) = \delta^*(q, w) \ . \tag{10}$$

Proof The implications "1. $\Rightarrow$ 2. $\Rightarrow$ 3. $\Rightarrow$ 4." are an immediate consequence of Corollary 2.4 and Theorem 4.1. The implication "4. $\Rightarrow$ 5." is immediate from the definition of L^R. The implication "5. $\Rightarrow$ 6." can be seen as follows. Choose v and w so that $w^i v \in L$ for all $i \geq 0$. Let $t = |Q|$ and consider the states $q_i = \delta^*(q_0, w^i)$ for all $i \geq 0$. According to the pigeon-hole principle, there must exist $0 \leq i < i + d \leq t$ such that $q_i = q_{i+d}$. It follows that, for all $j \geq i$, $q_j = q_{j+d}$. Let now $j \geq i$ be the smallest number that is a multiple of d. Then, $\delta(q_0, w^j) = q_j = q_{j+j} = \delta^*(q_j, w^j)$. Thus, condition (9) is satisfied with q_j in the role of q, w^j in the role of w, and v as chosen above. If M has no superfluous states, then from every state (except for non-final trap states) one can reach a final state. In this case, (9) clearly simplifies to (10).
The implication "6. $\Rightarrow$ 1." is pretty obvious because, with w, v chosen according to (9), we get $d^-_{max}(L) \geq d^-_{max}(L|w^\ell v) \geq \ell$ for all choices of ℓ, which implies that $d^-_{max}(L) = \infty$. $\square$

For the ease of later reference, we note the following. Condition (10) is satisfied with $q = q_0$ if and only if the following holds:

$$\exists w \in \Sigma^+ : \delta^*(q_0, w) = q_0 \ . \tag{11}$$

This condition can clearly be checked in linear time.

We move on and present an upper bound on the maximum outdegree in the finite case:

Theorem 4.3. *Let $L \subseteq \Sigma^*$ be a regular language. Let $M = (Q, \Sigma, \delta, q_0, F)$ be the Nerode automaton for L. Then $d^+_{max}(L) < \infty$ implies that $d^+_{max}(L) \leq |F \setminus \{q_0\}|$.*

Proof It suffices to show that $d^+_{max}(L) \geq 1+|F\setminus\{q_0\}|$ implies that $d^+_{max}(L) = \infty$. Let $\ell = 1 + |F|$. Suppose first (Case 1) that $q_0 \notin F$ so that $d^+_{max}(L) \geq \ell$. Choose $u_1, \ldots, u_\ell \in \Sigma^+$ according to condition (3) from Lemma 2.5. Thus, for all $i = 1, \ldots, \ell$, $\delta^*(q_0, u_1 \ldots u_i) \in F$. According to the pigeonhole principle, there must exist $1 \leq i < j \leq \ell$ such that $\delta^*(q_0, u_1 \ldots u_i) = \delta^*(q_0, u_1 \ldots u_j)$. But then the strings from $u_1 \ldots u_i (u_{i+1} \ldots u_j)^*$ witness that $d^+_{max}(L) = \infty$. Suppose now (Case 2) that $q_0 \in F$ so that $d^+_{max}(L) \geq \ell - 1$. According to the pigeonhole principle, there must exist $0 \leq i < j \leq \ell - 1$ such that $\delta^*(q_0, u_1 \ldots u_i) = \delta^*(q_0, u_1 \ldots u_j)$. Now we may argue as in Case 1. □

In combination with Corollary 2.4, we get the following

Corollary 4.4. *Let L be a regular language with a finite maximum indegree. Let Q be the set of states of the Nerode automaton for L, and let F^R be the set of final states of the Nerode automaton for L^R. Then,*

$$d^-_{max}(L) \leq \begin{cases} |F^R| - 1 \leq 2^{|Q|-1} - 1 & \text{if } \varepsilon \in L \\ |F^R| \leq 2^{|Q|-1} & \text{otherwise} \end{cases} .$$

Moreover, if L is prefix-closed, then $d^-_{max}(L) \leq |F^R| - 1 \leq 2^{|Q|-2} - 1$.

Proof The upper bound in terms of $|F^R|$ follows immediately from $d^-_{max}(L) = d^+_{max}(L^R)$ and from Theorem 4.3 (with L^R in the role of L). The upper bound in terms of $|Q|$ follows from Lemma 2.1. □

The next result shows that the upper bound on $\omega(L)$ from Theorem 3.2, the upper bound on $d^+_{max}(L)$ from Theorem 4.3 and the upper bound on $d^-_{max}(L)$ from Corollary 4.4 are tight, respectively (even when specialized to prefix-closed regular languages or their reverse).

Theorem 4.5. *For every $t \geq 1$, there exists a prefix-closed DFA $M = (Q, \Sigma, \delta, q_0, F)$ with $t+1$ final states and a non-final trap state q_- such that the following holds:*

1. $\omega(L(M)) = 2^{|F \setminus \{q_0\}|}$.
2. *Let F^R denote the set of final states of the Nerode automaton for $L(M)^R$. Then, $|F^R| = 2^{|Q|-2}$ and $d^-_{max}(L(M)) = d^+_{max}(L(M)^R) = |F^R| - 1$.*

Proof Let $M = (Q, \Sigma, \delta, q_0, F)$ be given by $F = \{q_0, q_1, \ldots, q_t\}$, $Q = F \cup \{q_-\}$, $\Sigma = \{a_1, \ldots, a_t\}$, $\delta(q_-, a) = q_-$ for all $a \in \Sigma$ and

$$\delta(q_i, a_j) = \begin{cases} q_j & \text{if } i < j \\ q_i & \text{if } i > j \\ q_- & \text{if } i = j \end{cases}$$

for all $0 \leq i \leq t$ and $1 \leq j \leq t$. M is obviously prefix-closed. Note that

$$|F \setminus \{q_0\}| = t \quad \text{and} \quad |F^R| \leq 2^{|Q|-2} = 2^t \tag{12}$$

where the latter inequality follows from Lemma 2.1. The next step in the proof is showing that $L(M)$ has a finite clique number and a finite maximum indegree (so that $L(M)^R$ has a finite maximum outdegree). To this end, we note that the following holds for any $q_k \in Q$ and any $w \in \Sigma^+$:

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

It follows that $\omega(L(M)) < \infty$ because the second statement in Theorem 3.1 cannot be true. Since M is prefix-closed, $L(M)$ is prefix-closed too so that Corollary 2.6 applies and we get $d^+_{max}(L(M)^R) = d^-_{max}(L(M)) = \omega(L(M)) - 1 < \infty$.
In the next stage of the proof, we inductively define strings $w(1), \ldots, w(t)$ that lead to graphs with a large maximum indegree:

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

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 $|w(t)| = 2^t - 1$. Consider the digraph $\vec{G}(L(M)|w(t))$ with nodes $1, \ldots, 2^t$.

Claim: For $i = 1, \ldots, 2^t - 1$, the computation of M on $w(t)_{i,2^t}$ avoids the trap state q_-.

Proof of the claim: The claim will become evident from the following observations:

1. All computations on the string $w(k)$ that are started in a letter position between 1 and $|w(k)|$ will avoid all states in $\{q_{k+1}, \ldots, q_t\} \cup \{q_-\}$.
2. All computations on the string $w(k)a_{k+1}$ that are started in a letter position between 1 and $1 + |w(k)|$ will end up in the state q_{k+1}.

The second observation immediately follows from the first one. The first observation for $w(k)$ follows inductively from the first observation for $w(k-1)$ and the second observation for $w(k-1)a_k$. Observation 1 applied to $k = t$ coincides with the above claim.

The claim implies that the node 2^t in $\vec{G}(L(M)|w(t))$ has indegree $2^t - 1$. We may therefore conclude that

$$d^+_{max}(L(M)^R) = d^-_{max}(L(M)) \geq d^-_{max}(L(M)|w(t)) \geq 2^t - 1 \overset{(12)}{\geq} |F^R| - 1 \quad (13)$$

and

$$\omega(L(M)) = d^-_{max}(L(M)) + 1 \geq 2^t \overset{(12)}{=} 2^{|F \setminus \{q_0\}|} \ . \quad (14)$$

Since the lower bounds match with the respective upper bounds, the inequalities (13) and (14) hold with equality, respectively. This concludes the proof. □

5. Complexity Issues

As we will show in Section 5.1, the finiteness of the parameters $\omega(L)$ and $d^-_{max}(L)$ can be decided in time $O(n^2)$, respectively, provided that L is given as a DFA with n states. Moreover, $d^+_{max}(L)$ can be determined in time $O(n)$ and $d^-_{max}(L)$ can be determined in time polynomial in the size n of the given DFA M and the size of the minimum DFA for L^R. These good news are complemented by the following bad news. The problems of computing $\omega(L)$ or $d^-_{max}(L)$, where L is given as a DFA, are infeasible. The same can

be said about the computation of $d^+_{max}(L)$ if L is given as an NFA. More precisely, we show in Section 5.2 that the decision variants of these problems are PSPACE-complete.

5.1. Efficiently Solvable Problems

A naive algorithm for checking the validity of (5) from the second statement in Theorem 3.1 or (10) from the sixth statement in Theorem 4.2 would take cubic time. The following result establishes a quadratic time bound:

Theorem 5.1. *Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$, it can be decided in $O(|Q|^2)$ steps whether $\omega(L(M)) < \infty$ (resp. whether $d^-_{max}(L) < \infty$).*

Proof We first discuss the test for finiteness of the clique number. We make use of the fact that $\omega(L(M)) = \infty$ iff condition (5) from the second statement in Theorem 3.1 is valid. As mentioned in connection with condition (6) already, it can be checked in linear time whether (5) holds with $q = q_0$. We may therefore assume that (5) cannot be satisfied by setting $q = q_0$. Under this assumption, condition (5) can be checked in quadratic time as follows:

1. Compute the DFA M^2 as the product of M with itself and represent it by its transition diagram, say D.
2. For each $q \in F \setminus \{q_0\}$, add a "special edge" $[(q, q), (q_0, q)]$ to D.
3. Compute the strongly connected components (hereafter simply called "strong components") of the digraph D (which, by now, contains the special edges).
4. If there exists some $q \in F \setminus \{q_0\}$ such that (q_0, q) and (q, q) belong to the same strong component, then return "infinite clique number" else return "finite clique number".

The product automaton M^2 has $O(|Q|^2)$ states. The time bound for the algorithm is dominated by the computation of M^2 resp. D, and the computation of the strong components of D. Both computations can be performed in time linear in the size of M^2, i.e., in time $O(|Q|^2)$. We have still to show that our procedure returns the correct classification. One direction is immediate: if there exists a state $q \in F \setminus \{q_0\}$ and a word $w \in \Sigma^+$ such that $q = \delta^*(q_0, w) = \delta^*(q, w)$, then the nodes (q_0, q) and (q, q) will end up in the same strong component of D, which leads to the correct classification "infinite clique number".

Suppose conversely that there exists some state $q \in F \setminus \{q_0\}$ such that (q_0, q)

and (q, q) belong to the same strong component of D, which leads to the classification "infinite clique number". Then there exists a path P from (q_0, q) to (q, q) in D (possibly containing special edges). It suffices to show that a shortest path from (q_0, q) to (q, q) does not contain special edges (which implies that condition (5) from the second statement in Theorem 3.1 holds and $L(M)$ has an infinite clique number, indeed). We will prove this indirectly by showing that a path P from (q_0, q) to (q, q) that contains at least one special edge is not a shortest one. Let $e = [(q', q'), (q_0, q')]$ be the last special edge on P so that P decomposes into an initial segment P_1, the edge e and a final segment P_2. The final segment P_2, leading from (q_0, q') to (q, q), does not contain any special edge. Thus there exists a word $w \in \Sigma^+$ such that P_2 corresponds to the state transitions $\delta^*(q_0, w) = q$ and $\delta^*(q', w) = q$. But then the special edge $[(q', q'), (q_0, q')]$ is superfluous because w gives us a shortcut: the concatenation of P_1 and P_2 (without the detour over (q_0, q')) leads directly from (q', q') to (q, q). Thus, P is not a shortest path, as desired.

As for the test for finiteness of the maximum indegree, we may assume that M has no superfluous states (because the latter can be removed in $O(|Q|)$ steps). It follows that $d^-_{max}(L(M)) = \infty$ iff Condition (10) from the sixth statement in Theorem 4.2 holds. A comparison of this condition with (5) from the second statement in Theorem 3.1 shows that we can almost proceed as in the test for finiteness of the clique number. Condition (11) is easily checked in linear time. We may therefore assume that (10) cannot be satisfied by setting $q = q_0$. Given this assumption, we have to check for each $q \in Q \setminus \{q_0\}$ whether (q_0, q) and (q, q) belong to the same strong component. This can be done in quadratic time by a procedure that is similar to the one we had used for checking the validity of condition (5). □

The computation of $d^+_{max}(L(M))$ takes linear time only:

Theorem 5.2. *Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$, the parameter $d^+_{max}(L(M))$ can be computed within $O(|Q|)$ steps.*

Proof We may assume that M has no superfluous states. We first check whether $d^+_{max}(L(M)) = \infty$. It follows from the second statement in Theorem 4.1 that we have to check Condition (8). This is easily done in time $O(|Q|)$. If $d^+_{max}(L) = \infty$, we are done. Otherwise, we know that each $q \in F$ forms a trivial strong component (having no arcs and consisting of node q only). In order to compute $d^+_{max}(L(M))$, an auxiliary directed acyclic "supergraph" G is computed from D as follows:

1. The "super-nodes" in G are the strong components of D. We refer to the super-nodes representing a single node from $F \setminus \{q_0\}$ as "F-nodes".
2. An arc e is drawn from a strong component K_1 to another strong component K_2 iff D contains an arc leading from a node in K_1 to a node in K_2.

It is fairly straightforward to see that $d^+_{max}(L(M))$ coincides with the maximum number of F-nodes that can be found on the same directed path through G. Since G is acyclic, this number is found in the obvious way while processing the nodes of G in topological order.

□

Combining Theorem 5.1 with Corollary 2.4 and Lemma 2.2 (and the short remark thereafter), we immediately get the following result:

Corollary 5.3. *Let M be a DFA with state set Q. Let M^R_{min} be the minimum DFA for the language $L(M)^R$, and let Q^R_{min} be the state set of M^R_{min}. Then M^R_{min} and $d^-_{max}(L(M))$ can be computed from M in time poly$(|Q| + |Q^R_{min}|)$.*

5.2. Inherently Hard Problems

The following result (combined with binary search) implies that each of the parameters $\omega(L)$, $d^-_{max}(L)$ and $d^+_{max}(L)$ can be computed by a space-efficient algorithm even if L is given as an NFA:

Theorem 5.4. *The following decision problems belong to PSPACE:*

- *Given an NFA M and $\ell \geq 1$, decide whether $\omega(L(M)) \geq \ell$.*
- *Given an NFA M and $\ell \geq 1$, decide whether $d^-_{max}(L(M)) \geq \ell$.*
- *Given an NFA M and $\ell \geq 1$, decide whether $d^+_{max}(L(M)) \geq \ell$.*

Proof For the sake of brevity, let $L = L(M)$. According to a well known theorem of Savitch (1970), every non-deterministic Turing machine with a space bound $S(n) \geq \log(n)$ can be simulated by a deterministic Turing machine with a space bound of order $S(n)^2$. Thus, it suffices to present space-efficient non-deterministic algorithms.
The following algorithm, denoted A_1, does the job for the clique number. A_1 guesses a string $w = a_1a_2a_3\ldots \in \Sigma^*$ letter by letter. At every letter position i, A_1 guesses whether a new (non-deterministic) simulation of the NFA M is

started or not. After a while several simulations of M will be running simultaneously. Let $Q = \{q_0, q_1, \ldots, q_t\}$ be the set of states of M. At every letter position i, A_1 makes a "snapshot" of the form $(m, m_0, m_1, \ldots, m_t)$ such that $m = m_0 + m_1 + \ldots + m_t$ with the interpretation that currently m simulations are running and, for $i = 0, 1, \ldots, t$, m_i of them are in state q_i when they reach the actual letter position. As for the start of a new simulation of M and as for the termination of the simulations, A_1 respects the following rules:

(R1) It never starts a new simulation if, for some $q_j \notin F$, $m_j \geq 1$.

(R2) If $m \geq \ell$ and, for all $q_j \notin F$, $m_j = 0$, it aborts all simulations and accepts.

It is obvious that A_1 can use its non-determinism to update the snapshot $(m, m_0, m_1, \ldots, m_t)$ when getting to the next letter position so that the above interpretation is valid all the time. Since a snapshot does not require much space and only the actual snapshot has to be stored, A_1 is certainly space-efficient. We claim that there exists an accepting computation of A_1 if and only if $\omega(L) \geq \ell$.

Suppose first that $\omega(L) \geq \ell$. Choose strings $u_1 \ldots u_{\ell-1} \in \Sigma^+$ according to condition (2) from Lemma 2.5. If $w = u_1 \ldots u_{\ell-1}$ is the string guessed by A_1, and if A_1 starts a new simulation of M precisely at the first letter of every subword u_j of w, and if A_1 guesses M's accepting computations for the strings $u_i \ldots u_j$, then A_1 will accept after having processed the last letter of w.

Suppose now that there exists an accepting computation of A_1 on input instance (M, ℓ). Let w be the string guessed by A_1. Let $w = u_1 \ldots u_m$ be the unique decomposition of w with the property that precisely at the beginning of each substring u_i a new simulation of M was started. Since A_1 respects rule (R1), it follows that condition (2) from Lemma 2.5 is valid. Since A_1 respects rule (R2), we may conclude that $m \geq \ell$. It follows that $\omega(L) \geq \ell$, as desired.

According to Lemma 2.5, checking whether $d^-_{max}(L) \geq \ell$ is equivalent to checking the validity of condition (4) from Lemma 2.5. This can be checked by a space-efficient non-deterministic algorithm A_2 which proceeds as A_1 except for rule (R1) which is ignored by A_2. (This reflects the fact that the substrings $u_i \ldots u_j$ with $j \leq \ell$ need not be in L.)

According to Lemma 2.5, checking whether $d^+_{max}(L) \geq \ell$ is equivalent to checking the validity of condition (3) from Lemma 2.5. This can be checked

by a space-efficient non-deterministic algorithm A_3 which runs a *single* non-deterministic simulation of M on the string w (where, as before, w is guessed letter by letter) and by maintaining a counter m which coincides with the number of (non-empty) prefixes of w that have been accepted so far. As soon as $m \geq \ell$, A_3 aborts the simulation and accepts. $\square$

Algorithm A_3 in the above proof looks conceptually simpler than the algorithms A_1 and A_2. Therefore, the following might be a good alternative to A_2:

1. Given the NFA M for the language L, compute the NFA M^R for the language L^R. This can be done by "reversing the transition diagram" of M as explained in Section 2.1.
2. Apply the algorithm A_3 to input instance (M^R, ℓ).

Corollary 2.4 implies that this is a correct test for $d^-_{max}(L) \geq \ell$.

Theorem 5.5. *The following decision problems are PSPACE-hard:*

(a) Given a DFA M and $\ell \geq 1$, decide whether $\omega(L(M)) \geq \ell$.

(b) Given a DFA M and $\ell \geq 1$, decide whether $d^-_{max}(L(M)) \geq \ell$.

(c) Given a DFA M and $\ell \geq 1$, decide whether $d^+_{max}(L(M^R)) \geq \ell$.

Moreover these problems remain PSPACE-hard even if M is prefix-closed.

Proof In order to show PSPACE-hardness, we present a polynomial time reduction from "Finite Automata Intersection (FAI)" to the decision problem (b) in the above list. At the end of the proof, we explain why this shows the PSPACE-hardness of all three problems. FAI, which is known to be PSPACE-complete (Kozen, 1977), 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 final state per DFA, does there exist an input string $w \in \Sigma^+$ that is accepted by every DFA in the list? Let $L_i = L(M_i)$ for $i = 1, \ldots, T$. Let $\vdash, \dashv \notin \Sigma$ be two new input symbols. Consider the following prefix-closed regular language:

$$L = \{\dashv\} \cup \bigcup_{i=1}^{T} \left\{\vdash^i w \dashv : w \in L_i \setminus \{\varepsilon\}\right\} \cup \mathrm{Pref}\left(\bigcup_{i=1}^{T} \left\{\vdash^i w : w \in \Sigma^*\right\}\right) \quad . \quad (15)$$

We claim that there is a word $w \in \Sigma^+$ which is accepted by all of $M_1, \ldots, M_T$ iff $d^-_{max}(L) \geq \ell := T+1$. Suppose first that there exists a string $w \in \cap_{i=1}^T L_i$. Then the word $\vdash^T w \dashv$ has $T+1$ (non-empty) suffixes belonging to L, namely the suffix $\dashv$ and, for each $i \in \{1, \ldots, T\}$, the suffix $\vdash^i w \dashv$. According to Lemma 2.5, this implies that $d^-_{max}(L) \geq T+1$. As for the converse direction, suppose that $d^-_{max}(L) \geq T+1$. According to Lemma 2.5, there must exist a word $u \in L$ which has T proper suffixes belonging to L too. Let S denote the set of suffixes of u that belong to L. The definition of L implies that each suffix $u' \in S \setminus \{\dashv\}$ must be of the form $\vdash^i w \dashv$ or $\vdash^i w$ for some $i \in \{1, \ldots, T\}$ and some $w \in \Sigma^*$. Moreover, if u' ends by $\dashv$, then w must be a non-empty word from L_i. Note that the same string w is shared by all suffixes in S. The only explanation why u actually has in total $T+1$ suffixes belonging to L is that the longest such suffix is of the form $\vdash^T w \dashv$ and w is a non-empty word belonging to all languages $L_1, \ldots, L_T$. This completes the proof of the claim.

We still have to show how a prefix-closed DFA M that recognizes L can be computed from $M_1, \ldots, M_T$ in polynomial time. For $i = 1, \ldots, T$, let $M_i = (Q_i, \Sigma, \delta_i, q_0^i, \{q_+^i\})$ where q_0^i denotes the initial state and q_+^i denotes the unique final state of M_i. We may assume that the sets $Q_1, \ldots, Q_T$ are pairwise disjoint (by renaming states if necessary). Let $Z = \{z_0^0, z_0^1, \ldots, z_0^T, z, z_-\}$ be a collection of new states. We build the DFA $M = (Q, \Sigma', \delta, z_0^0, F)$ from the given DFA $M_1, \ldots, M_T$ as follows:

- $Q = \bigcup_{i=1}^T Q_i \cup Z$, $\Sigma' = \Sigma \cup \{\vdash, \dashv\}$ and $F = Q \setminus \{z_-\}$. z_0^0 is the initial state and z_- is a non-final trap state.
- If M reads $\vdash$ in state z_0^i for some $0 \leq i \leq T-1$, then M moves to state z_0^{i+1}.
- For every $i \in \{1, \ldots, T\}$, every $q \in Q_i$ and every $a \in \Sigma$, we set $\delta(z_0^i, a) = \delta_i(q_0^i, a)$ and $\delta(q, a) = \delta_i(q, a)$.
- When reading $\dashv$ in state z_0^0 or in state q_+^i for some $1 \leq 1 \leq T$, then M moves to state z.
- In all cases not covered yet, M moves to the non-final trap state z_-.

Clearly the input strings ε and $\dashv$ are accepted by M. Suppose now that $u \notin \{\varepsilon, \dashv\}$. It is easy to see that M reaches its trap state z_- unless u is of the form $\vdash^i wb$ for some $1 \leq i \leq T$, some $w \in \Sigma^*$, and some $b \in \{\varepsilon, \dashv\}$ such that

w is non-empty if $b = \dashv$. Suppose now that u is of this form. The definition of M implies that, after having processed $\vdash^i$, M will simulate M_i on input string $w \in \Sigma^*$. At this stage, M can reach its trap state z_- only if $b = \dashv$ and $w \notin L_i$. The discussion shows that the strings from the language L, defined as in (15), are accepted by M while all other strings make M reaching its non-final trap state z_-.
The whole discussion so far can be summarized by saying that $M_1, \ldots, M_T \mapsto (M, T+1)$ is a polynomial time reduction of FAI to problem (b) in Theorem 5.5. However, since $d^-_{max}(L(M)) = d^+_{max}(L(M^R))$, it is a polynomial time reduction from FAI to problem (c) in Theorem 5.5 too. Moreover, since M is prefix-closed, it follows that $d^-_{max}(L(M)) = \omega(L(M)) - 1$. Thus $M_1, \ldots, M_T \mapsto (M, T+2)$ is a polynomial time reduction of FAI to problem (a) in Theorem 5.5. This completes the proof.

□

Theorems 5.4 and 5.5 immediately lead to the following

Corollary 5.6. *The problems listed in Theorem 5.5 are PSPACE-complete.*

Acknowledgements.. We thank the anonymous reviewers for their helpful comments on improving the presentation of the paper and for pointing out that Brzozowski's method for DFA minimization gives the minimum DFA for the reverse language as an intermediate result (provided that every state of the originally given DFA is reachable).

References

Bermudez, M. E., Schimpf, K. M., 1990. Practical arbitrary lookahead LR parsing. Journal of Computer and System Sciences 41 (2), 230–250.

Boullier, P., 1984. Contribution à la construction automatique d'analyseurs lexicographiques et syntaxiques. Ph.D. thesis, Université d'Orléans.

Brzozowski, J. A., 1962. Canonical regular expressions and minimal state graphs for definite events. Mathematical Theory of Automata 12 (6), 529–561.

Chung, F. R. K., Grinstead, C. M., 1983. A survey of bounds for classical Ramsey numbers. Journal of Graph Theory 7 (1), 25–37.

Farré, J., Gálvez, J. F., 2001. A bounded graph-connect construction for LR-regular parsers. In: Proceedings of the 10th International Conference on Compiler Construction. pp. 244–258.

Fredricksen, H., 1979. Schur numbers and the Ramsey numbers $N(3, 3, \ldots, 3; 2)$. Journal of Combinatorial Theory, Series A 27 (3), 376–377.

Gosling, J., Joy, B., Steele, G., 1996. The Java™ Language Specification. Addison-Wesley.

Knuth, D. E., 1965. On the translation of languages from left to right. Information and Control 8 (6), 607–639.

Konitzer, M., 2013. Laufzeitanalyse und Optimierung von Parsern für LR-reguläre Grammatiken. Ph.D. thesis, Ruhr-University Bochum.

Konitzer, M., Simon, H. U., 2014. DFA with a bounded activity level. In: Proceedings of the 8th International Conference on Language and Automata Theory and Applications. pp. 478–489.

Kozen, D., 1977. Lower bounds for natural proof systems. In: Proceedings of the 18th Symposium on Foundations of Computer Science. pp. 254–266.

Savitch, W. J., 1970. Relationships between nondeterministic and deterministic tape complexities. Journal of Computer and System Sciences 70 (2), 177–192.

Schmitz, S., 2007. Approximating context-free grammars for parsing and verification. Ph.D. thesis, Université de Nice-Sophia Antipolis.

Čulik, K., Cohen, R., 1973. LR-regular grammars - an extension of LR(k) grammars. Journal of Computer and System Sciences 7 (1), 66–96.

Wan, H., 1997. Upper bounds for Ramsey numbers R(3,3,...,3) and Schur numbers. Journal of Graph Theory 26 (3), 119–122.