Decidability and Shortest Strings in Formal Languages

Levent Alpoge\textsuperscript{1}, Thomas Ang\textsuperscript{2}, Luke Schaeffer\textsuperscript{2}, and Jeffrey Shallit\textsuperscript{2}

\textsuperscript{1} Harvard College, Cambridge, MA 02138, USA
\texttt{levent.alpoge@gmail.com}
\textsuperscript{2} School of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1 Canada
\texttt{angthomas@gmail.com, l3schaef@uwaterloo.ca, shallit@cs.uwaterloo.ca}

Abstract. Given a formal language $L$ specified in various ways, we consider the problem of determining if $L$ is nonempty. If $L$ is indeed nonempty, we find upper and lower bounds on the length of the shortest string in $L$.

1 Introduction

Given a formal language $L$ specified in some finite way, a common problem is to determine whether $L$ is nonempty. And if $L$ is indeed nonempty, then another common problem is to determine good upper and lower bounds on the length of the shortest string in $L$, which we write as $\text{lss}(L)$. Such bounds can be useful, for example, in estimating the state complexity of $L$, since $\text{lss}(L) < \text{sc}(L)$.

As an example, we start with a very simple result often stated in introductory classes on formal language theory.

**Proposition 1.** Let $L$ be accepted by an NFA $M$ with $n$ states and $t$ transitions. Then we can decide in time $O(n + t)$ whether $L \neq \emptyset$. If $L$ is nonempty, then $\text{lss}(L) < n$. Further, this bound is tight.

We now turn to a more challenging example. Here $L$ is specified as the complement of a language accepted by an NFA.

**Theorem 1.** Let $L$ be accepted by an NFA with $n$ states. Then it is PSPACE-complete to determine whether $\overline{L} \neq \emptyset$. If $\overline{L} \neq \emptyset$, then $\text{lss}(\overline{L}) < 2^n$. Further, for some constant $c$, $0 < c \leq 1$, there is an infinite family of examples with $n$ states such that $\text{lss}(\overline{L}) \geq 2^{cn}$.

**Proof.** For the PSPACE-completeness, see [1].

The upper bound is easy and follows from the subset construction. The lower bound is significantly harder; see [5]. \hfill $\Box$

These two examples set the theme of the paper. We examine several problems about shortest strings in regular languages and prove bounds for $\text{lss}(L)$. Some of the results have appeared in the master’s thesis of the second author [3].
2 The first problem

Recall the following classical result about intersections of regular languages.

**Proposition 2.** Let \( L_1 \) (resp., \( L_2 \)) be accepted by an NFA with \( s_1 \) states and \( t_1 \) transitions (resp., \( s_2 \) states and \( t_2 \) transitions) Then \( L_1 \cap L_2 \) is accepted by an NFA with \( s_1s_2 \) states and \( t_1t_2 \) transitions.

**Proof.** Use the usual direct product construction. \( \square \)

This suggests the following natural problems. Given NFA’s \( M_1 \) and \( M_2 \) as above, decide if \( L(M_1) \cap L(M_2) \neq \emptyset \). This can clearly be done in \( O(s_1s_2 + t_1t_2) \) time, by using the direct product construction followed by breadth-first or depth-first search.

Now assume \( L(M_1) \cap L(M_2) \neq \emptyset \). What is a good bound on \( \text{lss}(L(M_1) \cap L(M_2)) \)? Combining Propositions 1 and 2 we immediately get the upper bound \( \text{lss}(L(M_1) \cap L(M_2)) < s_1s_2 \).

However, is this bound tight? For \( \gcd(s_1, s_2) = 1 \) an obvious construction shows it is, even in the unary case: choose \( L_1 = a^{s_1-1}(a^*)^* \) and \( L_2 = a^{s_2-1}(a^*)^* \). However, this idea no longer works for \( \gcd(s_1, s_2) > 1 \). Nevertheless, the bound \( s_1s_2 - 1 \) is tight for binary and larger alphabets, as the following result shows.

**Theorem 2.** For all integers \( m, n \geq 1 \) there exist DFAs \( M_1, M_2 \) with \( m \) and \( n \) states, respectively, and with \( |\Sigma| = 2 \) such that \( L(M_1) \cap L(M_2) \neq \emptyset \), and \( \text{lss}(L(M_1) \cap L(M_2)) = mn - 1 \).

**Proof.** The proof is constructive. Without loss of generality, assume \( m \leq n \), and set \( \Sigma = \{0, 1\} \). Let \( M_1 \) be the DFA given by \( (Q_1, \Sigma, \delta_1, p_0, F_1) \), where \( Q_1 = \{p_0, p_1, p_2, \ldots, p_{m-1}\} \), \( F_1 = p_0 \), and for each \( a, 0 \leq a \leq m - 1 \), and \( c \in \{0, 1\} \) we set

\[
\delta_1(p_a, c) = p(a+c) \mod m.
\]

Then

\[
L(M_1) = \{ x \in \Sigma^* : |x|_1 \equiv 0 \pmod{m} \}.
\]

Let \( M_2 \) be the DFA \( (Q_2, \Sigma, \delta_2, q_0, F_2) \), shown in Figure 2, where \( Q_2 = \{q_0, q_1, \ldots, q_{n-1}\} \), \( F_2 = q_{n-1} \), and for each \( a, 0 \leq a \leq n - 1 \),

\[
\delta_2(q_a, c) = \begin{cases} 
q_{a+c}, & \text{if } 0 \leq a < m - 1; \\
q_{(a+1) \mod n}, & \text{if } c = 0 \text{ and } m - 1 \leq a \leq n - 1; \\
q_0, & \text{if } c = 1 \text{ and } m - 1 \leq a \leq n - 1. 
\end{cases}
\]
Fig. 1. The DFA $M_2$.

Focusing solely on the 1’s that appear in some accepting computation in $M_2$, we see that we can return to $q_0$

(a) via a simple path with $m$ 1’s, or

(b) (if we go through $q_{n-1}$), via a simple path with $(m - 1)$ 1’s and ending in

the transition $\delta(q_{n-1}, 0) = q_0$.

After some number of cycles through $q_0$, we eventually arrive at $q_{n-1}$. Letting $i$ denote the number of times a path of type (b) is chosen (including the last path that arrives at $q_{n-1}$) and $j$ denote the number of times a path of type (a) is chosen, we see that the number of 1’s in any accepted word must be of the form $i(m - 1) + jm$, with $i > 0$, $j \geq 0$. The number of 0’s along such a path is then at least $i(n - m + 1) - 1$, with the $-1$ in this expression arising from the fact that the last part of the path terminates at $q_{n-1}$ without taking an additional 0 transition back to $q_0$.

Thus

$$L(M_2) \subseteq \{ x \in \Sigma^* : \exists i,j \in \mathbb{N}, \text{ such that } i > 0, j \geq 0, \text{ and } \\
| x |_1 = i(m - 1) + jm, \ | x |_0 \geq i(n - m + 1) - 1 \}.$$  

Furthermore, for every $i,j \in \mathbb{N}$, such that $i > 0, j \geq 0$, there exists an $x \in L(M_2)$ such that $| x |_1 = i(m - 1) + jm$, and $| x |_0 = i(n - m + 1) - 1$. This is obtained, for example, by cycling $j$ times from $q_0$ to $q_{m-1}$ and then back to $q_0$ via a transition on 1, then $i - 1$ times from $q_0$ to $q_{n-1}$ and then back to $q_0$ via a transition on 0, and finally one more time from $q_0$ to $q_{n-1}$.

It follows then that

$$L(M_1) \cap L(M_2) \subseteq \{ x \in \Sigma^* : \exists i,j \in \mathbb{N}, \text{ such that } i > 0, j \geq 0, \text{ and } \\
| x |_1 = i(m - 1) + jm, \ | x |_0 \geq i(n - m + 1) - 1 \\
\text{and } i(m - 1) + jm \equiv 0 \pmod{m}. \}.$$  

Further, for every such i and j, there exists a corresponding element in \( L(M_1 \cap M_2) \). Since \( m-1 \) and \( m \) are relatively prime, the shortest such word corresponds to \( i = m, j = 0 \), and satisfies \( |x|_o = m(m - n + 1) - 1 \). In particular, a shortest accepted word is \( (1^{m-1}0^{n-m+1})^{m-1}1^{m-1}0^{n-m} \), which is of length \( mn - 1 \). 

We can also obtain a bound for the unary case. Let

\[
F(m, n) = \max_{1 \leq i, j \leq mn} \left( \max_{1 \leq i, j \leq n} (m - i, n - j) + \ellcm(i, j) \right),
\]
as defined in [7].

**Theorem 3.** Given unary DFA’s \( M_1 \) (resp., \( M_2 \)) with \( m \) (resp., \( n \)) states, accepting \( L_1 \) (resp., \( L_2 \)), we have \( \text{ls}(L_1 \cap L_2) \leq F(m, n) - 1 \). Furthermore, for all \( m, n \geq 1 \) there exist unary DFA’s of \( m \) and \( n \) states achieving this bound.

**Proof.** Follows from [7].

### 3 The second problem

Recall the Post correspondence problem: we are given two finite nonempty languages \( A = \{x_1, x_2, \ldots, x_n\} \) and \( B = \{y_1, y_2, \ldots, y_n\} \), and we want to determine if there exist \( r \geq 1 \) and a finite sequence of indices \( i_1, i_2, \ldots, i_r \) such that \( x_{i_1} \cdots x_{i_r} = y_{i_1} \cdots y_{i_r} \). As is well-known, this problem is undecidable.

Levent Alpoge [2] asked about the variant where we throw away the “correspondence”: determine if there exist \( r, s \geq 1 \) and two finite sequences of indices \( i_1, \ldots, i_r \) and \( j_1, \ldots, j_s \) such that \( x_{i_1} \cdots x_{i_r} = y_{j_1} \cdots y_{j_s} \). In other words, we want to decide if \( A^+ \cap B^+ \neq \emptyset \).

This variant is, of course, decidable. In fact, even a more general version is decidable, where the languages need not be finite.

**Proposition 3.** Suppose \( A \) is a language accepted by an NFA \( M_1 \) with \( s_1 \) states and \( t_1 \) transitions, and \( B \) is accepted by an NFA \( M_2 \) with \( s_2 \) states and \( t_2 \) transitions. Then we can decide in \( O(s_1s_2 + t_1t_2) \) time whether \( A^+ \cap B^+ \neq \emptyset \).

**Proof.** Given NFA \( M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) \) accepting \( A \), we can create an NFA-\( \epsilon \) \( M'_1 = (Q_1, \Sigma, \delta'_1, q_1, F'_1) \) accepting \( A^+ \) by adding an \( \epsilon \)-transition from every final state of \( M_1 \) back to \( q_0 \). We can apply a similar construction to create \( M'_2 = (Q_2, \Sigma, \delta'_2, q_2, F'_2) \) accepting \( B^+ \). Then we can create an NFA-\( \epsilon \) \( M \) accepting \( A^+ \cap B^+ \) using the usual direct product construction. Since this construction is crucial to what follows, and since there is one subtle point, we describe it in some detail.

Given \( M'_1 = (Q_1, \Sigma, \delta'_1, q_1, F'_1) \) and \( M'_2 = (Q_2, \Sigma, \delta'_2, q_2, F'_2) \) as above, \( M = (Q, \Sigma, \delta, q_0, F) \), where \( Q = Q_1 \times Q_2, q_0 = [q_1, q_2] \), and \( F = F_1 \times F_2 \). The transition function \( \delta \) is defined as follows:

For \( p \in Q_1, q \in Q_2 \), and \( a \in \Sigma \cup \{\epsilon\} \) we have \( [p', q'] \in \delta([p, q], a) \) if \( p' \in \delta'_1(p, a) \) and \( q' \in \delta'_2(q, a) \). These transitions correspond to the usual direct product edges of the transition diagram.
However, we also need edges in which one machine performs an explicit \( \varepsilon \)-transition, and the other machine performs an implicit \( \varepsilon \)-transition by simply staying in its own state. This corresponds to including the transitions \([p', q'] \in \delta([p, q], \varepsilon)\) if \(p' \in \delta'_1(p, \varepsilon)\) and \(q = q'\) or if \(p' = p\) and \(q' \in \delta'_2(q, \varepsilon)\).

This construction results in an NFA-\( \varepsilon \) accepting \( A^+ \cap B^+ \) and having at most \(t_1t_2 + 2s_1s_2\) transitions.

Now we can use the usual breadth-first or depth-first search to solve the emptiness problem. \(\square\)

**Corollary 1.** Given NFA’s \(M_1\) accepting \(L_1\) (resp., \(M_2\) accepting \(L_2\)) of \(m\) (resp., \(n\)) states, the shortest string in \(L_1^+ \cap L_2^+\) is of length at most \(mn - 1\).

Suppose \(m \geq n \geq 1\). Then there exists \(M_1\) accepting \(L_1\) (resp., \(M_2\) accepting \(L_2\)) of \(m\) (resp., \(n\)) states such that the shortest string in \(L_1^+ \cap L_2^+\) is of length \(\geq (m - 1)n\).

**Proof.** The first assertion follows from Proposition 3.

For the second assertion, we can take \(M_1\) and \(M_2\) as in the proof of Theorem 2. Clearly \(L_1 = L_1^+\). When we apply our construction to \(M_2\) to create \(L_2^+\), we add an \( \varepsilon \)-transition from \(q_{n-1}\) back to \(q_0\). The effect is to allow one less 0 in each cycle through the states. As in the proof of Theorem 2, to get the proper number of 1’s, we must have \(i = m\), and hence the shortest string in \(L_1^+ \cap L_2^+\) is of length \((m - 1)n\). \(\square\)

We can improve the upper bound to \(mn - 2\) as follows:

**Theorem 4.** For any \(m\)-state DFA \(M_1\) and \(n\)-state DFA \(M_2\) such that \(L(M_1)^+ \cap L(M_2)^+ \neq \emptyset\) we have \(\text{iss}(L(M_1)^+ \cap L(M_2)^+) < mn - 1\).

**Proof.** Assume, contrary to what we want to prove, that we have DFAs \(M_1\) and \(M_2\) with \(m\) and \(n\) states, respectively, such that \(\text{iss}(L(M_1)^+ \cap L(M_2)^+) = mn - 1\). Let \(M_1\) be the DFA given by \((Q_1, \Sigma, \delta_1, p_0, F_1)\), where \(Q_1 = \{p_0, p_1, p_2, \ldots, p_{m-1}\}\), and let \(M_2\) be the DFA given by \((Q_2, \Sigma, \delta_2, p_0, F_2)\), where \(Q_2 = \{q_0, q_1, q_2, \ldots, q_{n-1}\}\).

Then let \(M'_1\) and \(M'_2\) be the \( \varepsilon \)-NFAs obtained by adding \( \varepsilon \)-transitions from the final states to the start states in \(M_1\) and \(M_2\), respectively. Let \(M\) be the \( \varepsilon \)-NFA obtained by applying the cross-product construction to \(M'_1\) and \(M'_2\). Then \(M\) accepts \(L(M_1)^+ \cap L(M_2)^+\).

If \(M\) has more than one final state, a shortest accepting path would only visit one of them, and this immediately gives a contradiction. So, assume each of \(M_1\) and \(M_2\) have only one final state; that is \(F_1 = \{p_x \in Q_1\}\) and \(F_2 = \{q_y \in Q_2\}\).

Then \(M = (Q_1 \times Q_2, \Sigma, \delta, [p_0, q_0], [p_x, q_y])\), where for all \(p_i \in Q_1, q_j \in Q_2, a \in \Sigma, \delta([p_i, q_j], a) = [\delta_1(p_i, a), \delta_2(q_j, a)]\). Note that \(M\) has \( \varepsilon \)-transitions from \([p_x, q_j]\) to \([p_0, q_j]\) for all \(q_j \in Q_2\) and \([p_i, q_y]\) to \([p_i, q_0]\) for all \(p_i \in Q_1\).

Let \(w_1\) be a shortest word accepted by \(M_1\) and \(w_2\) be a shortest word accepted by \(M_2\). Then \(\delta([p_0, q_0], w_1) = [p_x, q_i]\) for some \(i\) such that \(q_i \in Q_2\), and while carrying out this computation we never pass through two states \([p_a, q_0]\) and \([p_c, q_0]\) such that \(a = c\). Likewise, \(\delta([p_0, q_0], w_2) = [p_j, q_y]\) for some \(j\) such that \(p_j \in Q_1\),
and while carrying out this computation we never pass through two states \([p_a, q_b]\) and \([p_c, q_d]\) such that \(b = d\). If both \(x = 0\) and \(y = 0\) the shortest accepted string is \(\epsilon\), so without loss of generality, assume \(x \neq 0\). Then \(\delta([p_0, q_0], w_1) = [p_x, q_0]\) or else we can visit \(|w_1| + 2\) states with \(|w_1|\) symbols by using an \(\epsilon\)-transition and we get a contradiction. If \(y = 0\), \(w_1\) is the shortest string accepted by \(M\) and we have a contradiction. So, \(y \neq 0\) and \(\delta([p_0, q_0], w_2) = [p_0, q_y]\). It follows that reading \(w_1\) from the initial state brings us to \([p_x, q_0]\) without passing through \([p_0, q_y]\), and reading \(w_2\) from the initial state brings us to \([p_0, q_0]\) without passing through \([p_x, q_0]\). So, a shortest accepting path need only visit one of \([p_x, q_0]\) and \([p_0, q_y]\), and again we have a contradiction.

**Proof.** Assume the input alphabet of both \(M_1\) and \(M_2\) is \(\Sigma = \{a\}\). Let \(c_1\) (resp., \(c_2\)) be the length of the shortest nonempty string in \(L_1\) (resp., \(L_2\)). Clearly \(c_1 \leq m\) and \(c_2 \leq n\). Furthermore, if \(c_1 = m\), then \(L_1 = (a^m)^∗\), and similarly if \(c_2 = n\) then \(L_2 = (a^n)^∗\). Hence if \((c_1, c_2) = (m, n)\), then \(\epsilon \in L_1^+ \cap L_2^+\), and hence \(\text{lss}(L_1^+ \cap L_2^+) = 0 \leq G'(m, n)\). Otherwise either \(c_1 < m\) or \(c_2 < n\). Without loss of generality, assume \(c_2 < n\). Then \(a^{\text{lcm}(c_1, c_2)} \in L_1^+ \cap L_2^+\), so \(\text{lss}(L_1^+ \cap L_2^+) \leq \text{lcm}(c_1, c_2) \leq G(m, n - 1) \leq G'(m, n)\).

Now suppose we are given \(m\) and \(n\). Let \(i, j\) be the integers maximizing \(\text{lcm}(i, j)\) over \(1 \leq i \leq m, 1 \leq j \leq n\) with \((i, j) \neq (m, n)\). If \(i < m\), choose \(L_1 = (a^i)^∗\), which can be accepted by a DFA with \(i + 1 \leq m\) states, and choose \(L_2 = (a^j)^∗\), which can be accepted by a DFA with \(j \leq n\) states. Otherwise, reverse the roles of \(m\) and \(n\). Thus we get DFA’s of \(m\) and \(n\) states, respectively, achieving \(\text{lss}(L_1^+ \cap L_2^+) = G'(m, n)\). □

### 4 The third problem

Another variation on the Post correspondence problem, also proposed by Alpoge [2], is more interesting. Here we throw away only part of the “correspondence”:
given \( A = \{x_1, x_2, \ldots, x_n\} \) and \( B = \{y_1, y_2, \ldots, y_n\} \), we want to decide if there exist \( r \geq 1 \) and two finite sequences of indices \( i_1, i_2, \ldots, i_r \) and \( j_1, j_2, \ldots, j_r \) such that \( x_{i_1} \cdots x_{i_r} = y_{j_1} \cdots y_{j_r} \). In other words, we only demand that the number of words on each side be the same.

This case is also efficiently decidable, even when \( A \) and \( B \) are possibly infinite regular languages.

**Theorem 6.** Let \( M_1 \) (resp., \( M_2 \)) be an NFA with \( s_1 \) states and \( t_1 \) transitions (resp., \( s_2 \) states and \( t_2 \) transitions). We can decide in polynomial time (in \( s_1, s_2, t_1, t_2 \)) whether there exists \( k \) such that \( L(M_1)^k \cap L(M_2)^k \neq \emptyset \).

**Proof.** First, we prove the (possibly surprising?) result that

\[
L = \bigcup_{k \geq 1} (L(M_1)^k \cap L(M_2)^k)
\]

is a context-free language.

We construct a pushdown automaton \( M \) accepting \( L \). On input \( x \), our PDA attempts to construct two same-length factorizations of \( x \): one into elements of \( L(M_1) \), and one into elements of \( L(M_2) \). To ensure the factorizations are really of the same length, we use the stack of the PDA to maintain a counter that records the absolute value of the difference between the number of factors in the first factorization and the number of factors in the second. The appropriate sign of the difference is maintained in the state of the PDA.

As we read \( x \), we simulate the NFA’s \( M_1 \) and \( M_2 \). If we reach a final state in either machine, then we have the option (nondeterministically) to deem this the end of a factor in the appropriate factorization, and update the stack accordingly, or continue with the simulation. We accept if the stack records a difference of \( 0 \) — that is, if the stack contains no counters and only the initial stack symbol \( Z_0 \) — and we are in a final state in both machines (indicating that the factorization is complete into elements of both \( L_1 \) and \( L_2 \)).

Thus we have shown that \( L \) is context-free. Furthermore, our PDA has \( O(s_1 s_2) \) states and \( O(t_1 t_2) \) transitions. It uses only two distinct stack symbols — the counter and the initial stack symbol — and never pushes more than one additional symbol on the stack in any transition. Such a PDA can be converted to a context-free grammar \( G \), using the standard “triple construction” \([6 \text{ Thm. 5.4}]\), using \( O(s_1^2 s_2^2) \) states and \( O(s_1^2 s_2^2 t_1 t_2) \) transitions. Now we can test the emptiness of the language generated by a context-free grammar of size \( t \) in \( O(t) \) time, by removing useless symbols and seeing if any productions remain \([6 \text{ Thm. 4.2}]\).

We conclude that it is decidable in polynomial time whether there exists \( k \) such that \( L(M_1)^k \cap L(M_2)^k \neq \emptyset \). \( \square \)

**Remark 1.** There exist simple examples where \( L = \bigcup_{k \geq 1} (L(M_1)^k \cap L(M_2)^k) \) is not regular. For example, take \( L(M_1) = b^* ab^* \) and \( L(M_2) = a^* ba^* \). Then \( L = \{ x \in \{a, b\}^* : |x|_a = |x|_b \geq 1 \} \), the language of nonempty strings with the same number of \( a \)'s and \( b \)'s.
Furthermore, if $M_1, M_2, M_3$ are all NFA’s, then the analogous language

\[ L = \bigcup_{k \geq 1} (L(M_1)^k \cap L(M_2)^k \cap L(M_3)^k) \]

need not be context-free. A counterexample is given by taking $L(M_1) = \{b, c\}^*a\{b, c\}^*$, $L(M_2) = \{a, c\}^*b\{a, c\}^*$, and $L(M_3) = \{a, b\}^*c\{a, b\}^*$. Then

\[ L = \{x \in \{a, b, c\}^* : |x|_a = |x|_b = |x|_c \geq 1\}, \]

which is clearly not context-free.

Remark 2. Mike Domaratzki (personal communication) observes that the decision problem “given $M_1, M_2$, does there exist $k \geq 1$ such that $L(M_1)^k \cap L(M_2)^k \neq \emptyset$” becomes undecidable if $M_1$ and $M_2$ are pushdown automata, by reduction from the problem “given CFG’s $G_1, G_2$, is $L(G_1) \cap L(G_2) \neq \emptyset$” [6, Theorem 8.10]. Given $G_1$ and $G_2$, we can easily create PDA’s accepting $L_1 := L(G_1)\#$ and $L_2 := L(G_2)\#$, where $\#$ is a new symbol not in the alphabet of either $G_1$ or $G_2$. Then $L_1^k \cap L_2^k \neq \emptyset$ for some $k \geq 1$ if and only if $L(G_1) \cap L(G_2) \neq \emptyset$. A similar result holds for the linear context-free languages [4].

We now turn to the question of, given regular languages $A$ and $B$, determining the shortest string in $L = \bigcup_{k \geq 1} (A^k \cap B^k)$, given that it is nonempty. Actually, we consider a more general problem, where we intersect more than two languages.

We start by proving a result about directed graphs.

Lemma 1. Suppose $G = (V, E)$ is a directed graph with edge weights in $\mathbb{Z}^d$, where the components of the edge weights are all bounded in absolute value by $K$. Let $\sigma(p)$ denote the weight of a path $p$, obtained by summing the weights of all associated edges. If $G$ contains a cycle $C$: $u \rightarrow u$ such that $\sigma(C) = \mathbf{0} = (0, 0, \ldots, 0)$, then $G$ also contains a cycle $C': u \rightarrow u$ with $\sigma(C') = \mathbf{0} = (0, 0, \ldots, 0)$ and length at most $|V|^{d+1}K^d d^{d/2}(|V|^2 + d)$.

Proof. For each vertex $v$ in the cycle $C$, break $C$ at the first occurrence of $v$. This gives us

\[ C = P_1P_2P_3 \cdots P_k \]

such that $P_1: v_1 \rightarrow v_2, P_2: v_2 \rightarrow v_3, \ldots, P_k: v_k \rightarrow v_{k+1}$ where $\{v_1, \ldots, v_k\}$ is the set of vertices visited by $C$. The final vertex, $v_{k+1}$, is the same as $v_1$ because $C$ is a cycle. Notice that $k \leq |V|$ because each vertex appears at most once in the list $v_1, \ldots, v_k$.

For each $P_i: v_i \rightarrow v_{i+1}$, generate a new path $\hat{P}_i: v_i \rightarrow v_{i+1}$ by removing all simple subcycles. The length of $\hat{P}_i$ is at most $|V|$; otherwise some vertex is repeated, so we have not removed all subcycles. Recombine the $\hat{P}_i$’s into a cycle $T = \hat{P}_1 \cdots \hat{P}_k$ having length $|T| \leq |V|^{|k|} \leq |V|^2$. In addition to $T$, we have a list of simple subcycles $B_1, \ldots, B_t$ that we removed while generating the $\hat{P}_i$’s.
Consider the cycles we can construct using \( T, B_1, B_2, \ldots, B_\ell \). For any \( B_i \), we know \( T \) visits the starting vertex of \( B_i \) because \( T \) visits all the vertices in \( C \). Therefore we can splice \( B_i \) into \( T \) at its starting vertex. Since \( B_i \) is a cycle, we can insert it into \( T \) any positive number of times. We can also append \( T \) to the whole cycle as many times as we like. These techniques allow us to construct a cycle with weight

\[
t \sigma(T) + b_1 \sigma(B_1) + \cdots + b_\ell \sigma(B_\ell)
\]

where \( t \geq 1 \) and \( b_1, \ldots, b_\ell \geq 0 \) are all integers.

Recall that \( T, B_1, \ldots, B_\ell \) were constructed by decomposing \( C \). Each edge from \( C \) exists somewhere in \( T, B_1, \ldots, B_\ell \), so we have

\[
0 = \sigma(C) = \sigma(T) + \sigma(B_1) + \cdots + \sigma(B_\ell).
\]

This shows that it is possible to write 0 as an integer linear combination of \( \sigma(T), \sigma(B_1), \ldots, \sigma(B_\ell) \). Unfortunately, for each nonzero \( b_i \) we have at least one copy of \( B_i \), with length at most \(|V|\). Since all the \( b_i \)'s are nonzero and \( \ell \) is unbounded, the corresponding cycle has unbounded length. If we hope to find a bounded cycle by this technique then we need to bound the number of nonzero \( b_i \)'s. Let us approach the problem with linear programming. Construct a matrix \( A \in \mathbb{R}^{d \times \ell} \) where the \( i \)th column is given by \( A(i) = \sigma(B_i) \). Let \( b \in \mathbb{R}^d \) be the column vector \( \sigma(T) \). We are looking for solutions to the problem

\[
Ax = b, \quad x \geq 0, \quad x \in \mathbb{R}^{\ell}.
\]

This is just the feasible set of a linear program in standard equality form. We saw earlier that it has the feasible solution \( x = (1 1 \cdots 1 1)^T \). Note that if \( A \) is not full rank then we remove linearly dependent rows until we have a full rank matrix, and proceed with a matrix of rank \( d' \leq d \).

Linear programming theory tells us a feasible problem of this form has a basic feasible solution \( x^* \) with at most \( d \) nonzero entries. Without loss of generality (relabelling if necessary), take all but the first \( d \) entries of \( x^* \) to be zero. Letting \( A \) be the first \( d \) columns of \( A \), the basic solution \( x^* \) satisfies the following equation:

\[
A \begin{pmatrix} x^*_1 \\ \vdots \\ x^*_d \end{pmatrix} = b;
\]

\[
\sigma(B_1)x^*_1 + \cdots + \sigma(B_d)x^*_d = -\sigma(T).
\]

We are not done yet because the \( x^*_i \)'s are real numbers and we need an integer linear combination. Cramer’s rule gives an explicit solution for each coefficient, \( x^*_i = \frac{\det(\hat{A}_i)}{\det(\hat{A})} \), where \( \hat{A}_i \) is the matrix \( \hat{A} \) with the \( i \)th column replaced by \( b \). Note that \( \hat{A} \) and \( \hat{A}_i \) are integer matrices, so their determinants are integers and \( x^*_i \) is a rational number. When we multiply through by \( |\det(\hat{A})| \), all the coefficients will be positive integers:

\[
\sigma(B_1)|\det(\hat{A}_1)| + \cdots + \sigma(B_d)|\det(\hat{A}_d)| + \sigma(T)|\det(\hat{A})| = 0.
\]
We can bound the determinants with Hadamard’s inequality, which says that the determinant of a matrix $M$ is bounded by the product of the norms of its columns. Each $B_i$ is a simple cycle, so $|B_i| \leq |V|$. It follows that any entry of $\sigma(B_i)$ is at most $|V|^2$, so $\|\sigma(B_i)\| \leq |V|^2 K \sqrt{d}$. On the other hand, $T$ has length at most $|V|$. Combining these estimates gives $|\det(\hat{A}_i)| \leq |V|^2 K d^{d/2}$ for all $i$ and $|\det(\hat{A})| \leq |V|^{d+1} K d^{d/2}$. Now we construct the cycle $C'$ from this linear combination, with $|\det(\hat{A}_i)|$ copies of $T$ and $|\det(\hat{A})|$ copies of each $B_i$. By construction, $C'$ has weight 0 and its length is bounded as follows:

$$|C'| = |\det A||T| + \sum_{i=1}^{d} |\det A_i||B_i|$$

$$\leq |V|^{d+1} K^d d^{d/2} |V|^2 + \sum_{i=1}^{d} |V|^d K^d d^{d/2} |V|$$

$$= |V|^{d+1} K^d d^{d/2} (|V|^2 + d).$$

□

Corollary 2. Consider a generalization of the third problem to $d$ languages $L_1, L_2, \ldots, L_d$ accepted by NFA’s having $s_1, \ldots, s_d$ states, respectively. If

$$\bigcup_{k \geq 1} (L_{k_1} \cap \cdots \cap L_{k_d})$$

is nonempty, then the shortest string in the language has length bounded by

$$O(s^d(d-1)^{(d-1)/2}(s^2 + d - 1)),$$

where $s := (s_1 + 1)(s_2 + 1)\ldots(s_n + 1)$.

Proof. We discuss the case $d = 2$, and then briefly indicate how this is generalized to the general case.

First we discuss an automaton $M_K = (Q_K, \Sigma, \delta_K, q_{K1}, F_K)$ accepting $K = A^* \cap B^*$ which is a slight variant of the construction given in the proof of Theorem 3 above.

Suppose we are given a regular language $A$ (resp., $B$) accepted by an NFA $M_1$ (resp., $M_2$). Without loss of generality, we will assume that $M_1$ (resp., $M_2$) has no transitions into its initial state. This can be accomplished, if necessary, by adding one new state with transitions out the same as the transitions out of the initial state, and redirecting any transitions into the initial state to the new state. If the original machine had $s$ states, then the new machine has at most $s + 1$ states. Call these new machines $M'_1 = (Q_1, \Sigma, \delta, q_1, F_1)$ and $M'_2 = (Q_2, \Sigma, \delta, q_2, F_2)$.

Next we create an NFA-ε $M''_1 = (Q_1, \Sigma, \delta', q_1, F'_1)$ by adding an ε-transition from every final state of $M'_1$ back to its initial state, and by changing the set of final states to be $F'_1 = \{q_1\}$. This new machine $M''_1$ accepts $A^*$. We carry out a similar construction on $M'_2$ obtaining $M''_2$ accepting $B^*$. 

□
Finally, mimicking the construction of Theorem 3, we create an NFA-ε $M_K$ accepting $K = A^* \cap B^*$ using the direct product construction outlined above on $M_1'$ and $M_2'$. Note that $M_K$ has at most $(s_1 + 1)(s_2 + 1)$ states and has exactly one accepting state, which is its initial state.

We define the edge weights of $M_k$ to be $\mathbb{Z}$ as follows. An explicit ε-transition in $M_1'$ or $M_2'$ marks the end of a word, so each explicit ε-transition taken in $M_1'$ back to the start gets weight +1, while each explicit ε-transition in $M_2'$ back to the start gets weight −1. In this way we keep track of the difference between the number of factors used in $L(M_1')$ and $L(M_2')$.

For the general case, we form the intersection automaton as before, and define the $i$'th coordinate of $\sigma(P)$, for $1 \leq i < d$, to be the difference in the number of ε-transitions taken in $M_1'$ and $M_i'$. Now just apply Lemma 1 to get the desired bound. □

When $d = 2$, we can improve on the result of the previous lemma:

**Theorem 7.** If $d = 2$, then the length of the cycle $C'$ in Lemma 1 is at most $2K|V|^2$.

**Proof.** Remove simple cycles $B_1, B_2, \ldots, B_\ell$ from $C$ until we are left with $R$, which has no proper subcycles. It follows that $R$ must be a simple cycle, so we have decomposed $C$ into simple subcycles. Note that the weight of $C$ is the sum of the weights of all the $B_i$’s and $R$.

If $R$ has weight 0 then take $C' = R$. We are done because $R$ has length at most $|V| \leq 2K|V|^2$. If $R$ has nonzero weight then the positive and negative cases are identical so take $R$ to have positive weight without loss of generality. Then there must be some $B_i$ with negative weight, otherwise the sum of the weights of the $B_i$’s and $R$ would be positive, but $C$ has weight 0. Call the negative weight cycle $S$.

If $R$ and $S$ have some vertex in common, then we can splice $\sigma(R)$ copies of $S$ into $-\sigma(S)$ copies of $R$ to get a cycle $C'$ of weight 0. Since $\sigma(R) \leq K|R|$ and $\sigma(S) \leq K|S|$, the cycle has length $|\sigma(R)||S| + |\sigma(S)||R| \leq 2K|R||S| \leq 2K|V|^2$.

Otherwise, $R$ and $S$ have no vertex in common so we need to find some way to get from $R$ to $S$ and back again. Clearly $C$ passes through every vertex in $R$ and $S$, but we want a shorter cycle. Let $T$ be the shortest cycle that passes through some vertex in $R$ and some vertex in $S$. We will split $T$ into $\alpha$, the piece from $R$ to $S$, and $\beta$, the piece from $S$ to $R$.

We know that $R, S$ are simple, and $\alpha, \beta$ must be simple or we could make a shorter cycle $T$ by making them shorter. Therefore, any vertex in $V$ occurs at most four times in $R, S$ and $T$, once for each of $R, S, \alpha, \beta$. But $R$ and $S$ have no vertices in common, so each vertex occurs at most three times in $R, S$ and $T$.

Now if some vertex $v$ occurs three times in $R, S$ and $T$, then it must be in $\alpha, \beta$ and either $R$ or $S$ (without loss of generality, let it be in $R$). Then we can remove a prefix of $\alpha$ up to $v$, producing $\hat{\alpha}$. Similarly, remove a suffix of $\beta$ starting from $v$, giving $\hat{\beta}$. Then $\hat{\alpha} \hat{\beta}$ is a shorter cycle that visits $v \in R$ and still visits $S$, contradicting the minimality of $T$. Therefore any vertex $v$ occurs at most twice in $R, S$ and $T$, so $|R| + |S| + |T| \leq 2|V|$. 


Let us combine $T$ with $R$ if $T$ has positive weight and $S$ if $T$ has negative weight to produce a cycle $Y$. Either $R$ or $S$ is left over, call it $X$. Note that $X$ and $Y$ have opposite sign weights, and also have a vertex in common. As before, we combine $|\sigma(X)|$ copies of $Y$ with $|\sigma(Y)|$ copies of $X$ to produce a cycle $C'$ of length at most $2K|X||Y|$. Under the constraint $|X|+|Y|=|R|+|S|+|T| \leq 2|V|$, the length $2K|X||Y|$ is maximized when $|X|=|Y|=|V|$, with maximum value $2K|V|^2$, completing the proof. \hfill \Box

Finally, we prove an improvement for the unary case.

**Proposition 4.** Let $A, B$ be nonempty finite languages over a unary alphabet, say $A = \{a^{m_1}, \ldots, a^{m_r}\}$ and $B = \{a^{n_1}, \ldots, a^{n_s}\}$. Then $A^k \cap B^k \neq \emptyset$ for some $k \geq 1$ iff $\min_{1 \leq i \leq r} m_i \leq \max_{1 \leq j \leq s} n_j$ and $\min_{1 \leq j \leq s} n_j \leq \max_{1 \leq i \leq r} m_i$. If both conditions hold, then $A^k \cap B^k \neq \emptyset$ for some $k < \max(m_1, \ldots, m_r, n_1, \ldots, n_s)$, and this bound is tight.

**Proof.** Suppose $\min_{1 \leq i \leq r} m_i > \max_{1 \leq j \leq s} n_j$. Then every element of $A^k$ will be of length greater than every element of $B^k$. Similarly, if $\min_{1 \leq j \leq s} n_j \leq \max_{1 \leq i \leq r} m_i$, then every element of $B^k$ will be of length greater than every element of $A^k$. Hence if either condition holds, we have $A^k \cap B^k = \emptyset$ for all $k \geq 1$.

Now suppose $\min_{1 \leq i \leq r} m_i \leq \max_{1 \leq j \leq s} n_j$ and $\min_{1 \leq j \leq s} n_j \leq \max_{1 \leq i \leq r} m_i$. Then there exist $a^l, a^m \in A$ and $a^n \in B$ such that $l \leq m \leq n$. Choose $i = n - m$ and $j = m - l$. Then $A^{i+j}$ contains $(a^l)^i(a^n)^j = a^{i(m-l)+j} = a^{l-n+m+n-l} = a^{m(n-l)}$. And $B^{i+j}$ contains $(a^m)^{i+j} = a^{m(n-l)}$. So for $k = i+j$ we get $A^k \cap B^k \neq \emptyset$. Now $i - j = n - l < n \leq \max(m_1, \ldots, m_r, n_1, \ldots, n_s)$.

The bound is tight, as can be seen by taking $A = \{a^n\}$ and $B = \{a^{n-1}\}$. Then the least $k$ such that $A^k \cap B^k \neq \emptyset$ is $k = n-1$. \hfill \Box

**References**

1. A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.
2. L. Alpoge. Post correspondence problem variant. Posted to http://cstheory.stackexchange.com/questions/4284/post-correspondence-problem-variant, January 12 2011.
3. T. Ang. Problems related to shortest strings in formal languages. M. Math. thesis, University of Waterloo, 2010.
4. B. S. Baker and R. V. Book. Reversal-bounded multipushdown machines. *J. Comput. System Sci.* 8 (1974), 315–332.
5. K. Ellul, B. Krawetz, J. Shallit, and M.-w. Wang. Regular expressions: new results and open problems. *J. Autom. Lang. Combin.* 10 (2005), 407–437.
6. J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.
7. G. Pighizzini and J. Shallit. Unary language operations, state complexity, and Jacobsthal’s function. *Int. J. Found. Comput. Sci.* 13 (2002), 145–159.