Errata for: Decidability Results on the Existence of Lookahead Delegators for NFA

Christof Löding

February 22, 2023

The paper [4] contains some statements that are wrong. In this note, I want to shed some light on these mistakes.

Lemma 3 in the paper is wrong. Thanks to Christopher Hugenroth and Dietrich Kuske for pointing that out. I first explain the problem with the lemma, and then have a look at the consequences for the other results stated in the paper.

Counter-Example for Lemma 3: Intuitively, the problem with the lemma is that not every state that is reachable in runs constructed by a delegator, is also reachable with every lookahead. This causes the problem in the proof of the direction that constructs a set $Q'$ from a delegator. Thanks to Christopher Hugenroth for providing a counter-example showing that the lemma is indeed false.

Consider the NFA in Figure 1 over the alphabet $\Sigma = \{a, b, c\}$. The accepted language is $a(a + b)\Sigma^*$. There is a 1-delegator that moves to $q_1$ if the input starts with $aa$ and to $q_2$ if the input starts with $ab$. So we can define $f$ as follows:

\[
\begin{align*}
  f(q_0, aa) &= q_1 \quad f(q_0, ab) = q_2 \\
  f(q_1, aw) &= q_5 \quad f(q_2, bw) = q_5
\end{align*}
\]

for each $w \in \{\varepsilon, a, b, c\}$, and an arbitrary definition of $f$ for all other combinations.

Now assume that there is a set $Q'$ of states as claimed in Lemma 3. It needs to contain $q_0$. Then it also needs to contain an $a$-successor $q$ of $q_0$. 

1
with \((ab)^{-1}L_{q_0} = b^{-1}L_q\). This must be \(q_2\). Then \(Q'\) also needs to contain an \(a\)-successor \(p\) of \(q_2\) with \((aa)^{-1}L_{q_3} = a^{-1}L_p\). Such a state \(p\) does not exist because \((aa)^{-1}L_{q_2} = a^*(b + c)\Sigma^*\) but \(a^{-1}L_{q_3} = a^*b\Sigma^*\) and \(a^{-1}L_{q_4} = a^*c\Sigma^*\).

Note that \(q_2\) is not reachable with the lookahead \(aa\) in runs constructed by \(f\), but the lemma requires the existence of corresponding states for all lookaheads.

As a consequence, all proofs that are based on Lemma 3 are faulty. However, not all results are wrong. Some have a proof without Lemma 3.

- **Lemma 7 and Corollary 8**: The results are correct. Further below I sketch my original proof, which does not use Lemma 3 (for \(k = 0\) this is the proof cited as personal communication in [1, Theorem 15]). I only prove Corollary 8 with a slightly simplified version of the game (compared to the game used in [4]).

- **Theorem 11 and 12**: Theorem 11 indeed seems to be wrong, and thus also the upper bound in Theorem 12. The hardness part of Theorem 12 does not use Lemma 3 and should be ok.

- **Theorem 15**: The proof is wrong. It should be possible to obtain a similar bound \(K\) with techniques as used in [2] and [3, Theorem 4.8]. However, since Theorem 15 is mainly used for proving the upper bound in Corollary 16, which is again based on Lemma 3, going into the details...
of proving a statement similar to the one in Theorem 15 seems only to be worth the effort in a context where this result can be used further. So for the moment the status of Theorem 15 is open.

- Corollary 16: The proof is based on Theorem 11 and thus on Lemma 3. The upper bound seems to be wrong. The hardness part does not use Lemma 3 and thus should be ok (as for Theorem 12).

Below I explain a correct argument for Corollary 8.

**Corollary 8** For each \( k \in \mathbb{N} \), the problem \( k\text{-DELEGATOR} \) can be solved in polynomial time.

**Proof.** We construct a safety game in which a winning strategy for Player 0 corresponds to a delegator, and vice versa. The game is similar to the one in [4] but a bit simpler without the parts on short lookaheads.

The rough idea for the game is the following: Player 1 chooses the input word, and Player 0 chooses the next transition based on the next \( k + 1 \) symbols of the input. Then Player 1 wins if he can play an input that is in the language but on which Player 0 fails to construct an accepting run. To check whether the run of Player 0 is accepting or not, it suffices to store the current state of the run in the game configuration. But the problem is to check whether the input played by Player 1 is accepted by the NFA. For this purpose, we also let Player 1 choose transitions for a run on the input that he plays. We have to ensure that Player 0 cannot use this information when she is choosing her transitions. It turns out that it is sufficient to let Player 1 choose the next transition after Player 0. This forces Player 0 to always choose a next state from which still all possible extensions of the current input that are in the language can be accepted. We formalize this idea below.

Let \( A = (Q, \Sigma, q_0, \Delta, F) \) be the NFA for which we want to check the existence of a \( k\)-delegator (where \( k \) is fixed). We write \( L(A, q) \) to denote the language accepted by \( A \) with starting state \( q \).

The vertices of the game graph are \( V = Q \times Q \times \Sigma^{k+1} \times \{0, 1\} \) with \( V_i = Q \times Q \times \Sigma^{k+1} \times \{i\} \). The first state component is for the run chosen by Player 0, the second state component for the run chosen by Player 1. The possible moves are as follows, where \( p, q, p', q' \in Q, a, a' \in \Sigma \) and \( w \in \Sigma^k \):

- \((p, q, aw, 0) \rightarrow (p', q, aw, 1)\) for all \((p, a, p') \in \Delta\).
\( (p', q, aw, 1) \rightarrow (p', q', wb, 0) \) for all \( b \in \Sigma \), and \( (q, a, q') \in \Delta \).

The possible starting vertices are those of the form \((q_0, q_0, aw, 0)\). One can think of Player 1 choosing the starting vertex and thus the first \( k + 1 \) symbols of the input.

The winning condition is a safety condition for Player 0: If a vertex \((p', q, aw, 1)\) is reached with \( w \notin L(A, p') \) and \( aw \in L(A, q) \), then Player 0 loses.

We claim that Player 0 has a winning strategy from all the starting vertices of this game if, and only if, there is a \( k \)-delegator for \( A \).

For the direction from left to right, assume that \( f \) is a \( k \)-delegator for \( A \). Then Player 0 can always choose the next transition according to the delegator, that is, from a vertex \((q_1, q_2, aw, 0)\), Player 0 chooses the transition given by \( f(q_1, aw) \).

Assume that Player 1 can win against this strategy of Player 0, that is, from some starting vertex, Player 1 can reach a position \((p', q, aw, 1)\) with \( w \notin L(A, p') \) and \( aw \in L(A, q) \). Let \( a_1 \cdots a_m \) be the sequence of symbols on which Player 0 has chosen transitions until then. So \( a = a_m \), and Player 1 has chosen transitions for \( a_1 \cdots a_{m-1} \), which means that there is a run of \( A \) on \( a_1 \cdots a_{m-1} \) that leads to \( q \). From \( a_m w = aw \in L(A, q) \) we can conclude that \( a_1 \cdots a_m w \in L(A) \).

Now consider the run produced by \( f \) for input \( a_1 \cdots a_m w \), which is a sequence of states \( q_0 q_1 \cdots q_\ell \) with \( \ell = |a_1 \cdots a_m w| \). Since Player 0 chooses the transitions according to \( f \), this run is in state \( p' \) after \( a_1 \cdots a_m \), that is, \( p' = q_m \). But \( w \notin L(A, p') \), and hence \( q_\ell \notin F \). Thus, \( f \) does produce an accepting run on \( a_1 \cdots a_m w \), contradicting the fact that \( f \) is a \( k \)-delegator.

For the direction from right to left, assume that Player 0 has a winning strategy \( \sigma \) from all the starting vertices. Consider a vertex \((p, p, aw, 0)\) that is reached from some starting vertex in a play where Player 0 plays according to \( \sigma \), and let \((p', p, aw, 1)\) be the next vertex according to \( \sigma \). We prove that

\[
w^{-1}L(A, p') = (aw)^{-1}L(A, p) \quad (*)
\]

or in other words that for all \( u \in \Sigma^* \): \( wu \) is accepted from \( p' \) iff \( awu \) is accepted from \( p \).

Clearly, \( w^{-1}L(A, p') \subseteq (aw)^{-1}L(A, p) \) because \( p' \) is reached from \( p \) by an \( a \)-transition, and thus if \( wu \) is accepted from \( p' \), then \( awu \) is accepted from \( p \).

It remains to show \( w^{-1}L(A, p') \supseteq (aw)^{-1}L(A, p) \). Assume to the contrary that there is a word \( u \) such that \( awu \) is accepted from \( p \) but \( wu \) is not accepted.
from \( p' \). If \( u = \varepsilon \) is the empty word, then the winning condition for Player 1 is satisfied in the current vertex \((p', p, aw, 1)\) of the play, contradicting the fact that \( \sigma \) is a winning strategy for Player 0. Otherwise, \( u \) contains at least one letter, and Player 1 can simply play the transitions of an accepting run on \( awu \) from \( p \), and choose the letters from \( u \) to be appended to the lookahead. After Player 1 has appended the last letter of \( u \) to the lookahead, the play is in a vertex of the form \((p_0, p_1, a'w', 0)\) such that \( a'w' \in L(A, p_1) \), where \( a'w' \) is the suffix of length \( k + 1 \) of \( wu \), that is, \( wu = xa'w' \) for some \( x \). However, \( a'w' \not\in L(A, p_0) \) because otherwise we would also have that \( wu \in L(A, p') \), since Player 0 was playing a run from \( p' \) to \( p_0 \) on \( x \). Hence, independent of the next move of Player 0 to some vertex \((p'_0, p_1, a'w', 1)\), we get that \( w' \not\in L(A, p'_0) \), and thus the winning condition for Player 1 is satisfied. Since Player 0 is playing according to a winning strategy, we can conclude that such a word \( u \) cannot exist, which finishes the proof of \((*)\).

We can construct a delegator \( f \) as follows. Since we consider a safety game for Player 0, we can choose \( \sigma \) to be a positional winning strategy for Player 0, that is, the choices of \( \sigma \) only depend on the current vertex of the game. For each state \( p \in Q \), letter \( a \in \Sigma \) and lookahead \( w \in \Sigma^k \), consider the game vertex \((p, p, aw, 0)\) and let \((p', p, aw, 1)\) be the next vertex chosen by \( \sigma \). Define \( f(p, a, w) = p' \). This defines the delegator for lookaheads of length \( k \). Further, for each \( v \in \Sigma^\ell \) for \( \ell < k \) such that \( av \in L(A, p) \), define \( f(p, a, v) = p' \) for some transition \((p, a, p')\) such that \( v \) is accepted from \( p' \).

Now let \( x \in L(A) \). We need to show that \( f \) constructs an accepting run on \( x \). Let \( x = a_1 \cdots a_n \) with \( a_i \in \Sigma \), and let the run constructed by the delegator be \( q_0a_1q_1 \cdots a_nq_n \). We show that \( a_i+1 \cdots a_n \in L(A, q_i) \) for each \( i \in \{1, \ldots, n\} \), which implies the claim with \( i = n \) (and the convention that \( a_{n+1} \cdots a_n = \varepsilon \)). The proof goes by induction on \( i \).

For the base case \( i = 0 \) we have that \( a_1 \cdots a_n = x \in L(A) = L(A, q_0) \). For the induction step let \( i \geq 1 \) with \( a_i \cdots a_n \in L(A, q_{i-1}) \). We make a case distinction depending on the length of the remaining suffix of \( x \).

**Case 1:** If \( n - i + 1 \geq k + 1 \), that is, \( |a_i \cdots a_n| \geq k + 1 \), then the next transition is chosen according to the winning strategy \( \sigma \) from the game vertex \((q_{i-1}, q_{i-1}, a_i a_{i+1} \cdots a_{i+k}, 0)\). Since the previous transitions also have been chosen according to the strategy, the game vertex \((q_{i-1}, q_{i-1}, a_i a_{i+1} \cdots a_{i+k}, 0)\) is reachable in a play in which Player 0 uses \( \sigma \). From \((*)\) we conclude that \( w^{-1}L(A, q_i) = (aw)^{-1}L(A, q_{i-1}) \) with \( a = a_i \) and \( w = a_{i+1} \cdots a_{i+k} \). Since \( a_i \cdots a_n \in L(A, q_{i-1}) \) by induction hypothesis, we obtain \( a_{i+1} \cdots a_n \in L(A, q_i) \) as desired.
Case 2: The second case is that \( n - i + 1 < k + 1 \). Since \( a_i \cdots a_n \in L(A, q_{i-1}) \), by definition of \( f \), the next state is chosen according to an accepting run from \( q_{i-1} \) on \( a_i \cdots a_n \). This implies that \( a_{i+1} \cdots a_n \in L(A, q_i) \) as desired.

For fixed \( k \), the game graph is of polynomial size in \( A \) and can be constructed in polynomial time. Furthermore, one can compute in polynomial time the set of vertices from which Player 0 has a winning strategy (and also a corresponding positional strategy), which finishes the proof.

References


