On minimising automata with errors

Paweł Gawrychowski[1,*], Artur Jeż[1,*], and Andreas Maletti[2,**]

[1] Institute of Computer Science, University of Wrocław
ul. Joliot-Curie 15, 50-383 Wrocław, Poland
`{gawry,aje}@cs.uni.wroc.pl`
[2] Institute for Natural Language Processing, Universität Stuttgart
Azenbergstraße 12, 70174 Stuttgart, Germany
`andreas.maletti@ims.uni-stuttgart.de`

Abstract. The problem of k-minimisation for a DFA M is the computation of a smallest DFA N (where the size $|M|$ of a DFA M is the size of the domain of the transition function) such that $L(M) \triangle L(N) \subseteq \Sigma^{<k}$, which means that their recognized languages differ only on words of length less than k. The previously best algorithm, which runs in time $\mathcal{O}(|M| \log^2 n)$ where n is the number of states, is extended to DFAs with partial transition functions. Moreover, a faster $\mathcal{O}(|M| \log n)$ algorithm for DFAs that recognise finite languages is presented. In comparison to the previous algorithm for total DFAs, the new algorithm is much simpler and allows the calculation of a k-minimal DFA for each k in parallel. Secondly, it is demonstrated that calculating the least number of introduced errors is hard: Given a DFA M and numbers k and m, it is NP-hard to decide whether there exists a k-minimal DFA N with $|L(M) \triangle L(N)| \leq m$. A similar result holds for hyper-minimisation of DFAs in general: Given a DFA M and numbers s and m, it is NP-hard to decide whether there exists a DFA N with at most s states such that $|L(M) \triangle L(N)| \leq m$.

Keywords: finite automaton, minimisation, lossy compression

1 Introduction

Deterministic finite automata (DFAs) are one of the simplest devices recognising languages. The study of their properties is motivated by (i) their simplicity, which yields efficient operations, (ii) their wide-spread applications, (iii) their connections to various other areas in theoretical computer science, and (iv) the apparent beauty of their theory. A DFA M is a quintuple $\langle Q, \Sigma, \delta, q_0, F \rangle$, where Q is its finite state-set, Σ is its finite alphabet, $\delta \colon Q \times \Sigma \to Q$ is its partial transition function, $q_0 \in Q$ is its starting state, and $F \subseteq Q$ is its set of accepting states. The DFA M is total if δ is total. The transition function δ is extended to $\delta \colon Q \times \Sigma^* \to Q$ in the standard way. The language $L(M)$ that is *recognised by the DFA M* is $L(M) = \{w \mid \delta(q_0, w) \in F\}$.

Two DFAs M and N are *equivalent* (written as $M \equiv N$) if $L(M) = L(N)$. A DFA M is *minimal* if all equivalent DFAs are larger. One of the classical DFA problems is the *minimisation problem*, which given a DFA M asks for the (unique) minimal equivalent DFA. The asymptotically fastest DFA minimisation algorithm runs in time $\mathcal{O}(|\Sigma|\, n \log n)$ and is due to HOPCROFT [9,7], where $n = |Q|$; its variant for partial DFAs is known to run in time $\mathcal{O}(|M| \log n)$.

Recently, minimisation was also considered for hyper-equivalence [2,3], which allows a finite difference in the languages. Two languages L and L' are *hyper-equivalent* if $|L \bigtriangleup L'| < \infty$, where $\bigtriangleup$ denotes the symmetric difference of two sets. The DFAs M and N are hyper-equivalent if their recognised languages are. The DFA M is *hyper-minimal* if all hyper-equivalent DFAs are larger. The algorithms for hyper-minimisation [3,2] were gradually improved over time to the currently best run-time $\mathcal{O}(|M| \log^2 n)$ [8,6], which can be reduced to $\mathcal{O}(|M| \log n)$ using a strong computational model (with randomisation or special memory access). Since classical DFA minimisation linearly reduces to hyper-minimisation [8], an algorithm that is faster than $\mathcal{O}(|M| \log n)$ seems unlikely. Moreover, according to the authors' knowledge, randomisation does not help HOPCROFT's [5] or any other DFA minimisation algorithm. Thus, the randomised hyper-minimisation algorithm also seems to be hard to improve.

Already [3] introduces a stricter notion of hyper-equivalence. Two languages L and L' are *k-similar* if they only differ on words of length less than k. Analogously, DFAs are k-similar if their recognised languages are. A DFA M is *k-minimal* if all k-similar DFAs are larger, and the *k-minimisation problem* asks for a k-minimal DFA that is k-similar to the given DFA M. The known algorithm [6] for k-minimisation of total DFAs runs in time $\mathcal{O}(|M| \log^2 n)$, however it is quite complicated and fails for non-total DFAs.

In this contribution, we present a simpler k-minimisation algorithm for general DFAs, which still runs in time $\mathcal{O}(|M| \log^2 n)$. This represents a significant improvement compared to the complexity for the corresponding total DFA if the transition table of M is sparse. Its running time can be reduced if we allow a stronger computational model. In addition, the new algorithm runs in time $\mathcal{O}(|M| \log n)$ for every DFA M that recognises a finite language. Finally, the new algorithm can calculate (a compact representation of) a k-minimal DFA for each possible k in a single run (in the aforementioned run-time). Outputting all the resulting DFAs might take time $\Omega(n|M| \log^2 n)$.

Although k-minimisation can be efficiently performed, no uniform bound on the number of introduced errors is provided. In the case of hyper-minimisation, it is known [10] that the *optimal* (i.e., the DFA committing the least number of errors) hyper-minimal DFA and the number of its errors m can be efficiently computed. However, this approach does not generalise to k-minimisation. We show that this is for a reason: already the problem of calculating the number m of errors of an optimal k-minimal automaton is NP-hard.

Finally, for some applications it would be beneficial if we could balance the number m of errors against the compression rate $\frac{|N|}{|M|}$. Thus, we also consider the question whether given a DFA M and two integers s and m there is a DFA N

with at most s states that commits at most m errors (i.e., $|L(M) \triangle L(N)| \leq m$). Unfortunately, we show that this problem is also NP-hard.

2   Preliminaries

We usually use the two DFAs $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ and $N = \langle P, \Sigma, \mu, p_0, F' \rangle$. We also write $\delta(w)$ for $\delta(q_0, w)$. The *right-language* $L_M(q)$ of a state $q \in Q$ is the language $L_M(q) = \{w \mid \delta(q, w) \in F\}$ recognised by M starting in state q. Minimisation of DFAs is based on calculating the equivalence $\equiv$ between states, which is defined by $q \equiv p$ if and only if $L_M(q) = L_N(p)$. Similarly, the *left language* of q is the language $\delta^{-1}(q) = \{w \mid \delta(w) = q\}$ of words leading to q in M.

For two languages L and L', we define their *distance* $d(L, L')$ as

$$d(L, L') = \min \{\ell \mid L \cap \Sigma^{\geq \ell} = L' \cap \Sigma^{\geq \ell}\} \ ,$$

where $\min \emptyset = \infty$. Actually, d is an ultrametric. The distance d can be extended to states: $d(q, p) = d(L_M(q), L_N(p))$ for $q \in Q$ and $p \in P$. It satisfies the simple recursive formula:

$$d(q, p) = \begin{cases} 0 & \text{if } q \equiv p, \\ 1 + \max \{d(\delta(q, a), \mu(p, a)) \mid a \in \Sigma\} & \text{otherwise.} \end{cases} \quad (1)$$
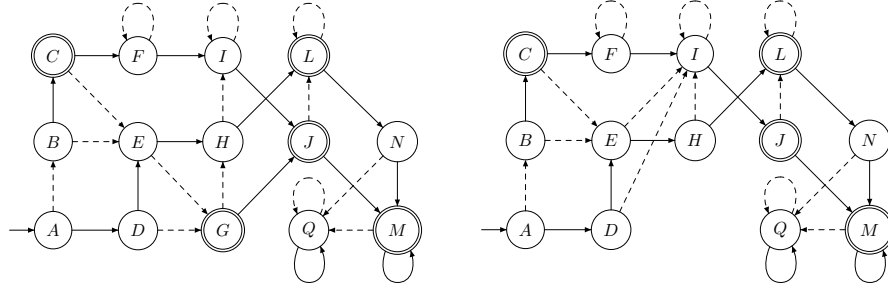
Since d is an ultrametric on languages, (1) yields that the distance $d(q_1, q_2)$ between $q_1, q_2 \in Q$ in the DFA M is either infinite or small. Formally, $d(q_1, q_2) = \infty$ or $d(q_1, q_2) < |Q|$.

The minimal DFAs considered in this paper are obtained mostly by state merging. We say that the DFA N is the result of *merging state q to state p* (assuming $q \neq p$) in M if N is obtained from M by changing all transitions ending in q to transitions ending in p and deleting the state q. If q was the starting state, then p is the new starting state. Formally, $P = Q \setminus \{q\}$, $F' = F \setminus \{q\}$, and

$$\mu(r, a) = \begin{cases} p & \text{if } \delta(r, a) = q \\ \delta(r, a) & \text{otherwise,} \end{cases} \qquad p_0 = \begin{cases} p & \text{if } q_0 = q \\ q_0 & \text{otherwise.} \end{cases}$$

The process is illustrated in Fig. 1.

Finally, let $in\text{-}level_M(q)$ be the length of the longest word leading to q in M. If there is no such longest word, then $in\text{-}level_M(q) = \infty$. Formally, $in\text{-}level_M(q) = \sup \{|w| \mid w \in \delta^{-1}(q)\}$ for every $q \in Q$. The structural characterisation of hyper-minimal DFAs [3, Sect. 3.2] relies on a state classification into *kernel* and *preamble* states. The set $\text{Ker}(M)$ of kernel states consists of all states $q \in Q$ with $in\text{-}level_M(q) = \infty$, whereas the remaining states are preamble states. Roughly speaking, the kernels of two hyper-equivalent and hyper-minimal automata are isomorphic in the standard sense, and their preambles are also isomorphic except for acceptance values.

Fig. 1. Merging state G into I.

3   Efficient k-minimisation

3.1 k-similarity and k-minimisation

Two languages L and L' are *k-similar* if they only differ on words of length smaller than k, and the two DFAs M and N are k-similar if their recognised languages are. The DFA M is *k-minimal* if all k-similar DFAs are larger. In this section, we first give a general simple algorithm k-Minimise that computes a k-minimal DFA that is k-similar to the input DFA M. Then we present a data structure that allows a fast, yet simple implementation of this algorithm.

Definition 1. *For two languages L and L', we let $L \sim_k L' \iff d(L, L') \leq k$.*

The hyper-equivalence relation [3] can be now defined as $\sim = \bigcup_k \sim_k$. Next, we extend k-similarity to states.

Definition 2. *Two states $q \in Q$ and $p \in P$ are k-similar, denoted by $q \sim_k p$, if*

$$d(q, p) + \min(k, \textit{in-level}_M(q), \textit{in-level}_N(p)) \leq k \ .$$

While $\sim_k$ is an equivalence relation on languages, it is, in general, only a compatibility relation (i.e., reflexive and symmetric) on states. On states the hyper-equivalence is not a direct generalisation of k-similarity. Instead, $p \sim q$ if and only if $L_M(q) \sim L_N(p)$. We use the k-similarity relation to give a simple algorithm k-Minimise(M), which constructs a k-minimal DFA (see Algorithm 1). In Section 3.2 we show how to implement it efficiently.

Theorem 1. *k-Minimise returns a k-minimal DFA that is k-similar to M.*

3.2   Distance forests

In this section we define distance forests, which capture the information of the distance between states of a given minimal DFA M. We show that k-minimisation can be performed in linear time, when a distance forest for M is supplied. We

---

Algorithm 1 k-MINIMISE(M) with minimal M

1: calculate $\sim_k$ on Q
2: $N \leftarrow M$
3: **while** $q \sim_k p$ for some $q, p \in P$ and $q \neq p$ **do**
4:     **if** $in\text{-}level_M(q) \geq in\text{-}level_M(p)$ **then**
5:         swap q and p
6:     $N \leftarrow$ MERGE(N, q, p)

---

start with a total DFA M because in this case the construction is fairly easy. In Section 3.3 we show how to extend the construction to non-total DFAs.

Let $\mathcal{F}$ be a forest (i.e., set of trees) whose leaves are enumerated by Q and whose edges are weighted by elements of $\mathbb{N}$. For convenience, we identify the leaf vertices with their label. For every $q \in Q$, we let tree$(q) \in \mathcal{F}$ be the (unique) tree that contains q. The level level(v) of a vertex v in $t \in \mathcal{F}$ is the maximal weight of all paths from v to a leaf, where the weights are added along a path. Finally, given two vertices v_1, v_2 of the same tree $t \in \mathcal{F}$, the lowest common ancestor of v_1 and v_2 is the vertex lca(v_1, v_2).

Definition 3 (Distance forest). *Let $\mathcal{F}$ be a forest whose leaves are enumerated by Q. Then $\mathcal{F}$ is a* distance forest *for M if for every $q, p \in Q$ we have*

$$d(q, p) = \begin{cases} \text{level}(\text{lca}(q, p)) & \text{if } \text{tree}(q) = \text{tree}(p), \\ \infty & \text{otherwise.} \end{cases}$$
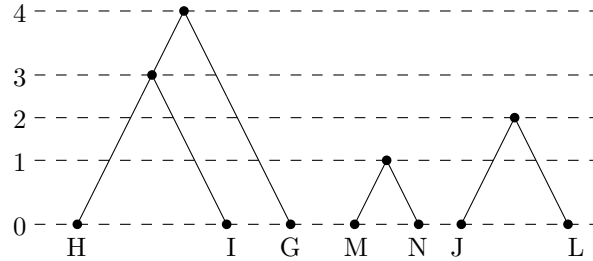

Fig. 2. A distance forest for the left DFA of Fig. 1. Single-node trees are omitted.

In order to construct a distance forest we use (1) to calculate the distance. Mind that M is minimal, so there are no states with distance 0. In phase ℓ, we merge all states at distance exactly ℓ into one state. Since we merged all states of distance at most $\ell - 1$ in the previous phases, we only need to identify the states of distance 1 in the merged DFA. Thus we simply group the states according to their vectors of transitions by letters from $\Sigma = \{a_1, \ldots, a_m\}$. To this end we store these vectors in a dictionary, organised as a trie of depth m. The leaf of

a trie corresponding to a path $(q_1, \ldots, q_m)$ keeps a list of all states q such that $\delta(q, a_i) = q_i$ for every $1 \le i \le m$. For each node v in the trie we keep a *linear dictionary* that maps a state q into a child of v. We demand that this linear dictionary supports search, insertion, deletion, and enumeration of all elements.

Theorem 2. *Given a total DFA M, we can build a distance forest for M using $\mathcal{O}(|M| \log n)$ linear-dictionary operations.*

We now shortly discuss some possible implementations of the linear dictionary. An implementation using balanced trees would have linear space consumption and the essential operations would run in time $\mathcal{O}(\log n)$. If we allow randomisation, then we can use dynamic hashing. It has a worst-case constant time look-up and an amortised expected constant time for updates [11]. Since it is natural to assume that $\log n$ is proportional to the size of a machine word, we can hash in constant time. We can obtain even better time bounds by turning to more powerful models. In the RAM model, we can use exponential search trees [1], whose time per operation is $\mathcal{O}(\frac{(\log \log n)^2}{\log \log \log n})$ in linear space. Finally, if we allow a quadratic space consumption, which is still possible in sub-quadratic time, then we can allocate (but not initialise) a table of size $|M| \times n$. Standard methods can be used to keep track of the actually used table entries, so that we obtain a constant run-time for each operation, but at the expense of $\Theta(|M| n)$ space; i.e., quadratic memory consumption.

We can now use a distance forest to efficiently implement k-MINIMISE. For each state q we locate its highest ancestor v_q with $\text{level}(v_q) \le k - \text{in-level}(q)$. Then q can be merged into any state that occurs in the subtree rooted in v_q (assuming it has a smaller *in-level*). This can be done using a depth-first traversal on the trees of the distance forest. A more elaborate construction based on this approach yields the following.

Theorem 3. *Given a distance forest for M, we can compute the size of a k-minimal DFA that is k-similar to M for all k in time $\mathcal{O}(|M|)$. For a fixed k, we can also compute a k-minimal DFA in time $\mathcal{O}(|M|)$. Finally, we can run the algorithm in time $\mathcal{O}(|M| \log n)$ such that it has a k-minimal DFA stored in memory in its k-th phase.*

3.3   Finite languages and partial transition functions

The construction of a distance forest was based on a total transition function δ, and the run-time was bounded by the size of δ. We now show a modification for the non-total case. The main obstacle is the construction of a distance forest for an acyclic DFA. The remaining changes are relatively straightforward.

Theorem 4. *For every acyclic DFA M we can build a distance forest in time $\mathcal{O}(|M| \log n)$.*

Proof (sketch). Since $L(M)$ is finite, we have that $m(p) = \max \{|w| \, | \, w \in L_M(p)\}$ is a natural number for every state p. Let $Q_i = \{p \, | \, m(p) = i\}$ and $Q_{<\infty} = \bigcup_i Q_i$.

Every state has a finite right-language, and thus every distance forest consists of a single tree. We iteratively construct the fragments of this tree by starting from a single leaf $\bot$, which represents the empty language and "undefinedness" of the transition function. Before we start to process Q_t, we have already constructed the distance tree for $\bigcup_{i<t} Q_i$. The constructed fragments are connected to a single path, called the *spine*, which ends at the leaf $\bot$ (see Fig. 3).

Let $Q_t = \{p_1, \ldots, p_s\}$, and let $v \in Q_t$. Moreover, let $\mathrm{f}(v)$ be the vector of states $\mathbf{v} = (\delta(v, a))_{a \in \Sigma}$, where the coordinates are sorted by a fixed order on Σ. Define the distance between those vectors as

$$d((p_a)_{a \in \Sigma}, (p'_a)_{a \in \Sigma}) = \max \{d(p_i, p'_i) + 1 \mid a \in \Sigma\} \ ,$$

where we know that $d(p_i, \bot) = m(p_i)$ and $d(\bot, p'_i) = m(p'_i)$. Similarly to the distance, we can define the father $\mathrm{f}(\mathbf{v})$ of a vector $\mathbf{v} = (p_a)_{a \in \Sigma}$ as $\mathrm{f}(\mathbf{v}) = (\mathrm{f}(p_a))_{a \in \Sigma}$. Then

$$\mathrm{f}^{\ell+1}(v) = \mathrm{f}^{\ell+1}(v') \iff \mathrm{f}^{\ell}(\mathbf{v}) = \mathrm{f}^{\ell}(\mathbf{v}').$$

We can now use a divide-and-conquer approach: First, for each vector we calculate its 2^k-th ancestor, where $k = \lceil \log s/2 \rceil$. Then all such vectors are sorted according to their ancestors, in particular they are partitioned into blocks with the same ancestors. After that we recurse onto those (bottom) blocks that have more than two entries and onto the upper block, which consists of the different 2^k-ancestors. The recursion ends for blocks containing at most two vectors, for which we calculate the distance tree directly. $\qquad\qquad\square$
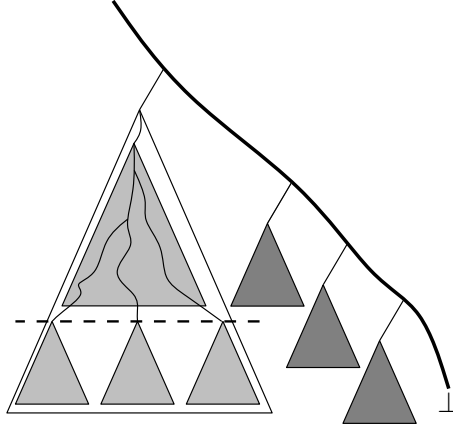

Fig. 3. Illustration for the construction of the distance tree. The spine is depicted using with a thicker line. Splitting one fragment into smaller recursive calls is shown.

For every state $q \in Q$, its *signature* $\mathrm{sig}(q)$ is $\{a \mid L_M(\delta(q, a))$ is infinite$\}$. If $\mathrm{sig}(q) \neq \mathrm{sig}(p)$, then $d(q, p) = \infty$, which allows us to keep a separate dictionary for each signature. Let us fix such a trie. To take into account also the transitions by letters outside the signature, we introduce a fresh letter $\$$, whose transitions are represented in the trie as well. We organize them such that in phase ℓ the $\$$-transitions for the states q and p are the same if and only if $\max \{d(\delta(q, a), \delta(p, a)) \mid a \notin \mathrm{sig}(q)\} \leq \ell - 1$. This is easily organised if the distance forest for all states with a finite right-language is supplied.

Theorem 5. *Given a (non-total) DFA M we can build a distance forest for it using $\mathcal{O}(|M|\log n)$ linear-dictionary operations.*

4   Hyper-equivalence and hyper-minimisation

When considering minimisation with errors, it is natural that one would like to impose a bound on the total number of errors introduced by minimisation. In this section, we investigate whether given $m, s \in \mathbb{N}$ and a DFA M we can construct a DFA N such that:

 (i) N is hyper-equivalent to M; i.e., $N \sim M$,
 (ii) N has at most s states, and
(iii) N commits at most m errors compared to M; i.e., $|L(N) \triangle L(M)| \leq m$.

Let us call the general problem 'error-bounded hyper-minimisation'. We show that this problem is intractable (NP-hard). Only having a bound on the number of errors allows us to return the original DFA, which commits no errors.

To show NP-hardness of the problem we reduce the 3-colouring problem to it. Roughly speaking, we construct the DFA M from a graph $G = \langle V, E \rangle$ as follows. Each vertex $v \in V$ is represented by a state $v \in Q$, and each edge $e \in E$ is represented by a symbol $e \in \Sigma$. We introduce additional states in a way such that their isomorphic copies are present in any minimal DFA that is hyper-equivalent to M. The additional states are needed to ensure that for every edge $e = \{v_1, v_2\} \in E$ the languages $L_M(\delta(v_1, e))$ and $L_M(\delta(v_2, e))$ differ. Now we assume that $m = |E| \cdot (|V| - 2)$ and $s = 14$. We construct the DFA M such that all vertices of $V \subseteq Q$ are hyper-equivalent to each other and none is hyper-equivalent to any other state. We can save $|V| - 3$ states by merging all states of V into at most 3 states. These merges will cause at least $|E| \cdot (|V| - 2)$ errors. Additionally, 3 states will become superfluous after the merges, so that we can save $|V|$ states. There are two cases:

 – If the input graph G is 3-colourable by $c \colon V \to [3]$, then we can merge all states of $c^{-1}(i)$ into a single state for every $i \in [3]$. Since c is proper, we never merge states $v_1, v_2 \in Q$ with $\{v_1, v_2\} \in E$, which avoids further errors.
 – On the other hand, if G is not 3-colourable, then we merge at least two states $v_1, v_2 \in Q$ such that $e = \{v_1, v_2\} \in E$. This merge additionally introduces 2 errors caused by the difference $L(\delta(v_1, e)) \triangle L(\delta(v_2, e))$.

Consequently, a DFA that (i) is hyper-equivalent to M, (ii) has at most s states, and (iii) commits at most m errors exists if and only if G is 3-colourable. This shows that error-bounded hyper-minimisation is NP-hard.

Definition 4. *We construct a DFA $M = \langle Q, \Sigma, \delta, \top, F \rangle$ as follows:*

 – $Q = \{\top, \bot, \infty, \odot, \ominus\} \cup V \cup \{\bigcirc_j \mid \bigcirc \in \{\odot, \bullet, \ominus\}, j \in [3]\}$,
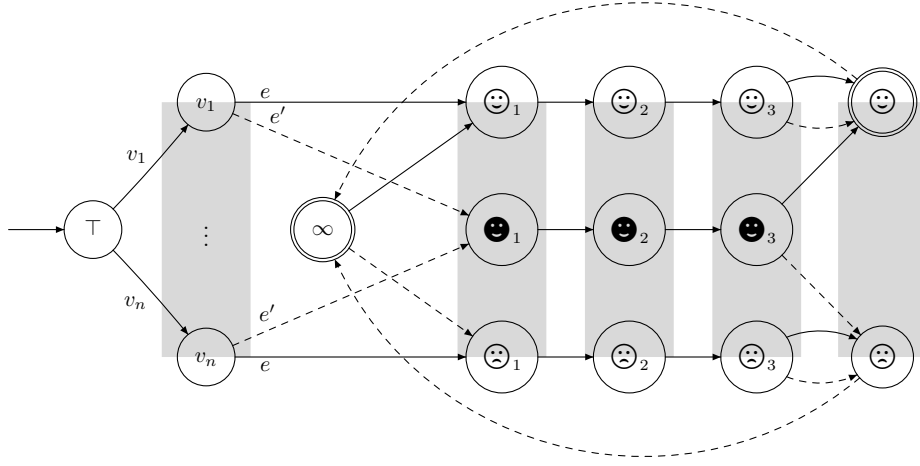 – $\Sigma = \{a, b\} \cup V \cup E$,
 – $F = \{\infty, \odot\}$,

Fig. 4. DFA M constructed in Section 4, where a-transitions are represented by unbroken lines (unless noted otherwise), b-transitions by dashed lines, and $e = \{v_1, v_n\}$ and $e' = \{v_2, v_3\}$ with $v_1 < v_2 < v_3 < v_n$. The hyper-equivalence $\sim$ is indicated.

– for every $v \in V$, $e = \{v_1, v_2\} \in E$ with $v \notin e$ and $v_1 < v_2$, $\bigcirc \in \{\odot, \odot\}$

$$\delta(\top, v) = v \qquad \delta(\infty, a) = \odot_1 \qquad \delta(\infty, b) = \odot_1$$
$$\delta(v, e) = \bullet_1 \qquad \delta(v_1, e) = \odot_1 \qquad \delta(v_2, e) = \odot_1$$

$$\delta(\bullet_1, a) = \bullet_2 \quad \delta(\bullet_2, a) = \bullet_3 \quad \delta(\bullet_3, a) = \odot \quad \delta(\bullet_3, b) = \odot$$
$$\delta(\bigcirc_1, a) = \bigcirc_2 \quad \delta(\bigcirc_2, a) = \bigcirc_3 \quad \delta(\bigcirc_3, a) = \bigcirc \quad \delta(\bigcirc_3, b) = \bigcirc \quad \delta(\bigcirc, b) = \infty$$

– For all remaining cases, we set $\delta(q, \sigma) = \bot$.

Consequently, the DFA M has $14 + |V|$ states (see Figure 4). Next, we show how to collapse hyper-equivalent states using a proper 3-colouring $c\colon V \to [3]$ to obtain only 14 states.

Definition 5. Let $c\colon V \to [3]$ be a proper 3-colouring for G. We construct the DFA $c(M) = \langle P, \Sigma, \mu, \top, F \rangle$ where

– $P = \{\top, \bot, \infty, \odot, \odot\} \cup [3] \cup \{\bigcirc_j \mid \bigcirc \in \{\odot, \odot\}, j \in [3]\}$,
– $\mu(p, \sigma) = \delta(p, \sigma)$ for all $p \in P \setminus \{\top, 1, 2, 3\}$ and $\sigma \in \Sigma$, and
– for every $v \in V$, $i \in [3]$, and $e = \{v_1, v_2\} \in E$ with $v_1 < v_2$

$$\mu(\top, v) = c(v) \qquad\qquad \mu(i, e) = \begin{cases} \odot_1 & , \text{ if } c(v_2) \neq i \\ \odot_1 & , \text{ otherwise.} \end{cases}$$

Lemma 1. There exists a DFA that has at most 14 states and commits at most $|E| \cdot (|V| - 2)$ errors when compared to M if and only if G is 3-colourable.

Corollary 1. 'Error-bounded hyper-minimisation' is NP-complete. More formally, given a DFA M and two integers $m, s \in \mathrm{poly}(|M|)$, it is NP-complete to decide whether there is a DFA N with at most s states and $|L(M) \triangle L(N)| \leq m$.

5   Error-bounded k-minimisation

In Section 3 the number of errors between M and the constructed k-minimal DFA was not calculated. In general, there is no unique k-minimal DFA for M and the various k-minimal DFAs for M can differ in the number of errors that they commit relative to M. Since several dependent merges are performed in the course of k-minimisation, the number of errors between the original DFA M and the resulting k-minimal DFA is not necessarily the sum of the errors introduced for each merging step. This is due to the fact that errors made in one merge might be cancelled out in a subsequent merge. It is natural to ask, whether it is nevertheless possible to *efficiently* construct an *optimal k-minimal* DFA for M (i.e., a k-minimal DFA with the least number of errors introduced). In the following we show that the construction of an optimal k-minimal DFA for M is intractable (NP-hard).

The intractability is shown by a reduction from the 3-colouring problem for a graph $G = \langle V, E \rangle$ in a similar, though much more refined, way as in Section 4. We again construct a DFA M with one state v for every vertex $v \in V$ and one letter e for each edge $e \in E$. We introduce three additional states $\{1_0, 2_0, 3_0\}$ (besides others) to represent the 3 colours. For the following discussion, let $N = \langle P, \Sigma, \mu, p_0, F' \rangle$ be a k-minimal DFA for M. Let us fix an edge $e = \{v_1, v_2\} \in E$. The DFA M is constructed such that the languages $L_M(\delta(v_1, e))$ and $L_M(\delta(v_2, e))$ have a large but finite symmetric difference; as in the previous section, if a proper 3-colouring $c\colon V \to [3]$ exists the DFA N can be obtained by merging each state v into $c(v)_0$. In addition, for every edge $e = \{v_1, v_2\} \in E$ and vertex $v \in e$, we let $\mu(c(v)_0, e) = \delta(v, e)$. On the other hand, if G admits no proper 3-colouring, then the DFA N is still obtained by state merges performed on M. However, because G has no proper 3-colouring, in the constructed DFA M there exist 2 states v_1, v_2 such that $e = \{v_1, v_2\} \in E$ and that both v_1 and v_2 are merged into the same state $p \in P$. Then the transition $\mu(p, e)$ cannot match both $\delta(v_1, e)$ and $\delta(v_2, e)$. In order to make such an error costly, the left languages of v and v' are designed to be large, but finite. In contrast, we can easily change the transitions of states $\{1_0, 2_0, 3_0\}$ by letters e because the left-languages of the states $\{1_0, 2_0, 3_0\}$ are small.

To keep the presentation simple, we will use two gadgets. The first one will enable us to make sure that two states cannot be merged: k-similar states are also hyper-equivalent, so we can simply avoid undesired merges by making states hyper-inequivalent. Another gadget will be used to increase the in-level of certain states to a desired value.

Lemma 2. *For every congruence $\simeq\, \subseteq Q \times Q$ on M, there exists a DFA N such that (i) $p_1 \nsim p_2$ for every $p_1 \in P \setminus Q$ and $p_2 \in P$ with $p_1 \neq p_2$, and (ii) $q_1 \nsim q_2$ in N for all $q_1 \not\simeq q_2$.*

In graphical illustrations, we use different shapes for q_1 and q_2 to indicate that $q_1 \nsim q_2$, because of the gadget of Lemma 2. Note that states with the same shape need not be k-similar.

Lemma 3. *For every subset $S \subseteq Q \setminus \{q_0\}$ of states and map* min-level$: S \to \mathbb{N}$, *there exists a DFA $N = \langle Q \cup I, \Sigma \cup \Delta, \mu, q_0, F \rangle$ such that $|\mu^{-1}(i)| = 1$ for every $i \in I$ and* in-level$_N(s) \geq$ min-level(s) *for every $s \in S$.*

We will indicate the level i below the state name in graphical illustrations. Moreover, we add a special feathered arrow to the state q, whenever the gadget is used for the state q to increase its level.

Next, let us present the formal construction. Let $G = \langle V, E \rangle$ be an undirected graph. Select $k, s \in \mathbb{N}$ such that $s > \log(|V|) + 2$ and $k > 4s$. Moreover, let $\ell = k - 2s$.
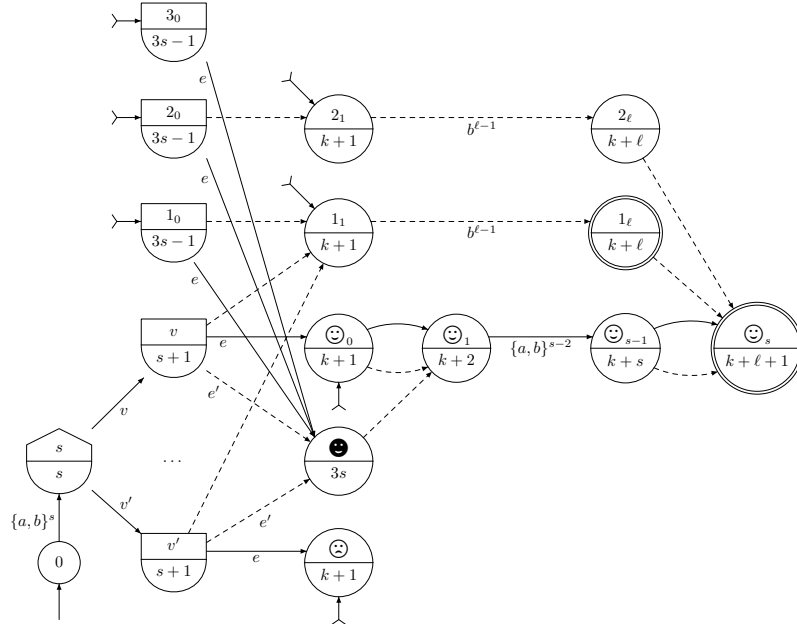

Fig. 5. Illustration of the DFA M of Section 5

Definition 6. *We construct the DFA $M = \langle Q, \Sigma, \delta, 0, F \rangle$ as follows:*

- $Q = \{\perp, \text{\ding{109}}, \odot, 3_0\} \cup \{i_j \mid i \in [2], j \in [\ell]\} \cup V \cup [0, s] \cup \{\odot_i \mid 0 \leq i \leq s\}$,
- $\Sigma = \{a, b\} \cup V \cup E$,
- $F = \{\odot_s, 1_\ell\}$, *and*
- *for every $v \in V$, $e = \{v_1, v_2\} \in E$ with $v \notin e$ and $v_1 < v_2$, $i \in [s]$, and $j \in [\ell]$*

$$
\begin{array}{llll}
\delta(i-1, a) = i & \delta(v_1, e) = \odot_0 & \delta(1_0, e) = \text{\ding{109}} & \delta(1_{j-1}, b) = 1_j \\
\delta(i-1, b) = i & \delta(v_2, e) = \odot & \delta(2_0, e) = \text{\ding{109}} & \delta(2_{j-1}, b) = 2_j \\
\delta(\odot_{i-1}, a) = \odot_i & \delta(v, e) = \text{\ding{109}} & \delta(3_0, e) = \text{\ding{109}} & \delta(1_\ell, b) = \odot_s \\
\delta(\odot_{i-1}, b) = \odot_i & \delta(v, a) = 1_1 & & \delta(2_\ell, b) = \odot_s \\
\delta(\text{\ding{109}}, a) = \odot_1 & \delta(s, v) = v & &
\end{array}
$$

- *For all remaining cases, we set $\delta(q, \sigma) = \perp$.*

Finally, we show how to collapse k-similar states using a proper 3-colouring $c \colon V \to [3]$. We obtain the k-similar DFA $c(M) = \langle P, \Sigma, \mu, 0, F \rangle$ from M by merging each state v into $c(v)_0$. In addition, for every edge $e = \{v_1, v_2\} \in E$, we let $\mu(c(v_1)_0, e) = \delta(v_1, e)$ and $\mu(c(v_2)_0, e) = \delta(v_2, e)$. Since the colouring c is proper, we have that $c(v_1) \neq c(v_2)$, which yields that μ is well-defined. For the remaining $i \in [3] \setminus \{c(v_1), c(v_2)\}$, we let $\mu(i_0, e) = \odot_0$. All equivalent states (i.e., $\bot$ and $\odot$) are merged. The gadgets that were added to M survive and are added to $c(M)$. Naturally, if a certain state does no longer exist, then all transitions leading to or originating from it are deleted too. This applies for example to ☻.

Lemma 4. *There exists a k-minimal DFA N for M with at most*

$$2^{2s-1} \cdot |E| \cdot (|V| - 2) + 3 \cdot 2^{s-1} \cdot |E| + 2^{s+1} \cdot |V|$$

errors if and only if the input graph G is 3-colourable.

Corollary 2. *'Error-bounded k-minimisation' is NP-complete.*

References

1. Andersson, A., Thorup, M.: Dynamic ordered sets with exponential search trees. J. ACM 54(3) (2007)
2. Badr, A.: Hyper-minimization in $\mathcal{O}(n^2)$. In: Proc. 13th Int. Conf. Implementation and Application of Automata. LNCS, vol. 5148, pp. 223–231. Springer (2008)
3. Badr, A., Geffert, V., Shipman, I.: Hyper-minimizing minimized deterministic finite state automata. RAIRO, Theoret. Inform. Appl. 43(1), 69–94 (2009)
4. Bender, M.A., Farach-Colton, M.: The level ancestor problem simplified. Theor. Comput. Sci. 321(1), 5–12 (2004)
5. Castiglione, G., Restivo, A., Sciortino, M.: Hopcroft's algorithm and cyclic automata. In: Proc. 2nd Int. Conf. Language and Automata Theory and Applications. LNCS, vol. 5196, pp. 172–183. Springer (2008)
6. Gawrychowski, P., Jeż, A.: Hyper-minimisation made efficient. In: Proc. 34th Int. Symp. Mathematical Foundations of Computer Science. LNCS, vol. 5734, pp. 356–368. Springer (2009)
7. Gries, D.: Describing an algorithm by Hopcroft. Acta Inf. 2(2), 97–109 (1973)
8. Holzer, M., Maletti, A.: An $n \log n$ algorithm for hyper-minimizing a (minimized) deterministic automaton. Theoret. Comput. Sci. 411(38-39), 3404–3413 (2010)
9. Hopcroft, J.E.: An $n \log n$ algorithm for minimizing states in a finite automaton. In: Kohavi, Z. (ed.) Theory of Machines and Computations, pp. 189–196. Academic Press (1971)
10. Maletti, A.: Better hyper-minimization — not as fast, but fewer errors. In: Proc. 15th Int. Conf. Implementation and Application of Automata. LNCS, vol. 6482, pp. 201–210. Springer (2011)
11. Pagh, R., Rodler, F.F.: Cuckoo hashing. J. Algorithms 51(2), 122–144 (2004)

A   Proofs and additional material for Section 2

Lemma 5. *If $p, q \in Q$ then $d(p, q) < +\infty$ implies that $d(p, q) < n$.*

Proof. Let D_i denote the equivalence relation defined as $D_i(p, q)$ iff $d(p, q) \leq i$. Let n_i be the number of equivalence classes of D_i, for $i = 0, 1, \ldots$. Note that if $n_i = n_{i+1}$ then $n_i = n_j$ for all $j > i$ and $d(q, q') > i$ implies $d(q, q') = +\infty$.

Since $n_0 \leq |Q|$ the sequence $n_0 \geq n_1 \geq \ldots$ stabilises at position $n_{|Q|-1}$, i.e., there are no states q, q' such that $D_{|Q|}(q, q')$ and $\neg D_{|Q|-1}(q, q')$. Hence $d(q, q') < +\infty$ implies $D_{|Q|-1}(q, q')$, i.e., $d(q, q') < n$. $\qquad\square$

B   Proofs and additional material for Section 3

B.1   Proofs and additional material for Section 3.1

It can be shown that if $M \sim_k N$ then the states reached after reading the same word are also k-similar, assuming that the word is short enough.

Lemma 6. *Let $M \sim_k N$, $q_1, q_2 \in Q$, and $w_1, w_2 \in \Sigma^*$ be such that $\delta(w_i) = q_i$ and $|w_i| = \text{in-level}_M(q_i)$ for $i \in [2]$. If $q_1 \not\sim_k q_2$, then $\mu(w_1) \not\sim_k \mu(w_2)$.*

Proof. First, suppose that $q_1 \not\sim q_2$. Then, $M \sim N$ yields that

$$\mu(w_1) \sim \delta(w_1) = q_1 \not\sim q_2 = \delta(w_2) \sim \mu(w_2)$$

and thus $\mu(w_1) \not\sim \mu(w_2)$, which proves that $\mu(w_1) \not\sim_k \mu(w_2)$.

Second, let $d(q_1, q_2) < \infty$. Since $q_1 \not\sim_k q_2$, we have

$$d(q_1, q_2) + \min(k, |w_1|, |w_2|) > k \ .$$

Clearly, there exists $u \in L_M(q_1) \triangle L_M(q_2)$ with $|u| \geq d(q_1, q_2) - 1$. Moreover, $|w_1 u| \geq k \leq |w_2 u|$. Since $M \sim_k N$, we have $w_1 u, w_2 u \notin L(M) \triangle L(N)$. Consequently

$$u \notin L_M(q_1) \triangle L_N(\mu(w_1)) \qquad \text{and} \qquad u \notin L_M(q_2) \triangle L_N(\mu(w_2)) \ .$$

By assumption, $u \in L_M(q_1) \triangle L_M(q_2)$ and thus $u \in L_N(\mu(w_1)) \triangle L_N(\mu(w_2))$, which shows that $d(\mu(w_1), \mu(w_2)) \geq |u| + 1$. Clearly, $\text{in-level}_N(\mu(w_1)) \geq |w_1|$ and $\text{in-level}_N(\mu(w_2)) \geq |w_2|$, which yields $\mu(w_1) \not\sim_k \mu(w_2)$. $\qquad\square$

We show some properties of k-MINIMISE, which are used to show that it properly constructs a k-minimal DFA. Let N denote the DFA constructed by k-MINIMISE at any particular point.

Lemma 7. *If $\delta_N(p', w) = p$ and $\text{in-level}_M(p) < k$ then*

$$|w| \leq \text{in-level}_M(p) - \text{in-level}_M(p'). \tag{2}$$

Proof. The assertion of the lemma is shown to be hold after each merge done by k-MINIMISE, i.e., by the induction on the number of merges done by k-MINIMISE. If there were no merges done yet then $N = M$ and the claim holds true. Let N denote the DFA before the merge and N' after it.

We focus on $w = a \in \Sigma$. So assume that $\delta_{N'}(p_1, a) = p_2$ after merging state p to q. The only non-trivial case is when $\delta_N(p_1, a) = p$ and $p_2 = q$, i.e., when something is changed after the merging. By induction assumption $in\text{-}level_M(p) - in\text{-}level_M(p_1) \geq 1$. As $in\text{-}level_M(q) \geq in\text{-}level_M(p)$ as guaranteed by k-MINIMISE, the claim is obtained.

When $|w| > 1$ it is enough to consider the states obtained after transitions after each letter of w and sum up the inequalities. $\square$

Lemma 8. *During the run of k-MINIMISE for all $p' \in Q(N)$,*

$$d(L_M(p'), L_N(p')) \leq \max(0, k - in\text{-}level_M(p')). \tag{3}$$

Proof. We establish this claim by induction. Let N' denote the DFA after merging p to q and N just before this merge. Note, that as $p \not\equiv q$ (in M), thus $d(p, q) > 0$ (in M). Thus $p \sim_k q$ implies $\min(k, in\text{-}level_M(p), in\text{-}level_M(q)) < k$. Since p is merged to q by k-MINIMISE, $in\text{-}level_M(p) \leq in\text{-}level_M(q)$ and as $p \sim_k$ q also $in\text{-}level_M(p) < k$. Then by Lemma 2 we conclude that there is no word leading from q to p in N: assume for the sake of contradiction that there is such a word w. Since $in\text{-}level_M(p) < k$, by Lemma 2

$$in\text{-}level_M(p) \geq in\text{-}level_M(q) + |w| > in\text{-}level_M(q) \ ,$$

contradiction. Thus there is now word leading from q to p in N and therefore $L_N(q) = L_{N'}(q)$.

For the other case, lest us first estimate $d(L_N(p), L_{N'}(q)) = d(L_N(p), L_N(q))$. As already noted, $\min(k, in\text{-}level_M(p), in\text{-}level_M(q)) = in\text{-}level_M(p)$, which allows us to reduce $p \sim_k q$ to

$$d(L_M(p), L_M(q)) + in\text{-}level_M(q) \leq k$$

and thus

$$d(L_M(p), L_M(q)) \leq k - in\text{-}level_M(p).$$

By induction assumption

$$d(L_M(q), L_N(q)) \leq \max(0, k - in\text{-}level_M(q))$$
$$d(L_M(p), L_N(p)) \leq \max(0, k - in\text{-}level_M(q))$$
$$= k - in\text{-}level_M(p)$$

and as d is an ultra metric

$$d(L_N(p), L_N(q)) \leq k - in\text{-}level_M(p) \ . \tag{4}$$

So consider an arbitrary state p'. If it has no word leading to p in N, then $L_N(p') = L_{N'}(p')$ and we are done. If it has a word w leading to p, then Lemma 7 can be applied, establishing:

$$d(L_N(p'), L_{N'}(p')) = \max_{w:\delta_N(p',w)=p} |w| + d(L_{N'}(p), L_N(q))$$

the former can be estimated by (3) and the latter by (4), yielding

$$d(L_N(p'), L_{N'}(p')) \leq (in\text{-}level_M(p) - in\text{-}level_M(p')) + (k - in\text{-}level_M(p))$$
$$\leq k - in\text{-}level_M(p'),$$

which ends the proof.   □

Proof (of Theorem 1). Let $q_0, q_1, \ldots, q_n$ be the starting states in DFAs $M = N_0$, $N_1, \ldots, N_n = N$. By Lemma 8, $d(L_M(q_i), L_{N_i}(q_i)) \leq k$. On the other hand, since q_i is merged to q_{i+1} then $d(L_M(q_i), L_M(q_{i+1})) \leq k$. So all the languages in question are within distance k of each other and therefore

$$d(L(M), L(N)) \leq k \ .$$

Thus $M \sim_k N$. It is left to show that N is k-minimal. Consider the set of states Q' of N and let M' be a DFA k-similar to M. By k-MINIMISE, they are pairwise k-dissimilar (as states in M). For a state $q \in Q'$ let w_q be the word such that $|w_q| \geq \min(in\text{-}level_M(q), k)$. Consider any two such words w_q and w_p. Then by Lemma 6 w_q and w_p cannot lead to the same state in M'. Hence the size of M' is at least $|Q'|$, which is exactly the size of N.   □

Corollary 3 (of Theorem 1). *Each maximal (with respect to the inclusion) set Q' of pairwise k-dissimilar states of a DFA M is of size of the k-minimal DFA for M.*

Proof. First note that without loss of generality we may assume that

$$p \in Q', \ q \notin Q' \text{ and } p \sim_k q \text{ implies } in\text{-}level_M(p) \geq in\text{-}level_M(q). \qquad (5)$$

If not, then we can replace p by q in Q', without loosing the assumed property of Q'. After finitely many such substitutions, Q' satisfying (5) is obtained.

Run k-MINIMISE for M, and whenever there are two states $p \sim_k q$ considered, merge the one outside Q' to the one in Q' (do arbitrarily, if none is in Q'). Since Q' is maximal with respect to the inclusion, k-MINIMISE terminates with the DFA with Q' as the set of states.   □

Now we are able to establish a structural characterisation of k-similar DFAs, analogous to characterisation of hyper-equivalent DFA's [3, Sect. 3.2]. In particular, we derive the analogue of [3, Theorem 3.8] for k-similar DFAs.

Corollary 4 (of Lemma 6 and Corollary 3). *Let $S \subseteq Q$ be a maximal set of pairwise k-dissimilar states of M, and let N be a k-minimal DFA for M. Then there exists a bijection $h: S \to P$ such that*

- $q \sim h(q)$ *for every* $q \in S$, *and*
- $q \equiv h(q)$ *for every* $q \in S$ *such that* $in\text{-}level_M(q) \geq k$.

Proof (of Corollary 4). We have $|S| = |P|$ by Corollary 3. For every $q \in S$, let $w_q \in \delta^{-1}(q)$ be such that $|w_q| = in\text{-}level_M(q)$. We define the mapping $h \colon S \to P$ by $h(q) = \mu(w_q)$ for every $q \in S$. Since $M \sim_k N$, which yields $M \sim N$, we have $q = \delta(w_q) \sim \mu(w_q) = h(q)$. Finally, suppose that $in\text{-}level_M(q) \geq k$. Then $L_M(q) = L_M(\delta(w_q)) = L_N(\mu(w_q)) = L_N(h(q))$ because $M \sim_k N$, which yields $q \equiv h(q)$.   □

B.2   Additional material for Section 3.2

We refer to the distance tree we construct for the DFA M using the notation $\mathcal{D}(M)$. We identify the leaves with the states of the DFA if this raises no confusion. To simplify the argument, we assume that $\bot$ is always in the $\mathcal{D}(M)$. We refer to a tree in a distance forest by a name of a *distance tree*. The vertices that are present in the compressed representation are called *explicit*, while those that were removed are called *implicit*. The standard terms *father* $f(v)$ of a vertex v and *ancestor* always refer to implicit vertices.

Algorithm 2 DISTANCE-TREE

1: **for** $p \in Q$ **do**
2:     $\text{state}(p) \leftarrow p$, $\text{level}(p) \leftarrow 0$, activate p
3: **for** $\ell = 1$ **to** $|n|$ **do**
4:     group active nodes according to $(\delta((\text{state}(v), a))_{a \in \Sigma})$
5:     **for** each group of nodes V' such that $|V'| > 1$ **do**
6:         choose $v \in V'$
7:         create active node v', $\text{level}(v') = \ell$, $\text{state}(v') = \text{state}(v)$
8:         **for** $v'' \in V'$ **do**
9:             join v'' to v', deactivate v''
10:            replace $\text{state}(v'')$ in entries of δ by $\text{state}(v')$

Lemma 9. *For a minimised DFA M if* DISTANCE-TREE *replaced the state q in δ by p at phase ℓ then $d(p,q) = \ell$.*

Proof. The proof proceeds by induction on ℓ. If $\ell = 1$ then q and p have the same successors. Since q and p are not equivalent, their distance is exactly 1, as claimed.

   Suppose that q was replaced by p in phase $\ell > 1$. Since q was not replaced by p in phase $\ell - 1$, by the induction assumption $d(q,p) > d - 1$. Let $(q_1, \ldots, q_{|\Sigma|})$ be the set of successors of q in the DFA and let $(q'_1, \ldots, q'_{|\Sigma|})$ be the vector of its successors in the representation in DISTANCE-TREE. Since q_i was replaced by

q_i' in phase $\ell - 1$ or earlier, $d(q_i, q_i') \leq \ell - 1$. Similarly, $d(p_i, q_i') \leq \ell - 1$. Then, by (1),

$$d(p, q) = 1 + \max_{i=1}^{|\Sigma|} d(q_i, p_i)$$

$$\leq 1 + \max_{i=1}^{|\Sigma|} \left(\max(d(q_i, q_i'), d(q_i', p_i)) \right)$$

$$\leq \ell.$$

Consider now any other state r such that r was replaced by q in earlier phases. Then $d(r, q) \leq \ell - 1$ and thus $d(r, p) = \ell$. $\qquad\square$

The bottleneck of DISTANCE-TREE is the replacing of occurrences of q in δ by some other state p. We show that such replacing can be done in a way so that a single entry in δ is modified at most $\log n$ times.

Lemma 10. DISTANCE-TREE *can be implemented so that it alters every value of δ at most $\log n$ times.*

Proof. The key modification needed in DISTANCE-TREE is the choice of node $v \in V'$. For each state we introduce a counter $c(p)$, initially set to 1, which keeps the track of how many states p represents. When we choose a node $v \in V'$ we take the one with the largest $c(\text{state}(v))$. We update the value accordingly $c(\text{state}(v')) \leftarrow \sum_{v'' \in V'} c(\text{state}(v''))$.

Note, that if we replace a value q by p in δ, then $c(p) \geq c(q)$ before the update of c and so $c(p) \geq 2c(q)$ after the change. Thus if we replace the entry in δ, the corresponding value of c at least doubles. Since $c(p)$ is upper-bounded by n, each entry is replaced at most $\log n$ times. $\qquad\square$

Proof (of Theorem 2). To allow fast replacing of entries q in the δ, for each state q used as the label in one of the dictionaries, we store an up-to-date list of its occurrences in all vectors in all dictionaries.

When state p is merged into q we update the tries: we use the up-to date list of occurrences. For each occurrence of p in an internal node v we have the following situations: if v does not have a child labelled by q then we remove p from the linear dictionary, and insert q into it, pointing at the same child as p used to. If v has both children p and q, we have to merge their corresponding subtrees, rooted at v_1 and v_2. We choose one of them, say v_1, and insert each child of v_1 into subtrie of v_2. Then we set pointer from q to v_2. This might result in yet another situation of the same type, we do so recursively until we get to the leaves.

The total cost of the case, when we did not need to merge linear dictionaries can be bounded similarly as in Lemma 10: note, that after inserting q and deleting p from the linear dictionary, $c(q) \geq 2c(p)$. Thus each such element is modified at most $\log n$ times and so the total cost is $\mathcal{O}(|\Sigma| n \log n)$.

If we do merge the linear dictionaries, we make a different analysis. For each linear dictionary we keep a counter, which calculates how many vertices were

inserted into this linear dictionary. When we merge two linear dictionaries, we remove the one with the smaller value of the counter and insert all its elements into the other dictionary. Then we sum the counters and update the remanding counter.

Since each time a vertex is reinserted, the value of the counter in its linear dictionary at least doubles, and the maximal value of such counter is n, each vertex is inserted into a dictionary at most $\log n$ times.

If we are to merge two leaves, we simply join their respective lists and remove one of the leaves.   □

Now we can use the distance forest to our benefit. Instead of finding pairs of states that are k-similar we proceed in another fashion: roughly speaking, for each state q we want to find the closest (with respect to d) state p satisfying $in\text{-}level(p) \geq in\text{-}level(q)$, then, using this state, we want to judge, whether q is ever going to be merged to other state. To this end, we label the nodes of the $\mathcal{D}(M)$ by states of the DFA M, formally we define state(v) for each node v of $\mathcal{D}(M)$. Since leaves of the $\mathcal{D}(M)$ are identified with leaves, we obviously set state$(q) = q$ for each such leaf. Then we label each inner node with one of the labels of its children, choosing the one with the maximal $in\text{-}level_M$.

For each state q let its *submit node* be the first node on the path from leaf q to the root labelled with a state different than q and let the *submit state* be the label of this node; let $d(q)$ be the depth of the submit node q, if q is the label of the root, then $d(q) = \infty$. Furthermore, define value$(q) = in\text{-}level_M(q) + d(q)$. The next lemma shows that value(q) can be used to approximate $\sim_k$.

Lemma 11. *If* value$(q) \leq k$ *then* q *is k-similar to every state appearing as the label on the path from its submit state to the root.*

If value(p), value$(q) > k$ *then* $p \sim_k q$.

Proof. Suppose that value$(q) \leq k$ and let q' be any node label above q's submit vertex (inclusively). Then

$$\min(in\text{-}level(q), in\text{-}level(q')) + \text{level}(\text{lca}(q, q')) = in\text{-}level(q) + \text{level}(\text{lca}(q, q'))$$
$$\leq in\text{-}level(q) + d(q)$$
$$= \text{value}(q)$$
$$\leq k \ ,$$

and so $q \sim_k q'$.

Let value(q), value$(q') > k$, without loss of generality we may assume that $in\text{-}level(q) \leq in\text{-}level(q')$ and q is not q''s submit node (which could happen if $in\text{-}level(q) = in\text{-}level(q')$. Then $d(q, q') = \text{level}(\text{lca}(q, q')) \geq d(q)$. Thus

$$d(q, q') + \min(in\text{-}level(q), in\text{-}level(q')) \geq d(q) + in\text{-}level(q)$$
$$= \text{value}(q)$$
$$> k \ ,$$

which concludes the proof.   □

For each state q let its k-ancestor node v be the first node on the path from q to the root such that value(state(v)) $> k$, and let k-ancestor state be the label of v.

Corollary 5 (of Lemma 11). *The DFA obtained by merging each state q to its k-ancestor state is k-minimal and k-similar to M.*

The following theorem states easy consequences of Corollary 5.

Proof (of Theorem 3). Calculate the labels in the $\mathcal{D}(M)$ as described earlier, this can be done using one depth-first traversal. Then calculate value(q) for each state q, this as well can be done using one depth-first traversal. Sort the pairs $(q, \text{value}(q))$ according to value(q), since value(q) $\leq 2n$, this can be done in linear time using CountingSort. Note, that the number of states of k-minimal DFA M_k equals $|\{q \mid \text{value}(q) > k\}|$, by Corollary 5, which can be now easily computed in linear time.

By Corollary 5 to obtain the k-minimal DFA it is enough to merge each q with value(q) $\leq k$ to its k-ancestor. A table of *ancestor* assigning to state q its k-ancestor can be computed in linear time. The merging can be performed in $\mathcal{O}(|\delta|)$ time, as we are only interested in the transition of the states q' such that value(q') $> k$. We look through δ and replace each entry $\delta(q', a) = q$ by *ancestor*(q).

There is a little subtlety: when replacing $\delta(q', a) = q$ by $\delta(q', a) = q''$ we should take care that $q \neq \bot$, as otherwise it would be impossible to bound the running time by $|\delta|$. However, note that if $q'' \sim_k \bot$ then the language of q'' is finite and therefore there is a path from q'' to $\bot$, hence *in-level*(q'') $\leq$ *in-level*($\bot$). Note, that when *in-level* for two states are equal, we arbitrarily choose one, so without loss of generality it can be assumed that $\bot$ is never merged to any other state.

We now present a $\mathcal{O}(|\delta| \log n)$ algorithm, which at step $k = 0, 1, \ldots, n$ has in memory the k-minimal DFA. As previously, in step k it will keep only states with value at greater than k: it merges each existing state q such that value(q) $= k$ into its k-ancestor q' which by the construction satisfies value(q') $\geq$ value(q)$+1 > k$. To obtain the proper running time, we need only need to organise the data structures properly. It is represented by a list of transition: for each state q we list the pairs (a, q') such that $\delta(a, q) = q'$, for all valid a. Moreover, each q has a list of incoming transition, i.e., list of pointers to the transitions to it. Moreover, each state q has a counter rank(q), which describes how many states were merged to it.

Assume that we merge p to q and rank(p) $\leq$ rank(q). Then the situation is easy. We redirect each transition to p into q, and perform the update rank(q) $\leftarrow$ rank(q) + rank(p). When rank(p) $>$ rank(q) then we redirect each transition to q into p, and replace the outgoing transition from p by outgoing transitions from q. Since they are given as a list, this is done in $\mathcal{O}(1)$ time. Then we rename p as q and update rank rank(q) $\leftarrow$ rank(q) + rank(p).

Note, that each time an entry in δ is modified, the rank of the target state doubles. Thus each transition is modified at most $\log n$ times and so the running time is $\mathcal{O}(|\delta| \log n)$.

Note, that while the running time $\mathcal{O}(|\delta| \log n)$, outputting the results for each k might take a time up to $\Omega(n|\delta| \log n)$.   □

B.3   Additional material for Section 3.3

Lemma 12. *The total cost of maintaining the transition by $\$$ in the tries is $\mathcal{O}(|\delta| \log n)$ linear dictionary operations.*

Proof. Because $L(M)$ is finite, $m(p) = \max\{|w| : w \in L(p)\}$ is defined for any state p, let us denote $Q_i = \{p \mid m(p) = i\}$ and $Q_{<\infty} = \bigcup_i Q_i$.

Consider a DFA M' built on states $Q_{<\infty}$ and their direct predecessors, i.e., $Q'_{<\infty} = \{q' : \exists a \in \Sigma \; \delta(q,a) \in Q_{<\infty}\} \cup \{\bot\}$. Take the δ' restricted to input from $Q'_{<\infty}$ and values in $Q_{<\infty}$. For each state q with infinite right-language and transitions into states in $Q_{<\infty}$ we insert into vectors of successors $(\$, q')$, or $(\$, \bot)$, if q has only undefined transitions. By Theorem 4 the cost of construction of $\mathcal{T}'$ for M' is $\mathcal{O}(|\delta| \log n)$. Using $\mathcal{T}'$ for M' in phase d we merge states from M' which are at distance $d - 1$ or less: it is enough to sort the nodes of $\mathcal{T}'$ according to their level and in phase d merge leaves in subtree of each node v such that $\text{level}(v) \le d$. To perform the merging efficiently, each state in M' is assigned rank, which denotes the number of states that it represents. When states $q_1, \ldots, q_r$ are to be merged, we choose the one with the maximal rank, say q_i and replace of occurrence of $q_1, \ldots, q_r$ in the trie by q_i. Then we update the rank: $\text{rank}(q_i) \leftarrow \sum_{j=1}^{r} \text{rank}(q_j)$. Thus each such entry in the trie is replaced at most $\log n$ times: whenever it is replaced, the corresponding rank doubles and it is upper bounded by n.   □

Proof (of Theorem 5). In linear time we can identify states such that their right-language is finite. By Theorem 4 their distance tree can be built in $\mathcal{O}(|\delta| \log n)$ time.

Grouping of states in DISTANCE-TREE is a by-product of using trie for the vectors of successors. Each state is deleted from set Q once, each such deletion results in an update of the trie.

The number of linear dictionary operations for the letters in the respective signature can be upper bounded as in Theorem 2, that is, by $\mathcal{O}(|\delta| \log n)$. By Lemma 12 the same bound applies to the construction and usage of the distance tree for states with finite right-language,   □

Next, we comment how the ancestors and nodes are represented for the algorithm, as some of them are implicit: they are represented by an explicit vertex directly below them with an offset, i.e., a pair (v, ℓ).

Recall, that we iteratively construct fragments of the distance forest, built on states $Q_t = \{p \mid m(p) = t\}$, i.e., recognising words of length at most t. For each already constructed fragment we do the preprocessing allowing efficient

computing the lca for any pair of vertices. There are known construction for doing this in constant time [4] however, they use the power of the full RAM model. For our purposes, the simple construction that keeps at every node a list of ancestors 2^0, 2^1, ..., $2^{\log n}$ higher allow performing the search in $\Theta(\log n)$ time, which does not influence the total running time in our case. As shown later, having the preprocessing performed for each such a fragment separately, is enough to execute lca-queries for the whole tree.

Firstly we argue that indeed adding fragments built on states from Q_t is reasonable. Moreover, if for some states $m(p) \neq m(q)$, $d(p,q)$ can be calculated easily.

Lemma 13. *If $m(p) \neq m(q)$ then $d(p,q) = \max(m(p), m(q)) + 1$.*
If $m(p) = m(q)$ then $d(p,q) \leq \max(m(p), m(q)) + 1$.

Recall, that the spine is the path joining the state $\bot$, which recognises an empty language, with the root of the $\mathcal{D}(M)$. Note, that the spine has no compressed fragments, as the distance between $\bot$ and a a state p is equal to $m(p)+1$, by Lemma 13, moreover, by (1) it is easy to see that $\{m(p) : p \in Q_{<\infty}\}$ is equal to $\{0, 1, 2, \ldots, \max\{p \in Q_{<\infty} : m(p)\}$. Therefore the spine is created beforehand as an uncompressed line of length $\max\{p \in Q_{<\infty} : m(p)+1\}$, which is at most $|Q_{<\infty}|$.

Lemma 14. *The distance between $\mathbf{v} = ((a_i, p_i))_{i \in I_1}$ and $\mathbf{v}' = ((a_i, p_i'))_{i \in I_2}$ can be computed using $\mathcal{O}(|I_1| + |I_2|)$ lca-queries.*

Proof. We calculate their distance straight from the definition. This can be done using $|I_1| + |I_2|$ lca queries, by going through consecutive elements of these vectors: as they are sorted, seeing (a_i, p_i) and $(a_{i'}, p_{i'}')$ we can decide whether $a_i = a_{i'}$, in which case $i = i' \in I_1 \cap I_2$, or $a_i > a_{i'}$, and thus $i \in I_1 \setminus I_2$; or $a_i < a_{i'}$, when $i' \in I_2 \setminus I_1$. In the first case, we calculate $d(p_i, p_{i'}')$, this can be done by comparing $m(p_i)$ and $m(p_{i'}')$ and by calculating $\mathrm{lca}(p_i, p_{i'}')$, if $m(p_i) = m(p_{i'}')$. This distance is compared with the current maximum. If $i \in I_1 \setminus I_2$, we compare the current maximum with $m(p_i) + 1$; the situation for $p_{i'}'$ is symmetric.

There are at most $|I_1| \cup |I_2|$ lca queries used and at most as much other operations, which are all performed in constant time. So the cost of the whole procedure can be charged to the $|I_1| \cup |I_2|$ lca queries. $\square$

To show that the total time of the construction of the the fragment for Q_t is $\mathcal{O}(|\delta_t| \log t)$, where δ_t is the transition function restricted to the input from Q_t, we estimate separately the cost of finding the ancestors of the vectors and the cost of grouping of the vectors, according to their 2^k successors. However, to properly estimate the time needed for that, we cannot use the whole already constructed part of the distance tree, as it may be vary large comparing to Q_t. Thus, as a preprocessing step, we extract out of the distance tree the distance tree induced by the states appearing in the vectors. Being more precise, we calculate the subtrees $\mathcal{T}_a$ for $a \in \Sigma$: it is a sub-distance tree for states $\delta(Q_t, a)$.

Lemma 15. *Constructing $\mathcal{T}_a$ for $a \in \Sigma$ can be done in time $\mathcal{O}(|\delta_t| \log |Q_t|)$.*

Proof. Firstly we calculate the states in $\delta(Q_t, a)$ for each a, in total time $\mathcal{O}(|\delta_t|)$. Let us a fix a letter a. We sort the leaves in $\mathcal{T}_a$ in time $(|\mathcal{T}_a|\log|Q_t|)$: we assume that they are in some arbitrary, but fixed, order in $\mathcal{D}(M)$. Let them be q_1, q_2, $\ldots$, q_s. We build $\mathcal{T}_a$ by successively adding leaves. Suppose that a tree for q_1, $\ldots$, q_i has already been built. The right-most path (names of nodes and their levels) is kept as a list. To add q_{i+1}, we calculate the $\mathrm{lca}(q_i, q_{i+1})$ and remove from the list all nodes with smaller level. If the last element of the list, call it v, has level greater than $\mathrm{lca}(q_i, q_{i+1})$ we create a new inner node v' in $\mathcal{T}_a$, insert it into the right-most path as the last element, make the right-most son of v the only child of v' and v' the new right-most son of v. Next we make q_{i+1} a right-most child of the last node in the list (which might be v or v').

Since each node is inserted and removed from the right-most path at most once, the total running time is $\mathcal{O}(|\mathcal{T}_a|)$. Summing up over all $a \in \Sigma$, we obtain that the total construction time is $\mathcal{O}(|\delta_t|)$. $\square$

Lemma 16. *The total time of finding ancestors of vectors is $\mathcal{O}(|\delta_t|\log n)$.*

Proof. Finding the ancestors is implemented naively: for each state q in the vector we traverse $\mathcal{D}(M)$ up 2^k steps up from q.

Consider a pair (a, p) in one of the vectors $\mathbf{v}$ and one of the edges e in $\mathcal{T}_a$ on the way from p to the spine. We show that e is traversed at most $\lceil \log t \rceil$ times when constructing the distance tree for Q_t.

Note, that the recursive calls are made for $k = \lceil \log t \rceil, \lceil \log t \rceil - 1, \ldots, 1$. So it is enough to show that e is traversed once for a fixed value of k.

Consider a given instance of a recursive call. Then when the sub-recursive calls are made, e goes to exactly one of these sub-calls, except when one of the 2^k-ancestor of leaves lays on e. However, in such a case there is no need to traverse e by any sub-call from the lower group: the lower end of e is an ancestor of all vertices in this tree. So we can modify the algorithm a bit: as soon as some edge is to be traversed, we check if the root lies on this edge. If so, this edge is not traversed, as searched node is implicit and therefore represented by the lower end and an offset. And so e is traversed at most once for each k, which concludes the proof. $\square$

Lemma 17. *Given k, the total size of grouping vectors in the recursive calls for k is linear in their size plus an additional cost, which is $\mathcal{O}(|\delta_t|\log|Q_t|)$ summed over all k*

Proof. We want to sort lexicographically vectors $\mathbf{v}_1, \ldots, \mathbf{v}_i$ of integers in the range $1, 2, \ldots, n + |\Sigma|$. This can be done in a standard way (say, usinf RADIX-SORT) in time $\mathcal{O}(n + |\Sigma| + \sum_{j=1}^{i} |\mathbf{v}_j|)$. This is too much, as n and $|\Sigma|$ might be large compared to $\sum_{j=1}^{i} |\mathbf{v}_j|$. Thus we can do the following. In each $\mathcal{T}_a$ we can replace each node by a number from range $1, \ldots, 2|Q_t| - 1$. Then instead of sorting according to names in $\mathcal{D}(M)$, we use the local names from $\mathcal{T}_a$. In this way the running time is $\mathcal{O}(|Q_t| + |\Sigma| + \sum_{j=1}^{i} |\mathbf{v}_j|)$

Still, for small instances, $|Q_t| + |\Sigma|$ can be substantially larger than $\sum_{j=1}^{i} |\mathbf{v}_j|$. To avoid this problem, we process all the recursive call for a fixed k in parallel.

Then the sorting is done for all vectors in the recursive calls. Note, that if vectors come from different recursive calls, they cannot have the same non-trivial ancestors (and trivial, i.e., empty, ones can be identified and removed from the sorting beforehand). Thus the additional cost $\mathcal{O}(|Q_t| + |\Sigma|)$ is included once for each $k = 1, \ldots, \log |Q_t|$, and so in total gives $\mathcal{O}((|Q_t| + |\Sigma|) \log |Q_t|)$ time, which is $\mathcal{O}(|\delta_t| \log |Q_t|)$. $\qquad\square$

Two previous lemmata allow calculating the whole recursion time

Lemma 18. *The cost of the procedure for vectors $\mathbf{v_1}, \ldots, \mathbf{v_\ell}$, excluding the additional cost of $\mathcal{O}(|\delta_t| \log |Q_t|)$ from Lemma 17, is $\mathcal{O}((\sum_{i=1}^{\ell} |\mathbf{v_i}| - |\mathbf{v}|) \log |Q_t|)$, where $\mathbf{v}$ is a lowest common ancestor of $\mathbf{v_1}, \ldots, \mathbf{v_\ell}$.*

Proof. First note, that if $\mathbf{v}$ is the lowest ancestor of $\mathbf{v_1}, \ldots, \mathbf{v_\ell}$ and $\mathbf{v}'$ is some ancestor of these vectors, then $|\mathbf{v}| \geq |\mathbf{v}'|$, and so the estimation using $|\mathbf{v}'|$ is weaker than the one using $|\mathbf{v}|$.

The claim is shown by an induction. Fix a constant c, for which it is shown that the total cost is at most $c(\sum_{i=1}^{\ell} |\mathbf{v_i}| - |\mathbf{v}|) \log s$.

The basis of the induction are calls for at most two vectors. No recursive call is made for one vector, and so we do not consider it. As observed in Lemma 14, when there are only two vectors $\mathbf{v}, \mathbf{v}'$, the calculation is done using $\mathcal{O}(\mathbf{v} + \mathbf{v}')$ lca queries. So c can be chosen in advance so that this is at most $c(\mathbf{v} + \mathbf{v}') \log t$.

When there are more than two vectors, by Lemma 17 we can group them according to their ancestors in time at most $\sum_{i=1}^{\ell} |\mathbf{v_i}|$. Since $|\mathbf{v}| \leq \min_{i=1}^{\ell} |\mathbf{v_i}|$, constant c can be chosen in advance in the way that this is at most $c(\sum_{i=1}^{\ell} |\mathbf{v_i}| - |\mathbf{v}|)$ time.

Then there are sub-calls made. Let $\mathbf{u_1}, \ldots, \mathbf{u_{\ell'}}$ be the ancestors of vectors. Then the 'upper' recursive call, by the induction assumption, takes at most $c(\log s - 1)(\sum_{i=1}^{\ell'} \mathbf{u_i} - \mathbf{v})$ (note, that $\mathbf{v}$ is a common ancestor of $\mathbf{u_1}, \ldots, \mathbf{u_{\ell'}}$). The 'lower' subcalls take, in total, time $c(\sum_{i=1}^{\ell} |\mathbf{v_i}| - \sum_{i=1}^{\ell'} |\mathbf{u_i}|)(\log s - 1)$, since each $\mathbf{u_i}$ is a common ancestor of vectors for one of these subcalls. Hence all the calls take at most $c(\sum_{i=1}^{\ell} |\mathbf{v_i}| - |\mathbf{v}|)(\log s - 1)$ time.

Summing the cost of the recursive calls and the grouping:

$$c(\sum_{i=1}^{\ell} |\mathbf{v_i}| - |\mathbf{v}|) + c(\sum_{i=1}^{\ell} |\mathbf{v_i}| - |\mathbf{v}|)(\log s - 1) \leq c(\sum_{i=1}^{\ell} |\mathbf{v_i}| - |\mathbf{v}|) \log s,$$

as claimed. $\qquad\square$

Proof (of Theorem 4). By Lemma 17 together with Lemma 18.

C Additional material for Section 4

The DFA $c(M)$ has 14 states, and it can easily be verified that it is hyper-equivalent to M and has no different, but equivalent kernel states. It is depicted in Figure 6.
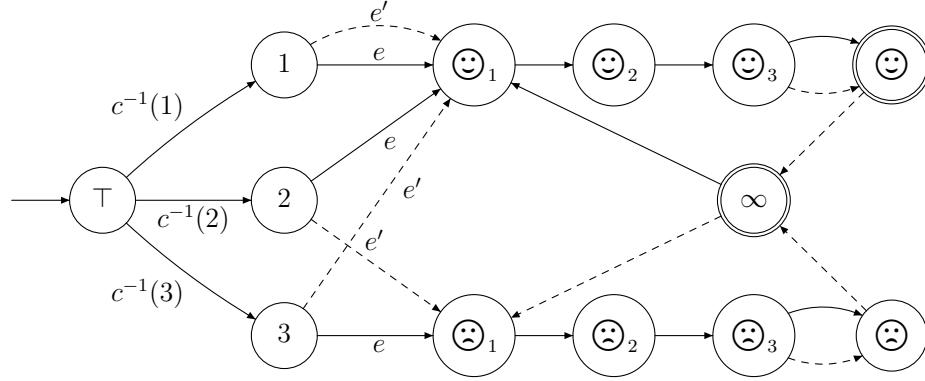
Fig. 6. DFA $c(M)$ constructed in Section 4, where a-transitions are represented by unbroken lines (unless noted otherwise), b-transitions are represented by dashed lines, and $e = \{v_1, v_2\}$ and $e' = \{v_3, v_4\}$ with $v_1 < v_2 < v_3 < v_4$ and $c(v_1) = 1$, $c(v_2) = 3$, $c(v_3) = 1$, and $c(v_4) = 2$. The state $\bot$ is not depicted.

Proof (of Lemma 1). Let $N = \langle P, \Sigma, \mu, p_0, F' \rangle$ be a DFA such that $M \sim N$ and $|P| \leq 14$. Without loss of generality we may assume that there are no different, but equivalent kernel states in N. Note that the DFA M is minimal provided that G has no vertex without any incident edge. By [3, Theorem 3.8][3] there exists a mapping $h \colon Q \to P$ such that $q \sim h(q)$ for every $q \in Q$, which additionally is an isomorphism $h \colon \mathrm{Ker}(M) \to \mathrm{Ker}(N)$ between the kernel states. Since $\mathrm{Ker}(M) = \{\bot, \infty, \odot, \otimes\} \cup \{\bigcirc_j \mid \bigcirc \in \{\odot, \otimes\}, j \in [3]\}$, we have $\{\infty, \odot\} \subseteq F'$ and 10 distinct states in N that behave like their counterparts in M [i.e., $L_M(q) = L_N(h(q))$ for every $q \in \mathrm{Ker}(M)$]. Since $\top \not\sim p$ and $\delta(v) \not\sim p$ for every $p \in \mathrm{Ker}(N)$ and $v \in V$, we conclude that $h(\top) = p_0 \notin \mathrm{Ker}(N)$ and $h(\delta(v)) \notin \mathrm{Ker}(N)$, which means that we identified the 11th and 12th state in N because they are not hyper-equivalent to each other (i.e., $\top \not\sim \delta(v)$ for every $v \in V$). Consequently, there are at most 2 other unidentified states.

Claim 1. The DFA N commits at least $|E| \cdot (|V| - 2)$ errors with prefix ve such that $v \in V$ is a vertex and $e \in E$ is a non-incident edge. Exactly $|E| \cdot (|V| - 2)$ such errors are committed if $\mu(ve) \in \{h(\odot_1), h(\otimes_1)\}$ for every $v \in V$ and $e \in E$ with $v \notin e$.

Proof. Let $v \in V$ be a vertex and $e \in E$ be a non-incident edge, which means that $v \notin e$. Clearly, $\delta(vea^{j-1}) = \bullet_j$ for all $j \in [3]$ and $\bullet_i \not\sim \bullet_j$ for all different $i, j \in [3]$. Consequently, also the 3 states of $S = \{\mu(vea^{j-1}) \mid j \in [3]\}$ are pairwise hyper-inequivalent (and hence different). Since there are at most 2 unidentified states, at least one state of S is already identified. Moreover,

---

[3] Actually, the cited theorem assumes N to be minimal, but the original proof also works in our relaxed setting (i.e., when there are no different, but equivalent states in the kernel).

$\delta(vea^{j-1}) \sim \mu(vea^{j-1})$ for all $j \in [3]$ yields that $\mu(vea^{j-1}) \in \{h(\odot_j), h(\odot_j)\}$ for at least one $j \in [3]$, since $\odot_j$ and $\odot_j$ are the only states hyper-equivalent to $\odot_j$ and $\mu(vea^{j-1}) \sim \odot_j$. Since $\sim$ is a congruence and h is an isomorphism on $\mathrm{Ker}(M)$, we obtain that $\mu(vea^2) \in \{h(\odot_3), h(\odot_3)\}$ in any case. However, $\delta(vea^2) = \bullet_3$, so we obtain the error word $vea^2 x$ for some $x \in \{a, b\}$ because

- $L_N(h(\odot_3)) = L_M(\odot_3)$ and $L_N(h(\odot_3)) = L_M(\odot_3)$,
- $L_M(\bullet_3) \triangle L_N(h(\odot_3)) = L_M(\bullet_3) \triangle L_M(\odot_3) = \{b\}$, and
- $L_M(\bullet_3) \triangle L_N(h(\odot_3)) = L_M(\bullet_3) \triangle L_M(\odot_3) = \{a\}$.

Moreover, if $\mu(ve) \in \{h(\odot_1), h(\odot_1)\}$, then there is exactly one error word with prefix ve. Overall, this yields at least one error for every $v \in V$ and $e \in E$ with $v \notin e$. The total number of such errors is at least $|E| \cdot (|V| - 2)$, and it is exactly $|E| \cdot (|V| - 2)$ if $\mu(ve) \in \{h(\odot_1), h(\odot_1)\}$ for every $v \in V$ and $e \in E$ with $v \notin e$. $\qquad\square$

Let $S = \{\mu(v) \mid v \in V\}$. For every $v \in V$, the state $\delta(v)$ is not hyper-equivalent to any state of $\mathrm{Ker}(N)$ and $\delta(v) \not\sim p_0$, which yields that $|S| \leq 3$; without loosing generality we can assume that $|S| = 3$, as this is the hardest case. Let $S = \{p_1, p_2, p_3\}$, and let $c: V \to [3]$ be such that for every $v \in V$ we have $c(v) = i$ if and only if $\mu(v) = p_i$. Thus, we deduced a 3-colouring from the transitions of N. Next, we investigate colouring violations and its connection with the number of errors with respect to M. For every edge $e = \{v_1, v_2\} \in E$ such that $c(v_1) = c(v_2)$, we have $L_N(\mu(v_1 e)) = L_N(\mu(v_2 e))$, but

$$L_M(\delta(v_1 e)) \triangle L_M(\delta(v_2 e)) = L_M(\odot_1) \triangle L_M(\odot_1) = \{aaa, aab\} \ .$$

Consequently, we obtain at least 2 additional errors for every such edge e because the corresponding error words start with ve where $v \in e$. Those two errors together with the errors described in Claim 1 yield that N commits strictly more than $|E| \cdot (|V| - 2)$ errors if G is not 3-colourable.

Finally, we show that the DFA $c(M) = \langle P, \Sigma, \mu, \top, F \rangle$, which has exactly 14 states, commits exactly $|E| \cdot (|V| - 2)$ errors, if G is 3-colourable by the proper 3-colouring $c: V \to [3]$. By Claim 1, $c(M)$ commits exactly $|E| \cdot (|V| - 2)$ errors with prefix ve for $v \in V$ and $e \in E$ with $v \notin e$ because $\mu(ve) \in \{\odot_1, \odot_1\}$ for all such v and e. Clearly, all remaining error words must start with ve for some $v \in V$ and $e \in E$ such that $v \in e$. However, if $v \in e$, then $\delta(ve) = \mu(c(v)e)$ by the definition of $c(M)$, which yields that no error word with prefix ve exists. $\quad\square$

D Additional material for Section 5

Recall that $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ is our DFA and that we can always rename states to avoid conflicts. In particular, we assume that $Q \cap \mathbb{N} = \emptyset$.

Lemma 19 (full version of Lemma 2). *For every congruence $\simeq \subseteq Q \times Q$ on M, there exists a DFA $N = \langle P, \Delta, \mu, q_0, F' \rangle$ such that*

(i) $L_M(q) = L_N(q) \cap \Sigma^*$ and $\delta^{-1}(q) = \mu^{-1}(q)$ for every $q \in Q$,

(ii) $L_M(q_1) \triangle L_M(q_2) = L_N(q_1) \triangle L_N(q_2)$ *for all* $q_1 \simeq q_2$,
(iii) $p_1 \not\sim p_2$ *for every* $p_1 \in P \setminus Q$ *and* $p_2 \in P$ *with* $p_1 \neq p_2$, *and*
(iv) $q_1 \not\sim q_2$ *in* N *for all* $q_1 \not\simeq q_2$.

Proof. Let n be the index of $\simeq$ and $\theta\colon (Q/\simeq) \to [n]$ be an arbitrary bijection. Moreover, let

- $P = Q \cup [n]$ and $F' = F \cup \{n\}$,
- $\Delta = \Sigma \cup \{t, x\}$ where $t, x \notin \Sigma$ are different, new letters,
- $\mu(q, \sigma) = \delta(q, \sigma)$ for every $q \in Q$ and $\sigma \in \Sigma$,
- $\mu(q, t) = \theta([q]_\simeq)$ for every $q \in Q$, and
- $\mu(i, x) = i + 1$ for every $i \in [n-1]$ and $\mu(n, x) = 1$.

Next, we discuss hyper-equivalence and k-similarity in the DFA N. Clearly, $i \not\sim j$ for all different $i, j \in [n]$. Since $\sim$ is a congruence, we obtain that $q_1 \not\sim q_2$ for all $q_1, q_2 \in Q$ such that $q_1 \not\simeq q_2$. Moreover, $i \not\sim q$ for every $i \in [n]$ and $q \in Q$. The (i) is obvious. Finally, we can prove by an induction on the length of $w \in \Delta^*$ that $w \in L_M(q_1) \triangle L_M(q_2)$ if and only if $w \in L_N(q_1) \triangle L_N(q_2)$ for all $q_1 \simeq q_2$. $\qquad\square$

In the future, we will use the construction in the proof of Lemma 19 as a gadget with n states and simply assume that we can enforce that $q_1 \not\sim q_2$ for every $q_1 \not\simeq q_2$ and every congruence $\simeq$ with index n. This can be done since none of the newly added states is k-similar to an existing state and all newly added states are pairwise dissimilar. Moreover, since we will only merge k-similar states and $\mu(q_1, t) = \mu(q_2, t)$ for all $q_1 \sim_k q_2$, the gadget does not introduce new errors.

Lemma 20 (full version of Lemma 3). *For every subset* $S \subseteq Q \setminus \{q_0\}$ *of states and mapping* min-level$\colon S \to \mathbb{N}$, *there is a DFA* $N = \langle Q \cup I, \Sigma \cup \Delta, \mu, q_0, F \rangle$ *such that*

- $L_M(q) = L_N(q)$ *for every* $q \in Q \setminus \{q_0\}$,
- $|\mu^{-1}(i)| = 1$ *for every* $i \in I$,
- $|\delta^{-1}(s) \triangle \mu^{-1}(s)| = 1$ *for* $s \in S$ *with* min-level$(s) \geq 2$ *and* $\delta^{-1}(q) = \mu^{-1}(q)$ *for all remaining* $q \in Q$, *and*
- *in-level*$_N(s) \geq$ min-level(s) *for every* $s \in S$.

Proof. Let $n = \max \{\text{min-level}(s) \mid s \in S\}$ be the maximal requested level. We construct the DFA N such that

- $I = [n]$ (supposing that $Q \cap \mathbb{N} = \emptyset$),
- $\Delta = \{d\} \cup \{d_s \mid s \in S\}$ are new, different letters,
- $\mu(q, \sigma) = \delta(q, \sigma)$ for every $q \in Q$ and $\sigma \in \Sigma$, and
- $\mu(q_0, d) = 1$ and $\mu(i, d) = i + 1$ for every $i \in [n-1]$,
- $\mu(\text{min-level}(s) - 1, d_s) = s$ for every $s \in S$ such that min-level$(s) \geq 2$.

Clearly, $L_M(q) = L_N(q)$ for every $q \in Q \setminus \{q_0\}$ and *in-level*$_N(s) \geq$ min-level(s) for every $s \in S$. Finally, $\mu^{-1}(i) = \{d^i\}$ for every $i \in [n]$, which can be used to prove the remaining statements. $\qquad\square$

We can use the first gadget to make sure that all newly introduced states in the previous construction are k-dissimilar. Mind that a renaming can be used to ensure that $Q \cap \mathbb{N} = \emptyset$.

We use the gadgets of Lemmata 19 and 20 as follows: We select the states $S = \{1_0, 2_0, 3_0, 1_1, 2_1, ☺_0, ☺\}$ and let min-level: $S \to \mathbb{N}$ be such that

$$\text{min-level}(1_0) = \text{min-level}(2_0) = \text{min-level}(3_0) = 3s - 1$$

$$\text{min-level}(☺_0) = \text{min-level}(☺) = \text{min-level}(1_1) = \text{min-level}(2_1) = k + 1 \ .$$

Moreover, let $\simeq \ \subseteq Q \times Q$ be the equivalence induced by the partition (single-element classes are omitted)

$$\{\bot, ●, ☺\} \cup \{☺_i \mid 0 \le i \le s\} \cup \{i_j \mid i \in [2], j \in [\ell]\} \quad \{v \mid v \in V\} \cup \{1_0, 2_0, 3_0\} \ .$$

It can easily be checked that $\simeq$ is a congruence. Let $M = \langle Q, \Sigma, \delta, 0, F \rangle$ be the DFA obtained after adding the gadgets of Lemmata 19 and 20 using the above parameters. The obtained DFA M is illustrated in Fig. 5. We state some simple properties that follow immediately from the gadgets.

Lemma 21. *We observe the following simple properties:*

- *Hyper-inequivalence is as indicated in Fig. 5; i.e., $q_1 \not\sim q_2$ for every $q_1 \not\simeq q_2$.*
- *The levels are as indicated in Fig. 5. More specifically, for $j \in [\ell]$, $i \in [s-1]$, and $v \in V$.*

$$in\text{-}level_M(1_0) = in\text{-}level_M(2_0) = 3s - 1 \quad in\text{-}level_M(\bot) = \infty$$
$$in\text{-}level_M(3_0) = 3s - 1$$
$$in\text{-}level_M(1_j) = in\text{-}level_M(2_j) = k + j \quad in\text{-}level_M(v) = s + 1$$

$$in\text{-}level_M(☺_i) = k + i + 1 \qquad\qquad in\text{-}level_M(☺_s) = k + \ell + 1$$
$$in\text{-}level_M(●) = 3s \qquad\qquad\qquad\quad in\text{-}level_M(i) = i$$
$$in\text{-}level_M(☺) = k + 1 \qquad\qquad\quad in\text{-}level_M(s) = s$$

Proof. Both properties follow immediately from Lemmata 19 and 20 and trivial inductions. □

Lemma 22.

(i) *The difference between the right-languages of the states $\{1_0, 2_0, 3_0\}$ is small. More precisely,*

$$L_M(1_0) \triangle L_M(2_0) = \{ab^{\ell-1}\}$$
$$L_M(1_0) \triangle L_M(3_0) = \{ab^{\ell-1}, ab^\ell\}$$
$$L_M(2_0) \triangle L_M(3_0) = \{ab^\ell\} \ .$$

(ii) *Additionally, $|\delta^{-1}(v)| = 2^s$ for every $v \in V$, and*

$$|L_M(☺_i) \triangle L_M(☺)| = 2^{s-i} \qquad d(☺_i, ☺) = s - i + 1$$

$$|L_M(\copyright_i) \,\triangle\, L_M(\newmoon)| \geq 2^{s-1} \qquad d(\copyright_i, \newmoon) = s+1$$

$$|L_M(\newmoon) \,\triangle\, L_M(\odot)| = 2^{s-1} \qquad d(\newmoon, \odot) = s+1$$

$$|L_M(1_j) \,\triangle\, L_M(\newmoon)| \geq |L_M(\newmoon)| = 2^{s-1} \qquad d(1_j, \newmoon) = \max(s, \ell - j + 1) + 1$$

$$|L_M(2_j) \,\triangle\, L_M(\newmoon)| \geq |L_M(\newmoon)| = 2^{s-1} \qquad d(2_j, \newmoon) = \max(s, \ell - j + 1) + 1$$

for every $0 \leq i \leq s$ and $j \in [\ell]$.
(iii) Finally, $\sim_k$ is the reflexive and symmetric closure of

$$\equiv \cup \{(i_0, v) \mid i \in [3], v \in V\} \cup \{(v_1, v_2) \mid v_1, v_2 \in V\} \cup$$
$$\cup \{(\newmoon, \copyright_i) \mid 0 \leq i \leq s\} \cup \{(\newmoon, \odot), (\newmoon, \perp)\} \cup \{(\newmoon, i_j) \mid i \in [2], s + 1 < j \leq \ell\}.$$

Proof. Properties (i) and (ii) can be observed easily. We turn to (iii): let $i, i' \in [3]$ such that $i \neq i'$. Since $\textit{in-level}_M(i_0) = \textit{in-level}_M(i'_0) = 3s - 1$ by Lemma 21 and $d(i_1, i'_1) \geq \ell - 1$ by (i), we obtain $d(i_0, i'_0) + 3s \geq (k - 2s) - 1 + 3s = k + s - 1 \geq k$, which proves that $i_0 \nsim_k i'_0$. Similarly, we can compute $d(i_0, v) + s + 1 \leq (k - 2s + 1) + s + 2 = k - s + 3 \leq k$ for every $v \in V$, which proves $i_0 \sim_k v$. Now, let $v_1, v_2 \in V$. Then

$$\textit{in-level}_M(v_1) = \textit{in-level}_M(v_2) = s + 1 \qquad \text{and} \qquad d(v_1, v_2) \leq s + 2 \ ,$$

which yields $2s + 3 < k$ and proves that $v_1 \sim_k v_2$. Since $\textit{in-level}(\newmoon) = 3s$ and $d(\copyright_0, \newmoon) = s + 1$, we obtain $3s + s + 1 = 4s + 1 \leq k$, which proves that $\copyright_0 \sim_k \newmoon$. In essentially the same way, we can prove all similarities to $\newmoon$, which yields that we proved all similarities. Clearly, two states $q_1, q_2 \in Q$ such that $\min(\textit{in-level}_M(q_1), \textit{in-level}_M(q_2)) \geq k$ are k-similar if and only if $q_1 \equiv q_2$, which proves the nontrivial dissimilarities. $\qquad\square$

Using a proper 3-colouring $c \colon V \to [3]$ we define the DFA $c(m)$.

Definition 7. *Let $c \colon V \to [3]$ be a 3-colouring and $c(M) = \langle P, \Sigma, \mu, 0, F \rangle$ be the DFA such that*

- $P = \{\perp\} \cup \{i_j \mid i \in [2], j \in [\ell]\} \cup [0, s] \cup \{\copyright_i \mid 0 \leq i \leq s\}$
- *for every $v \in V$, $e = \{v_1, v_2\} \in E$ with $v \notin e$ and $v_1 < v_2$, $i \in [s]$, $j \in [\ell]$, and $j' \in [3]$*

$$\mu(i - 1, a) = i \qquad\qquad \mu(\copyright_{i-1}, a) = \copyright_i \quad \mu(1_{j-1}, a) = 1_j$$
$$\mu(i - 1, b) = i \qquad\qquad \mu(\copyright_{i-1}, b) = \copyright_i \quad \mu(2_{j-1}, a) = 2_j$$
$$\mu(s, v) = c(v)_0 \qquad\qquad\quad \mu(1_\ell, a) = \copyright_s \qquad \mu(2_\ell, a) = \copyright_s$$
$$\mu(j'_0, e) = \begin{cases} \copyright_0 & , \textit{ if } c(v_2) \neq j' \\ \perp & , \textit{ otherwise.} \end{cases}$$

- *For all remaining cases, we set $\mu(q, \sigma) = \perp$.*

The DFA $c(M)$ is illustrated in Figure 7. We first show that $c(M)$ is k-similar to M and then that it is k-minimal.

Lemma 23. *The constructed DFA $c(M) = \langle P, \Sigma, \mu, 0, F \rangle$ is k-similar to M.*

Proof. Let Δ be the alphabet of letters without the letters used by the gadgets, which are collected in Γ. We first observe the equalities

$$
\begin{aligned}
L_N(\copyright_s) &= L_M(\copyright_s) & L_N(\copyright_s) \cap \Delta^* &= \{\varepsilon\} \\
L_N(\bot) &= L_M(\bot) = L_M(\copyright) & L_N(\bot) \cap \Delta^* &= \emptyset \\
L_N(1_j) &= L_M(1_j) & L_N(1_j) \cap \Delta^* &= \{b^{\ell-j}, b^{\ell-j+1}\} \\
L_N(2_j) &= L_M(2_j) & L_N(2_j) \cap \Delta^* &= \{b^{\ell-j+1}\} \\
L_N(\copyright_i) &= L_M(\copyright_i) & L_N(\copyright_i) \cap \Delta^* &= \{a, b\}^{s-i} \\
\mu^{-1}(i) &= \delta^{-1}(i) & \mu^{-1}(i) &= \{a, b\}^i \;.
\end{aligned}
$$

In addition, since $\{i_j \mid i \in [2], j \in [\ell]\}$ forms a single class of $\simeq$ in both $c(M)$ and M it holds that

$$
L_{c(M)}(i_j) \cap \Delta^* \Gamma \Sigma^* = L_M(i'_{j'}) \cap \Delta^* \Gamma \Sigma^*
$$

for all $i, i' \in [2]$ and $j, j' \in [\ell]$. From those statements and simple applications of the statements of Lemmas 21 and 22 we can easily conclude that M and $c(M)$ are k-similar. $\qquad\square$
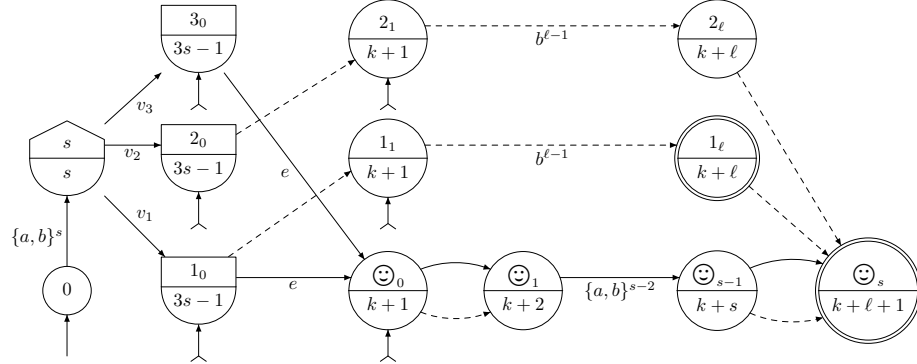

Fig. 7. Illustration of the DFA $c(M)$ of Section 5

Note, that all states in $c(M)$ are pairwise k-dissimilar by Lemma 22. Consequently, they form a maximal set of pairwise k-dissimilar states in M, which yields that $|P|$ coincides with the number of states of all k-minimal DFA for M by Lemma 3. Thus DFA M is k-minimal.

We finally show that our construction is correct.

Proof (of Lemma 4). Let $N = (P, \Sigma, p_0, \mu, F')$ be a DFA that is k-minimal for M. We select a maximal set Q' of pairwise k-dissimilar states by

$$
Q' = \{\bot\} \cup \{i_j \mid i \in [2], j \in [\ell]\} \cup [0, s] \cup \{\copyright_i \mid 0 \le i \le s\} \;.
$$

Let $h \colon Q' \to P$ be the bijection of Corollary 4 using the maximal set Q'. Let

$$S = \{\bot\} \cup \{\circledcirc_i \mid 0 \le i \le s\} \cup \{i_j \mid i \in [2], j \in [\ell]\} \ .$$

Consequently, $q \equiv h(q)$ for every $q \in S$. From $M \sim_k N$, which yields $M \sim N$, we can conclude that $\delta(w) \sim \mu(w)$ for every $w \in \Sigma^*$. Let $w \in \{a,b\}^i$ for some $0 \le i \le s$. Then $\mu(w) = h(i)$ because $\mu(w) \sim \delta(w) = i$ and $h(i)$ is the only state $p \in P$ such that $p \sim i$. Now, let $w = uv$ with $u \in \{a,b\}^s$ and $v \in V$. Since $\delta(uv) = v$ and $v \sim i_0$ only for $i \in [3]$, we obtain $\mu(uv) \in \{h(1_0), h(2_0), h(3_0)\}$. From this behaviour we deduce a colouring $c \colon V \to [3]$ by $c(v) = i$ if and only if $\mu(uv) = h(i_0)$ for every $v \in V$. Note that this definition does not depend on the choice of u because $\mu(u) = \mu(u') = h(s)$ for all $u, u' \in \{a,b\}^s$.

Claim 2. The DFA N commits at least

$$m = 2^{2s-1} \cdot |E| \cdot (|V| - 2) + 3 \cdot 2^{s-1} \cdot |E|$$

errors with prefix

- uve where $u \in \{a,b\}^*$, $v \in V$ is a vertex, and $e \in E$ such that $v \notin e$, or
- dwe where d is the main symbol of the level gadget, $w \in \Sigma^*$, and $e \in E$.

Exactly m such errors are committed if the following two conditions, called (†), are fulfilled:

(i) $\mu(uve) \in \{h(\circledcirc_1), h(\bot)\}$ for every $u \in \{a,b\}^*$, $v \in V$ and $e \in E$ with $v \notin e$.
(ii) $\mu(dwe) \in \{h(\circledcirc_1), h(\bot)\}$ for every $w \in \Sigma^*$ and $e \in E$.

Proof. We distinguish two cases: whether the error word goes through i_0 for some $i \in [3]$ or it goes through v for some $v \in V$:

- The only way to arrive at i_0 in M is via the level gadget. Then by Lemma 20 there exists exactly one $w \in \delta^{-1}(i_0)$ such that $|w| = 3s - 1$ and w does not end with $v \in V$. In addition, $\delta(i_0, e) = \bullet$ for every $e \in E$. However, in N we have $\mu(w) \in \{h(1_0), h(2_0), h(3_0)\}$ and $\mu(h(j_0), e) \in h(S)$ for every $j \in [3]$ because these are the only hyper-equivalent states. Since $q \equiv h(q)$ for every $q \in S$, we have

$$|L_N(h(q)) \triangle L_M(\bullet)| \ge 2^{s-1}$$
$$|L_N(h(\bot)) \triangle L_M(\bullet)| = |L_N(h(\circledcirc_0)) \triangle L_M(\bullet)| = 2^{s-1}$$

for every $q \in S$ by Lemma 22(ii). This yields at least 2^{s-1} error words that start with we, and exactly 2^{s-1} such error words under condition (†). Consequently, there are at least $3 \cdot 2^{s-1} \cdot |E|$ such error words in total, and exactly that many under condition (†), because the level gadget is reproduced exactly.

– Next, we consider a word $w = uve$ with $u \in \{a,b\}^s$, $v \in V$, and $e \in E$ such that $v \notin e$. Clearly, $\delta(w) = \newmoon$, but $\mu(w) \in h(S)$ because $h(S)$ are the only states of P that are hyper-equivalent to $\newmoon$. Since $q \equiv h(q)$ for every $q \in S$ and $\mu(u) = h(s)$ independently of u, there are 2^s such words u and $|L_M(\newmoon) \triangle L_N(h(q))| \geq 2^{s-1}$ for every $q \in S$ by Lemma 22(ii) because $L_N(h(q)) = L_M(q)$. The same lemma also allows us to conclude that $|L_M(\newmoon) \triangle L_N(h(q))| = 2^{s-1}$ under condition (†). Overall, there are $2^s \cdot 2^{s-1} \cdot |E| \cdot (|V| - 2)$ such error words, and exactly that many under condition (†).

This proves the claim. □

Now we investigate the number of errors introduced for a colouring violation in the 3-colouring c. Let $e = \{v_1, v_2\} \in E$ be a violating edge, i.e., $c(v_1) = c(v_2)$. Then we have $\mu(uv_1e) = \mu(uv_2e) \in h(S)$ for every $u \in \{a,b\}^s$. However, $\{\delta(uv_1e), \delta(uv_2e)\} = \{\circledcirc_0, \circledcirc\}$, which yields at least $2^s \cdot 2^s$ errors by Lemma 22(ii). Since $s > \log_2(|V|) + 2$, we have that $2^{2s} > 2^{s+1} \cdot |V|$, which, together with Claim 2 proves our statement if the graph is not 3-colourable.

Finally, let us consider the errors of $c(M) = \langle P, \Sigma, \mu, 0, F \rangle$ provided that $c \colon V \to [3]$ is a proper 3-colouring for G. Recall the properties mentioned in the proof of Lemma 23. Since $c(M)$ fulfils property (†), we already identified exactly the errors of Claim 2. Clearly, all error words pass a state $h(i_0)$ in N with $i \in [3]$, which yields that we only have to consider error words with prefix uve or uva where $u \in \{a,b\}^*$, $v \in V$, and $e \in E$ such that $v \in e$.

– We start with the prefix uva. As already observed we have that $\mu^{-1}(s) = \{a,b\}^s = \delta^{-1}(s)$. Moreover, $\mu(s, v) = i_0$ for all $v \in c^{-1}(i)$, whereas we have $\delta(s, v) = v$. Thus, we consider $L_M(v) \triangle L_{c(M)}(i_0)$. Since $\delta(v, a) = 1_1$ and $\mu(i_0, a) = i_1$, we obtain the potential errors

$$\{uvab^j \mid j \in \{\ell - 1, \ell\}, u \in \{a,b\}^s\} \ .$$

Consequently, we have $2^{s+1} \cdot |V|$ potential errors, all of which are of length at most $s + 2 + \ell = k - s + 2 < k$.
– Finally, we have to consider the prefixes uve. If $v \in e$, then $\mu(i_0, e) \equiv \delta(v, e)$ by the construction of $c(M)$ and so no errors are introduced.

Summing up all identified error words, we obtain that the DFA $c(M)$ commits at most
$$2^{2s-1} \cdot |E| \cdot (|V| - 2) + 3 \cdot 2^{s-1} \cdot |E| + 2^{s+1} \cdot |V|$$

errors, all of which are of length smaller than k, which also proves that the DFA $c(M)$ is k-similar to M. Moreover, since all of its states are pairwise k-dissimilar, it is also k-minimal. □