International Journal of Foundations of Computer Science

Hyper-Minimization for Deterministic Tree Automata*

ARTUR JEŻ†
Max-Planck-Institut für Informatik
Campus E1 4, 66123 Saarbrücken, Germany
aje@cs.uni.wroc.pl

ANDREAS MALETTI‡
Institute for Natural Language Processing, Universität Stuttgart
Pfaffenwaldring 5b, 70569 Stuttgart, Germany
andreas.maletti@ims.uni-stuttgart.de

Received (19 November 2012)
Accepted (31 March 2013)
Communicated by (xxxxxxxxxx)

Hyper-minimization is a recent automaton compression technique that can reduce the size of an automaton beyond the limits imposed by classical minimization. The additional compression power is enabled by allowing a finite difference in the represented language. The necessary theory for hyper-minimization is developed for (bottom-up) deterministic tree automata. The hyper-minimization problem for deterministic tree automata is reduced to the hyper-minimization problem for deterministic finite-state string automata, for which fast algorithms exist. The fastest algorithm obtained in this way runs in time $\mathcal{O}(m \log n)$, where m is the size of the transition table and n is the number of states of the input tree automaton.

1. Introduction

In many application areas (such as speech processing [17], transliteration, parsing, and machine translation [13]) large finite-state devices are used. Since the final devices are often shipped to customers, where they have to fit into limited memory, procedures that reduce the size are essential in those setups. The classical minimization problem for deterministic finite-state automata (dfa) [21] asks for the smallest (measured in the number of states) dfa that recognizes the same language as the input dfa. The asymptotically fastest dfa minimization algorithm is Hopcroft's

*This article is an extended and revised version of [Jeż, Maletti: *Hyper-Minimization for Deterministic Tree Automata*. In Proc. Implementation and Application of Automata, LNCS 7381, pages 217–228, Springer, 2012].
†Currently on leave from the Institute of Computer Science, University of Wrocław, ul. Joliot-Curie 15, 50-383 Wrocław, Poland.
‡Financial support provided by the German Research Foundation (DFG) grant MA 4959 / 1–1.

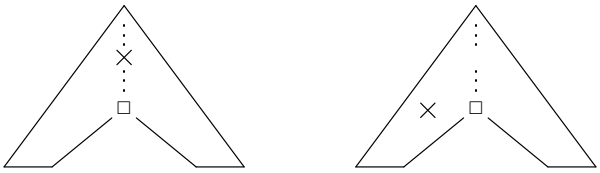

Fig. 1. Illustration of the difference locations in a context: along the path to the root (left) and off this path (right).

algorithm [9], which runs in time $\mathcal{O}(m \log n)$, in which m is the number of transitions and n is the number of states of the input dfa. Similar minimization algorithms exist for finite-state tree automata [5, 6], which recognize tree languages.

Hyper-minimization is a recent technique that allows to reduce the size of a dfa at the expense of a finite number of errors. Such lossy compression can be essential if the memory facilities of the user device are severely limited (e.g., on mobile phones). Hyper-minimization was introduced in [2], in which both its theoretical foundations and an inefficient hyper-minimization algorithm were presented. More efficient algorithms were subsequently developed [1] and the currently fastest algorithms achieve the time complexity $\mathcal{O}(m \log n)$ [4, 8], which coincides with the complexity of the fastest classical dfa minimization algorithms. Since hyper-minimization reduces to minimization [8], any faster hyper-minimization algorithm would automatically yield a faster algorithm for classical dfa minimization. Although no theoretical result precludes the existence of such faster minimization algorithms, to this day no such algorithms are known despite heavy research for more than 50 years. Most of the mentioned applications actually use weighted dfa [18], so hyper-minimization was generalized to this setting [16] and to dfa over infinite strings [19]. A detailed survey of the theory and existing algorithms can be found in [15].

Here, we generalize hyper-minimization to (bottom-up) deterministic tree automata (dta) [5, 6], which have applications in XML [10] and natural language processing [12]. To this end, we faithfully generalize the approach and lift the basic definitions from dfa to dta. The main notion for hyper-minimization of dfa is the state almost-equivalence, which generalizes the classical forward-language equivalence used in dfa minimization. Consequently, our central notion is the almost equivalence for states of a dta, which generalizes the context-language equivalence used in dta minimization [3]. The fastest known algorithm for dta minimization [7] runs in time $\mathcal{O}(m \log n)$, where m is the size and n is the number of states of the input dta. Our hyper-minimization algorithm achieves the same asymptotic run-time complexity using a reduction to hyper-minimization for dfa.

The slightly non-standard reduction shows that dta hyper-minimization is not a straightforward adjustment of dfa hyper-minimization. Their overall structure is identical, but the licensing properties differ. The main difference concerns the location of the errors in the recognized context language. While for dfa the errors in

the forward languages of a state all occur along its successor states, in a dta the errors can also occur in sibling states, which we illustrate in Fig. 1. Consequently, several foundational results (and in particular, those on which the currently fastest hyper-minimization algorithms rely) for dfa hyper-minimization [2] do not faithfully generalize to dta. We provide alternatives for these statements and develop a hyper-minimization algorithm based on them. Finally, we provide a reduction to hyper-minimization for dfa, which allows us to easily obtain the run time $\mathcal{O}(m \log n)$ for our hyper-minimization algorithm for dta. Consequently, we asymptotically match the time complexity of dta minimization, so further improvements would automatically yield improvements for dta minimization, which currently seem unlikely. The reduction required for the last statement is presented in the final section.

2. Preliminaries

The set of all nonnegative integers is $\mathbb{N}$. The cardinality of a finite set S is denoted by $|S|$. The symmetric difference $S \ominus T$ of sets S and T is $(S - T) \cup (T - S)$. If it is finite, then S and T are almost equal. A binary relation $\cong$ on S is an equivalence relation if it is reflexive, symmetric, and transitive.

An alphabet Σ is a finite set of symbols. The set of all strings over Σ is Σ^*, which contains the empty string ε. The length of a string $w \in \Sigma^*$ is $|w|$. A deterministic finite automaton (dfa) [21] is a tuple $M = (Q, \Sigma, q_0, \delta, F)$, in which Q is the finite set of states, Σ is the alphabet of input symbols, $q_0 \in Q$ is the initial state, $\delta\colon Q \times \Sigma \to Q$ is the (partial) transition function, and $F \subseteq Q$ is the set of final states. The transition function extends to a mapping $\delta\colon Q \times \Sigma^* \to Q$ by $\delta(q, \varepsilon) = q$ and $\delta(q, w\sigma) = \delta(\delta(q, w), \sigma)$ for all $q \in Q$, $w \in \Sigma^*$, and $\sigma \in \Sigma$. The dfa M recognizes the language $L(M) = \{w \in \Sigma^* \mid \delta(q_0, w) \in F\}$. The states $q, q' \in Q$ are equivalent, written $q \equiv q'$, if $\delta(q, w) \in F$ if and only if $\delta(q', w) \in F$ for all $w \in \Sigma^*$. The (trim) dfa M is minimal if it does not contain two different, but equivalent states. For every $k \in \mathbb{N}$, let $q \simeq_k q'$ if $\delta(q, w) \equiv \delta(q', w)$ for all $w \in \Sigma^*$ with $|w| \geq k$. If $q \simeq_k q'$ for some $k \in \mathbb{N}$, then q and q' are almost equivalent, which we denote by $q \simeq q'$.

Together with a mapping $\mathrm{rk}\colon \Sigma \to \mathbb{N}$ the alphabet Σ forms the ranked alphabet (Σ, rk). For every $k \in \mathbb{N}$, we let $\Sigma_k = \mathrm{rk}^{-1}(k)$ be the set of all symbols of rank k. We typically denote the ranked alphabet (Σ, rk) by just Σ and write $\sigma^{(k)}$ to indicate that σ has rank k. For a set T, we let $\Sigma(T) = \{\sigma(t_1, \ldots, t_k) \mid \sigma \in \Sigma_k, t_1, \ldots, t_k \in T\}$. The set $T_\Sigma(Q)$ of Σ-trees with states Q is the smallest set T such that $Q \cup \Sigma(T) \subseteq T$. We write T_Σ for $T_\Sigma(\emptyset)$. Let $n \in \mathbb{N}$ be such that $n \geq \mathrm{rk}(\sigma)$ for every $\sigma \in \Sigma$. The set of positions $\mathrm{pos}(t) \subseteq \{1, \ldots, n\}^*$ is recursively defined by $\mathrm{pos}(q) = \{\varepsilon\}$ for every $q \in Q$ and

$$\mathrm{pos}(\sigma(t_1, \ldots, t_k)) = \{\varepsilon\} \cup \{iw \mid 1 \leq i \leq k, w \in \mathrm{pos}(t_i)\}$$

for every $\sigma \in \Sigma_k$ and $t_1, \ldots, t_k \in T_\Sigma(Q)$. For every $\sigma \in \Sigma \cup Q$, the set of all σ-labeled positions of a tree $t \in T_\Sigma(Q)$ is $\mathrm{pos}_\sigma(t)$. Given such a position $w \in \mathrm{pos}_\sigma(t)$ and a tree $t' \in T_\Sigma(Q)$, the tree $t[t']_w$ is obtained from t by replacing the occurrence of σ

at w by t'. The height $\mathrm{ht}(t)$ and size $|t|$ of $t \in T_\Sigma(Q)$ are $\mathrm{ht}(t) = \max_{w \in \mathrm{pos}(t)} |w|$ and $|t| = |\mathrm{pos}(t)|$, respectively.

A context c is a tree of $T_{\Sigma \cup \{\square^{(0)}\}}(Q)$ such that $\mathrm{pos}_\square(c)$ is a singleton; i.e., the special nullary symbol $\square$ occurs exactly once. The set of all contexts is $C_\Sigma(Q)$, which we abbreviate to C_Σ if $Q = \emptyset$. For every $c \in C_\Sigma(Q)$ and $t \in T_{\Sigma \cup \{\square\}}(Q)$, we let $c[t] = c[t]_w$ with $\mathrm{pos}_\square(c) = \{w\}$. If $c, c' \in C_\Sigma(Q)$ are contexts, then $c[c'] \in C_\Sigma(Q)$ is a context, and we set $c^0 = \square$ and $c^{n+1} = c[c^n]$ for every $n \in \mathbb{N}$. The depth of a context $c \in C_\Sigma(Q)$ with $\mathrm{pos}_\square(c) = \{w\}$ is $\mathrm{dp}(c) = |w|$.

A deterministic tree automaton (dta) [5, 6] is a tuple $M = (Q, \Sigma, \delta, F)$, in which Q is the finite set of states, Σ is the ranked alphabet of input symbols, $\delta \colon \Sigma(Q) \to Q$ is the (partial) transition function, and $F \subseteq Q$ is the set of final states. The transition function δ extends to $\delta \colon T_\Sigma(Q) \to Q$ by $\delta(q) = q$ for every $q \in Q$ and

$$\delta(\sigma(t_1, \dots, t_k)) = \delta(\sigma(\delta(t_1), \dots, \delta(t_k)))$$

for every $\sigma \in \Sigma_k$ and $t_1, \dots, t_k \in T_\Sigma(Q)$. For every $q, q' \in Q$, we let

- $L(M)^q_{q'} = \{c \in C_\Sigma \mid \delta(c[q']) = q\}$ be the contexts that take M from q' to q,
- $L(M)_{q'} = \bigcup_{f \in F} L(M)^f_{q'}$ be those that take it from q' into a final state, and
- $L(M)^q = \delta^{-1}(q) \cap T_\Sigma$ be the (stateless) trees that take M into the state q.

The dta M recognizes the tree language $L(M) = \bigcup_{f \in F} L(M)^f$. The size $|M|$ of M is $\sum_{s \in \mathrm{dom}(\delta)} |s|$. The dta is *trim* if $L(M)^q \neq \emptyset$ for every $q \in Q$.

An equivalence relation $\cong$ on Q is a congruence (on the dta M) if we have that $\delta(\sigma(q_1, \dots, q_k)) \cong \delta(\sigma(q'_1, \dots, q'_k))$ for every $\sigma \in \Sigma_k$ and $q_1 \cong q'_1, \dots, q_k \cong q'_k$. Two states $q, q' \in Q$ are equivalent, which is denoted by $q \equiv_M q'$ (or just $q \equiv q'$), if $L(M)_q = L(M)_{q'}$. Note that $\equiv_M$ is a congruence, and actually, the coarsest (i.e., least refined) congruence on M that respects F, which means that a final state cannot be equivalent to a nonfinal state. The trim dta M is minimal if it does not have two different, but equivalent states. For every dta M, an equivalent minimal dta can be computed efficiently using an adaptation [7] of HOPCROFT's algorithm [9], which runs in time $O(m \log n)$ where $m = |M|$ and $n = |Q|$.

3. Foundations

In this section, we investigate the foundations required for the minimization of dta, where in contrast to the standard setting we do not require that the recognized tree language is preserved, but rather we allow a finite difference. This type of minimization is called *hyper-minimization*. The theoretical properties that enable the reduction to the hyper-minimization problem for dfa are investigated in Section 4, but the general approach is established here. Since it coincides with the approach for dfa hyper-minimization, we transfer the relevant notions from dfa to dta and prove variants of the relevant theorems. We refer the reader to [2] for the foundations of dfa hyper-minimization and a detailed account of the dfa-versions of most of the results in this section.

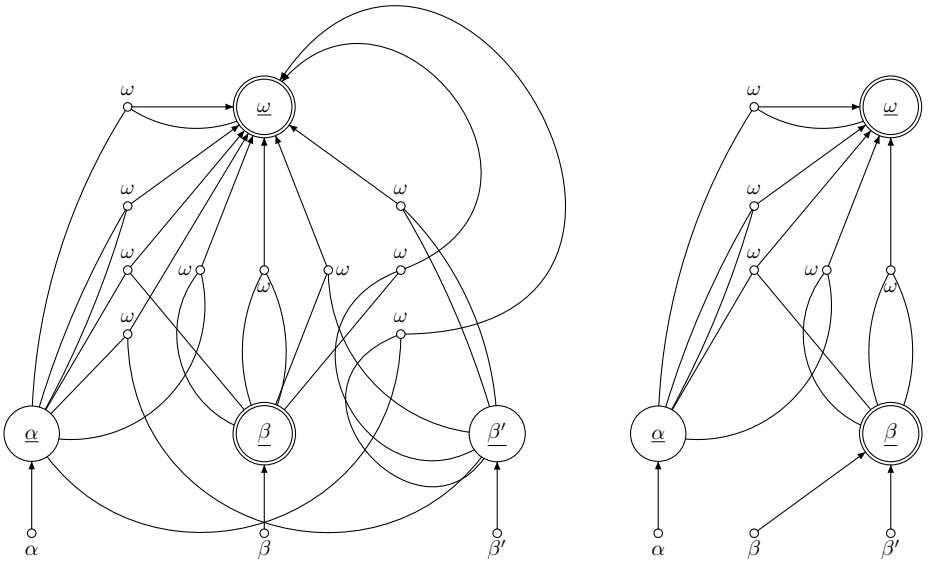

Fig. 2. Illustration of the dta M_{ex} [left] and merge$(M_{\text{ex}}, \underline{\beta'} \to \underline{\beta})$ [right] of Examples 2 and 5, respectively.

In the rest of this paper, $M = (Q, \Sigma, \delta, F)$ is a minimal dta of size m. To avoid a detailed discussion of undefinedness or partialness, we imagine $\text{dom}(\delta) = \Sigma(Q)$. This is achieved by imagining a non-final sink state $\perp$ as the target of all missing transitions. However, these transitions and the state $\perp$ itself are imaginary and only used for theoretical convenience, so they will not count towards $|M|$ and $|Q|$.

Given the minimal dta M, the goal of hyper-minimization is the (efficient) computation of a dta M' with as few states as possible such that $L(M)$ and $L(M')$ are almost equal. If $L(M)$ and $L(M')$ are almost equal, then the dta M and M' are *almost equivalent*. Moreover, if there does not exist a dta with strictly fewer states than M that is almost equivalent to M, then M is *hyper-minimal*. Using these notions, the goal of hyper-minimization is the computation of a hyper-minimal dta that is almost equivalent to M.

We approach this problem in the same spirit as classical minimization. Recall, that a minimal dta is obtained by identifying and merging all equivalent states. We plan to obtain a hyper-minimal dta by identifying and merging certain almost equivalent states, which we introduce next.

Definition 1 (cf. Definition 2.2 of [2]) *Two states $q, q' \in Q$ are* almost equivalent *if $L(M)_q \ominus L(M)_{q'}$ is finite. The almost equivalence on Q is denoted by $\sim$.*

Example 2. *We use the dta $M_{\text{ex}} = (Q, \Sigma, \delta, F)$ as running example, where*

- $\Sigma = \{\alpha^{(0)}, \beta^{(0)}, \beta'^{(0)}, \omega^{(2)}\}$ *and* $\Gamma = \{\alpha, \beta, \beta'\}$,

- $Q = \{\underline{\sigma} \mid \sigma \in \Sigma\}$ *and* $F = \{\underline{\beta}, \underline{\omega}\}$, *and*
- δ *is undefined except in the following cases: for every* $\gamma, \gamma' \in \Gamma$

$$\delta(\gamma) = \underline{\gamma} \qquad \delta(\omega(\underline{\gamma}, \underline{\gamma'})) = \delta(\omega(\underline{\alpha}, \underline{\omega})) = \underline{\omega} \ .$$

The dta M_{ex} *recognizes the tree language* $\{\beta\} \cup \{c^n[\omega(\gamma, \gamma')] \mid n \in \mathbb{N}, \gamma, \gamma' \in \Gamma\}$ *with* $c = \omega(\alpha, \square)$. *The dta* M_{ex} *is illustrated in Fig. 2. Note that* M_{ex} *is minimal and has* 4 *states and size* $3 + 27 + 3 = 33$. *The states* $\underline{\beta}$ *and* $\underline{\beta'}$ *are almost equivalent because*

$$L(M_{\mathrm{ex}})_{\underline{\beta'}} = \{c^n[\omega(\square, \gamma)], c^n[\omega(\gamma, \square)] \mid n \in \mathbb{N}, \gamma \in \Gamma\}$$
$$L(M_{\mathrm{ex}})_{\underline{\beta}} = \{\square\} \cup L(M_{\mathrm{ex}})_{\underline{\beta'}} \ ,$$

whereas the state $\underline{\alpha}$ *is neither almost equivalent to* $\underline{\beta}$ *nor to* $\underline{\omega}$.

Next, we demonstrate that the almost equivalence $\sim$ is a congruence on M. In contrast to the state equivalence $\equiv$, which is used for classical minimization and respects F, the almost equivalence $\sim$ clearly need not respect F (see Example 2 because $\underline{\beta} \sim \underline{\beta'}$ but $\underline{\beta} \in F$ and $\underline{\beta'} \notin F$).

Lemma 3 (see Lemma 2.10 in [2]) *For all almost equivalent states* $q \sim q'$ *and all contexts* $c \in C_\Sigma$, *we have* $\delta(c[q]) \sim \delta(c[q'])$. *In particular,* $\sim$ *is a congruence.*

Proof. Let $C' = L(M)_{\delta(c[q])} \ominus L(M)_{\delta(c[q'])}$ and $C = L(M)_q \ominus L(M)_{q'}$ be the sets of contexts that distinguish between $\delta(c[q])$ and $\delta(c[q'])$ and between q and q', respectively. We know that C is finite and need to prove that C' is finite as well. Let $\imath\colon C' \to C_\Sigma$ be such that $\imath(c') = c'[c]$ for every context $c' \in C'$. Clearly, $\imath(c') = c'[c] \neq c''[c] = \imath(c'')$ for all $c', c'' \in C'$ such that $c' \neq c''$. Moreover, $\imath(c') = c'[c] \in C$ for all $c' \in C'$, so $\imath\colon C' \to C$ is injective. Consequently, $|C'| \leq |C|$, which proves that C' is finite.

The congruence property can now be deduced via particular contexts of depth 1 and the standard piecewise replacement. Let $\sigma \in \Sigma_k$ and $q_1 \sim q_1', \ldots, q_k \sim q_k'$ be pairwise almost equivalent states. Moreover, for each $q \in Q$, let $t_q \in L(M)^q$ be arbitrary; such a tree exists for all $q \in Q$ because M is minimal. Then

$$\begin{aligned}
&\delta(\sigma(q_1, \ldots, q_k)) = \delta(\sigma(\square, t_{q_2}, \ldots, t_{q_k})[q_1]) \\
&\sim \delta(\sigma(\square, t_{q_2}, \ldots, t_{q_k})[q_1']) = \delta(\sigma(q_1', q_2, \ldots, q_k)) = \delta(\sigma(t_{q_1'}, \square, t_{q_3}, \ldots, t_{q_k})[q_2]) \\
&\sim \ldots \\
&\sim \delta(\sigma(t_{q_1'}, \ldots, t_{q_{k-1}'}, \square)[q_k']) = \delta(\sigma(q_1', \ldots, q_k')) \ .
\end{aligned}$$

□

To confirm that our strategy of merging almost equivalent states yields a hyper-minimal dta, we first need to characterize hyper-minimal dta. To this end, we first show that the two states reached after processing the same tree in two almost equivalent dta behave like almost equivalent states.

Lemma 4. *As usual, let* M *be the minimal dta under consideration. Moreover, let* $M' = (Q', \Sigma, \delta', F')$ *be another minimal dta that is almost equivalent to* M. *Then* $L(M')_{\delta'(t)}$ *and* $L(M)_{\delta(t)}$ *are almost equal for all* $t \in T_\Sigma$.

Proof. For every tree language $L \subseteq T_\Sigma$ and $t \in T_\Sigma$, let $t^{-1}L = \{c \in C_\Sigma \mid c[t] \in L\}$ be the set of contexts that wrap around t to form a tree in L. Since M and M' are almost equivalent, we know that $L(M)$ and $L(M')$ are almost equal. Consequently, also $t^{-1}L(M) = L(M)_{\delta(t)}$ and $t^{-1}L(M') = L(M')_{\delta'(t)}$ are almost equal, which proves the statement. □

We already announced that we plan to obtain a hyper-minimal dta by merging certain almost equivalent states. As in classical minimization, a *merge* of the state $q \in Q$ into the state $q' \in Q$ redirects all transitions leading to q into q'. Formally, for two different states $q, q' \in Q$, the dta $\mathrm{merge}(M, q \to q')$ is $(Q - \{q\}, \Sigma, \delta', F - \{q\})$, where for every $s \in \Sigma(Q - \{q\})$

$$\delta'(s) = \begin{cases} q' & \text{if } \delta(s) = q \\ \delta(s) & \text{otherwise.} \end{cases}$$

Example 5. *Recall the dta* $M_{\mathrm{ex}} = (Q, \Sigma, \delta, F)$ *of Example 2. If we merge* $\underline{\beta'}$ *into* $\underline{\beta}$*, then we obtain the dta* $\mathrm{merge}(M_{\mathrm{ex}}, \underline{\beta'} \to \underline{\beta})$*, which is* $(Q - \{\underline{\beta'}\}, \Sigma, \delta', F)$ *where* δ' *is undefined except in the following cases: for every* $\gamma, \gamma' \in \{\alpha, \beta\}$

$$\delta'(\alpha) = \underline{\alpha} \qquad \delta'(\beta) = \delta'(\beta') = \underline{\beta} \qquad \delta'(\omega(\underline{\gamma}, \underline{\gamma'})) = \delta'(\omega(\underline{\alpha}, \underline{\omega})) = \underline{\omega} \ .$$

The merged dta is illustrated in Fig. 2.

Before we proceed, we recall a central notion from [2]. A state $q \in Q$ is a kernel state if $L(M)^q$ is infinite. Otherwise q is a preamble state. The set of kernel states is $\mathrm{Ker}(M)$. In contrast to the standing assumption, we do not assume that M is minimal in the next lemma. This adjustment is necessary because we want to perform many merges in sequence, but we cannot guarantee that the dta resulting from a merge in a minimal dta is again minimal.

Lemma 6. *Let* $M = (Q, \Sigma, \delta, F)$ *be a not necessarily minimal dta, and let* $q \sim q'$ *be different almost equivalent states such that* q *is a preamble state. Then the dta* $\mathrm{merge}(M, q \to q')$ *and* M *are almost equivalent.*

Proof. Let $\mathrm{merge}(M, q \to q') = (Q', \Sigma, \delta', F')$. Since q and q' are almost equivalent, the set $C = L(M)_q \ominus L(M)_{q'}$ is finite. Let $\ell \in \mathbb{N}$ be such that $\ell > \mathrm{ht}(c)$ for every $c \in C$. Moreover, let $t \in T_\Sigma$ be such that $\mathrm{ht}(t) \geq \ell + |Q|$. Since q is a preamble state, the elements of the finite set $L(M)^q$ are pairwise not subtrees (i.e., $t' \in T_\Sigma$ is a subtree of $t \in T_\Sigma$ if there exists a context $c \in C_\Sigma$ such that $t = c[t']$) of each other. Consequently, we obtain the uniquely determined tree $u \in T_\Sigma(Q)$ from t by replacing all subtrees from $L(M)^q$ in t by just the state q. Since we replaced subtrees by the state that accepts them, we obtain that $\delta(t) = \delta(u)$. Furthermore, $\mathrm{ht}(u) \geq \ell$ because $\mathrm{ht}(t') \leq |Q|$ for all $t' \in L(M)^q$ since q is a preamble state. Let $\mathrm{pos}_q(u) = \{w_1, \ldots, w_n\}$ with $w_1 < \cdots < w_n$ be the occurrences of q in u. For each $i \in [n]$, let $c_i = (u[q']_{w_1} \cdots [q']_{w_{i-1}})[\Box]_{w_i}$ be the context obtained from u by

replacing the first $i-1$ occurrences $w_1, \dots, w_{i-1}$ of q by the merging target state q' and the occurrence w_i by $\Box$. Clearly, $\mathrm{ht}(c_i) = \mathrm{ht}(u) \geq \ell$ for all $i \in [n]$, and thus

$$\delta(t) = \delta(u) = \delta(c_1[q]) \overset{\dagger}{\equiv} \delta(c_1[q']) = \delta(c_2[q]) \overset{\dagger}{\equiv} \cdots \overset{\dagger}{\equiv} \delta(c_n[q']) \overset{\ddagger}{=} \delta'(t) \ ,$$

where $\dagger$ holds because $\mathrm{ht}(c_i) \geq \ell$, which yields that the states q and q' reach equivalent states after processing the contexts $c_1, \dots, c_n$, and $\ddagger$ holds because δ and δ' coincide on all transitions not involving q. Consequently, $\delta(t) \equiv \delta'(t)$, and thus $\mathrm{merge}(M, q \to q')$ and M agree on all suitably tall trees. Since there are only finitely many small trees (mind that the rank of symbols is fixed), we obtain that $\mathrm{merge}(M, q \to q')$ and M are almost equivalent. $\Box$

Lemma 6 shows how to obtain a smaller dta that remains almost equivalent to the original dta. Since we did not require minimality in Lemma 6, the procedure can be iterated. It remains to demonstrate that if no further merges are possible (subject to the conditions of Lemma 6), then the obtained dta is hyper-minimal. To prove this, we characterize hyper-minimality in the spirit of [2]. Recall that a trim dta is minimal if and only if it does not have two different, but equivalent states. The condition for hyper-minimality replaces equivalence by almost equivalence and additionally requires that one state is a preamble state (see Lemma 6).

Theorem 7. *The minimal dta M is hyper-minimal if and only if every pair of different, but almost equivalent states consists of only kernel states.*

Proof. Lemma 6 proves the "only if"-direction by contra-position because we can merge almost equivalent states with at least one preamble state to obtain a smaller, not necessarily minimal, but almost equivalent dta. For the "if"-direction, let $M' = (Q', \Sigma, \delta', F')$ be an almost equivalent dta that has strictly fewer states (i.e., $|Q'| < |Q|$). We construct the product dta $M \times M' = (Q \times Q', \Sigma, \delta \times \delta', F \times F')$, where $(\delta \times \delta') \colon \Sigma(Q \times Q') \to Q \times Q'$ is such that

$$(\delta \times \delta')(\sigma(\langle q_1, q_1' \rangle, \dots, \langle q_k, q_k' \rangle)) = \langle \delta(\sigma(q_1, \dots, q_k)), \delta'(\sigma(q_1', \dots, q_k')) \rangle$$

for every $\sigma \in \Sigma_k$ and $\langle q_1, q_1' \rangle, \dots, \langle q_k, q_k' \rangle \in Q \times Q'$. Since M is minimal, we have $L(M)^q \neq \emptyset$ for every $q \in Q$. If $q \in \mathrm{Ker}(M)$, then we select $t_q \in L(M)^q$ such that $\mathrm{ht}(t_q) \geq |Q|^2$. For all preamble states $q \in Q$, we select $t_q \in L(M)^q$ arbitrarily. By the pigeon-hole principle with $|Q'| < |Q|$, there must exist different $q_1, q_2 \in Q$ and $q' \in Q'$ such that $(\delta \times \delta')(t_q) = \langle q, q' \rangle$ for $q \in \{q_1, q_2\}$. Consequently, $q_1 \sim q_2$ because $L(M)_{q_1}$ and $L(M')_{q'}$ as well as $L(M)_{q_2}$ and $L(M')_{q'}$ are almost equal by Lemma 4. By the assumption, q_1 and q_2 must be kernel states of M because q_1 and q_2 are different, but almost equivalent. Moreover, $\langle q_1, q' \rangle$ and $\langle q_2, q' \rangle$ are kernel states of $M \times M'$ by the selection of the access trees t_{q_1} and t_{q_2} with $\mathrm{ht}(t_{q_1}) \geq |Q|^2 \leq \mathrm{ht}(t_{q_2})$, which can be pumped [5, 6]. Now, for the sake of a contradiction, let $c \in L(M)_{q_1} \ominus L(M')_{q'}$. Then $\{c[t] \mid t \in L(M \times M')^{\langle q_1, q' \rangle}\} \subseteq L(M) \ominus L(M')$. Since $\langle q_1, q' \rangle$ is a kernel state of $M \times M'$, the set $L(M \times M')^{\langle q_1, q' \rangle}$ is infinite, which yields that

Algorithm 1 Structure of our dta hyper-minimization algorithm.

Require: a minimal dta $M = (Q, \Sigma, \delta, F)$ of size m with n states
Return: an almost equivalent hyper-minimal dta

$K \leftarrow \mathrm{Ker}(M)$ // complexity: $\mathcal{O}(m)$; see Section 4
2: $\sim \leftarrow$ AlmostEquivalence(M) // complexity: $\mathcal{O}(m \log n)$; see Section 4
for all $B \in (Q/\!\sim)$ **do**
4: select $q_B \in B$ such that $q_B \in K$ if possible
for all $q \in B - K$ **do**
6: $M \leftarrow \mathrm{merge}(M, q \to q_B)$ // complexity: $\mathcal{O}(1)$
return M

$L(M) \ominus L(M')$ is infinite. This contradicts that M and M' are almost equivalent, so consequently, $L(M)_{q_1} \ominus L(M')_{q'} = \emptyset$, and similarly $L(M)_{q_2} \ominus L(M')_{q'} = \emptyset$. Thus, $L(M)_{q_1} = L(M')_{q'} = L(M)_{q_2}$, which proves that q_1 and q_2 are equivalent. Since M is minimal, we obtain that $q_1 = q_2$ contradicting the assumption that q_1 and q_2 are different. Consequently, the dta M' with strictly fewer states cannot exist, which proves the statement. □

Example 8. *The dta M_{ex} of Example 2 is not hyper-minimal since $\underline{\beta} \sim \underline{\beta'}$ and both states are preamble states. However, the dta* $\mathrm{merge}(M_{\mathrm{ex}}, \underline{\beta'} \to \underline{\beta})$ *of Example 5 is hyper-minimal.*

The overall hyper-minimization approach is demonstrated in Algorithm 1. First, we determine the kernel states and the almost equivalence with the help of the methods described in Section 4. With these two pieces of information, we can merge states (by simply changing a reference to obtain the constant run-time) subject to the conditions of Lemma 6, which shows that the merged dta is almost equivalent to the original. Once we stop the merges, there are no more preamble states that are almost equivalent to another state, which by Theorem 7 means that the returned dta is hyper-minimal.

Corollary 9 (of Lemma 6 and Theorem 7) *Given a correct computation of the kernel states* $\mathrm{Ker}(M)$ *and the almost equivalence $\sim$, Algorithm 1 returns a hyper-minimal dta that is almost equivalent to M.*

4. Computation of the kernel states and the almost equivalence

In this section, we show how to compute the kernel states $\mathrm{Ker}(M)$ as well as the almost equivalence $\sim$ efficiently. In both cases, we use a reduction to the corresponding problem for dfa. We start with the calculation of the kernel states. Recall that $m = |M|$. It is known [4, 8] that the kernel states of a dfa can be computed using any fast algorithm for computing strongly connected components in a directed graph (e.g., Tarjan algorithm [20]). The next proposition shows the trivial problem

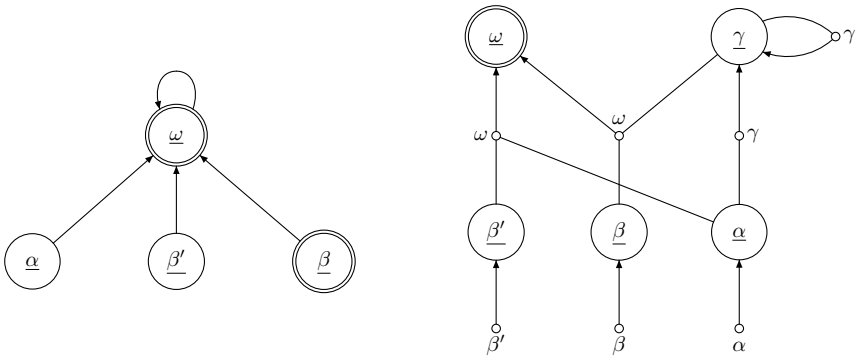

Fig. 3. The graph (Q, E) [left] derived from the dta M_{ex} of Example 2 and the dta [right] of Example 12.

translation.

Lemma 10. $\mathrm{Ker}(M)$ *can be computed in time* $\mathcal{O}(m)$.

Proof. We turn the dta M into the graph (Q, E), where

$$E = \{\langle q, \delta(\sigma(q_1, \ldots, q_k))\rangle \mid q \in \{q_1, \ldots, q_k\},\, \sigma(q_1, \ldots, q_k) \in \mathrm{dom}(\delta)\} \ .$$

Consequently, $|E| \leq m$. A strongly connected component of this graph is trivial, when it consists of a single node without a loop. It is simple to observe that $q \in \mathrm{Ker}(M)$ if and only if it is reachable from a non-trivial strongly connected component of the graph (Q, E) [see [8] for details]. Since Tarjan's algorithm [20] for strongly connected components of (Q, E) runs in time $\mathcal{O}(|Q| + |E|)$ and $|Q| \leq m$ by minimality and $|E| \leq m$, we obtain the stated complexity $\mathcal{O}(m)$. □

Example 11. *The only kernel state of the dta* M_{ex} *of Example 2 is* $\underline{\omega}$*, which is easily determined from the graph* (Q, E) *displayed in Fig. 3.*

It remains to compute the almost equivalence efficiently. Since almost equivalent states q and q' have almost equal context languages $L(M)_q$ and $L(M)_{q'}$, there exists an integer $k \in \mathbb{N}$ such that $k > \mathrm{dp}(c)$ for all contexts $c \in L(M)_q \ominus L(M)_{q'}$. A simple pumping argument shows that we can select k such that $k \leq |Q|^2$. In fact, a slightly more elaborate argument proves that even $k \leq |Q|$ can be required. In contrast to the string case, the set of contexts of bounded depth can be (and typically is) infinite. Nevertheless, for every context $c \in C_\Sigma$ with $\mathrm{dp}(c) \geq k$ the states $\delta(c[q])$ and $\delta(c[q'])$ are equivalent because no deeper context is in the symmetric difference $L(M)_q \ominus L(M)_{q'}$. Since M is minimal, and thus has no different, but equivalent states, we can conclude that $\delta(c[q]) = \delta(c[q'])$ for all $c \in C_\Sigma$ with $\mathrm{dp}(c) \geq k$. Since we need this property in the sequel, for every $k \in \mathbb{N}$ we let $q \sim_k q'$ if $q \sim q'$ and $\delta(c[q]) = \delta(c[q'])$ for all $c \in C_\Sigma$ with $\mathrm{dp}(c) \geq k$. In particular, $q \sim q'$

implies $q \sim_k q'$ for some $k \in \mathbb{N}$. Roughly speaking, almost equivalent states always eventually yield the same state after processing suitably deep (or large) contexts.

The converse of the previous statement is particularly interesting for the computation of almost equivalent states because it would offer a way to determine the almost equivalence of two states with the help of $\sim_0, \sim_1, \ldots$ For minimal dfa this converse trivially holds because there are only finitely many strings of bounded length; i.e., two states that always eventually yield the same state are almost equivalent, which is the main property used in the fastest hyper-minimization algorithms [4, 8] for dfa. Unfortunately, the converse does not hold for dta because there can be infinitely many contexts of bounded depth, so that not only the successor, but also the sibling states determine the almost equivalence (see Fig. 1). Let us illustrate this difficulty on an example.

Example 12. *Let us consider the dta* (Q, Σ, δ, F) *with* $Q = \{\underline{\sigma} \mid \sigma \in \Sigma\}$*, input symbols* $\Sigma = \{\alpha^{(0)}, \beta^{(0)}, \beta'^{(0)}, \gamma^{(1)}, \omega^{(2)}\}$*, and the final states* $F = \{\underline{\omega}\}$*. The transition function* δ *is given by*

$$\delta(\sigma) = \underline{\sigma} \qquad \delta(\gamma(\underline{\alpha})) = \delta(\gamma(\underline{\gamma})) = \underline{\gamma} \qquad \delta(\omega(\underline{\beta}, \underline{\gamma})) = \delta(\omega(\underline{\beta'}, \underline{\alpha})) = \underline{\omega}$$

for every $\sigma \in \{\alpha, \beta, \beta'\}$*. This dta is illustrated in Fig. 3. Although* $\underline{\beta}$ *and* $\underline{\beta'}$ *agree on all contexts of depth at least 2 (both states reject all such contexts), they are not almost equivalent since they differ on the infinitely many contexts* $\omega(\Box, t)$ *with* $t \in T_{\{\gamma^{(1)}, \alpha^{(0)}\}}$*, which lead to acceptance in* $\underline{\beta}$*, but are rejected in* $\underline{\beta'}$*.*

Already in the conference version [11], we announced an adjusted version of the converse for the efficient computation of the almost equivalence. We use two new notions: skinny contexts of $C_\Sigma(Q)$ and a distinction between preamble and kernel contexts. We begin with their definition. The *skinny* contexts $\widehat{C}_\Sigma(Q) \subseteq C_\Sigma(Q)$ are the smallest set C such that $\Box \in C$ and $\sigma(q_1, \ldots, q_{i-1}, c, q_{i+1}, \ldots, q_k) \in C$ for all $\sigma \in \Sigma_k$, $q_1, \ldots, q_k \in Q$, integers $1 \leq i \leq k$, and contexts $c \in C$. In other words, skinny contexts only contain states outside the path from the root to the unique occurrence of the symbol $\Box$. We remark that $\delta(c[q]) = \delta(c'[q])$ for all $c, c' \in C_\Sigma(Q)$ such that c' is obtained from c by replacing an occurrence of a state q' by a tree of $\delta^{-1}(q')$. Consequently, we can replace any occurrence of a state q' by a tree in the tree language $\delta^{-1}(q')$ without effect on the behavior of M. The obtained context c' is an *instance of* c. The essential property is that each context $c' \in C_\Sigma(Q)$ is an instance of exactly one skinny context, which allows us to reduce $C_\Sigma(Q)$ to $\widehat{C}_\Sigma(Q)$. This move from C_Σ to $\widehat{C}_\Sigma(Q)$ is a common strategy, which is also used in dta minimization [7, 14]. Finally, a context $c \in C_\Sigma(Q)$ is a *kernel context* if $\mathrm{pos}_q(c) \neq \emptyset$ for some $q \in \mathrm{Ker}(M)$; i.e., the context c contains a kernel state of M.

Lemma 13. *We have* $q \sim q'$ *if and only if there is an integer* $k \in \mathbb{N}$ *such that*

- $\delta(c[q]) = \delta(c[q'])$ *for all* $c \in \widehat{C}_\Sigma(Q)$ *with* $\mathrm{dp}(c) \geq k$*, and*
- $\delta(c[q]) = \delta(c[q'])$ *for all kernel contexts* $c \in \widehat{C}_\Sigma(Q)$*.*

Proof. We start with the "if"-direction. By assumption, q and q' agree on skinny kernel contexts. By the argumentation above, they also agree on all instances of such contexts. Moreover, they disagree only on finitely many skinny non-kernel contexts because there are only finitely many skinny contexts of a given depth. Since only preamble states occur in disagreement contexts, we know that those contexts have only finitely many instances of C_Σ, on which q and q' also disagree. This proves that q and q' agree on almost all contexts of C_Σ, which yields $q \sim q'$.

For the remaining "only if" direction, we know that q and q' disagree on finitely many contexts of C_Σ. Let C be the finite set of skinny contexts, of which the disagreement contexts are instances of. We select k such that it is strictly larger than the depth of any context of C. Since q and q' agree on every instance $c' \in C_\Sigma$ of a skinny context $c \in \widehat{C}_\Sigma(Q)$ with depth at least k, they also agree on c, which proves the first item. Now, let c be a skinny kernel context. Clearly, c has infinitely many instances in C_Σ, on all of which M behaves equally. Thus, q and q' cannot disagree on c because they would have to disagree also on all infinitely many instances of c, which proves the second item. □

Since the parameter k in Lemma 13 can always be limited to $|Q|$, we now have an effective procedure to check for almost equivalence of two states. To determine the full almost equivalence on Q, we also prove a variant that utilizes the known almost equivalence of pairs of states. A context $c \in C_\Sigma(Q)$ is *shallow* if $c \in \Sigma(Q \cup \{\Box\})$. The set of all shallow contexts for M is denoted by C_M. Intuitively, shallow contexts are transitions with exactly one $\Box$ in the list of source states. In particular, they are skinny contexts.

Corollary 14 (of Lemma 3 and Lemma 13) *We have $\sim_0 = \{(q, q) \mid q \in Q\}$, and for every $k \in \mathbb{N}$ we have $q \sim_{k+1} q'$ if and only if for each shallow context $c \in C_M$*

- *$\delta(c[q]) \sim_k \delta(c[q'])$ and*
- *$\delta(c[q]) = \delta(c[q'])$ if c is a kernel context.*

Proof. For the "only-if" direction, $q \sim_{k+1} q'$ implies $q \sim q'$, which by Lemma 13 implies the second item. Moreover, using Lemma 3 we conclude that $\delta(c[q]) \sim \delta(c[q'])$. Naturally $\delta(c[q])$ and $\delta(c[q'])$ agree on all contexts of depth at least k because q and q' agree on all contexts of depth at least $k+1$. For the converse, it is clear that q and q' agree on all contexts of depth at least $k+1$. It remains to prove that $q \sim q'$. By induction using the first item, we can obtain that $\delta(c[q]) = \delta(c[q'])$ for all $c \in \widehat{C}_\Sigma(Q)$ with $\mathrm{dp}(c) \geq k$ because $\sim_0$ is the identity. The second item of Lemma 13 is similarly obtained using both items of this statement, which then proves that $q \sim q'$. □

Corollary 14 allows us to implement the computation of the almost equivalence efficiently. In the conference version [11] we presented an efficient algorithm based on a similar property, but did not prove its correctness. Here we present a reduction

to dfa, for which we know how to compute the almost equivalence efficiently [4, 8]. However, the statement corresponding to Corollary 14 for dfa does not have the stronger condition on kernel contexts (since kernel contexts do not exist if all symbols are at most unary), so our translation into a dfa will include a nonstandard part.

Definition 15. *The reduction dfa* $\mathrm{Red}(M)$ *is the dfa*

$$(Q \cup \overline{\mathrm{Ker}(M)} \cup \{q_0\},\, C_M \cup \Sigma_0 \cup \overline{\mathrm{Ker}(M)},\, q_0,\, \delta',\, F \cup \overline{\mathrm{Ker}(M)}) \ ,$$

such that $\overline{\mathrm{Ker}(M)} = \{\overline{q} \mid q \in \mathrm{Ker}(M)\}$ *is a copy of* $\mathrm{Ker}(M)$ *and*

- $\delta'(q_0, \alpha) = \delta(\alpha)$ *for every* $\alpha \in \Sigma_0$,
- *for every* $q \in Q$ *and* $c \in C_M$, *let*

$$\delta'(q, c) = \begin{cases} \overline{\delta(c[q])} & \text{if } c \text{ is a kernel context} \\ \delta(c[q]) & \text{otherwise} \end{cases}$$

- $\delta'(\overline{q}, \overline{q}) = \overline{q}$ *for every* $q \in \mathrm{Ker}(M)$, *and*
- *all remaining entries in* δ' *are undefined.*

Thus, by definition, the dfa $\mathrm{Red}(M)$ has at most $2 \cdot |Q| + 1$ states and at most $m + |Q|$ transitions. Since M is minimal, we also know that $|Q| \leq m$, which yields that $\mathrm{Red}(M)$ has $\mathcal{O}(|Q|)$ states and $\mathcal{O}(m)$ transitions.

Example 16. *We construct the dfa* $\mathrm{Red}(M_{\mathrm{ex}}) = (Q', \Sigma', q_0, \delta', F')$ *for the dta* M_{ex} *of Example 2. Given* $\Gamma = \{\alpha, \beta, \beta'\}$, *the components are*

- $Q' = \{q_0, \underline{\alpha}, \underline{\beta}, \underline{\beta'}, \underline{\omega}, \overline{\underline{\omega}}\}$,
- $\Sigma' = \{\alpha, \beta, \beta', \overline{\underline{\omega}}, \omega(\underline{\alpha}, \square), \omega(\square, \underline{\omega})\} \cup \{\omega(\square, \underline{\gamma}), \omega(\underline{\gamma}, \square) \mid \gamma \in \Gamma\}$,
- $F' = \{\underline{\beta}, \underline{\omega}, \overline{\underline{\omega}}\}$, *and*
- *the following transitions are in* δ': *for every* $\gamma, \gamma' \in \Gamma$

$$\begin{array}{lll} \delta'(q_0, \gamma) = \underline{\gamma} & \delta'(\underline{\gamma}, \omega(\square, \underline{\gamma'})) = \underline{\omega} & \delta'(\underline{\gamma}, \omega(\underline{\gamma'}, \square)) = \underline{\omega} \\ \delta'(\overline{\underline{\omega}}, \overline{\underline{\omega}}) = \overline{\underline{\omega}} & \delta'(\underline{\omega}, \omega(\underline{\alpha}, \square)) = \underline{\omega} & \delta'(\underline{\alpha}, \omega(\square, \underline{\omega})) = \overline{\underline{\omega}} \ . \end{array}$$

The dfa $\mathrm{Red}(M_{\mathrm{ex}})$ *is illustrated in Fig. 4. It has* 6 *states and* 24 *transitions.*

Lemma 17. $\mathrm{Red}(M)$ *is minimal.*

Proof. The minimality of M immediately yields the minimality of $\mathrm{Red}(M)$ as any difference context also yields a difference string in $\mathrm{Red}(M)$. □

Finally, we need to demonstrate how computing the almost equivalence $\simeq$ for $\mathrm{Red}(M)$ helps us compute the almost equivalence $\sim$ for M. The next theorem shows that both are basically the same. We only need to disregard the additional states introduced in the reduction [i.e., those in $\{q_0\} \cup \overline{\mathrm{Ker}(M)}$].

Theorem 18. *Let* $\simeq$ *be the almost equivalence for* $\mathrm{Red}(M)$. *Then* $\sim \, = \, \simeq \cap Q^2$.

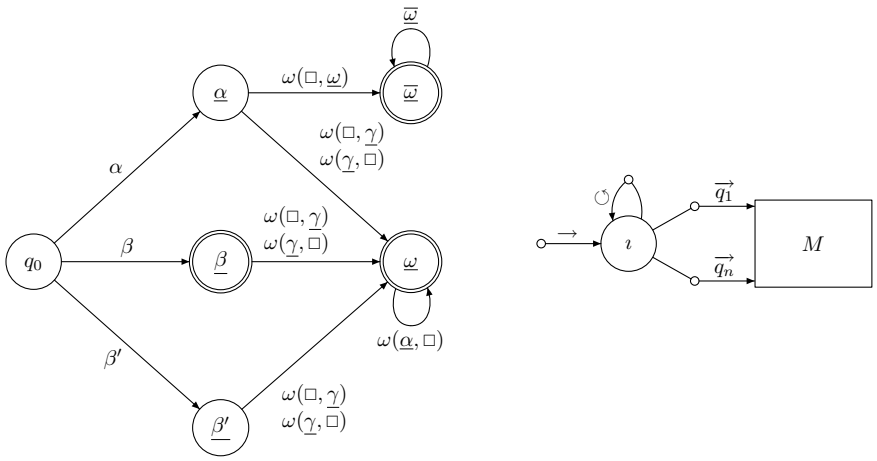

Fig. 4. The dfa $\mathrm{Red}(M_{\mathrm{ex}})$ [left] of Example 16 and an illustration [right] of the reduction discussed in Section 5.

Proof. We prove that $\sim_k = \simeq_k \cap Q^2$ by induction on k. Since both M and $\mathrm{Red}(M)$ are minimal by Lemma 17, we have that $\sim_0 = \{(q, q) \mid q \in Q\} = \simeq_0 \cap Q^2$, which completes the induction base. In the induction step, we suppose that $\sim_k = \simeq_k \cap Q^2$.

Let $q \sim_{k+1} q'$. By Corollary 14 we have $\delta(c[q]) \sim_k \delta(c[q'])$ for every shallow context $c \in C_M$ with equality if c is a kernel context. In the latter case, we obtain $\delta'(q, c) = \overline{\delta(c[q])} = \overline{\delta(c[q'])} = \delta'(q', c)$, so $\delta'(q, c) \simeq_k \delta'(q, c)$. Similarly, for non-kernel contexts c we obtain $\delta'(q, c) = \delta(c[q]) \sim_k \delta(c[q']) = \delta'(q', c)$, which by the induction hypothesis yields $\delta'(q, c) \simeq_k \delta'(q', c)$. For all remaining letters $\sigma \in \Sigma_0 \cup \overline{\mathrm{Ker}(M)}$ we have $\delta'(q, \sigma) = \bot = \delta'(q', \sigma)$, so $\delta'(q, \sigma) \simeq_k \delta'(q', \sigma)$ for every letter $\sigma \in C_M \cup \Sigma_0 \cup \overline{\mathrm{Ker}(M)}$, which proves that $q \simeq_{k+1} q'$.

For the converse, let $q \simeq_{k+1} q'$ with $q, q' \in Q$. Consequently, $\delta'(q, \sigma) \simeq_k \delta'(q, \sigma)$ for all letters $\sigma \in C_M \cup \Sigma_0 \cup \overline{\mathrm{Ker}(M)}$. Let $c \in C_M$ be a shallow context. If c is not a kernel context, then $\delta(c[q]) = \delta'(q, c) \simeq_k \delta'(q', c) = \delta(c[q'])$, which by induction hypothesis yields $\delta(c[q]) \sim_k \delta(c[q'])$. Finally, if c is a kernel context, then $\overline{\delta(c[q])} = \delta'(q, c) \simeq_k \delta'(q', c) = \overline{\delta(c[q'])}$. It is easy to observe that

$$\simeq_k \cap \overline{\mathrm{Ker}(M)} = \{(\overline{q}, \overline{q}) \mid q \in \mathrm{Ker}(M)\} \ .$$

Since $\delta'(q, c) \simeq_k \delta'(q', c)$ and both are in $\overline{\mathrm{Ker}(M)}$, we can conclude that they are equal, which in turn proves that $\delta(c[q]) = \delta(c[q'])$. Thus, we proved both conditions of Corollary 14, which yields $q \sim_{k+1} q'$ and concludes the induction. □

Example 19. *With the existing algorithms we determine the almost equivalence* $\simeq$ *on the dfa* $\mathrm{Red}(M_{\mathrm{ex}})$ *of Example 16, which is displayed in Fig. 4:*

$$\simeq = \{(q, q) \mid q \in Q'\} \cup \{\langle \underline{\beta}, \underline{\beta'} \rangle, \langle \underline{\beta'}, \underline{\beta} \rangle\} \ .$$

Consequently, Theorem 18 yields that $\underline{\beta}$ *and* $\underline{\beta'}$ *are the only different, but almost equivalent states in* M_{ex} *of Example 2.*

We already proved in Corollary 9 that Algorithm 1 is correct. Now we complete its run-time analysis, which leads to our main theorem.

Theorem 20. *Hyper-minimization of M can be performed in time $\mathcal{O}(m \log |Q|)$.*

Proof. Algorithm 1 runs in time $\mathcal{O}(m \log |Q|)$ because we can determine

- the kernel states $\mathrm{Ker}(M)$ in $\mathcal{O}(m)$ by Lemma 10 and
- the almost equivalence $\sim$ using Theorem 18 and the almost equivalence $\simeq$ of $\mathrm{Red}(M)$, which can be computed in time $\mathcal{O}(m' \log n')$ by Theorem 13 of [8], where $m' \in \mathcal{O}(m)$ is the number of transitions of $\mathrm{Red}(M)$ and $n' \in \mathcal{O}(|Q|)$ is the number of states of $\mathrm{Red}(M)$.

Overall, this yields the desired run-time complexity, and the correctness of Algorithm 1 was already established in Corollary 9. □

5. Relation to minimization

In this section, we demonstrate that dta minimization can be reduced in linear time to dta hyper-minimization. For dfa this is achieved [8] with a new distinguished symbol that takes every state back to the initial state, thus making all states kernel states. Since we do not have a single initial state in a dta, we use a slightly different construction. Let $M = (Q, \Sigma, \delta, F)$ be a dta that is not necessarily minimal. For every $\alpha \in \Sigma_0$, let $\overrightarrow{\delta(\alpha)}$ be a new symbol of rank 1. Moreover, we use two new symbols $\rightarrow^{(0)}$ and $\circlearrowleft^{(1)}$, and a new state $\imath \notin Q$ that acts similar to an initial state in a dfa. We construct the dta $M' = (Q \cup \{\imath\}, \Sigma', \delta', F)$, in which

- $\Sigma' = \Sigma \cup \{\overrightarrow{q}^{(1)} \mid \alpha \in \Sigma_0, q = \delta(\alpha)\} \cup \{\rightarrow^{(0)}, \circlearrowleft^{(1)}\}$,
- $\delta'(s) = \delta(s)$ for all $s \in \mathrm{dom}(\delta)$,
- $\delta'(\rightarrow) = \imath$ and $\delta'(\circlearrowleft(\imath)) = \imath$, and
- $\delta'(\overrightarrow{q}(\imath)) = q$ for all $\alpha \in \Sigma_0$ and $q = \delta(\alpha)$.
- All remaining transitions are undefined.

Clearly, M' can be constructed in time $\mathcal{O}(|M|)$. We illustrate the construction in Fig. 4. All reachable states in M' are kernel states. It is easy to see that such a dta is hyper-minimal if and only if it is minimal. Consequently, we can hyper-minimize M' to obtain a minimal dta M'' for the tree language $L(M')$. We can turn this dta M'' into a minimal dta for $L(M)$ by dropping all transitions involving the newly introduced symbols. Thus, we have reduced minimization to hyper-minimization, which shows that the complexity of dta minimization is a lower bound on the complexity of hyper-minimization.

References

[1] A. Badr, Hyper-minimization in $O(n^2)$, *Int. J. Found. Comput. Sci.* **20**(4) (2009) 735–746.

[2] A. Badr, V. Geffert and I. Shipman, Hyper-minimizing minimized deterministic finite state automata, *RAIRO Theor. Inf. Appl.* **43**(1) (2009) 69–94.
[3] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison and M. Tommasi, Tree automata: Techniques and applications Available on: `http://tata.gforge.inria.fr/`, (October 12, 2007).
[4] P. Gawrychowski and A. Jeż, Hyper-minimisation made efficient, *Proc. 34th Int. Symp. Mathematical Foundations of Computer Science*, LNCS **5734**, (Springer, 2009), pp. 356–368.
[5] F. Gécseg and M. Steinby, *Tree Automata* (Akadémiai Kiadó, Budapest, 1984).
[6] F. Gécseg and M. Steinby, Tree languages, *Beyond Words*, eds. G. Rozenberg and A. Salomaa, *Handbook of Formal Languages* **3** (Springer, 1997), pp. 1–68.
[7] J. Högberg, A. Maletti and J. May, Backward and forward bisimulation minimization of tree automata, *Theoret. Comput. Sci.* **410**(37) (2009) 3539–3552.
[8] M. Holzer and A. Maletti, An $n \log n$ algorithm for hyper-minimizing a (minimized) deterministic automaton, *Theoret. Comput. Sci.* **411**(38–39) (2010) 3404–3413.
[9] J. E. Hopcroft, An $n \log n$ algorithm for minimizing states in a finite automaton, *Theory of Machines and Computations*, eds. Z. Kohavi and A. Paz (Academic Press, 1971), pp. 189–196.
[10] H. Hosoya, *Foundations of XML Processing: The Tree-Automata Approach* (Cambridge University Press, 2011).
[11] A. Jeż and A. Maletti, Hyper-minimization for deterministic tree automata, *Proc. 17th Int. Conf. Implementation and Application of Automata*, LNCS **7381**, (Springer, 2012), pp. 217–228.
[12] K. Knight, Capturing practical natural language transformations, *Machine Translation* **21**(2) (2007) 121–133.
[13] K. Knight and J. May, Applications of weighted automata in natural language processing, *Handbook of Weighted Automata*, eds. M. Droste, W. Kuich and H. Vogler, *EATCS Monographs on Theoret. Comput. Sci.* (Springer, 2009), pp. 571–596.
[14] A. Maletti, Minimizing deterministic weighted tree automata, *Inform. and Comput.* **207**(11) (2009) 1284–1299.
[15] A. Maletti, Notes on hyper-minimization, *Proc. 13th Int. Conf. Automata and Formal Languages*, (Nyíregyháza College, 2011), pp. 34–49.
[16] A. Maletti and D. Quernheim, Hyper-minimisation of deterministic weighted finite automata over semifields, *Proc. 13th Int. Conf. Automata and Formal Languages*, (Nyíregyháza College, 2011), pp. 285–299.
[17] M. Mohri, Finite-state transducers in language and speech processing, *Comput. Linguist.* **23**(2) (1997) 269–311.
[18] J. Sakarovitch, Rational and recognisable power series, *Handbook of Weighted Automata*, eds. M. Droste, W. Kuich and H. Vogler, *EATCS Monographs on Theoret. Comput. Sci.* (Springer, 2009), pp. 105–174.
[19] S. Schewe, Beyond hyper-minimisation — Minimising DBAs and DPAs is NP-complete, *Proc. 30th Int. Conf. Foundations of Software Technology and Theoretical Computer Science*, *LIPIcs* **8**, (Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2010), pp. 400–411.
[20] R. E. Tarjan, Depth-first search and linear graph algorithms, *SIAM J. Comput.* **1**(2) (1972) 146–160.
[21] S. Yu, Regular languages, *Word, Language, Grammar*, eds. G. Rozenberg and A. Salomaa, *Handbook of Formal Languages* **1** (Springer, 1997), pp. 41–110.