Hyper-Optimization for Deterministic Tree Automata[☆]

Andreas Maletti[a,1,*]

[a] *Institute for Natural Language Processing, Universität Stuttgart, Pfaffenwaldring 5b, 70569 Stuttgart, Germany*

Abstract

Hyper-minimization is a lossy minimization technique that allows a finite number of errors. It was already demonstrated that hyper-minimization can be performed efficiently for deterministic string automata and (bottom-up) deterministic tree automata (DTAs). The asymptotically fastest DTA hyper-minimization algorithms run in time $O(m \cdot \log n)$, where m is the size of the DTA and n is the number of its states. In this contribution, the committed errors are investigated. First, the structure of all hyper-minimal DTAs for a given tree language is characterized, which also yields a formula for the number of such hyper-minimal DTAs. Second, an algorithm is developed that computes the number of errors that a given hyper-minimal DTA commits when compared to a given reference DTA. Third, it is shown that optimal hyper-minimization (i.e., computing a hyper-minimal DTA that commits the least number of errors of all hyper-minimal DTAs) can be achieved in time $O(m \cdot n)$. Finally, a discussion of various other error measures (besides only their number) is provided.

Keywords: hyper-minimization, deterministic tree automaton, lossy compression, error analysis, error optimization
2010 MSC: 68Q45, 68R05, 68Q25

1. Introduction

In many application areas, large finite-state models are approximated automatically from data. Classical examples in the area of natural language processing include the estimation of probabilistic tree automata [1–3] for parsing [4], probabilistic finite-state automata [5] for speech recognition [6], and probabilistic finite-state transducers [7] for machine translation [8]. Since those models tend to become huge, various efficient minimization techniques (such as the merge-steps in the Berkeley parser training [4] or bisimulation minimization [9, 10]) are used. Unfortunately, already the computation of an equivalent minimal nondeterministic finite-state automaton (NFA) [11] given an input NFA is PSPACE-complete [12] and thus inefficient; this remains true even if the input NFA is deterministic. However, given a deterministic finite-state automaton (DFA) the computation of an equivalent minimal DFA is very efficient [13]. Consequently, we restrict our focus to deterministic finite-state devices. Exactly, the same situation exhibits itself for tree automata [14, 15], which are the finite-state models used in this contribution. In addition, (bottom-up) deterministic tree automata are as expressive as (nondeterministic) tree automata, which recognize exactly the regular tree languages.

In several applications (e.g., digital audio compression) it is beneficial to reduce the size at the expense of errors. This approach is generally called 'lossy compression', and naturally, there are application-specific constraints on the permitted errors. In hyper-minimization [16] we simply allow any finite number of errors; i.e., the obtained representation might recognize a language that has a finite symmetric difference to the language recognized by the original representation. This constraint is rather simplistic, but it allows a convenient theoretical treatment [16] and

[☆] This is an extended and revised version of [A. Maletti: *Hyper-optimization for deterministic tree automata.* Proc. 18th CIAA, LNCS 7982, p. 244–255, Springer 2013].

[*] Corresponding author
Email address: andreas.maletti@ims.uni-stuttgart.de (Andreas Maletti)

[1] The author was financially supported by the German Research Foundation (DFG) grant MA 4959 / 1-1.

efficient minimization algorithms [17–20]. Besides there are models for which finitely many errors are absolutely inconsequential [21]. Moreover, refined constraints often yield NP-hard minimization problems [22] and thus inefficient minimization procedures. Recently, an efficient hyper-minimization algorithm [23] for (bottom-up) deterministic tree automata (DTAs) was developed. It runs in time $O(m \cdot \log n)$, where m is the size of the input DTA and n is the number of its states. Thus, it is asymptotically as efficient as the fastest classical minimization algorithms [10] for DTAs.

All existing hyper-minimization algorithms for DTAs are purely qualitative in the sense that they guarantee that the returned DTA meets the error constraint and is hyper-minimal, which means that all DTAs with strictly fewer states[2] than the returned DTA violate the error constraint (i.e., recognize tree languages that have infinite difference to the tree language recognized by the input DTA). Consequently, the returned DTA is as small as possible subject to our error constraint. However, there is no (non-trivial) bound on the number of errors it might commit. In general, there are many (non-isomorphic) hyper-minimal DTAs for a given tree language. Our first result characterizes the differences between such alternatives in the spirit of [16, Theorems 3.8 and 3.9]. It shows that two such hyper-minimal DTAs permit a bijection between their states such that the distinction into preamble (i.e., those states that can only be reached by finitely many trees) and non-preamble (or kernel) states is preserved. Moreover, the two DTAs behave equivalently on their preambles except for their acceptance decisions and isomorphically on their kernels. In summary, such DTAs can only differ in two aspects:

- the finality (i.e., whether the state is final or not) of preamble states, and
- transition targets for transitions from exclusively preamble states to a kernel state.

Since we can easily count the number of permissible options, our characterization allows us to predict how many such hyper-minimal DTAs exist for the given input DTA.

The main application of the characterization is the calculation of the number of errors committed by a given hyper-minimal DTA inspired by the approach of [24]. In this sense we deliver a much more refined, quantitative analysis of hyper-minimization. To calculate the errors we first associate the errors to the states in which they occur. For preamble states the choice of their finality directly impacts which errors are caused. Due to the close connection between such DTAs we can easily compute the tree language recognized in each preamble state. Consequently, all errors associated to preamble states can be easily explained by the choice of finality. The analysis is slightly more complicated for errors associated to kernel states. Clearly, such error trees must have subtrees in which they took a special transition, which is a transition from exclusively preamble states to a kernel state, at the root. All kernel errors are then attributed to the special transitions. Since an error tree might have several such subtrees, we need to select a unique special transition for each error tree. Otherwise we would count the same error tree several times. We chose to select the left-most special transition, but other selections are possible. Fortunately, the restriction to left-most special transitions can be incorporated easily into the approach of [24]. Overall, we obtain an algorithm that computes the number of errors committed by a hyper-minimal DTA N in comparison to a reference DTA M (that recognizes a finitely different tree language) in time $O(m \cdot n)$, where m is the size of M and n is the number of states of M. Since our association and attribution of errors is such that the choices do not impact another, we can simply optimize each choice such that it causes the least number of errors locally. In this way we obtain a DTA that also globally commits the least number of errors. Thus, we can also compute an optimal hyper-minimal DTA N' in time $O(m \cdot n)$, where 'optimal' means that it commits the least number of errors among all hyper-minimal DTAs that recognize a finitely different tree language. Naturally, we can also compute the exact number of errors committed by this optimal DTA. Finally, we quickly discuss alternative optimality criteria since the number of errors is only one natural criterion. We focus on criteria that score the error trees since this approach meets our requirements. We discuss more realistic scoring functions might add up the error tree sizes or calculate the maximal height of an error tree. Unfortunately, not all such scoring functions can easily be incorporated into our algorithm.

2. Preliminaries

The set $\mathbb{N}$ contains all nonnegative integers, and we let $[k] = \{i \in \mathbb{N} \mid 1 \leq i \leq k\}$ for all $k \in \mathbb{N}$. The symmetric difference $S \ominus T$ of sets S and T is $(S - T) \cup (T - S)$. If a set S is finite, then we write $|S|$ for its cardinality. A binary relation $\cong \subseteq S \times S$ is an equivalence relation if it is reflexive (i.e., $s \cong s$ for all $s \in S$), symmetric (i.e.,

[2]Since we consider only deterministic devices, we can as well use the number of transitions as a size measure.

$s \cong s'$ implies $s' \cong s$ for all $s, s' \in S$), and transitive (i.e., $s \cong s'$ and $s' \cong s''$ imply $s \cong s''$ for all $s, s', s'' \in S$). Given such an equivalence relation $\cong$, the equivalence class $[s]_\cong$ of $s \in S$ is $[s]_\cong = \{s' \in S \mid s \cong s'\}$. Moreover, $(S/{\cong}) = \{[s]_\cong \mid s \in S\}$.

An alphabet Σ is a finite set, of which the elements are usually called symbols. A *ranked alphabet* (Σ, rk) consists of an alphabet Σ and a mapping $\mathrm{rk}\colon \Sigma \to \mathbb{N}$ that assigns a rank to each symbol of Σ. The set of all symbols of rank k is $\Sigma_k = \{\sigma \in \Sigma \mid \mathrm{rk}(\sigma) = k\}$ for every $k \in \mathbb{N}$. We typically denote a ranked alphabet (Σ, rk) by just Σ and assume that the ranks are clear from the context (or irrelevant). Similarly, we often omit universal quantifications like "for all $k \in \mathbb{N}$" especially for ranks. Whenever we want to explicitly indicate the rank of a symbol, we write $\sigma^{(k)}$ for a k-ary symbol σ. For every set T, we let $\Sigma(T) = \{\sigma(t_1, \dots, t_k) \mid \sigma \in \Sigma_k, t_1, \dots, t_k \in T\}$ contain all trees with a root node labeled by a symbol from Σ (used according to its rank) and direct subtrees that are taken from 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 $T_\Sigma(\emptyset)$. Given a unary symbol $\gamma \in \Sigma$, $n \in \mathbb{N}$, and $t \in T_\Sigma(Q)$, we write $\gamma^n(t)$ for the tree consisting of n γ-symbols on top of the tree t. The mapping *height* $\mathrm{ht}(t)\colon T_\Sigma(Q) \to \mathbb{N}$ is defined by $\mathrm{ht}(q) = 0$ for all $q \in Q$ and $\mathrm{ht}(\sigma(t_1, \dots, t_k)) = 1 + \max\{\mathrm{ht}(t_i) \mid i \in [k]\}$ for all $\sigma \in \Sigma_k$ and $t_1, \dots, t_k \in T_\Sigma(Q)$. The subset $C_\Sigma(Q) \subseteq T_{\Sigma \cup \{\square\}}(Q)$ of *contexts* contains all trees in which the special nullary symbol $\square$ occurs exactly once. Again, we write C_Σ for $C_\Sigma(\emptyset)$. For all $c \in C_\Sigma(Q)$ and $t \in T_{\Sigma \cup \{\square\}}(Q)$, we write $c[t]$ for the tree obtained from c by replacing $\square$ by t. The tree t is a *subtree* of $c[t]$ for all contexts $c \in C_\Sigma(Q)$.

A (total bottom-up) *deterministic* (finite-state) *tree automaton* (DTA) [1, 2] is a tuple $M = (Q, \Sigma, \delta, F)$ where Q is the finite, nonempty set of states, Σ is the ranked alphabet of input symbols, $\delta\colon \Sigma(Q) \to Q$ is the *transition mapping*, and $F \subseteq Q$ is the set of final states. The transition mapping δ extends to $\delta\colon T_\Sigma(Q) \to Q$ by $\delta(q) = q$ for all $q \in Q$ and

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

for all $\sigma \in \Sigma_k$ and $t_1, \dots, t_k \in T_\Sigma(Q)$. We let $L(M)^q_{q'} = \{c \in C_\Sigma \mid \delta(c[q']) = q\}$ for all $q, q' \in Q$. Moreover, $L(M)_{q'} = \bigcup_{q \in F} L(M)^q_{q'}$ contains all (stateless) contexts that take q' into a final state, and $L(M)^q = \{t \in T_\Sigma \mid \delta(t) = q\}$ contains all (stateless) trees that are recognized in the state q. A state $q \in Q$ is a *kernel* (respectively, *preamble*) state [16] if $L(M)^q$ is infinite (respectively, finite). The sets $\mathrm{Ker}(M)$ and $\mathrm{Pre}(M)$ contain all kernel and preamble states, respectively. The DTA M recognizes the tree language $L(M) = \bigcup_{q \in F} L(M)^q$, and all DTAs that recognize the same tree language are equivalent. A DTA is *minimal* if there exists no equivalent DTA with strictly fewer states. We can compute a minimal DTA that is equivalent to M using an adaptation [10] of Hopcroft's algorithm [13], which runs in time $O(|M| \cdot \log n)$ where $|M| = \sum_{k \in \mathbb{N}} k \cdot |\Sigma_k| \cdot n^k$ is the size of M and $n = |Q|$.

Example 1. Let us establish a running example. Let M_{ex} be the DTA $M_{\mathrm{ex}} = (Q, \Sigma, \delta, F)$, where

- $Q = \{q_\alpha, q_\beta, q_\eta, q_\gamma, q_\Gamma, q_\sigma, q_\Sigma, \bot\}$ and $F = \{q_\eta, q_\Gamma, q_\Sigma\}$,
- $\Sigma = \{\alpha^{(0)}, \beta^{(0)}, \eta^{(0)}, \gamma^{(1)}, \sigma^{(2)}\}$, and
- δ is defined by[3]

$$\begin{array}{lllll}
\delta(\alpha) = q_\alpha & \delta(\gamma(q_\alpha)) = q_\gamma & \delta(\gamma(q_\gamma)) = q_\Gamma & \delta(\sigma(q, q')) = q_\sigma & \delta(\sigma(q_\alpha, q_\sigma)) = q_\Sigma \\
\delta(\beta) = q_\beta & \delta(\gamma(q_\beta)) = q_\gamma & & & \delta(\sigma(q_\alpha, q_\Sigma)) = q_\Sigma \\
\delta(\eta) = q_\eta & \delta(\gamma(q_\eta)) = q_\gamma & & &
\end{array}$$

for every $q, q' \in Q - \{q_\sigma, q_\Sigma, \bot\}$.

For the sake of compactness, let $\Delta = \Sigma - \{\sigma\}$. The DTA M_{ex} recognizes the tree language

$$L(M_{\mathrm{ex}}) = L \cup \{c^n[\sigma(t_1, t_2)] \mid t_1, t_2 \in T_\Delta, n \geq 1\} \qquad \text{with} \qquad L = T_\Delta - \{\alpha, \beta, \gamma(\alpha), \gamma(\beta), \gamma(\eta)\} \ ,$$

where $c^1 = \sigma(\alpha, \square)$ and $c^{n+1} = c^n[c^1]$ for every $n \geq 1$. We illustrate the shape of the recognized trees in Figure 1. Moreover, we easily observe that $\mathrm{Ker}(M_{\mathrm{ex}}) = \{q_\Gamma, q_\sigma, q_\Sigma, \bot\}$, so all other states are preamble states. More precisely,

$$L(M_{\mathrm{ex}})^{q_\alpha} = \{\alpha\} \qquad L(M_{\mathrm{ex}})^{q_\beta} = \{\beta\} \qquad L(M_{\mathrm{ex}})^{q_\eta} = \{\eta\} \qquad L(M_{\mathrm{ex}})^{q_\gamma} = \{\gamma(\alpha), \gamma(\beta), \gamma(\eta)\} \ .$$

[3]We generally assume that $\bot$ is a sink states, so all transitions that are not explicitly stated return $\bot$.

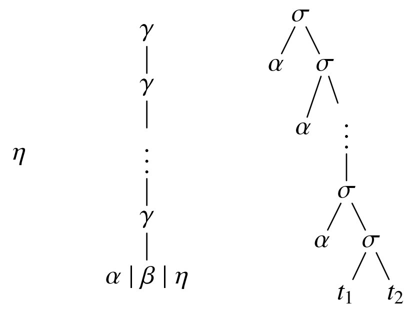


Figure 1: Illustration of the tree language $L(M_{\mathrm{ex}})$. Trees of the second shape need to have at least two γ-symbols and trees of the third shape need to contain at least two σ-symbols. Moreover, t_1 and t_2 are arbitrary trees that do not contain any σ-symbol.

In fact, M_{ex} is minimal, which can be established by exhibiting the context languages $L(M_{\mathrm{ex}})_q$ for all $q \in Q$:

$$\begin{aligned}
L(M_{\mathrm{ex}})_{q_\sigma} &= \{c^n \mid n \geq 1\} & L(M_{\mathrm{ex}})_{q_\Sigma} &= L(M)_{q_\sigma} \cup \{\square\} \\
L(M_{\mathrm{ex}})_{q_\Gamma} &= \{c^n[\sigma(\gamma^m(\square), t)] \mid m \in \mathbb{N}, n \geq 1, t \in T_\Delta\} \cup & L(M_{\mathrm{ex}})_{q_\gamma} &= L(M_{\mathrm{ex}})_{q_\Gamma} - \{\square\} \\
&\quad \cup \{c^n[\sigma(t, \gamma^m(\square))] \mid m \in \mathbb{N}, n \geq 1, t \in T_\Delta\} \cup & L(M_{\mathrm{ex}})_{q_\beta} &= L(M_{\mathrm{ex}})_{q_\Gamma} - \{\square, \gamma(\square)\} \\
&\quad \cup \{\gamma^m(\square) \mid m \in \mathbb{N}\} & L(M_{\mathrm{ex}})_{q_\eta} &= L(M_{\mathrm{ex}})_{q_\Gamma} - \{\gamma(\square)\} \\
L(M_{\mathrm{ex}})_{q_\alpha} &= L(M_{\mathrm{ex}})_{q_\beta} \cup \{c^n[\sigma(\square, t)] \mid n \in \mathbb{N}, t \in L(M_{\mathrm{ex}})\} \enspace ,
\end{aligned}$$

where $c^0 = \square$. Since the context languages are all different, we can conclude that M_{ex} is minimal [15, Section 1.5].

3. Structural characterization of hyper-minimal DTAs

The goal of this section is the development of a characterization of almost equivalent, hyper-minimal DTAs. In the string case, such a characterization was proved in [16, Theorems 3.8 and 3.9]. The characterization will tell us how to modify a given hyper-minimal DTA such that it remains hyper-minimal and its recognized tree language remains almost equivalent to the tree language recognized by the input DTA. Thus, the characterization shows us the alternatives that we need to consider when we search for an *optimal* hyper-minimal DTA, which is the DTA that commits the least number of errors.[4] However, before we start, we recall the basic notions (e.g., *almost equivalence* and *hyper-minimality*).

Throughout the paper, let $M = (Q, \Sigma, \delta, F)$ and $N = (P, \Sigma, \mu, G)$ be minimal DTAs over the same ranked alphabet Σ of input symbols. Since we ultimately want to compare two DTAs, we introduce all basic notions for M and N. However, this does not exclude the case that $M = N$, which we will actually use often. For every $q \in Q$ and $p \in P$, let $E_{q,p} = L(M)_q \ominus L(N)_p$ be the set of errors in the context languages recognized by q and p. The states q and p are *almost equivalent* [23, Definition 1], written $q \sim p$ or $p \sim q$, if $E_{q,p}$ is finite.[5] Naturally, we also say that q and p *disagree* on each element of $E_{q,p}$. If $E_{q,p} = \emptyset$, then q and p are in fact *equivalent*, which is written $q \equiv p$ or $p \equiv q$. It is well-known [15, Section 1.5] that minimal DTAs do not have different, but equivalent states. Correspondingly, the DTAs M and N are *almost equivalent*, also written $M \sim N$, if $E = L(M) \ominus L(N)$ is finite. As before, M and N are said to *disagree* on each element of the error set E.

Example 2. Recall the DTA M_{ex} of Example 1. The states q_β, q_η, q_γ, and q_Γ are obviously almost equivalent (see Example 1 for their context languages). Similarly, we can observe that the state q_α is not almost equivalent to q_Γ because $\{c[\sigma(\square, t)] \mid t \in L(M_{\mathrm{ex}})\} \subseteq L(M_{\mathrm{ex}})_{q_\alpha}$ demonstrates an infinite difference to $L(M_{\mathrm{ex}})_{q_\Gamma}$. In addition, the states q_σ and q_Σ are almost equivalent.

[4]We discuss other optimality criteria briefly in Section 5.

[5]We often consider $\sim$ as an equivalence relation on $Q \cup P$, which then includes pairs of almost equivalent states inside the same DTA.

Lemma 3 (see [23, Lemma 4]). *If $M \sim N$, then $\delta(t) \equiv \mu(t)$ and $\delta(t') \sim \mu(t')$ for all $t, t' \in T_\Sigma$ with $\mathrm{ht}(t) > |Q \times P|$.*[6]

Proof. The property $\delta(t') \sim \mu(t')$ is proven in [23, Lemma 4]. For the equivalence on sufficiently tall trees, we consider the product DTA $M \times N = (Q \times P, \Sigma, \delta \times \mu, F \times G)$, where

$$(\delta \times \mu)(\sigma(\langle q_1, p_1\rangle, \dots, \langle q_k, p_k\rangle)) = \langle \delta(\sigma(q_1, \dots, q_k)), \mu(\sigma(p_1, \dots, p_k))\rangle$$

for all $\sigma \in \Sigma_k$ and $\langle q_1, p_1\rangle, \dots, \langle q_k, p_k\rangle \in Q \times P$. Clearly, $(\delta \times \mu)(t) = \langle \delta(t), \mu(t)\rangle$ for all $t \in T_\Sigma$. If $\mathrm{ht}(t) > |Q \times P|$, then $(\delta \times \mu)(t)$ is a kernel state of $M \times N$ because the tree t can be pumped [1, 2]. For the sake of a contradiction, suppose that $\delta(t) \not\equiv \mu(t)$; i.e., there exists a disagreement context $c \in E_{\delta(t),\mu(t)}$. Since $\langle \delta(t), \mu(t)\rangle$ is a kernel state of $M \times N$, there exist infinitely many $u \in T_\Sigma$ such that $\langle \delta(u), \mu(u)\rangle = \langle \delta(t), \mu(t)\rangle$. However, for each such tree u we have $c[u] \in L(M)$ if and only if $c[u] \notin L(N)$, which yields $c[u] \in E$. Consequently, the set E is infinite, which contradicts the assumption that $M \sim N$. □

Example 4. Recall the DTA $M_{\mathrm{ex}} = (Q, \Sigma, \delta, F)$ of Example 1. In addition, consider the DTA $N_{\mathrm{ex}} = (P, \Sigma, \mu, G)$ given by

- $P = \{p_\alpha, p_\Gamma, p_\sigma, p_\Sigma, \bot\}$ and $G = \{p_\Gamma, p_\Sigma\}$, and
- the transition function μ as specified below.

$$\begin{array}{llll} \mu(\alpha) = p_\alpha & \mu(\gamma(p)) = p_\Gamma & \mu(\sigma(p_\alpha, p_\Gamma)) = p_\sigma & \mu(\sigma(p_\alpha, p_\alpha)) = p_\Sigma \\ \mu(\beta) = p_\Gamma & & \mu(\sigma(p_\Gamma, p)) = p_\sigma & \mu(\sigma(p_\alpha, p')) = p_\Sigma \\ \mu(\eta) = p_\Gamma & & & \end{array}$$

for all $p \in \{p_\alpha, p_\Gamma\}$ and $p' \in \{p_\sigma, p_\Sigma\}$.

It is easily established that M_{ex} and N_{ex} are almost equivalent because $L(N_{\mathrm{ex}}) = L(M_{\mathrm{ex}}) \cup \{\beta, \gamma(\alpha), \gamma(\beta), \gamma(\eta), \sigma(\alpha, \alpha)\}$. Moreover, both DTAs are minimal, so we can apply Lemma 3 to them to obtain that

$$q_\alpha \sim p_\alpha \qquad q_\beta \sim p_\Gamma \qquad q_\eta \sim p_\Gamma \qquad q_\gamma \sim p_\Gamma \qquad q_\Gamma \equiv p_\Gamma \qquad q_\sigma \equiv p_\sigma \qquad q_\Sigma \equiv p_\Sigma \qquad \text{and} \qquad \bot \equiv \bot \ .$$

Lemma 3 shows that almost equivalent DTAs are in almost equivalent states after processing the same (stateless) tree. If the tree is tall, then they are even in equivalent states. Before we proceed with the structural characterization of almost equivalent, hyper-minimal DTAs, we recall the notion of hyper-minimality and the main result used to check whether a given DTA is hyper-minimal. Formally, a DTA is *hyper-minimal* if all almost equivalent DTAs have at least as many states.

Theorem 5 ([23, Theorem 7]). *A minimal DTA is hyper-minimal if and only if all pairs of different, but almost equivalent states consist only of kernel states.*

Using Theorem 5 we can easily conclude that the DTA N_{ex} of Example 4 is in fact hyper-minimal, whereas the DTA M_{ex} of Example 1 is clearly not hyper-minimal. With the main notions established, we can now investigate how almost equivalent, hyper-minimal DTAs differ. The central relating notion is the transition homomorphism. Given a mapping $h\colon Q \to P$ we can extend it to a mapping $h\colon T_\Sigma(Q) \to T_\Sigma(P)$ by $h(\sigma(t_1, \dots, t_k)) = \sigma(h(t_1), \dots, h(t_k))$ for every $\sigma \in \Sigma_k$ and $t_1, \dots, t_k \in T_\Sigma(Q)$. Such a mapping $h\colon Q \to P$ is a *transition homomorphism*[7] if $h(\delta(s)) = \mu(h(s))$ for every $s \in \Sigma(Q)$. Transition homomorphisms are relations between states of two DTA that are preserved by taking transitions. Moreover, the mapping h is a *DTA homomorphism* if additionally $h(q) \in G$ if and only if $q \in F$. Thus, DTA homomorphisms additionally need to preserve finality of the states. As usual, a bijective homomorphism is called *isomorphism*. Next, we show that two almost equivalent hyper-minimal DTAs have DTA-isomorphic kernels and transition-isomorphic preambles.

Theorem 6. *If $M \sim N$ and both M and N are hyper-minimal, then there exists a bijection $h\colon Q \to P$ such that*

[6]If $M = N$, then $\delta = \mu$, so in particular $\delta(t) = \mu(t)$ for all $t \in T_\Sigma$. In this case, $\mathrm{ht}(t) > |Q|$ is actually sufficient.

[7]or a homomorphism between the Σ-algebras [1, 2] associated with M and N

1. *h is bijective on* $\mathrm{Ker}(M) \times \mathrm{Ker}(N)$*,*
2. *$h(q) \in G$ if and only if $q \in F$ for all $q \in$* $\mathrm{Ker}(M)$*, and*
3. *$h(\delta(s)) = \mu(h(s))$ for every $s \in \Sigma(Q) - \{s \in \Sigma(\mathrm{Pre}(M)) \mid \delta(s) \in \mathrm{Ker}(M)\}$.*

Proof. Clearly, $|Q| = |P|$ since M and N are both hyper-minimal. For every $q \in Q$ we select $t_q \in L(M)^q$ such that $\mathrm{ht}(t_q) > |Q \times P|$ whenever $q \in \mathrm{Ker}(M)$.[8] In other words, we select an access tree for each state of Q such that access trees for the kernel states $\mathrm{Ker}(M)$ are tall. The bijection will be established by evaluating the access trees in N. Formally, we let $h\colon Q \to P$ be such that $h(q) = \mu(t_q)$ for every $q \in Q$, which immediately proves that $h(q) \in \mathrm{Ker}(N)$ for all $q \in \mathrm{Ker}(M)$ because $\mathrm{ht}(t_q) > |Q \times P|$. Moreover, for each $q \in \mathrm{Ker}(M)$ the facts $M \sim N$ and $\mathrm{ht}(t_q) > |Q \times P|$ imply $q = \delta(t_q) \equiv \mu(t_q) = h(q)$ by Lemma 3. Thus, h is injective on $\mathrm{Ker}(M) \times \mathrm{Ker}(N)$ because $h(q_1) \equiv q_1 \not\equiv q_2 \equiv h(q_2)$ for all different $q_1, q_2 \in \mathrm{Ker}(M)$.[9] Finally, for every $p \in \mathrm{Ker}(N)$, select $u_p \in L(N)^p$ such that $\mathrm{ht}(u_p) > |Q \times P|$. Clearly, $\delta(u_p) \in \mathrm{Ker}(M)$ and by Lemma 3 we obtain $p = \mu(u_p) \equiv \delta(u_p) \equiv h(\delta(u_p))$. Since N is minimal, we can conclude that $p = h(\delta(u_p))$, which shows that h is surjective on $\mathrm{Ker}(M) \times \mathrm{Ker}(N)$, thereby proving the first item.

Recall from the previous paragraph that $q \equiv h(q)$ for every $q \in \mathrm{Ker}(M)$. Thus, $h(q) \in G$ if and only if $q \in F$, which proves the second item. For the third objective, let $s = \sigma(q_1, \dots, q_k) \in \Sigma(Q)$. Then

$$\delta(s) = \delta(\sigma(t_{q_1}, \dots, t_{q_k})) \overset{\dagger}{\sim} \mu(\sigma(t_{q_1}, \dots, t_{q_k})) = \mu(\sigma(\mu(t_{q_1}), \dots, \mu(t_{q_k}))) = \mu(\sigma(h(q_1), \dots, h(q_k))) = \mu(h(s)) \enspace ,$$

where the step marked † is due to Lemma 3. In the following, assume that $s \notin \Sigma(\mathrm{Pre}(M))$, which yields that there exists $i \in [k]$ such that $q_i \in \mathrm{Ker}(M)$. Consequently, $\mathrm{ht}(\sigma(t_{q_1}, \dots, t_{q_k})) > |Q \times P|$ by the selection of t_{q_i}, which can be used in Lemma 3 to show that the step marked † is actually equivalence ($\equiv$). We obtain that $\delta(s) \equiv \mu(h(s))$ and $\delta(s) \in \mathrm{Ker}(M)$. Thus, $h(\delta(s)) \equiv \delta(s) \equiv \mu(h(s))$ by the argument in the previous paragraph. Since N is minimal, we can conclude that $h(\delta(s)) = \mu(h(s))$ as desired.

Before we prove the missing case, in which $s \in \Sigma(\mathrm{Pre}(M))$ with $\delta(s) \in \mathrm{Pre}(M)$, we prove that h is bijective on $\mathrm{Pre}(M) \times \mathrm{Pre}(N)$, which automatically also proves that $h\colon Q \to P$ is a bijection. Since h is bijective on $\mathrm{Ker}(M) \times \mathrm{Ker}(N)$ by the proven first item, which yields $|\mathrm{Ker}(M)| = |\mathrm{Ker}(N)|$, and additionally $|Q| = |P|$, we obtain that $|\mathrm{Pre}(M)| = |\mathrm{Pre}(N)|$. Suppose that $h(q) \in \mathrm{Ker}(N)$ for some $q \in \mathrm{Pre}(M)$. Then $q \sim h(q) = \mu(u_{h(q)}) \sim \delta(u_{h(q)})$ with $\mathrm{ht}(u_{h(q)}) > |Q \times P|$ because $h(q) \in \mathrm{Ker}(N)$, where the almost equivalences are due to Lemma 3. Clearly, $\delta(u_{h(q)})$ is a kernel state of M, which yields that $q \neq \delta(u_{h(q)})$. Together with $q \sim \delta(u_{h(q)})$ and $q \in \mathrm{Pre}(M)$, these facts contradict the hyper-minimality of M by Theorem 5. It remains to prove that h is injective on $\mathrm{Pre}(M) \times \mathrm{Pre}(N)$, which due to $|\mathrm{Pre}(M)| = |\mathrm{Pre}(N)|$ also proves that h is surjective. For the sake of a contradiction, let $q_1, q_2 \in \mathrm{Pre}(M)$ be such that $q_1 \neq q_2$ but $h(q_1) = h(q_2)$. Using Lemma 3 we obtain $q_1 \sim h(q_1) = h(q_2) \sim q_2$, which together with $q_1 \neq q_2$ contradicts the hyper-minimality of M by Theorem 5. Consequently, h is bijective on $\mathrm{Pre}(M) \times \mathrm{Pre}(N)$.

Now we return to the final missing objective, which requires us to show that $h(\delta(s)) = \mu(h(s))$ if $\delta(s) \in \mathrm{Pre}(M)$. Recall that $\delta(s) \sim \mu(h(s))$ for all $s \in \Sigma(Q)$. Moreover, if $\delta(s) \in \mathrm{Pre}(M)$, then $h(\delta(s)) \in \mathrm{Pre}(N)$ by the results of the previous paragraph and additionally $h(\delta(s)) \sim \delta(s) \sim \mu(h(s))$ by Lemma 3. Consequently, we have a preamble state $h(\delta(s))$ of N that is almost equivalent to $\mu(h(s))$. Since N is hyper-minimal, we have $h(\delta(s)) = \mu(h(s))$ by Theorem 5. □

The previous theorem states that two almost equivalent, hyper-minimal DTAs are indeed very similar. They have a bijection between their states that preserves the distinction between preamble and kernel states. Moreover, via this bijection the two DTAs behave equally (besides acceptance) on the preamble states and isomorphically on the kernel states. Thus, two such DTAs can only differ in two aspects, which mirror the corresponding aspects for deterministic finite-state string automata [16]:

1. the finality (i.e., whether the state is final or not) of preamble states, and
2. transitions from exclusively preamble states to a kernel state.

This is slightly surprising in light of the changes required to generalize the hyper-minimization algorithm [19] from deterministic finite-state string automata to DTAs [23]. Several key statements required for hyper-minimization had to be adjusted in [23] because their natural generalizations do not hold for DTAs. However, Theorem 6 is the natural

[8]Such trees exist because each state is reachable (by hyper-minimality) and $L(M)^q$ is infinite for each kernel state q.

[9]We have $q_1 \not\equiv q_2$ because M is minimal.

generalization of the corresponding characterization of [16]. For a detailed discussion of the required adjustments, we refer the reader to [23].

Example 7. Let us illustrate Theorem 6 using the DTA N_{ex} of Example 4. We already remarked that N_{ex} is hyper-minimal. We can immediately conclude that each almost equivalent, hyper-minimal DTA M must also have 4 kernel states (corresponding to the 4 kernel states $\{p_\Gamma, p_\sigma, p_\Sigma, \bot\}$ of N_{ex}) and 1 preamble state q (corresponding to the preamble state p_α of N_{ex}). Due to the transition homomorphism property and the DTA isomorphism on the kernels, we immediately have that M has exactly the same structure as N_{ex} except for the following:

1. its preamble state q might be final, and
2. the transition targets $\delta(\gamma(\alpha))$ and $\delta(\sigma(\alpha, \alpha))$ need not coincide with the corresponding transition targets in μ.

However, by Lemma 3 we know that $\delta(\gamma(\alpha)) \sim \mu(\gamma(\alpha)) = p_\Gamma \in \text{Ker}(N_{\text{ex}})$. By Theorem 6 for each of the 4 kernel states of N_{ex} there is an equivalent kernel state of M (and vice versa), so in particular, there exists a state $q' \equiv p_\Gamma$. None of the other kernel states is almost equivalent to p_Γ, so the transition target can only be q' or the preamble state q. However, if the transition target is q, then q and q' are almost equivalent, which contradicts the hyper-minimality of M by Theorem 5 because $q \in \text{Pre}(M)$. Thus, we have $\delta(\gamma(\alpha)) = q'$. In summary, the structure of M must exactly match that of N_{ex} except for (i) the finality of q, and (ii) the transition target $\delta(\sigma(\alpha, \alpha))$, which can either be the state that is equivalent to p_σ or the state that is equivalent to p_Σ. Consequently, for the tree language $L(N_{\text{ex}})$ and thus also for the tree language $L(M_{\text{ex}})$ of the DTA M_{ex} of Example 1 there are only 4 almost equivalent, hyper-minimal DTAs (up to isomorphism).

We end this section by showing that the characterization trivially allows us to compute the number of almost equivalent, hyper-minimal DTAs for the given minimal DTA M. To this end, we simply compute one almost equivalent, hyper-minimal DTA N using the algorithms of [23] (see Algorithm 1). Then we can easily count the number of variations of the finality of preamble states of N and the target variations for the transitions from exclusively preamble states to a kernel state of N. In fact, once we know the almost equivalence $\sim$ for M we can easily predict the hyper-minimal DTA N (see Algorithm 1), so we can even compute the number based on M and $\sim$ alone.

Corollary 8 (of Theorem 6 and Lemma 3). *For every minimal DTA M there are exactly $2^{|\mathcal{B}|} \cdot g$ almost equivalent, hyper-minimal DTAs, where*

- $\mathcal{B} = \{B \in (Q/{\sim}) \mid B \subseteq \text{Pre}(M)\}$ *is the set of exclusively preamble state blocks, and*
- *g is the number of kernel state alternatives for the transitions from exclusively preamble states to a kernel state; i.e.*

$$g = \prod_{\substack{\sigma \in \Sigma_k, [q_1]_\sim, \dots, [q_k]_\sim \in \mathcal{B} \\ [\delta(\sigma(q_1, \dots, q_k))]_\sim \notin \mathcal{B}}} |[\delta(\sigma(q_1, \dots, q_k))]_\sim \cap \text{Ker}(M)| \enspace .$$

Example 9. Recall the DTA M_{ex} of Example 1. Since we only have one block $[q_\alpha]_\sim = \{q_\alpha\}$ of exclusively preamble states, we have $\mathcal{B} = \{\{q_\alpha\}\}$ and thus $|\mathcal{B}| = 1$. In addition, we have to consider the transitions β, η, $\gamma(q_\alpha)$, and $\sigma(q_\alpha, q_\alpha)$ for the second item in Corollary 8. However, $[\delta(\beta)]_\sim = [\delta(\eta)]_\sim = [\delta(\gamma(q_\alpha))]_\sim = \{q_\beta, q_\eta, q_\gamma, q_\Gamma\}$, which contains only the kernel state q_Γ, so those transitions give factors 1. For the transition $\sigma(q_\alpha, q_\alpha)$ we obtain $[\delta(\sigma(q_\alpha, q_\alpha))]_\sim = \{q_\sigma, q_\Sigma\}$, which contains two kernel states, so we obtain the factor 2. Consequently, there are $2^1 \cdot 2 = 4$ hyper-minimal DTAs that are almost equivalent to M_{ex} according to Corollary 8.

A reviewer of the conference version [25] suggested to consider those languages $L \subseteq T_\Sigma$, for which the minimal DTA is already hyper-minimal. Since the minimal DTA is a faithful representation of L, it commits no errors and is thus automatically optimal (for most sensible measures). In other words, the reviewer asked to consider the tree languages recognized by hyper-minimal DTAs. Clearly, such tree languages exist (e.g., T_Σ is such a tree language), and in the string case such languages are called *canonical regular languages* [16]. They were investigated a bit in [26, Section 5] and those (rather negative) results all naturally translate to DTAs.

4. Computing the number of errors

Now we want to compute the number of errors made by a particular hyper-minimal DTA N that is almost equivalent to the reference DTA M. Once we solve this problem, we can hope to use the characterization of Theorem 6 to obtain a hyper-minimal DTA that commits the least number of errors among all almost equivalent hyper-minimal DTAs. Such a DTA is also called a *hyper-optimal* DTA for M, and the problem of computing such a DTA is called *hyper-optimization*. It is shown in [27] how to reduce DTA hyper-minimization to hyper-minimization for deterministic finite-state string automata (DFAs), for which efficient algorithms exist [18, 19]. Similarly, efficient DFA hyper-optimization algorithms exist [24]. However, the reduction mentioned in [27] fails to work for hyper-optimization. Moreover, to the author's knowledge we can only use the existing hyper-optimization for DFAs using a reduction that constructs a huge DFA for the given DTA, which naturally hurts the efficiency.[10] It is possible to adjust the existing hyper-optimization for DFAs such that an efficient reduction is possible, but this would require us to prove that the adjustment remains correct. For these reasons, we will not attempt a reduction to the DFA case here.

For the rest of the section, let N be hyper-minimal and almost equivalent to M, which itself is not necessarily hyper-minimal. Recall that $E = L(M) \ominus L(N)$ is the set of error trees. Our goal is to determine the cardinality of E efficiently. To this end, we first partition E into $E = \bigcup_{p \in P} E_p$, where $E_p = L(N)^p \cap E$ for every $p \in P$. In other words, we associate each error tree $t \in E$ with the state $\mu(t)$. In the following development, we distinguish errors associated to preamble and kernel states of N. Theorem 6 shows that the preamble-kernel error distinction is stable among all almost equivalent, hyper-minimal DTAs.[11] Finally, [23, Section 4] shows how to obtain one hyper-minimal DTA N' that is almost equivalent to M (see Algorithm 1). Roughly speaking, we identify the almost equivalence $\sim$ on M and then merge preamble states into almost equivalent states. Formally, for every two different states $q, q' \in Q$, the DTA $\mathrm{merge}(M, q \to q')$ is $(Q - \{q\}, \Sigma, \delta', F - \{q\})$ where $\delta'(s) = q'$ if $\delta(s) = q$ and $\delta'(s) = \delta(s)$ otherwise for every $s \in \Sigma(Q - \{q\})$.

Example 10. Let us reconsider the DTA M_{ex} of Example 1. We already remarked in Example 2 that q_β, q_η, q_γ, and q_Γ are almost equivalent. Since only q_Γ is a kernel state, we would merge q_β, q_η, and q_γ into q_Γ according to Algorithm 1. Note that we cannot merge q_σ and q_Σ since both are kernel states. In fact, after those merges we obtain the DTA $M' = (Q', \Sigma, \delta', G')$ given by

- $Q' = \{q_\alpha, q_\Gamma, q_\sigma, q_\Sigma, \bot\}$ and $G' = \{q_\Gamma, q_\Sigma\}$, and
- the transition function δ' as specified below.

$$\begin{array}{llll}
\delta'(\alpha) = q_\alpha & \delta'(\gamma(q)) = q_\Gamma & \delta'(\sigma(q, q')) = q_\sigma & \delta'(\sigma(q_\alpha, q_\sigma)) = q_\Sigma \\
\delta'(\beta) = q_\Gamma & & & \delta'(\sigma(q_\alpha, q_\Sigma)) = q_\Sigma \\
\delta'(\eta) = q_\Gamma & & &
\end{array}$$

for all $q, q' \in \{q_\alpha, q_\Gamma\}$.

We already know from Example 4 that $|L(N_{\mathrm{ex}}) \ominus L(M_{\mathrm{ex}})| = \{\beta, \gamma(\alpha), \gamma(\beta), \gamma(\eta), \sigma(\alpha, \alpha)\}$, so N_{ex} commits 5 errors. However, $|L(M') \ominus L(M_{\mathrm{ex}})| = \{\beta, \gamma(\alpha), \gamma(\beta), \gamma(\eta)\}$, so it only commits 4 errors (with the same number of states as N_{ex}) and would thus be preferred over N_{ex}. In fact, as we will see later, M' is hyper-optimal.

4.1. *Errors recognized in preamble states*

We start with the errors E_p associated to a preamble state $p \in \mathrm{Pre}(N)$. Let N' be another almost equivalent, hyper-minimal DTA. In our application the DTA N' will be obtained from Algorithm 1 applied to M and $\sim$. Since the preambles of N and N' are transition-isomorphic by Theorem 6, we can essentially compute with N' and only need to remember that the preamble states of N and N' can differ in finality.

Lemma 11. *Let $N' = \mathrm{merge}(M, q \to q')$ for some $q \sim q'$ with $[q]_\sim \subseteq \mathrm{Pre}(M)$. Then $L(N')^{q'} = L(M)^q \cup L(M)^{q'}$. If N'' is the DTA returned by Algorithm 1 applied to M and $\sim$ and $B \in (Q/{\sim})$ is such that $B \subseteq \mathrm{Pre}(M)$, for which the state q_B was selected in Algorithm 1, then $L(N'')^{q_B} = \bigcup_{q \in B} L(M)^q$.*

Proof. Clearly, $L(M)^q \subseteq L(N')^{q'}$ because the preamble state q is reached only at the root in all trees of $L(M)^q$. To show that $L(M)^{q'} \subseteq L(N')^{q'}$, we first show that no subtree of a tree of $L(M)^{q'}$ can be processed in q. For a contradiction,

[10] In fact, the author believes that the constructed DFA might even be exponentially larger than the given DTA.

[11] An error associated to a preamble state of N can only be associated to a preamble state of another almost equivalent hyper-minimal DTA N'. Naturally, the error can be avoided in N', but the same error cannot be associated to a kernel state of N'.

Algorithm 1 Merging step in the DTA hyper-minimization algorithm [23].

```
Require: a minimal DTA $M$ and its almost equivalence $\sim \subseteq Q \times Q$
Return: an almost equivalent, hyper-minimal DTA
    for all $B \in (Q/{\sim})$ do
2:      select $q_B \in B$ such that $q_B \in \mathrm{Ker}(M)$ if possible
        for all $q \in B - \mathrm{Ker}(M)$ do
4:          $M \leftarrow \mathrm{merge}(M, q \to q_B)$
    return $M$
```

suppose that there exists $c \in L(M)_q^{q'}$; i.e., $q' = \delta(c[q])$. Then $q \sim q' = \delta(c[q]) \sim \delta(c[q']) = \delta(c^2[q]) \sim \delta(c^2[q']) = \cdots$ by [23, Lemma 3], and thus, $q \sim q''$ for some $q'' \in \mathrm{Ker}(M)$, which contradicts the assumption $[q]_\sim \subseteq \mathrm{Pre}(M)$. Consequently, $L(M)_q^{q'} = \emptyset$, which also proves that $L(M)^{q'} \subseteq L(N')^{q'}$. The inclusion $L(N')^{q'} \subseteq L(M)^q \cup L(M)^{q'}$ is easy to establish. □

Example 12. If we apply the second part of Lemma 11 to the DTAs M_{ex} and N_{ex} of Examples 1 and 4, respectively, then we obtain that $L(N_{\mathrm{ex}})^{p_\alpha} = L(M_{\mathrm{ex}})^{q_\alpha} = \{\alpha\}$. Note that we cannot compute $L(N_{\mathrm{ex}})^{p_\Gamma}$ using Lemma 11 because $\{q_\beta, q_\eta, q_\gamma, q_\Gamma\} \nsubseteq \mathrm{Pre}(M_{\mathrm{ex}})$.

The next lemma shows that the tree languages recognized in corresponding preamble states of two almost equivalent, hyper-minimal DTAs are always equal. Together with Lemma 11 we thus know exactly which trees are recognized in the preamble states $\mathrm{Pre}(N)$ of N.

Lemma 13. *Let $N = (P, \Sigma, \mu, G)$ and $N' = (P', \Sigma, \mu', G')$ be almost equivalent, hyper-minimal DTAs. Moreover, let $h\colon P \to P'$ be the bijection of Theorem 6. Then $L(N)^p = L(N')^{h(p)}$ for all $p \in P$.*

Proof. Let us first prove the inclusion $L(N)^p \subseteq L(N')^{h(p)}$ for all $p \in P$ by induction. Suppose that $\sigma(t_1, \dots, t_k) \in L(N)^p$ for some $\sigma \in \Sigma_k$ and $t_1, \dots, t_k \in T_\Sigma$. Moreover, let $p_i = \mu(t_i)$ for every $i \in [k]$. By the induction hypothesis we know that $t_i \in L(N')^{h(p_i)}$ for all $i \in [k]$. In addition, we have $\mu(\sigma(p_1, \dots, p_k)) = p$ and due to the transition homomorphism property of h we also have $h(p) = \mu'(\sigma(h(p_1), \dots, h(p_k)))$. Consequently,

$$\mu'(\sigma(t_1, \dots, t_k)) = \mu'(\sigma(\mu'(t_1), \dots, \mu'(t_k))) = \mu'(\sigma(h(p_1), \dots, h(p_k))) = h(p) \enspace ,$$

as desired. The converse inclusion can simply be proven by exchanging the roles of N and N'. □

Now let N' be the DTA returned by Algorithm 1 when applied to M and $\sim$. By Theorem 6 there exists a mapping $h\colon P' \to P$ such that N' and N are transition-isomorphic on their preambles via h. Putting Lemmata 11 and 13 together we have

$$L(N)^{h(q_B)} = L(N')^{q_B} = \bigcup_{q \in B} L(M)^q \tag{‡}$$

for every $B \in (Q/{\sim})$ with $B \subseteq \mathrm{Pre}(M)$,[12] where q_B is the state selected for block B in Algorithm 1. Thus, we have now expressed the recognized tree languages for all preamble states of N in terms of unions of the recognized tree languages of certain preamble states of M. To compute the number of errors E_p we still need to compute which trees in $L(N)^p$ are now incorrectly classified. We first demonstrate how to compute the number of recognized trees for each preamble state of M. To simplify the notation, let $a_q = |L(M)^q|$ for every $q \in \mathrm{Pre}(M)$, so we need to compute $|L(M)^q|$ for each state $q \in \mathrm{Pre}(M)$.

Lemma 14. *For every $q \in \mathrm{Pre}(M)$*

$$a_q = \sum_{\substack{\sigma \in \Sigma_k, q_1, \dots, q_k \in Q \\ \delta(\sigma(q_1, \dots, q_k)) = q}} (a_{q_1} \cdot \ldots \cdot a_{q_k}) \enspace .$$

Algorithm 2 Implementation of the counting argument of Lemma 14.

```
Require: a minimal DTA $M$
Return: a vector $a\colon Q \to \mathbb{N}$ such that $a(q) = a_q = |L(M)^q|$ for every $q \in \mathrm{Pre}(M)$
    for all $q \in Q$ do
2:      $a(q) \leftarrow 0$                                  // initialize vector to $\vec{0}$
    for all $q \in \mathrm{Pre}(M)$ do
4:      COMPUTE$(a, q)$       // compute the entry $a(q)$; the vector $a$ is a reference here
    return $a$

6:  COMPUTE$(a, q)$:          // function COMPUTE fills the vector $a$ and returns $a(q)$
    if $a(q) = 0$ then
8:      $a(q) \leftarrow \sum_{\substack{\sigma \in \Sigma_k, q_1, \dots, q_k \in Q \\ \delta(\sigma(q_1, \dots, q_k)) = q}} \mathrm{COMPUTE}(a, q_1) \cdot \ldots \cdot \mathrm{COMPUTE}(a, q_k)$   // use formula of Lemma 14, store value
    return $a(q)$             // and return it (or just return value if already computed)
```

It is clear that the equations in Lemma 14 yield a linear-time algorithm (i.e., a $O(|M|)$ time algorithm) if we avoid the recomputation of already computed values. A straightforward implementation is shown in Algorithm 2.

Example 15. For our example DTA M_{ex} of Example 1 we obtain

$$|L(M_{\text{ex}})^{q_\alpha}| = a_{q_\alpha} = 1 \qquad |L(M_{\text{ex}})^{q_\beta}| = a_{q_\beta} = 1 \qquad |L(M_{\text{ex}})^{q_\eta}| = a_{q_\eta} = 1 \qquad \text{and} \qquad |L(M_{\text{ex}})^{q_\gamma}| = a_{q_\gamma} = 3 \enspace .$$

With the help of (‡) we can now compute the number of errors made due to the finality of preamble states. For every $p \in \mathrm{Pre}(N)$ there exists a preamble state $q_B \in \mathrm{Pre}(N')$ by Theorem 6, for which we know that it was selected in Algorithm 1 for a block $B \subseteq \mathrm{Pre}(M)$ of exclusively preamble states of M. Consequently, Lemma 11 can be applied to compute the number of errors associated to p.

Theorem 16. *For every $p \in \mathrm{Pre}(N)$, let $u_p \in L(N)^p$ be arbitrary. Then*

$$|E_p| = \begin{cases} \sum_{q \in [\delta(u_p)]_\sim - F} a_q & \text{if } p \in G \\ \sum_{q \in [\delta(u_p)]_\sim \cap F} a_q & \text{otherwise.} \end{cases}$$

Proof. By definition we have $E_p = L(N)^p \cap E = L(N)^p \cap (L(M) \ominus L(N)) = (L(N)^p \cap L(M)) \ominus (L(N)^p \cap L(N))$ and $L(N)^p = \bigcup_{q \in [\delta(u_p)]_\sim} L(M)^q$ by Theorem 6 and (‡). Thus, if $p \in G$, then $L(N)^p \subseteq L(N)$ and

$$E_p = \Bigl(\bigcup_{q \in [\delta(u_p)]_\sim \cap F} L(M)^q \Bigr) \ominus \Bigl(\bigcup_{q \in [\delta(u_p)]_\sim} L(M)^q \Bigr) = \bigcup_{q \in [\delta(u_p)]_\sim - F} L(M)^q \enspace .$$

Since the sets $L(M)^q$ with $q \in Q$ are pairwise disjoint, we can simply add their cardinalities, and thus we proved the first case. On the other hand, if $p \notin G$, then $L(N)^p \cap L(N) = \emptyset$ and

$$E_p = \Bigl(\bigcup_{q \in [\delta(u_p)]_\sim \cap F} L(M)^q \Bigr) \ominus \emptyset = \bigcup_{q \in [\delta(u_p)]_\sim \cap F} L(M)^q \enspace ,$$

which proves the statement. □

Since $L(N)^p$ and $L(N)^{p'}$ are disjoint for two different states $p \neq p'$, the total number of errors associated to preamble states is $\sum_{p \in \mathrm{Pre}(N)} |E_p|$. To obtain the minimal number of errors, we simply select the finality of p such that $|E_p|$ is minimal (see Algorithm 4).[13]

[12]The union is actually disjoint as $T_\Sigma = \bigcup_{q \in Q} L(M)^q$ is a partition of T_Σ.

[13]Clearly, the DTA remains hyper-minimal and almost equivalent as only a finite number of errors is introduced by making p final or non-final.

Example 17. The hyper-minimal DTA N_{ex} of Example 4 only has the preamble state p_α, which corresponds to the block $[q_\alpha]_\sim \subseteq \text{Pre}(M)$. Correspondingly, $|E_{p_\alpha}| = 0$ because $[q_\alpha]_\sim \cap F = \emptyset$. If p_α would be in G (i.e., a final state in N_{ex}), then $|E_{p_\alpha}|$ would be 1 because $[q_\alpha]_\sim - F = \{q_\alpha\}$ and $a_{q_\alpha} = |L(M)^{q_\alpha}| = 1$ by Example 15. Consequently, having p_α as a non-final state already represents the optimal choice (with respect to the number of caused errors).

4.2. Errors recognized in kernel states

It remains to compute the number $\sum_{p\in\text{Ker}(N)}|E_p|$ of errors associated to kernel states of N. Recall that N' is the hyper-minimal DTA returned by Algorithm 1 applied to the reference DTA M and its almost equivalence $\sim$. In addition, the DTA N is an arbitrary almost equivalent, hyper-minimal DTA, for which we want to compute the number of committed errors in comparison to M. Theorem 6 shows that the preambles of N and N' are transition-isomorphic and the kernels are DTA-isomorphic, but the transitions from exclusively preamble states to a kernel state are not covered in this characterization. Thus, for a given kernel state $p \in \text{Ker}(N)$, it is difficult to express $L(N)^p$ in terms of $(L(M)^q)_{q\in Q}$ because we lose the tight connection at the transitions from exclusively preamble states to kernel states. Consequently, instead of assigning the errors made in kernel states to the kernel states themselves, we will attribute them to the transitions from exclusively preamble states to a kernel state. This approach works because

- we know what happens before taking such a transition as we know $L(N)^p$ for each $p \in \text{Pre}(N)$, and
- we know what happens after taking such a transition as we have a DTA-isomorphism on the kernels of N and N'.

This approach was also successful in the string case [24], however in the tree case it is complicated by the fact that a single error tree can use such exclusively-preamble-states-to-kernel-state-transitions, also called *special transitions*, several times as the next example demonstrates.

Example 18. Let us again consider the DTA N_{ex} of Example 4. We know that it commits the error $\sigma(\alpha, \alpha)$ in the kernel state q_Σ. According to our strategy of attributing kernel state errors to special transitions, we would attribute this error to the transition $\mu(\sigma(p_\alpha, p_\alpha)) = q_\Sigma$ because this transition is used at the root of $\sigma(\alpha, \alpha)$ and uses only the preamble state p_α as source and the kernel state q_Σ as target.

In general, the attribution is not that straightforward. To see this, let us consider the DTA $M = (Q, \Sigma, \delta, \{q\})$ with $Q = \{q, q_\beta, q_{\sigma(\beta,\beta)}\}$, $\Sigma = \{\alpha^{(0)}, \beta^{(0)}, \sigma^{(2)}\}$, and

$$\begin{array}{cccc} \delta(\alpha) = q & \delta(\sigma(q,q)) = q & \delta(\sigma(q,q_\beta)) = q & \delta(\sigma(q,q_{\sigma(\beta,\beta)})) = q \\ \delta(\beta) = q_\beta & \delta(\sigma(q_\beta,q)) = q & \delta(\sigma(q_\beta,q_\beta)) = q_{\sigma(\beta,\beta)} & \delta(\sigma(q_\beta,q_{\sigma(\beta,\beta)})) = q \\ & \delta(\sigma(q_{\sigma(\beta,\beta)},q)) = q & \delta(\sigma(q_{\sigma(\beta,\beta)},q_\beta)) = q & \delta(\sigma(q_{\sigma(\beta,\beta)},q_{\sigma(\beta,\beta)})) = q \ . \end{array}$$

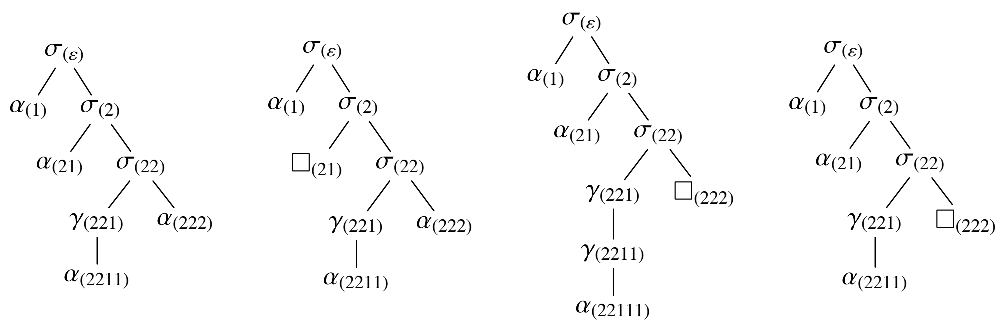
Obviously, $L(M) = T_\Sigma - \{\beta, \sigma(\beta,\beta)\}$. An almost equivalent, hyper-minimal DTA is $N = (\{\top\}, \Sigma, \mu, \{\top\})$, where μ is such that $\mu(\alpha) = \mu(\beta) = \mu(\sigma(\top, \top)) = \top$. Since $L(N) = T_\Sigma$, we have that $L(M) \ominus L(N) = \{\beta, \sigma(\beta,\beta)\}$. Clearly, N has only the kernel state $\top$, so both errors occur in a kernel state. Moreover, when N processes the error tree $\sigma(\beta,\beta)$, it will take two special transitions (both times $\mu(\beta) = \top$).

To avoid counting errors multiple times, we need to make the attribution of the kernel state errors to the special transitions unique. We solve this problem by selecting the left-most occurrence of a special transition, but any other selection would work equally well. In order to define left-most occurrences, we first need to introduce positions. Let Δ be a ranked alphabet and $t \in T_\Delta(Q)$. The set $\text{pos}(t) \subseteq \mathbb{N}^*$ of *positions in t* is defined by $\text{pos}(q) = \{\varepsilon\}$ for every $q \in Q$ and $\text{pos}(\sigma(t_1, \ldots, t_k)) = \{\varepsilon\} \cup \{iw \mid i \in [k], w \in \text{pos}(t_i)\}$ for all $\sigma \in \Delta_k$ and $t_1, \ldots, t_k \in T_\Delta(Q)$. For every $w \in \text{pos}(t)$, we write $t|_w$ for the subtree of t that is rooted in position w. A position $u \in \text{pos}(t)$ is *to the left of* another position $w \in \text{pos}(t)$, written $u \sqsubset w$, if $u \leq w$ and $u \npreceq w$, where $\leq$ and $\preceq$ are the lexicographic and prefix order on $\mathbb{N}^*$, respectively.

Example 19. Let $t = \sigma(\alpha, \sigma(\alpha, \sigma(\gamma(\alpha), \alpha)))$, which we also display left in Figure 2. The positions of t are

$$\text{pos}(t) = \{\varepsilon, 1, 2, 21, 22, 221, 2211, 222\} \ .$$

- The position 1 is to the left of position 21.
- The position 2211 is to the left of position 222.
- The position 2 is *not* to the left of position 222.



$\sigma_{(\varepsilon)}$ $\alpha_{(1)}$ $\sigma_{(2)}$ $\alpha_{(21)}$ $\sigma_{(22)}$ $\gamma_{(221)}$ $\alpha_{(222)}$ $\alpha_{(2211)}$
$\sigma_{(\varepsilon)}$ $\alpha_{(1)}$ $\sigma_{(2)}$ $\square_{(21)}$ $\sigma_{(22)}$ $\gamma_{(221)}$ $\alpha_{(222)}$ $\alpha_{(2211)}$
$\sigma_{(\varepsilon)}$ $\alpha_{(1)}$ $\sigma_{(2)}$ $\alpha_{(21)}$ $\sigma_{(22)}$ $\gamma_{(221)}$ $\square_{(222)}$ $\gamma_{(2211)}$ $\alpha_{(22111)}$
$\sigma_{(\varepsilon)}$ $\alpha_{(1)}$ $\sigma_{(2)}$ $\alpha_{(21)}$ $\sigma_{(22)}$ $\gamma_{(221)}$ $\square_{(222)}$ $\alpha_{(2211)}$

Figure 2: A tree with indicated positions (left) and three contexts.

Recall that M is our minimal reference DTA. In a context $c \in C_\Sigma(Q)$ the unique position of $\square$ is denoted by $\mathrm{pos}_\square(c)$. Now we can define the set LC of *left-most contexts*, which have no subtree to the left of (the occurrence of) $\square$ that is recognized (by M) in a state that is almost equivalent to a kernel state, as follows:

$$\mathrm{LC} = \{c \in C_\Sigma \mid \forall w \in \mathrm{pos}(c) \colon w \sqsubset \mathrm{pos}_\square(c) \text{ implies } [\delta(c|_w)]_\sim \subseteq \mathrm{Pre}(M)\} \ .$$

Example 20. We show three contexts right in Figure 2. For the purpose of the example, let $M = M_{\text{ex}}$ of Example 1, so the evaluation is done with M_{ex}.

- The first context (leftmost in Figure 2) is in LC because the only position that is to the left of $\square$ is processed in state q_α, which is not almost equivalent to a kernel state.
- The second context (middle in Figure 2) is *not* in LC because the position 221 is to the left of $\square$ and processed in state q_Γ, which is a kernel state.
- The third context (rightmost in Figure 2) is also *not* in LC because position 221 is again to the left of $\square$, but this time processed in state q_γ, which is almost equivalent to the kernel state q_Γ.

With the help of the set LC we can now make the error attribution more formal. We already know that each remaining error tree has a special transition that switches from exclusively preamble states to a kernel state. Moreover, we will now prove that every such error tree decomposes uniquely into a context of LC, which encodes the part of the tree that is processed after the left-most occurrence of a special transition, and a tree that uses a special transition at the root.

Lemma 21. *Every error tree $t \in E_p$ with $p \in \mathrm{Ker}(N)$ decomposes uniquely via $t = c[u]$ into a left-most context $c \in \mathrm{LC}$ and $u \in T_\Sigma$ such that $\mu(u) \in \mathrm{Ker}(N)$, but $\mu(u|_w) \in \mathrm{Pre}(N)$ for all $w \in \mathrm{pos}(u)$ with $w \neq \varepsilon$.*

Proof. Since $\mu(t) \in \mathrm{Ker}(N)$, there must exist positions $w \in \mathrm{pos}(t)$ such that $\mu(t|_w) \in \mathrm{Ker}(N)$ but $\mu(t|_{wv}) \in \mathrm{Pre}(N)$ for all $wv \in \mathrm{pos}(t)$ with $v \neq \varepsilon$. Let w be the left-most such position (i.e., a minimal such position with respect to $\sqsubset$). It remains to show that $t[\square]_w \in \mathrm{LC}$, where $t[\square]_w$ denotes the context obtained from t by replacing the subtree rooted at w by $\square$. By the selection of w, all positions $v \sqsubset w$ are such that $\mu(t|_v) \in \mathrm{Pre}(N)$. Thus, $[\delta(t|_v)]_\sim \subseteq \mathrm{Pre}(M)$ by Theorem 6 and (‡), which proves that $t[\square]_w \in \mathrm{LC}$. Thus, we obtain the suitable decomposition $t = c[t|_w]$ with $c = t[\square]_w$. The uniqueness is also easy to show as each other suitable position w' obeys $w \sqsubset w'$ and $\mu(t|_w) \in \mathrm{Ker}(N)$, which by Lemma 3 yields that $\delta(t|_w) \sim q$ for some $q \in \mathrm{Ker}(M)$. Consequently, $t[\square]_{w'} \notin \mathrm{LC}$ for all other suitable positions $w' \neq w$. $\square$

Example 22. Let us consider the DTAs M_{ex} and N_{ex} of Examples 1 and 4, respectively. We already observed that $\sigma(\alpha, \alpha)$ is an error tree of $L(M_{\text{ex}}) \ominus L(N_{\text{ex}})$. Moreover, $\mu(\sigma(\alpha, \alpha)) = p_\Sigma \in \mathrm{Ker}(N_{\text{ex}})$, so the error is committed in a kernel state of N. Its unique decomposition is $c = \square$ and $u = \sigma(\alpha, \alpha)$. Similarly, the kernel state errors β and $\gamma(\alpha)$ decompose such that $c = \square$. Finally, the errors $\gamma(\beta)$ and $\gamma(\eta)$ decompose such that $c = \gamma(\square)$.

The decomposition $t = c[u]$ of Lemma 21 already hints at the next steps. We can compute $\delta(u)$ and $\mu(u)$, for which we know that $\delta(u) \sim \mu(u)$ by Lemma 3. For every $q \in Q$ and $p \in P$, let $E_{q,p} = L(M)_q \ominus L(N)_p$ be the difference contexts between states q and p in M and N, respectively. Consequently, all errors attributed to the last transition used when processing u by N are made between those two states, so we have $c \in E_{\delta(u),\mu(u)}$ for the context c of the decomposition. Moreover, we know that $c \in \mathrm{LC}$ by the decomposition of Lemma 21. Let $e \in E_p$ with $p \in \mathrm{Ker}(N)$ be an error committed in a kernel state of N, and let $e = c[u]$ be its unique decomposition according to Lemma 21. Moreover, let $\sigma(p_1, \dots, p_k)$ be the transition that is used at the root of u; i.e., $u = \sigma(u_1, \dots, u_k)$ and $p_i = \mu(u_i)$ for every $i \in [k]$. In fact, it is clear from the decomposition of Lemma 21 that $p_1, \dots, p_k \in \mathrm{Pre}(N)$ and $\mu(\sigma(p_1, \dots, p_k)) \in \mathrm{Ker}(N)$. Formally, we say that the error e is *attributed to* the transition $\sigma(p_1, \dots, p_k)$. Clearly, each error is attributed to a unique special transition by Lemma 21.

Lemma 23. *For each error $e \in E_p$ with $p \in \mathrm{Ker}(N)$ that is attributed to the special transition $\sigma(p_1, \dots, p_k)$, we have $c \in E_{\delta(u),\mu(u)}$, where $e = c[u]$ is the unique decomposition of e according to Lemma 21. Moreover, for each $q \in Q$ and $p \in P$, every $c \in E_{q,p}$ and $u \in L(M)^q \cap L(N)^p$ yields an error tree $e \in E = L(M) \ominus L(N)$.*

Proof. Both properties are trivial observations on error trees. □

Example 24. Let us again consider the DTAs M_{ex} and N_{ex} of Examples 1 and 4, respectively. In particular, let us look at the errors committed in kernel states of N_{ex} and attribute them to the special transitions of N_{ex}.

- The error $\sigma(\alpha, \alpha)$ is attributed to the transition $\sigma(p_\alpha, p_\alpha)$.
- The error β is attributed to the transition β.
- The errors $\gamma(\alpha)$, $\gamma(\beta)$, and $\gamma(\eta)$ are all attributed to $\gamma(p_\Gamma)$.

It remains to compute the number of attributed errors. To this end, we first translate the problem to M as we did in the approach for the errors committed in preamble states. Note that for any decomposition $e = c[u]$ according to Lemma 21 we have $\mu(u) \in \mathrm{Ker}(N)$, which with the help of Theorem 6 yields that there exists a state $q \in Q$ such that $q \equiv \mu(u)$. Consequently, instead of computing $E_{\delta(u),\mu(u)} \cap \mathrm{LC}$ as required by Lemma 23, we can also compute $E_{\delta(u),q} \cap \mathrm{LC}$ because $E_{\delta(u),q} = E_{\delta(u),\mu(u)}$ due to $L(M)_q = L(N)_{\mu(u)}$. Clearly, we only need the cardinality of those error sets $E_{\delta(u),q} \cap \mathrm{LC}$, or more generally, the cardinality of $E_{q,q'} \cap \mathrm{LC}$ for all $q \sim q'$ with $q' \in \mathrm{Ker}(M)$. Thus, for all $q \sim q'$, let $d(q, q') = |E_{q,q'} \cap \mathrm{LC}|$ and

$$\begin{aligned} C_M &= C_\Sigma(Q) \cap \Sigma(Q \cup \{\Box\}) \\ \overline{C}_M &= \Big\{\sigma(q_1, \dots, q_i, \Box, q_{i+1}, \dots, q_k) \in C_M \;\Big|\; \bigcup_{j \in [i]} [q_j]_\sim \subseteq \mathrm{Pre}(M), q_{i+1}, \dots, q_k \in \mathrm{Pre}(M)\Big\} \enspace , \end{aligned}$$

of which the elements are called *transition* and *(left-most) preamble transition contexts*, respectively. Roughly speaking, transition contexts are contexts that have the shape of a transition, in which one source state was replaced by $\Box$. Since we are mostly interested in special transitions, the preamble transition contexts are only those, in which all the (remaining) source states are preamble states of M, and in particular, those to the left of $\Box$ are such that their whole equivalence class contains only preamble states (as in the definition of LC). To compute d, we adjust the straightforward counting procedure of [24].

Lemma 25. *For all $q, q' \in Q$ with $q \sim q'$ we have*

$$\begin{aligned} d(q, q) &= 0 \\ d(q, q') &= \Bigg(\sum_{\substack{c \in \overline{C}_M \\ c = \sigma(q_1, \dots, q_i, \Box, q_{i+1}, \dots, q_k)}} a_{q_1} \cdot \ldots \cdot a_{q_k} \cdot d(\delta(c[q]), \delta(c[q']))\Bigg) + \begin{cases} 1 & \text{if } q \in F \text{ xor } q' \in F \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

Proof. The first equation is trivial and the second equation straightforwardly formalizes $|E_{q,q'}|$, but only counts the error contexts of LC. More precisely, the final summand checks whether $\Box \in \mathrm{LC}$ is in the difference $E_{q,q'}$. Every other difference context $c'' \in E_{q,q'}$ decomposes via $c'' = c'[\xi]$ into

(i) a context ξ obtained from a transition context $c = \sigma(q_1, \dots, q_i, \Box, q_{i+1}, \dots, q_k)$ of $\overline{C}_M$ by replacing the states $q_1, \dots, q_k \in \mathrm{Pre}(M)$ by $t_1 \in L(M)^{q_1}, \dots, t_k \in L(M)^{q_k}$, respectively, which yields the factors $a_{q_1}, \dots, a_{q_k}$, and
(ii) an error context $c' \in E_{\delta(c[q]),\delta(c[q'])}$ for the states $\delta(c[q])$ and $\delta(c[q'])$, which yields the factor $d(\delta(c[q]), \delta(c[q']))$.

Algorithm 3 Implementation of the counting argument of Lemma 25.

```
Require: a minimal DTA $M$, its almost equivalence $\sim$, and $a_q$ for all preamble states $q \in \mathrm{Pre}(M)$
Return: the matrix $d\colon Q^2 \to \mathbb{N} \cup \{\infty\}$ such that $d(q,q') = \infty$ for all $q \not\sim q'$
    for all $(q,q') \in Q^2$ do
2:      $d(q,q') \leftarrow \begin{cases} 0 & \text{if } q = q' \\ \infty & \text{otherwise} \end{cases}$      // initialize diagonal to 0; otherwise ∞
    for all $q, q' \in Q$ with $q \sim q'$ do
4:      Compute′$(d, q, q')$      // compute the entry $d(q,q')$; the matrix $d$ is a reference here
    return $d$

6:  Compute′$(d, q, q')$:      // function Compute′ fills the matrix $d$ and returns $d(q,q')$
    if $d(q,q') = \infty$ then
8:      $d(q,q') \leftarrow \begin{cases} 1 & q \in F \text{ xor } q' \in F \\ 0 & \text{otherwise} \end{cases}$      // check difference context □
        $d(q,q') \leftarrow d(q,q') + \sum_{\substack{c \in \overline{C}_M \\ c = \sigma(q_1,\dots,q_i,\Box,q_{i+1},\dots,q_k)}} a_{q_1} \cdot \ldots \cdot a_{q_k} \cdot$ Compute′$(d, \delta(c[q]), \delta(c[q']))$      // use formula of Lemma 25
10: return $d(q,q')$      // and return it (or just return value if already computed)
```

We can immediately restrict ourselves to preamble transition contexts because for all $c \in C_M - \Sigma(\mathrm{Pre}(M) \cup \{\Box\})$ we have $\delta(c[q]) = \delta(c[q'])$ by [23, Proposition 18], which yields that $d(\delta(c[q]), \delta(c[q'])) = 0$. Moreover, if the states $q_1, \dots, q_i$ to the left of □ are not such that $[q_1]_\sim, \dots, [q_i]_\sim \subseteq \mathrm{Pre}(M)$, then the context c'' is not in LC and thus discarded. □

Example 26. Let us calculate the relevant differences on the DTA M_{ex} of Example 1. According to Example 2 we have that $q_\beta \sim q_\eta \sim q_\gamma \sim q_\Gamma$ and $q_\sigma \sim q_\Sigma$. As a demonstration, let us calculate $d(q_\beta, q_\Gamma)$. First, we notice that $q_\beta \notin F$ whereas $q_\Gamma \in F$, so we already identified the difference context □. For the remaining differences we have to consider the preamble transition contexts $\gamma(\Box)$, $\sigma(q_\alpha, \Box)$, and $\sigma(\Box, q)$ for all $q \in \{q_\alpha, q_\beta, q_\eta\}$. The first context yields $d(q_\gamma, q_\Gamma) = 1$ difference, whereas the remaining contexts all yield $d(q_\sigma, q_\sigma) = 0$ differences. In total, we thus obtain $d(q_\beta, q_\Gamma) = 2$, and the exact differences are $E_{q_\beta, q_\Gamma} = \{\Box, \gamma(\Box)\}$.

Since we now have a recursive procedure to compute d, let us quickly analyze its time complexity. The analysis is based on the idea that entries in d are never recomputed once they have been computed (see Algorithm 3).

Corollary 27 (of Lemmata 14 and 25). *We can compute $d(q,q')$ in time $O(m \cdot n)$ for all $q, q' \in Q$ with $q \sim q'$, where $m = |M|$ and $n = |Q|$.*

Proof. We can compute all a_q with $q \in \mathrm{Pre}(M)$ in time $O(m)$ by Algorithm 2, and we can compute each entry in d in time $O(\frac{m}{n})$ without the time needed to compute the recursive calls because there are $\sum_{k \geq 1} k \cdot |\Sigma_k| \cdot |Q|^{k-1}$ transition contexts.[14] Since there are at most n^2 entries in d, we obtain the stated time-bound. □

Now we can identify and count the errors caused in kernel states of N. We look at all the special transitions in N and compute the number of errors attributed to it. Let $s = \sigma(p_1, \dots, p_k) \in \Sigma(\mathrm{Pre}(N))$ with $\mu(s) \in \mathrm{Ker}(N)$ be a special transition. The set E_s of errors caused by this transition s is

$$E_s = \{c[\sigma(t_1, \dots, t_k)] \mid c \in E_{\delta(\sigma(t_1,\dots,t_k)),\mu(s)} \cap \mathrm{LC}, t_1 \in L(N)^{p_1}, \dots, t_k \in L(N)^{p_k}\} \ ,$$

[14]This actually needs another trick. Given a transition $\sigma(q_1, \dots, q_k) \in \Sigma(Q)$ we obtain k transition contexts $c_1, \dots, c_k$ by replacing in turn each state $q_1, \dots, q_k$ by □. To avoid a multiplication effort of $O(k)$ we once compute $a_{q_1} \cdot \ldots \cdot a_{q_k}$ and then compute the value for the context c_i by dividing this product by the value a_{q_i}.

which contains all errors that use the transition s as left-most special transition. This representation is still a bit cumbersome to work with, but fortunately we know from Lemmata 11 and 13 and (‡) that for all $i \in [k]$ there exists $B_i \subseteq \mathrm{Pre}(M)$ such that $L(N)^{p_i} = \bigcup_{q \in B_i} L(M)^q$. With this knowledge we can conclude that

$$\begin{aligned}
E_s &= \{c[\sigma(t_1, \dots, t_k)] \mid c \in E_{\delta(\sigma(t_1,\dots,t_k)),\mu(s)} \cap \mathrm{LC}, t_1 \in L(N)^{p_1}, \dots, t_k \in L(N)^{p_k}\} \\
&= \Big\{c[\sigma(t_1, \dots, t_k)] \,\Big|\, c \in E_{\delta(\sigma(t_1,\dots,t_k)),\mu(s)} \cap \mathrm{LC}, t_1 \in \bigcup_{q_1 \in B_1} L(M)^{q_1}, \dots, t_k \in \bigcup_{q_k \in B_k} L(M)^{q_k}\Big\} \\
&= \{c[\sigma(t_1, \dots, t_k)] \mid q_1 \in B_1, \dots, q_k \in B_k, c \in E_{\delta(\sigma(q_1,\dots,q_k)),\mu(s)} \cap \mathrm{LC}, t_1 \in L(M)^{q_1}, \dots, t_k \in L(M)^{q_k}\} \\
&= \bigcup_{q_1 \in B_1,\dots,q_k \in B_k} \{c[\sigma(t_1, \dots, t_k)] \mid c \in E_{\delta(\sigma(q_1,\dots,q_k)),\mu(s)} \cap \mathrm{LC}, t_1 \in L(M)^{q_1}, \dots, t_k \in L(M)^{q_k}\} \enspace ,
\end{aligned}$$

where the unions are disjoint. Hence we obtain a straightforward way to calculate $|E_s|$.

Lemma 28. *For every $s = \sigma(p_1, \dots, p_k) \in \Sigma(\mathrm{Pre}(N))$ with $\mu(s) \in \mathrm{Ker}(N)$*

$$|E_s| = \sum_{q_1 \in [\delta(u_{p_1})]_\sim, \dots, q_k \in [\delta(u_{p_k})]_\sim} a_{q_1} \cdot \ldots \cdot a_{q_k} \cdot d(\delta(\sigma(q_1, \dots, q_k)), q) \enspace ,$$

where $u_p \in L(N)^p$ is arbitrary for every $p \in \mathrm{Pre}(N)$ and $q \in \mathrm{Ker}(M)$ is such that $q \equiv \mu(s)$.

Proof. Let $N' = (P', \Sigma, \mu', G')$ be the DTA returned by Algorithm 1 when applied to M and $\sim$. Moreover, for every $i \in [k]$ let $p'_i = \mu'(u_{p_i})$. Then $L(N)^{p_i} = L(N')^{p'_i} = \bigcup_{q_i \in [\delta(u_{p_i})]_\sim} L(M)^{q_i}$ by Theorem 6 and (‡) [see also Lemmata 11 and 13]. Moreover, $L(N)_{\mu(s)} = L(M)_q$ by assumption. Together with these statements, the equation is a straightforward implementation of the definition of E_s as demonstrated earlier. □

Example 29. Let M_{ex} and N_{ex} once more be the DTAs of Examples 1 and 4. By inspection of N_{ex} it has the following special transitions:[15]

$$s_1 \colon \mu(\beta) = p_\Gamma \qquad s_2 \colon \mu(\eta) = p_\Gamma \qquad s_3 \colon \mu(\gamma(p_\alpha)) = p_\Gamma \qquad \text{and} \qquad s_4 \colon \mu(\sigma(p_\alpha, p_\alpha)) = p_\sigma \enspace .$$

Now let us compute $|E_{s_i}|$ for $i \in [4]$. This is trivial for $i \in \{1, 2\}$ because we just need to compute $d(q_\beta, q_\Gamma)$ and $d(q_\eta, q_\Gamma)$, respectively. From Example 26 we know that $d(q_\beta, q_\Gamma) = 2$, so $|E_{s_1}| = 2$, and more precisely, $E_{s_1} = \{\beta, \gamma(\beta)\}$. Similarly, we can show that $d(q_\eta, q_\Gamma) = 1$ because q_η and q_Γ are both final states (more precisely, $E_{q_\eta,q_\Gamma} = \{\gamma(\square)\}$). Thus, $|E_{s_2}| = 1$ and $E_{s_2} = \{\gamma(\eta)\}$. For the third transition, we fortunately have $[q_\alpha]_\sim = \{q_\alpha\}$ and $a_{q_\alpha} = 1$, so the computation again reduces to $d(q_\gamma, q_\Gamma)$, which is 1 by Example 26. Hence, $|E_{s_3}| = 1$ and $E_{s_3} = \{\gamma(\alpha)\}$. Finally, in the fourth transition we have essentially the same situation, so we only need to compute $d(q_\sigma, q_\Sigma)$, which is 1. More precisely, $E_{q_\sigma,q_\Sigma} = \{\square\}$, so consequently, $|E_{s_4}| = 1$ and $E_{s_4} = \{\sigma(\alpha, \alpha)\}$. In this way, we identified all 5 errors and attributed them as demonstrated in Example 24.

By Lemma 21 the sets E_s for special transitions $s \in \Sigma(\mathrm{Pre}(N))$ are pairwise disjoint, so the errors just add up. In addition, any state $p \in P$ such that $p \sim \mu(s)$ is a valid transition target, so to optimize the number of errors, we can simply select the transition target $p \in P$ with $p \sim \mu(s)$ that minimizes the number of caused errors. In summary, this yields our main theorem (and Algorithm 4).

Theorem 30. *Let $m = |M|$ and $n = |Q|$. For every hyper-minimal DTA N that is almost equivalent to M we can determine $|L(M) \ominus L(N)|$ in time $O(m \cdot n)$. Moreover, we can compute a hyper-minimal DTA N' that minimizes the number $|L(M) \ominus L(N')|$ of errors in time $O(m \cdot n)$.*

[15]We immediately also show the transition target, which simplifies the later discussion.

Algorithm 4 Hyper-optimization algorithm.

```
Require: a minimal DTA $M = (Q, \Sigma, \delta, F)$, its almost equivalence $\sim$, and
    an almost equivalent, hyper-minimal DTA $N = (P, \Sigma, \mu, G)$
Return: an almost equivalent, hyper-optimal DTA $N$
    select $u_p \in L(N)^p$ for all $p \in \mathrm{Pre}(N)$        // select access tree for each preamble state
 2: $G \leftarrow (G - \mathrm{Pre}(N)) \cup \{p \in \mathrm{Pre}(N) \mid \sum_{q \in [\delta(u_p)]_\sim - F} a_q < \sum_{q \in [\delta(u_p)]_\sim \cap F} a_q\}$        // use Theorem 16 to decide finality
    for all $s = \sigma(p_1, \dots, p_k) \in \Sigma(\mathrm{Pre}(N))$ with $\mu(s) \in \mathrm{Ker}(N)$ do
 4:     select $p \in \mathrm{Ker}(N)$ such that $p \equiv \arg\min_{q \in \mathrm{Ker}(M), q \sim \mu(s)} (\sum_{q_1 \in [\delta(u_{p_1})]_\sim, \dots, q_k \in [\delta(u_{p_k})]_\sim} a_{q_1} \cdot \ldots \cdot a_{q_k} \cdot d(\delta(\sigma(q_1, \dots, q_k)), q)$
        $\mu(s) \leftarrow p$        // reroute transition according to the optimal choice according to Lemma 28
 6: return $N$
```

5. Variations of the optimality criterion

Certainly, the number of errors is only one possible criterion that we can optimize. We can easily imagine other sensible criteria. In fact, a reviewer of the conference version [25] suggested to use the sum of the error tree sizes instead of their number as a more realistic measure. In other applications it might be better to keep the error trees as small as possible, whereas the actual number of errors might be rather unimportant.

We tried to present our approach independent of the actual error scoring function as much as possible (without obfuscating the actual approach). In principle, we can also compute the set $E = L(M) \ominus L(N)$ of errors directly and then score them. In this way, given a scoring $\mathrm{eval}(N) = \bigoplus_{e \in E} f(e)$ for some domain D, associative and commutative operator $\oplus \colon D \times D \to D$, and function $f \colon T_\Sigma \to D$, we could calculate an explicit representation of E and then apply the scoring. Many imaginable scoring functions fit this template. However, this approach will typically be expensive as computing a list representation for E could have exponential length in the size of M.[16] Finally, there are scoring approaches (e.g., those based on edit distances) that do not fit our approach at all because the score might not be an amalgamation over individually scored error trees $e \in E$.

For our hyper-optimization we kept the approach efficient in order to get efficient optimization algorithms. We associated and attributed the actual errors and demonstrated how to use that knowledge to derive efficient formulas for our optimality criterion. However, the general approach is much more flexible, which we showcase in 4 other scenarios, which can be (potentially only inefficiently) scored and optimized using our approach.

(A) $\mathrm{eval}_1(N) = \sum_{e \in E} |e|$ where $|e| = |\mathrm{pos}(e)|$.
(B) $\mathrm{eval}_2(N) = \sum_{e \in E} (\sum_{\sigma \in \Sigma} m_\sigma \cdot |e|_\sigma)$, where $|e|_\sigma$ is the number of occurrences of σ in e.
(C) $\mathrm{eval}_3(N) = \max_{e \in E} |e|$.
(D) $\mathrm{eval}_4(N) = \max_{e \in E} \mathrm{ht}(e)$.

In Scenario (A) we optimize the sum of the sizes of the error trees, which was suggested by a reviewer of the conference version [25]. In Scenario (B) we score in a similar way as in Scenario (A), but different symbols now are weighted using factors $(m_\sigma)_{\sigma \in \Sigma}$. This score can be used to avoid error trees containing selected symbols by making those symbols very expensive. Finally, in Scenarios (C) and (D) we just want to keep the size and the height of the error trees small at the expense of making potentially more errors, respectively.

For Scenario (A), the author is not aware of any efficient calculation of the score. We can abstract from the explicit representation of a set T of trees to a vector showing how many trees of each size are present in T. However, this representation still suffers from the inefficiencies of the explicit representation because the vector length can be exponential in the size of M and the sum of the entries can also be exponential in M, so the vector need not be sparse. Thus, we will not present a solution using this abstraction as it would be similarly inefficient as the trivial solution, which consists of explicitly calculating the set E of errors and then scoring them. Optimization is possible in this scenario as the best local choices are guaranteed to yield the best global result. Naturally, the process remains inefficient as we have to score all alternatives, which we can efficiently represent, but the author does not know how to efficiently score them. Thus, overall the inefficient scoring procedure makes the whole approach inefficient. Exactly

[16]The size of the trees in the list could also be exponential in the size of M.

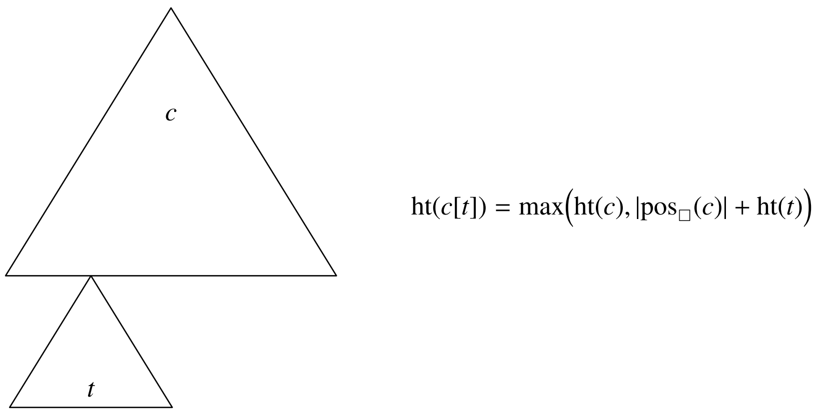

c
t
$\mathrm{ht}(c[t]) = \max\big(\mathrm{ht}(c), |\mathrm{pos}_\square(c)| + \mathrm{ht}(t)\big)$

Figure 3: Height of a tree $c[t]$.

the same problems occur in Scenario (B), so the author also does not know to efficiently score or optimize DTA using this criterion.

Scenario (C) is much simpler. Here, we do not add the sizes, but rather just want to keep the maximal size of the error trees small. In this case we can represent a set T of trees by just the maximal size of a tree in T. Clearly, this representation is efficient, and we will demonstrate that it is sufficient to compute the score and to optimize DTA. To this end, we adjust the definition of $(a_q)_{q\in\mathrm{Pre}(M)}$ with $a_q \in \mathbb{N}$ for all $q \in \mathrm{Pre}(M)$ as follows:

$$a_q = \max\Big\{1 + \sum_{i\in[k]} a_{q_i} \;\Big|\; \sigma \in \Sigma_k, q_1, \dots, q_k \in Q, \delta(\sigma(q_1, \dots, q_k)) = q\Big\} \enspace .$$

Then $a_q = \max\{|t| \mid t \in L(M)^q\}$ for all $q \in \mathrm{Pre}(M)$. Similarly, for every $q, q' \in Q$ with $q \sim q'$ we let

$$d(q, q') = \max\Big\{1 + \sum_{i\in[k]} a_{q_i} + d(\delta(c[q]), \delta(c[q'])) \;\Big|\; c = \sigma(q_1, \dots, q_i, \square, q_{i+1}, \dots, q_k) \in \overline{C}_M\Big\} \enspace .$$

Consequently, $d(q, q') = \max\{|c| - 1 \mid c \in L(M)_q \ominus L(M)_{q'}\}$. Finally, for every special transition s of N we obtain

$$\max_{e\in E_s} |e| = \max\Big\{1 + \sum_{i\in[k]} a_{q_i} + d(\delta(\sigma(q_1, \dots, q_k)), q) \;\Big|\; q_1 \in [\delta(u_{p_1})]_\sim, \dots, q_k \in [\delta(u_{p_k})]_\sim\Big\} \enspace ,$$

where $s = \sigma(p_1, \dots, p_k)$ with $p_1, \dots, p_k \in \mathrm{Pre}(N)$, $\mu(s) \in \mathrm{Ker}(N)$, $u_p \in L(N)^p$ arbitrary for every $p \in P$, and $q \in \mathrm{Ker}(M)$ such that $q \equiv \mu(s)$. Again, the current approach (make the best local decisions) works for the optimization with respect to this scoring function.

Finally, Scenario (D) is similar to Scenario (C), but with the height of the error trees instead of their size. Also in this case we can represent a set T of trees by just the maximal height of a tree in it, which is again an efficient representation. Again, we need to slightly adjust the major scoring definitions. This time, we let $(a_q)_{q\in\mathrm{Pre}(M)}$ with $a_q \in \mathbb{N}$ for all $q \in \mathrm{Pre}(M)$ be such that

$$a_q = \max\Big\{1 + \max_{i\in[k]} a_{q_i} \;\Big|\; \sigma \in \Sigma_k, q_1, \dots, q_k \in Q, \delta(\sigma(q_1, \dots, q_k)) = q\Big\} \enspace .$$

Then $a_q = \max\{\mathrm{ht}(t) \mid t \in L(M)^q\}$ for all $q \in \mathrm{Pre}(M)$. For the difference between almost equivalent states, we need a slightly more elaborate representation. More precisely, we need to store two values: (i) the largest height of an error context and (ii) the largest depth of the symbol $\square$ in an error context (see Figure 3 for an illustration). Given these two values, we can predict the largest height of a combined error tree (i.e., tree substituted into a difference context). For every $q, q' \in Q$ with $q \sim q'$, let $d(q, q') \in \mathbb{N} \times \mathbb{N}$ be given such that

$$\begin{aligned} d(q, q') = \Big\langle &\max\Big\{\max(d + 1 + \max_{i\in[k]} a_{q_i}, h) \;\Big|\; c = \sigma(q_1, \dots, q_i, \square, q_{i+1}, \dots, q_k) \in \overline{C}_M, \langle h, d\rangle = d(\delta(c[q]), \delta(c[q']))\Big\}, \\ &\max\Big\{1 + d \mid c \in \overline{C}_M, \langle h, d\rangle = d(\delta(c[q]), \delta(c[q']))\Big\}\Big\rangle \enspace . \end{aligned}$$

It is easy to show that

$$d(q,q') = \left\langle \max\,\{\mathrm{ht}(c) \mid c \in L(M)_q \ominus L(M)_{q'}\}\,,\ \max\,\{|\mathrm{pos}_\Box(c)| \mid c \in L(M)_q \ominus L(M)_{q'}\}\right\rangle$$

for all $q, q' \in Q$ with $q \sim q'$. Finally, for every special transition s in N we obtain

$$\max_{e \in E_s} \mathrm{ht}(e) = \max\left\{\max(d + 1 + \max_{i\in[k]} a_{q_i}, h) \,\middle|\, q_1 \in [\delta(u_{p_1})]_\sim, \ldots, q_k \in [\delta(u_{p_k})]_\sim, \langle h, d\rangle = d(\delta(\sigma(q_1,\ldots,q_k)), q)\right\} ,$$

where $s = \sigma(p_1,\ldots,p_k)$ with $p_1,\ldots,p_k \in \mathrm{Pre}(N)$, $\mu(s) \in \mathrm{Ker}(N)$, $u_p \in L(N)^p$ arbitrary for every $p \in P$, and $q \in \mathrm{Ker}(M)$ such that $q \equiv \mu(s)$. Our approach of making the best local decisions works for the optimization with respect to this scoring function.

Future work

Recently, [28] showed results in the string case for other regular languages of allowed differences (i.e., not just finite difference). These results should translate straightforwardly to tree automata. The difference in the number of errors between the optimal DTA and the worst DTA can be exponential, so the optimization can avoid a large number of errors. A practical evaluation for DTA remains future work, but a simple experiment was already conducted in [24, Section 6] for the string case.

References

[1] F. Gécseg, M. Steinby, Tree Automata, Akadémiai Kiadó, Budapest, 1984.

[2] F. Gécseg, M. Steinby, Tree languages, in: G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, Vol. 3, Springer, 1997, Ch. 1, pp. 1–68.

[3] Z. Fülöp, H. Vogler, Weighted tree automata and tree transducers, in: M. Droste, W. Kuich, H. Vogler (Eds.), Handbook of Weighted Automata, EATCS Monographs on Theoretical Computer Science, Springer, 2009, Ch. 9, pp. 313–403.

[4] S. Petrov, L. Barrett, R. Thibaux, D. Klein, Learning accurate, compact, and interpretable tree annotation, in: Proc. 44th Ann. Meeting of the ACL, Association for Computational Linguistics, 2006, pp. 433–440.

[5] J. Sakarovitch, Rational and recognisable power series, in: M. Droste, W. Kuich, H. Vogler (Eds.), Handbook of Weighted Automata, EATCS Monographs on Theoretical Computer Science, Springer, 2009, Ch. IV, pp. 105–174.

[6] M. Mohri, Finite-state transducers in language and speech processing, Comput. Linguist. 23 (2) (1997) 269–311.

[7] M. Mohri, Weighted finite-state transducer algorithms: An overview, in: C. Martín-Vide, V. Mitrana, G. Păun (Eds.), Formal Languages and Applications, no. 148 in Studies in Fuzziness and Soft Computing, Springer, 2004, Ch. 29, pp. 551–564.

[8] G. Iglesias, C. Allauzen, W. Byrne, A. de Gispert, M. Riley, Hierarchical phrase-based translation representations, in: Proc. 2011 Conf. Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2011, pp. 1373–1383.

[9] J. Högberg, A. Maletti, J. May, Bisimulation minimisation for weighted tree automata, in: T. Harju, J. Karhumäki, A. Lepistö (Eds.), Proc. 11th Int. Conf. Developments in Language Theory, Vol. 4588 of LNCS, Springer, 2007, pp. 229–241.

[10] J. Högberg, A. Maletti, J. May, Backward and forward bisimulation minimization of tree automata, Theoret. Comput. Sci. 410 (37) (2009) 3539–3552.

[11] S. Yu, Regular languages, in: G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, Vol. 1, Springer, 1997, Ch. 2, pp. 41–110.

[12] G. Gramlich, G. Schnitger, Minimizing nfa's and regular expressions, J. Comput. System Sci. 73 (6) (2007) 908–923.

[13] J. E. Hopcroft, An $n \log n$ algorithm for minimizing states in a finite automaton, in: Z. Kohavi, A. Paz (Eds.), Theory of Machines and Computations, Academic Press, 1971, pp. 189–196.

[14] M. Nivat, A. Podelski (Eds.), Tree Automata and Languages, Studies in Computer Science and Artificial Intelligence, North-Holland, 1992.

[15] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, M. Tommasi, Tree automata: Techniques and applications, `http://tata.gforge.inria.fr/`, release October, 12 (2007).

[16] A. Badr, V. Geffert, I. Shipman, Hyper-minimizing minimized deterministic finite state automata, RAIRO Theor. Inf. Appl. 43 (1) (2009) 69–94.

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

[18] P. Gawrychowski, A. Jeż, Hyper-minimisation made efficient, in: Proc. 34th Int. Symp. Mathematical Foundations of Computer Science, Vol. 5734 of LNCS, Springer, 2009, pp. 356–368.

[19] M. Holzer, A. Maletti, An $n \log n$ algorithm for hyper-minimizing a (minimized) deterministic automaton, Theoret. Comput. Sci. 411 (38–39) (2010) 3404–3413.

[20] A. Maletti, D. Quernheim, Hyper-minimisation of deterministic weighted finite automata over semifields, in: Proc. 13th Int. Conf. Automata and Formal Languages, Nyíregyháza College, 2011, pp. 285–299.

[21] S. Schewe, Beyond hyper-minimisation — minimising DBAs and DPAs is NP-complete, in: Proc. 30th Int. Conf. Foundations of Software Technology and Theoretical Computer Science, Vol. 8 of LIPIcs, Schloss Dagstuhl, 2010, pp. 400–411.

[22] P. Gawrychowski, A. Jeż, A. Maletti, On minimising automata with errors, in: Proc. 36th Int. Symp. Mathematical Foundations of Computer Science, Vol. 6907 of LNCS, Springer, 2011, pp. 327–338.

[23] A. Jeż, A. Maletti, Hyper-minimization for deterministic tree automata, in: Proc. 17th Int. Conf. Implementation and Application of Automata, Vol. 7381 of LNCS, Springer, 2012, pp. 217–228.
[24] A. Maletti, D. Quernheim, Optimal hyper-minimization, Int. J. Found. Comput. Sci. 22 (8) (2011) 1877–1891.
[25] A. Maletti, Hyper-optimization for deterministic tree automata, in: Proc. 18th Int. Conf. Implementation and Application of Automata, Vol. 7982 of LNCS, Springer, 2013, pp. 244–255.
[26] A. Maletti, D. Quernheim, Unweighted and weighted hyper-minimization, Int. J. Found. Comput. Sci. 23 (6) (2012) 1207–1225.
[27] A. Jeż, A. Maletti, Hyper-minimization for deterministic tree automata, Int. J. Found. Comput. Sci.To appear.
[28] M. Holzer, S. Jakobi, From equivalence to almost-equivalence, and beyond — minimizing automata with errors, in: Proc. 16th Int. Conf. Developments in Language Theory, Vol. 7410 of LNCS, Springer, 2012, pp. 190–201.