Backward and Forward Bisimulation Minimisation of Tree Automata

Johanna Högberg
Department of Computing Science
Umeå University, S–90187 Umeå, Sweden

Andreas Maletti
Faculty of Computer Science
Technische Universität Dresden, D–01062 Dresden, Germany

Jonathan May
Information Sciences Institute
University of Southern California, Marina Del Rey, CA 90292

Abstract. We improve an existing bisimulation minimisation algorithm for tree automata by introducing backward and forward bisimulations and developing minimisation algorithms for them. Minimisation via forward bisimulation is effective for deterministic automata and faster than the previous algorithm. Minimisation via backward bisimulation generalises the previous algorithm and is thus more effective but just as fast. We demonstrate implementations of these algorithms on a typical task in natural language processing.

1 Introduction

Automata minimisation has a long and studied history. For deterministic string automata (dfa) efficient algorithms exist. The well-known algorithm by Hopcroft [8] runs in time $O(n \log n)$ where n is the number of states of the input automaton. The situation is worse for non-deterministic automata (nfa). The minimisation problem for nfa is PSPACE-complete [13] and cannot even be efficiently approximated within the factor $o(n)$ unless P = PSPACE [7]. The problem must thus be restricted to allow algorithms of practical value, and one possibility is to settle for a partial minimisation. This was done in [2] for non-deterministic *tree automata* (nta), which are a generalisation of nfa that recognise tree languages and are used in applications such as model checking [1] and natural language processing [10].

The minimisation algorithm in [2] was inspired by a partitioning algorithm due to Paige and Tarjan [14], and relies heavily on *bisimulation*; a concept introduced by R. Milner as a formal tool for investigating transition systems. Intuitively, two states are bisimilar if they can simulate each other, or equivalently, the observable behaviour of the two states must coincide. Depending on the capacity of the observer, we obtain different types of bisimulation. In all cases we assume that the observer has the capacity to observe the reaction to a given input (i.e., the given tree is either accepted or rejected), so the presence of bisimilar states in an automaton indicates redundancy. Identifying bisimilar states allows us to reduce the size of the input automaton, but we are not guaranteed to obtain the smallest possible automaton. In this work we extend the approach of [2] in two ways: (i) we relax the constraints for state equivalence, and (ii) we introduce a new bisimulation relation that can be applied to deterministic bottom-up tree

automata [5], unlike [2], which is ineffective on dta. Thus we are able to find smaller automata than previously possible.

The two ways correspond, respectively, to two types of bisimulation: *backward* and *forward* bisimulation [4]. Let us explain both types on a bottom-up tree automaton. In a forward bisimulation on an automaton M, bisimilar states are restricted to have identical futures (i.e., the observer can inspect what will happen next). The future of a state q is the set of contexts[1] that would be recognised by M, if the computation starts with the state q at the unique $\Box$-labelled node in the context. By contrast, backward bisimulation uses a local condition on the transitions to enforce that the past of any two bisimilar states is equal (i.e., the observer can observe what already happened). The past of a state q is the language that would be recognised by the automaton if q were its only final state.

Both types of bisimulation yield efficient minimisation procedures and can be applied to arbitrary nta. Some special cases deserve mention: Forward bisimulation minimisation is useful on deterministic bottom-up tree automata (dta) where it computes the unique (up to isomorphism) minimal dta recognising the same language as the input dta (see Theorem 4.25). More importantly, it is shown in Theorem 4.23 that the asymptotic time-complexity of our minimisation algorithm is $O(\hat{r}\, m \log n)$, where $\hat{r}$ is the maximal rank of the input symbols, m is the size of the transition table, and n is the number of states. Thus our algorithm supersedes the currently best minimisation algorithm [5], whose complexity is $O(mn)$, for deterministic bottom-up tree automata.

Backward bisimulation, though slightly harder to compute, has great practical value as well. The backward bisimulation presented in this paper is weaker than the bisimulation of [2]. Consequently, the nta obtained by our backward bisimulation minimisation algorithm will have at most as many states as the automaton obtained by the minimisation algorithm of [2]. In addition, the asymptotic time-complexity of our algorithm (see Theorem 3.31), which is $O\big(\hat{r}^2\, m \log n\big)$ is the same as the one for the minimisation algorithm of [2][2].

Finally, there are advantages that support having two types of bisimulation. First, forward and backward bisimulation minimisation only yield nta that are minimal with respect to the respective type of bisimulation. Thus applying forward and backward bisimulation minimisation in an alternating fashion commonly yields even smaller nta (see Section 5). Second, in certain domains only one type of bisimulation minimisation is effective. For example, on dta without useless states backward bisimulation minimisation is ineffective because no two states of a dta have the same past.

2 Preliminaries

We write $\mathbb{N}$ to denote the set of natural numbers including zero. The set $\{k, k+1, \ldots, n\}$ is abbreviated to $[k, n]$, and the cardinality of a set S is denoted by $|S|$. We abbreviate $Q \times Q$ as Q^2, and the inclusion $q_i \in D_i$ for all $i \in [1, k]$ as $q_1 \cdots q_k \in D_1 \cdots D_k$.

Let $\mathcal{R}$ and $\mathcal{P}$ be equivalence relations on S. We say that $\mathcal{R}$ is *coarser* than $\mathcal{P}$ (or equivalently: $\mathcal{P}$ is a *refinement* of $\mathcal{R}$), if $\mathcal{P} \subseteq \mathcal{R}$. The *equivalence class* (or *block*) of an element s in S with respect to $\mathcal{R}$ is the set $[s]_{\mathcal{R}} = \{s' \mid (s, s') \in \mathcal{R}\}$. Whenever $\mathcal{R}$ is obvious from the context, we simply write $[s]$ instead of $[s]_{\mathcal{R}}$. It should be clear that $[s]$ and $[s']$ are equal if s and s' are in relation $\mathcal{R}$, and disjoint otherwise, so $\mathcal{R}$ induces a partition $(S/\mathcal{R}) = \{[s] \mid s \in S\}$ of S.

A *ranked alphabet* is a finite set of symbols $\Sigma = \bigcup_{k \in \mathbb{N}} \Sigma_{(k)}$ which is partitioned into pairwise

[1] A context is a tree in which there is a unique leaf labelled by the special symbol $\Box$.

[2] The $O(\hat{r} m \log n)$ run time reported in [2] assumes a total transition table, thus absorbing a factor $\hat{r}$ into m.

disjoint subsets $\Sigma_{(k)}$. The set T_Σ of *trees over* Σ is the smallest language over Σ such that $f\,t_1 \cdots t_k$ is in T_Σ for every f in $\Sigma_{(k)}$ and all $t_1, \ldots, t_k$ in T_Σ. To improve readability we write $f[t_1, \ldots, t_k]$ instead of $f\,t_1 \cdots t_k$ unless k is zero. Any subset of T_Σ is called a *tree language*.

A *non-deterministic tree automaton* (for short: nta) is a tuple $M = (Q, \Sigma, \delta, F)$, where Q is a finite set of *states*, Σ is a ranked alphabet, and δ is a finite set of *transitions* of the form $f(q_1, \ldots, q_k) \to q_{k+1}$ for some symbol f in $\Sigma_{(k)}$ and $q_1, \ldots, q_{k+1} \in Q$. Finally, $F \subseteq Q$ is a set of *accepting* states. To indicate that a transition $f(q_1, \ldots, q_k) \to q_{k+1}$ is in δ, we write $f(q_1, \ldots, q_k) \xrightarrow{\delta} q_{k+1}$. In the obvious way, δ extends to trees yielding a mapping $\delta\colon \mathrm{T}_\Sigma \to \mathfrak{P}(Q)$; i.e., for $t = f[t_1, \ldots, t_k]$ in T_Σ,

$$\delta(t) = \{q \mid f(q_1, \ldots, q_k) \xrightarrow{\delta} q \text{ and } q_i \in \delta(t_i) \text{ for all } i \in [1, k]\} \ .$$

For every $q \in Q$ we denote $\{t \in \mathrm{T}_\Sigma \mid q \in \delta(t)\}$ by $\mathcal{L}(M)_q$. The tree language *recognised* by M is $\mathcal{L}(M) = \bigcup_{q \in F} \mathcal{L}(M)_q$. Finally, we say that a state q in Q is *useless* if $\mathcal{L}(M)_q = \emptyset$.

3 Backward bisimulation

3.1 Foundation

We first introduce the notion of *backward bisimulation* for a nta M. This type of bisimulation requires bisimilar states to recognise the same tree language. Clearly, multiple states recognising the same language are not necessary, so we show how to collapse a block of bisimilar states into just a single state to obtain a potentially smaller nta M'. The construction is such that M' recognises exactly $\mathcal{L}(M)$. Finally, we show that there exists a coarsest backward bisimulation on M, which leads to the smallest collapsed nta.

We need the following notation. Let Π be a partition of a set Q. For every integer $k \in \mathbb{N}$, we write $\Pi_{(k)}$ for the set of all strings of blocks of Π with length k; i.e.,

$$\Pi_{(k)} = \{D_1 \times \cdots \times D_k \mid D_1, \ldots, D_k \in \Pi\} \ .$$

For the set of such strings of length at most k we write $\Pi_{(\leq k)}$; i.e., $\Pi_{(\leq k)} = \Pi_{(0)} \cup \cdots \cup \Pi_{(k)}$.

Definition 3.1 (cf. [4, Definition 4.1]) Let $M = (Q, \Sigma, \delta, F)$ be a nta, and let $\mathcal{R}$ be an equivalence relation on Q. We say that $\mathcal{R}$ is a *backward bisimulation on* M if for every $(p, q) \in \mathcal{R}$, symbol f of $\Sigma_{(k)}$, and $L \in (Q/\mathcal{R})_{(k)}$

$$\bigvee_{w \in L} f(w) \xrightarrow{\delta} p \quad \text{if and only if} \quad \bigvee_{w \in L} f(w) \xrightarrow{\delta} q \ . \qquad \square$$

Example 3.2 Suppose we want to recognise the tree language $L = \{f[a, b], f[a, a]\}$ over the ranked alphabet $\Sigma = \Sigma_{(2)} \cup \Sigma_{(0)}$ with $\Sigma_{(2)} = \{f\}$ and $\Sigma_{(0)} = \{a, b\}$. A canonical way to obtain a nta N recognising L is to construct nta N_1 and N_2 that recognise only $f[a, b]$ and $f[a, a]$, respectively. Then we construct N by disjoint union of N_1 and N_2. In this manner we could obtain the nta $N = ([1, 6], \Sigma, \delta, \{3, 6\})$ with

$$a() \xrightarrow{\delta} 1 \qquad b() \xrightarrow{\delta} 2 \qquad f(1, 2) \xrightarrow{\delta} 3 \qquad a() \xrightarrow{\delta} 4 \qquad a() \xrightarrow{\delta} 5 \qquad f(4, 5) \xrightarrow{\delta} 6 \ .$$

Let $\mathcal{P} = \{1, 4, 5\}^2 \cup \{2\}^2 \cup \{3\}^2 \cup \{6\}^2$. We claim that $\mathcal{P}$ is a backward bisimulation on N. In fact, we only need to check the transitions leading to 1, 4, or 5 in order to justify the claim. Trivially, the condition of Definition 3.1 is met for such transitions because (i) $a() \to q$ is in δ and (ii) $b() \to q$ is not in δ for every state $q \in \{1, 4, 5\}$. $\square$

Next we describe how a nta $M = (Q, \Sigma, \delta, F)$ may be collapsed with respect to an equivalence relation $\mathcal{R}$ on Q. In particular, we will invoke this construction for some $\mathcal{R}$ that is a backward (in the current section) or forward (in Section 4) bisimulation on M.

Definition 3.3 (cf. [4, Definition 3.3]) Let $M = (Q, \Sigma, \delta, F)$ be a nta, and let $\mathcal{R}$ be an equivalence relation on Q. The *aggregated nta (with respect to M and $\mathcal{R}$)*, denoted by $(M/\mathcal{R})$, is the nta $((Q/\mathcal{R}), \Sigma, \delta', F')$ given by $F' = \{[q] \mid q \in F\}$ and

$$\delta' = \{f([q_1], \ldots, [q_k]) \to [q] \mid f(q_1, \ldots, q_k) \xrightarrow{\delta} q\} \ . \qquad \square$$

The nta $(M/\mathcal{R})$ has as many states as there are equivalence classes with respect to $\mathcal{R}$. Thus $(M/\mathcal{R})$ cannot have more states than M. Clearly, our aim is to collapse as many states as possible into a single state.

Example 3.4 Let N be the nta and $\mathcal{P}$ the backward bisimulation of Example 3.2. According to Definition 3.3, the aggregated nta $(N/\mathcal{P})$, which should recognise the language $\{f[a, b], f[a, a]\}$, is $(Q', \Sigma, \delta', F')$ where $Q' = \{[1], [2], [3], [6]\}$ and $F' = \{[3], [6]\}$ and

$$a() \xrightarrow{\delta'} [1] \qquad b() \xrightarrow{\delta'} [2] \qquad f([1], [2]) \xrightarrow{\delta'} [3] \qquad f([1], [1]) \xrightarrow{\delta'} [6] \ . \qquad \square$$

For the rest of this section we let M be an arbitrary but fixed nta and $\mathcal{R}$ be a backward bisimulation on M. Next we prepare for Corollary 3.7, which follows from Lemma 3.5. This Corollary shows that M and $(M/\mathcal{R})$ recognise the same tree language. The linking property is that the states q and $[q]$ (in their respective nta) recognise the same tree language. In fact, this also proves that bisimilar states in M recognise the same tree language.

Lemma 3.5 (cf. [4, Theorem 4.2] and [3, Lemma 5.2]) For any state q of M we have $\mathcal{L}((M/\mathcal{R}))_{[q]} = \mathcal{L}(M)_q$.

Proof Let $(M/\mathcal{R}) = (Q', \Sigma, \delta', F')$. We inductively prove the statement for every tree t of T_Σ. Suppose that $t = f[t_1, \ldots, t_k]$ for some symbol f in $\Sigma_{(k)}$ and trees $t_1, \ldots, t_k$ from T_Σ.

$$\begin{aligned}
& f[t_1, \ldots, t_k] \in \mathcal{L}((M/\mathcal{R}))_{[q]} \\
\iff & [q] \in \delta'(f[t_1, \ldots, t_k]) \\
\iff & \bigvee_{q_1, \ldots, q_k \in Q} \Big(f([q_1], \ldots, [q_k]) \xrightarrow{\delta'} [q] \wedge \bigwedge_{i \in [1,k]} [q_i] \in \delta'(t_i) \Big) \\
\iff & \text{(by Definitions 3.1 and 3.3 and induction hypothesis applied } k \text{ times)} \\
& \bigvee_{q_1, \ldots, q_k \in Q} \Big(f(q_1, \ldots, q_k) \xrightarrow{\delta} q \wedge \bigwedge_{i \in [1,k]} q_i \in \delta(t_1) \Big) \\
\iff & q \in \delta(f[t_1, \ldots, t_k]) \\
\iff & f[t_1, \ldots, t_k] \in \mathcal{L}(M)_q
\end{aligned}$$

■

Let us make the important property that backward bisimilar states recognise the tree language explicit here. Essentially, this property yields that deterministic nta (without useless states) admit only the identity as backward bisimulation because no two states in a deterministic nta recognise the same tree language.

Corollary 3.6 (of Lemma 3.5) For every $(p, q) \in \mathcal{R}$ we have $\mathcal{L}(M)_p = \mathcal{L}(M)_q$. $\square$

With the help of Lemma 3.5 we can now prove that M and $(M/\mathcal{R})$ recognise the same tree language.

Corollary 3.7 (cf. [4, Theorem 4.2] and [3, Lemma 5.3]) $\mathcal{L}((M/\mathcal{R})) = \mathcal{L}(M)$.

Proof A tree t is in $\mathcal{L}((M/\mathcal{R}))$ if and only if there exists a state q of M such that $q \in F$ and $t \in \mathcal{L}((M/\mathcal{R}))_{[q]}$. By Lemma 3.5, the latter holds precisely when there exists an accepting state q of M such that $t \in \mathcal{L}(M)_q$, which is equivalent to $t \in \mathcal{L}(M)$. ∎

Clearly, among all backward bisimulations on M the coarsest one yields the smallest aggregated nta. Let us show that such a coarsest backward bisimulation on M exists. We prove this statement by showing that for every two backward bisimulations on M there exists a backward bisimulation on M that is coarser than both.

Lemma 3.8 (cf. [4, Theorem 3.5]) Let $\mathcal{R}$ and $\mathcal{P}$ be backward bisimulations on M. Then there exists a backward bisimulation $\mathcal{R}'$ on M such that $\mathcal{R} \cup \mathcal{P} \subseteq \mathcal{R}'$.

Proof Let $\mathcal{R}'$ be the smallest equivalence containing $\mathcal{R} \cup \mathcal{P}$. We now show that $\mathcal{R}'$ is a backward bisimulation on M. Let $(p, q) \in \mathcal{R}'$. Thus there exist $n \in \mathbb{N}$ and

$$(p_1, p_2), (p_2, p_3), \ldots, (p_{n-2}, p_{n-1}), (p_{n-1}, p_n) \in \mathcal{R} \cup \mathcal{P}$$

such that $p_1 = p$ and $p_n = q$. Clearly, every block $D \in (Q/\mathcal{R}')$ is a union of blocks of $(Q/\mathcal{R})$ as well as a union of blocks of $(Q/\mathcal{P})$. Thus the condition of Definition 3.1 is trivially met for all $(p_i, p_{i+1}) \in \mathcal{R} \cup \mathcal{P}$ with $i \in [1, n-1]$. Now we verify the condition for p and q. Thus, let f be a symbol of $\Sigma_{(k)}$, and $L \in (Q/\mathcal{R}')_{(k)}$.

$$\bigvee_{w \in L} f(w) \xrightarrow{\delta} p_1 \iff \bigvee_{w \in L} f(w) \xrightarrow{\delta} p_2 \iff \ldots \iff \bigvee_{w \in L} f(w) \xrightarrow{\delta} p_{n-1} \iff \bigvee_{w \in L} f(w) \xrightarrow{\delta} p_n \quad ∎$$

Finally, let us conclude that the coarsest backward bisimulation $\mathcal{P}$ on M exists and that the identity is the only backward bisimulation on $(M/\mathcal{P})$.

Theorem 3.9 (cf. [4, Theorem 3.2]) There exists a coarsest backward bisimulation $\mathcal{P}$ on the nta M, and the identity is the only backward bisimulation on $(M/\mathcal{P})$.

Proof The existence of the coarsest backward bisimulation on M is a direct consequence of Lemma 3.8 because only finitely many backward bisimulations may exist on M. We prove the latter statement by contradiction. Suppose that $(M/\mathcal{P}) = (Q', \Sigma, \delta', F')$ admits a backward bisimulation $\mathcal{R}'$ that is not the identity. Whenever we write $[p]$ with $p \in Q$, we mean $[p]_{\mathcal{P}}$. We construct the relation $\mathcal{P}' = \{(p, q) \mid ([p], [q]) \in \mathcal{R}'\}$. Trivially, $\mathcal{P}'$ is an equivalence relation on Q. Further, we claim that $\mathcal{P}'$ is a backward bisimulation on M. To validate the claim, let $(p, q) \in \mathcal{P}'$. Moreover, let $f \in \Sigma_{(k)}$ be a symbol and $D_1 \cdots D_k \in (Q/\mathcal{P}')^k$ be a sequence of blocks. Clearly, for every $i \in [1, k]$ the block D_i is a union $D_{i,1} \cup \cdots \cup D_{i,n_i}$ of pairwise different blocks $D_{i,1}, \ldots, D_{i,n_i} \in (Q/\mathcal{P})$. Moreover, $D'_i = \{D_{i,1}, \ldots, D_{i,n_i}\}$ is a block in $(Q'/\mathcal{R}')$.

$$\bigvee_{w \in D_1 \cdots D_k} f(w) \xrightarrow{\delta} p \iff \bigvee_{D''_1 \cdots D''_k \in D'_1 \cdots D'_k} \Big(\bigvee_{w \in D''_1 \cdots D''_k} f(w) \xrightarrow{\delta} p \Big)$$

$$\iff \bigvee_{D''_1 \cdots D''_k \in D'_1 \cdots D'_k} f(D''_1, \ldots, D''_k) \xrightarrow{\delta'} [p] \qquad \text{(by Definition 3.3)}$$

$$\iff \bigvee_{D''_1 \cdots D''_k \in D'_1 \cdots D'_k} f(D''_1, \ldots, D''_k) \xrightarrow{\delta'} [q] \qquad \text{(by } ([p],[q]) \in \mathcal{R}'\text{)}$$

$$\iff \bigvee_{D''_1 \cdots D''_k \in D'_1 \cdots D'_k} \Big(\bigvee_{w \in D''_1 \cdots D''_k} f(w) \xrightarrow{\delta} q \Big) \qquad \text{(by Definition 3.3)}$$

$$\iff \bigvee_{w \in D_1 \cdots D_k} f(w) \xrightarrow{\delta} q$$

Thus $\mathcal{P}'$ is a backward bisimulation on M. Moreover, $\mathcal{P}'$ is coarser than $\mathcal{P}$, and since $\mathcal{R}'$ is not the identity it follows that $\mathcal{P} \subset \mathcal{P}'$. This contradicts the assumption that $\mathcal{P}$ is the coarsest backward bisimulation on M. Consequently, the identity is the only backward bisimulation on $(M/\mathcal{P})$. ∎

3.2 Minimisation algorithm

We now present a minimisation algorithm for nta that draws on the ideas presented in Section 3.1. Algorithm 1 searches for the coarsest backward bisimulation $\mathcal{R}$ on the input nta M by producing increasingly refined equivalence relations $\mathcal{R}_0, \mathcal{R}_1, \mathcal{R}_2, \ldots$. The first of these is the coarsest possible candidate solution (see Definition 3.17). The relation $\mathcal{R}_{i+1}$ is derived from $\mathcal{R}_i$ by removing pairs of states that prevent $\mathcal{R}_i$ from being a backward bisimulation. The algorithm also produces an auxiliary sequence of relations $\mathcal{P}_0, \mathcal{P}_1, \mathcal{P}_2, \ldots$ that are used to find these offending pairs. The algorithm terminates when $\mathcal{R}_i$ and $\mathcal{P}_i$ coincide, as $\mathcal{R}_i$ is then the coarsest backward bisimulation of M.

Before we discuss the algorithm, its correctness, and its time complexity, we extend our notation. For the rest of this section, let M be an arbitrary but fixed nta and $\hat{r} = \max\{k \mid \Sigma_{(k)} \neq \emptyset\}$.

Definition 3.10 For every $q \in Q$ and $k \in \mathbb{N}$ let $obs^k_q : \Sigma_{(k)} \times \mathfrak{P}(Q)^k \to \mathbb{N}$ be the mapping given for every $f \in \Sigma_{(k)}$ and $D_1 \cdots D_k \in \mathfrak{P}(Q)^k$ by

$$obs^k_q(f, D_1 \cdots D_k) = |\{q_1 \cdots q_k \in D_1 \cdots D_k \mid f(q_1, \ldots, q_k) \xrightarrow{\delta} q\}| \ . \qquad \square$$

Intuitively, $obs^k_q(f, D_1 \cdots D_k)$, the *observation*, is a count of the number of f-transitions that lead from blocks $D_1, \ldots, D_k$ to q, and thus a local observation of the properties of q (cf. Definition 3.1). As we will shortly see, we discard (q, q') from our maintained set of bisimilar states should obs^k_q and $obs^k_{q'}$ disagree in the sense that one is positive whereas the other is zero.

Definition 3.11 Let B, C be subsets of Q, $i \in \mathbb{N}$, and $L, L' \subseteq \mathfrak{P}(Q)^*$ be languages.

- The notation L_i will abbreviate $(Q/\mathcal{P}_i)^0 \cup \cdots \cup (Q/\mathcal{P}_i)^{\hat{r}}$.
- We use $L(B)$ to abbreviate $\{D_1 \cdots D_k \in L \mid D_i = B \text{ for some } i \in [1,k]\}$.
- We let $L(B, \neg C) = \{D_1 \cdots D_k \in L \mid D_1 \cdots D_k \in L(B) \text{ and } D_i \neq C \text{ for all } i \in [1,k]\}$.
- We write $cut(B)$ for the subset $(Q^2 \setminus B^2) \setminus (Q \setminus B)^2$ of $Q \times Q$.
- We write $split(L)$ for the set of all (q, q') in $Q \times Q$ for which there exist $f \in \Sigma_{(k)}$ and a word $w \in L$ of length k such that exactly one of $obs^k_q(f, w)$ and $obs^k_{q'}(f, w)$ is zero.
- We write L^w, where $w = D_1 \cdots D_k \in L'$, to denote set of words in L of the form $C_1 \cdots C_k$ where $C_i \subseteq D_i$, for all $i \in [1,k]$.

- Finally, we write $splitn(L', L)$ for the set of all (q, q') in $Q \times Q$ such that there exist a symbol f in $\Sigma_{(k)}$ and a word $w = D_1 \cdots D_k \in L'$ of length k such that

$$obs_p^k(f, w) = \sum_{C_1 \cdots C_k \in L^w} obs_p^k(f, C_1 \cdots C_k)$$

holds for either $p = q$ or $p = q'$ but not both. □

```
input:    a nta M = (Q, Σ, δ, F);

initially:
    P_0   := Q × Q;
    R_0   := P_0 \ split(L_0);
    i     := 0;

while R_i ≠ P_i:
    choose S_i ∈ (Q/P_i) and B_i ∈ (Q/R_i) such that
           B_i ⊂ S_i and |B_i| ≤ |S_i| /2;

    P_{i+1} := P_i \ cut(B_i);
    R_{i+1} := (R_i \ split(L_{i+1}(B_i))) \ splitn(L_i(S_i), L_{i+1}(B_i));
    i       := i + 1;

return:   the nta (M/R_i);
```

Algorithm 1: A minimisation algorithm for non-deterministic tree automata.

Let us briefly discuss how the sets L_0, L_1, L_2, ... that are generated by Algorithm 1 relate to each other. The set L_0 contains a single word of length k, for each $k \in [0, \hat{r}]$, namely Q^k. Every word w of length k in the set L_{i+1} is in either in L_i, or of the form $D_1 \cdots D_k$, where $D_j \in \{B_i, S_i \setminus B_i\}$ for some $j \in [1, k]$.

Example 3.12 We trace the execution of the minimisation algorithm on the automaton N of Example 3.2. Let us start with the initialisation. State 2 can be separated from $[1, 6]$ since only obs_2^0 is non-zero for the symbol b and the empty word $\epsilon \in \mathrm{L}_0$. Similarly, states 3 and 6 differ from 1, 4, and 5, as obs_3^2 and obs_6^2 are both non-zero for the symbol f and word QQ. Thus we have:

$$\mathcal{P}_0 = Q \times Q \text{ and } \mathcal{R}_0 = \{1, 4, 5\}^2 \cup \{2\}^2 \cup \{3, 6\}^2 \ .$$

In the first iteration, we let $S_0 = Q$ and $B_0 = \{2\}$. The algorithm can now use the symbol f and word $w = (Q \setminus \{2\})\{2\}$ in $\mathrm{L}_1(B_1)$ to distinguish between state 3 and state 6, as $obs_3^2(f, w) > 0$ whereas $obs_6^2(f, w) = 0$. The next pair of relations is then:

$$\mathcal{P}_1 = \{2\}^2 \cup (Q \setminus \{2\})^2 \text{ and } \mathcal{R}_1 = \{1, 4, 5\}^2 \cup \{2\}^2 \cup \{3\}^2 \cup \{6\}^2 \ .$$

As the states in $\{1, 4, 5\}$ do not appear at the left-hand side of any transition, this block will not be further divided. However, another two iterations are needed before $\mathcal{P}_3$ equals $\mathcal{R}_3$. □

Next we establish the that algorithm really computes the coarsest backward bisimulation on M. We use the notations introduced in the algorithm.

Lemma 3.13 The relation $\mathcal{R}_i$ is a refinement of $\mathcal{P}_i$, for all $i \in \{0, 1, 2, \dots\}$.

Proof The proof is by induction on i. The base case is satisfied by the initialisation of $\mathcal{P}_0$ to $Q \times Q$. For the induction step, we proceed as follows. By definition, $\mathcal{R}_{i+1} \subseteq \mathcal{R}_i$ and $\mathcal{P}_{i+1} = \mathcal{P}_i \setminus cut(B_i)$. Since $B_i \in (Q/\mathcal{R}_i)$, we also have the equality $\mathcal{R}_i \cap cut(B_i) = \emptyset$, and by the induction hypothesis, the inclusion $\mathcal{R}_i \subseteq \mathcal{P}_i$. It follows that

$$\mathcal{R}_{i+1} \;\subseteq\; \mathcal{R}_i \;=\; \mathcal{R}_i \setminus cut(B_i) \;\subseteq\; \mathcal{P}_i \setminus cut(B_i) \;=\; \mathcal{P}_{i+1} \;. \qquad \blacksquare$$

Lemma 3.13 assures that $\mathcal{R}_i$ is a proper refinement of $\mathcal{P}_i$, for all $i \in \{0, \ldots, t-1\}$ where t is the value of i at termination. Up to the termination point t, we can always find blocks $B_i \in (Q/\mathcal{R}_i)$ and $S_i \in (Q/\mathcal{P}_i)$ such that B_i is contained in S_i, and the size of B_i is at most half of that of S_i. This means that checking the termination criterion can be combined with the choice of S_i and B_i, because we can only fail to choose these blocks if $\mathcal{R}$ and $\mathcal{P}$ are equal. Lemma 3.14 captures this line of thought.

Lemma 3.14 There exists a $t < |Q|$ such that $\mathcal{R}_t = \mathcal{P}_t$.

Proof Clearly, the algorithm only terminates if $\mathcal{R}_t$ and $\mathcal{P}_t$ coincides for some t in $\mathbb{N}$. Up until termination, i.e. for all i less than t, we have that

$$|(Q/\mathcal{R}_i)| > |(Q/\mathcal{P}_i)| \quad \text{and} \quad |(Q/\mathcal{P}_{i+1})| > |(Q/\mathcal{P}_i)|$$

hold by Lemma 3.13. The size of both $(Q/\mathcal{R}_i)$ and $(Q/\mathcal{P}_i)$ is bound from above by $|Q|$. Should the algorithm reach iteration $|Q|-1$ before terminating, we have by necessity that both $|(Q/\mathcal{P}_{|Q|-1})|$ and $|(Q/\mathcal{R}_{|Q|-1})|$ are equal to $|Q|$, so $\mathcal{R}_{|Q|-1}$ and $\mathcal{P}_{|Q|-1}$ coincide. Consequently, there exists an integer t less than $|Q|$ such that $\mathcal{R}_t$ and $\mathcal{P}_t$ are equal. $\blacksquare$

Observation 3.15 For every i in $[0, t-1]$ and word w of length k in $\mathrm{L}_i(S_i)$, it holds that

$$obs_q^k(f, w) = \sum_{q_1 \cdots q_k \in \mathrm{L}_{i+1}^w(B_i)} obs_q^k(f, q_1 \cdots q_k) \;+ \sum_{q_1 \cdots q_k \in \mathrm{L}_{i+1}^w(S_i \setminus B_i, \neg B_i)} obs_q^k(f, q_1 \cdots q_k) \;.$$

Observation 3.16 For every $D_1 \cdots D_k$ in $\mathrm{L}_i(S_i)$, where i in $[0, t-1]$, there is a unique word $D'_1 \cdots D'_k$ in $\mathrm{L}_{i+1}(S_i \setminus B_i, \neg B_i)$ such that $D'_j \subseteq D_j$ for all $j \in [1, k]$.

Definition 3.17 Let $\mathcal{R}$ and $\mathcal{P}$ be two equivalence relations on Q such that $\mathcal{P}$ is coarser than $\mathcal{R}$. We say that $\mathcal{R}$ *is stable with respect to* $\mathcal{P}$ if, for every pair (q, q') in $\mathcal{R}$, $obs_q^k(f, w)$ is zero if and only if $obs_{q'}^k(f, w)$ is zero, for every $f \in \Sigma_{(k)}$ and $w \in (Q/\mathcal{P})^k$. We say that $\mathcal{R}$ is *stable* if it is stable with respect to itself. $\square$

Lemma 3.18 The relation $\mathcal{R}_i$ is stable with respect to $\mathcal{P}_i$, for all $i \in \{0, \ldots, t\}$.

Proof By Lemma 3.13, the relation $\mathcal{P}_i$ is coarser than $\mathcal{R}_i$. The remaining proof is by induction on i. The base case follows from the definitions of $\mathcal{R}_0$ and $\mathcal{P}_0$. Now, let (q, q') be a pair in $\mathcal{R}_{i+1}$, and let f be a symbol in $\Sigma_{(k)}$. We show that $obs_q^k(f, w)$ is zero if and only if $obs_{q'}^k(f, w)$ is zero, for every $f \in \Sigma$ and $w \in \mathrm{L}_{i+1}$. Let f be a symbol in Σ and let w be a word in L_{i+1}. Depending on w, there are three cases, and we examine each of them.

The first case is when w is in L_i. The fact that (q, q') is an element of $\mathcal{R}_{i+1}$ means that (q, q') is also an element of the coarser relation $\mathcal{R}_i$. Supporting ourselves on the induction hypothesis, we have that $obs_q^k(f, w)$ is zero if only if $obs_{q'}^k(f, w)$ is zero.

Alternatively, w is in $\mathrm{L}_{i+1}(B_i)$, and here the desired equality follows from the fact that (q, q') is not in $\mathit{split}(\mathrm{L}_{i+1}(B_i))$.

The third and remaining case is when w is in $\mathrm{L}_{i+1}(S_i \setminus B_i, \neg B_i)$. We compute as follows.

$$
\begin{aligned}
& \mathit{obs}_q^k(f, w) = 0 \\
\iff & \mathit{obs}_q^k(f, [w]_i) = \sum_{w' \in \mathrm{L}_{i+1}^{[w]_i}(B)} \mathit{obs}_q^k(f, w') && \text{(by Obs. 3.16 and 3.15)} \\
\iff & \mathit{obs}_{q'}^k(f, [w]_i) = \sum_{w' \in \mathrm{L}_{i+1}^{[w]_i}(B)} \mathit{obs}_{q'}^k(f, w') && \text{(as } (q, q') \text{ is in } \mathcal{R}_{i+1}\text{)} \\
\iff & \mathit{obs}_{q'}^k(f, w) = 0 && \text{(by Obs. 3.16 and 3.15) .}
\end{aligned}
$$

■

Lemma 3.19 Let $\mathcal{R}$ be a stable relation, and let $D_1, \ldots, D_k$ be a sequence of sets of states, each the union of one or more blocks in $(Q/\mathcal{R})$. For every $(q, q') \in \mathcal{R}$, we have that $\mathit{obs}_q^k(f, D_1 \cdots D_k)$ is zero if and only if $\mathit{obs}_{q'}^k(f, D_1 \cdots D_k)$ is zero, for every $f \in \Sigma$.

Proof Let $\mathcal{R}$ be a stable relation, let $D_1, \ldots, D_k$ be a sequence of sets of states, each the union of one or more blocks in $(Q/\mathcal{R})$, let (q, q') be a pair of states in $\mathcal{R}$, and let f be a symbol in Σ. We compute as follows:

$$
\begin{aligned}
& \mathit{obs}_q^k(f, D_1 \cdots D_k) = 0 \\
\iff & \text{(by definition of the } \mathit{obs}_q^k \text{ function)} \\
& \{q_1 \cdots q_k \in D_1 \cdots D_k \mid f(q_1, \ldots, q_k) \xrightarrow{\delta} q\} = \emptyset \\
& \bigcup_{C_1 \cdots C_k \in (D_1/\mathcal{R}) \cdots (D_k/\mathcal{R})} \{q_1 \cdots q_k \in C_1 \cdots C_k \mid f(q_1, \ldots, q_k) \xrightarrow{\delta} q\} = \emptyset \\
\iff & \text{(because } (q, q') \text{ is in } \mathcal{R}\text{, and } \mathcal{R} \text{ is a stable relation)} \\
& \bigcup_{C_1 \cdots C_k \in (D_1/\mathcal{R}) \cdots (D_k/\mathcal{R})} \{q_1 \cdots q_k \in C_1 \cdots C_k \mid f(q_1, \ldots, q_k) \xrightarrow{\delta} q'\} = \emptyset \\
\iff & \text{(since } D_i \text{ is the union of one or more blocks in } (Q/\mathcal{R})\text{)} \\
& \{q_1 \cdots q_k \in D_1 \cdots D_k \mid f(q_1, \ldots, q_k) \xrightarrow{\delta} q'\} = \emptyset \\
\iff & \text{(by definition of the } \mathit{obs}_{q'}^k \text{ function)} \\
& \mathit{obs}_{q'}^k(f, D_1 \cdots D_k) = 0
\end{aligned}
$$

■

Lemma 3.20 If an equivalence relation $\mathcal{R}$ is a stable refinement of $\mathcal{R}_0$, then $\mathcal{R}$ is also a refinement of $\mathcal{R}_i$ for every $i \in \{0, \ldots, t\}$.

Proof The proof is by induction on i, and the base case is trivial. To cover the induction step, we show that if (q, q') is in $\mathcal{R}$, then (q, q') is also in $\mathcal{R}_{i+1}$. This is done by examining how the minimisation algorithm obtains $\mathcal{R}_{i+1}$ from $\mathcal{R}_i$. It is required that (q, q') is in $\mathcal{R}_i$, and this is satisfied by the induction hypothesis. Moreover, it must hold that $\mathit{obs}_q^k(f, w)$ is zero if and only if $\mathit{obs}_{q'}^k(f, w)$ is zero, for every symbol $f \in \Sigma_{(k)}$, and word $w = D_1 \cdots D_k$ in $\mathrm{L}_{i+1}(B_i)$. Since D_i is the union of one or more blocks in $(Q/\mathcal{R}_i)$ and hence in $(Q/\mathcal{R})$, for all $i \in [1, k]$, this condition is satisfied by Lemma 3.19.

The following computation shows that (q, q') is not in $\mathit{split}(\mathrm{L}_{i+1}(S_i \setminus B_i, \neg B_i))$. By Observation 3.16 we can write $\mathrm{L}^w_{i+1}(S_i \setminus B_i, \neg B_i)$ as $\{w''\}$.

$$
\begin{aligned}
& obs^k_q(f, w) = \sum_{w' \in \mathrm{L}^w_{i+1}(B)} obs^k_q(f, w') \\
\iff & \sum_{w' \in \mathrm{L}^w_{i+1}(S_i \setminus B_i, \neg B_i)} obs^k_q(f, w') = 0 && \text{(by Observation 3.15)} \\
\iff & obs^k_q(f, w'') = 0 && \text{(by Observation 3.16)} \\
\iff & obs^k_{q'}(f, w'') = 0 && \text{(by Lemma 3.19)} \\
\iff & \sum_{w' \in \mathrm{L}^w_{i+1}(S_i \setminus B_i, \neg B_i)} obs^k_{q'}(f, w') = 0 && \text{(by Observation 3.16)} \\
\iff & obs^k_{q'}(f, w) = \sum_{w' \in \mathrm{L}^w_{i+1}(B)} obs^k_{q'}(f, w') && \text{(by Observation 3.15)}
\end{aligned}
$$

■

Theorem 3.21 $\mathcal{R}_t$ is the coarsest backward bisimulation on M.

Proof Lemma 3.14 guarantees that the algorithm terminates and Lemma 3.18 that $\mathcal{R}_t$ is stable with respect to $\mathcal{P}_t$. Since $\mathcal{R}_t$ is equal to $\mathcal{P}_t$, the relation $\mathcal{R}_t$ is stable. In combination with Lemma 3.20, we obtain the main result of this section. ■

Let us now analyse the running time of the minimisation algorithm on M. We use n and m to denote the size of the sets Q and δ, respectively. In the complexity calculations, we write δ_L, where $L \subseteq \mathfrak{P}(Q)^*$, for the subset of δ that contains entries of the form $f(q_1, \ldots, q_k) \to q$, where $f \in \Sigma_{(k)}$, $q \in Q$, and $q_1 \cdots q_k$ is in $B_1 \cdots B_k$ for some $B_1 \cdots B_k \in L$. Our computation model is the random access machine [15], which supports indirect addressing, and thus allows the use of pointers. This means that we can represent each block in a partition $(Q/\mathcal{R})$ as a record of two-way pointers to its elements, and that we can link each state to its occurrences in the transition table. Given a state q and a block B, we can then determine $[q]_\mathcal{R}$ in constant time, and δ_L, where $L \subseteq \mathfrak{P}(Q)^*$, in time proportional to the number of entries.

To avoid pairwise comparison between states, we hash each state q in Q using $(obs^k_q)_{k \in [0, \hat{r}]}$ as key, and then inspect which states end up at the same positions in the hash table. Since a random access machine has unlimited memory, we can always implement a collision free hash h; i.e., by interpreting the binary representation of $(obs^k_q)_{k \in [0, \hat{r}]}$ as a memory address, and the time required to hash a state q is then proportional to the size of the representation of $(obs^k_q)_{k \in [0, \hat{r}]}$.

During the complexity calculations we maintain the following data structures:

$\mathcal{R}$-*blocks* A linked list where each entry represents a block in the partition $(Q/\mathcal{R}_i)$, i being the current iteration of the algorithm. Initially, $\mathcal{R}$-*blocks* contains the entries F and $Q \setminus F$.

$\mathcal{P}$-*blocks* A linked list where each entry S represents a block in the partition $(Q/\mathcal{P}_i)$. S contains a pointer to each entry B in $\mathcal{R}$-*blocks* such that $B \subseteq S$, labelled with the size of B. Initially, $\mathcal{P}$-*blocks* contains the single block Q, which has pointers to F and $Q \setminus F$ labelled with $|F|$ and $|Q \setminus F|$, respectively.

$\mathcal{P}$-*compound* A linked list of pointers to those blocks in $\mathcal{P}$-*blocks* that are composed of more than one block in $(Q/\mathcal{R}_i)$. This list is empty only if $\mathcal{R}_i = \mathcal{P}_i$.

Observation 3.22 The overall time complexity of the algorithm is

$$O\Big(\textsc{Init} + \sum\nolimits_{i\in[0,t-1]} (\textsc{Select}_i + \textsc{Cut}_i + \textsc{Split}_i + \textsc{Splitn}_i) + \textsc{Aggregate}\Big) \ ,$$

where $\textsc{Init}$, $\textsc{Select}_i$, $\textsc{Cut}_i$, $\textsc{Split}_i$, $\textsc{Splitn}_i$, and $\textsc{Aggregate}$ are the complexity of: the initialisation phase; the choice of S_i and B_i; the computation of $\mathcal{P}_i \setminus \mathit{cut}(B_i)$; the computation of $\mathcal{R}_i \setminus \mathit{split}(\mathrm{L}_{i+1}(B_i))$; the subtraction of $\mathit{splitn}(\mathrm{L}_i(S_i), \mathrm{L}_{i+1}(B_i))$; and the construction of the aggregated automaton $(M/\mathcal{R}_t)$; respectively.

Lemma 3.23 $\textsc{Init}$ is in $O(\hat{r}m + n)$.

Proof The relation $\mathcal{P}_0$ can clearly be initialised in $O(n)$. In the computation of $\mathcal{P}_0 \setminus \mathit{split}(\mathrm{L}_0)$, the value of $\mathit{obs}^k_q(f, w)$ must be calculated for every $q \in Q$, $f \in \Sigma$, and $w \in \mathrm{L}_0$. By Lemma 3.27, this can be done in time $O\big(\hat{r}\,|\delta_{\mathrm{L}_0}|\big) = O(\hat{r}m)$. To make the upcoming calculation of $\textsc{Splitn}_1$ more efficient, we save the value of $\mathit{obs}^k_q(f, w)$ in the counter $\Delta^0_q(f, w)$. ∎

Observation 3.24 $\textsc{Select}_i$ is in $O(1)$.

Lemma 3.25 $\textsc{Cut}_i$ is in $O(|B_i|)$.

Proof The entry S_i is removed from $\mathcal{P}$-*compound*, and the corresponding entry S_i in $\mathcal{P}$-*blocks* is split into two entries, $S_i \setminus B_i$ and B_i, by iterating over the states pointed to by B_i in time $O(|B_i|)$. The entry representing $S_i \setminus B_i$ points to every block in $(Q/\mathcal{R}_i)$ that the entry representing S_i pointed to, except of course B_i. If $S_i \setminus B_i$ is still compound, then it is added to $\mathcal{P}$-*compound*. The entry representing B_i only points to B_i. ∎

Lemma 3.26 Let $\mathcal{R}$ and $\mathcal{P}$ be equivalence relations on Q. The computation of $\mathcal{R} \setminus \mathit{split}(L)$, where $L \subseteq (Q/\mathcal{P})^*$, is in $O\big(\hat{r}\,|\delta_L|\big)$.

Proof Let $L_{(k)} = \{w \in L \mid w \text{ is of length } k\}$. We begin the computation by adjusting the value of $\mathit{obs}^k_q(f, [q_1]_{\mathcal{P}_{i+1}} \cdots [q_k]_{\mathcal{P}_{i+1}})$ for each transition $f(q_1, \ldots, q_k) \to q$ in δ_L, at the cost of $O(\hat{r}\,|\delta_L|)$ computation steps. The elements of $\mathcal{R}_i$ that are to be retained in $\mathcal{R}_{i+1}$ can thereafter be determined by hashing each state q in Q using $(\mathit{obs}^k_q)_{k\in\mathbb{N}}$ on the domain $\cup_{k\in[0,\hat{r}]}(\Sigma_{(k)} \times L_{(k)})$ as key, and then keeping those pairs of states that end up at the same memory address.

We now make the observation that if the pair $f \in \Sigma_{(k)}$ and $D_1 \cdots D_k \in L_{(k)}$ is in the support of obs^k_q for some $q \in Q$, then there is a transition $f(q_1 \cdots q_k) \to q$ in δ such that $q_1 \cdots q_k \in D_1 \cdots D_k$. In other words, size of the support of $(\mathit{obs}^k_q)_{q\in Q, k\in\mathbb{N}}$ on the domain $\cup_{k\in[0,\hat{r}]}(\Sigma_{(k)} \times L_{(k)})$ does not exceed the size of δ_L. We conclude that the total time required to hash the states is determined by the time it takes to calculate their hash key. ∎

Lemma 3.27 is immediate from Lemma 3.26.

Lemma 3.27 $\textsc{Split}_i$ is in $O\big(\hat{r}\,|\delta_{\mathrm{L}_{i+1}(B_i)}|\big)$.

Lemma 3.28 $\textsc{Splitn}_i$ is in $O\big(\hat{r}\,|\delta_{\mathrm{L}_{i+1}(B_i)}|\big)$.

Proof In the proof, we assume that we have access to a family of functions $(\Delta^i_q)_{q\in Q}$ such that $\Delta^i_q(f, w) = \mathit{obs}^k_q(f, w)$ for every $f \in \Sigma$ and word w in L_i. The computation of Δ^0 is carried out in the initialisation phase and is discussed in the proof of Lemma 3.23.

To prepare for the computation of $\textsc{Splitn}_i$, the set $\delta_{\mathrm{L}_{i+1}(B_i)}$ is traversed, and for each transition $f(q_1 \cdots q_k) \to q$,

- $\Delta^i_q(f, [q_1]_{\mathcal{P}_i} \cdots [q_k]_{\mathcal{P}_i})$ is decreased by one, and
- $\Delta^{i+1}_q(f, [q_1]_{\mathcal{P}_{i+1}} \cdots [q_k]_{\mathcal{P}_{i+1}})$ is increased by one.

When the last element of $\delta_{\mathrm{L}_{i+1}(B_i)}$ has been examined, $\Delta^{i+1}_q(f, w')$ is identified with $\Delta^i_q(f, w)$, where $\{w'\} = \mathrm{L}^w_{i+1}(S_i \setminus B_i, \neg B_i)$, for every $w \in \mathrm{L}_i$. The computation so far is in $O\big(\hat{r}\,|\delta_{\mathrm{L}_{i+1}(B_i)}|\big)$. The subtraction of $splitn(\mathrm{L}_i(S_i), \mathrm{L}_{i+1}(B_i))$ from $\mathcal{R}_i$ can then be determined by hashing Q into equivalence classes, using the value of Δ^i_q on the domain

$$\{(f, [q_1]_{\mathcal{P}_i} \cdots [q_k]_{\mathcal{P}_i}) \mid f(q_1, \ldots, q_k) \to q \in \delta_{\mathrm{L}_{i+1}(B_i)}\}$$

as key for q. The total time needed to calculate $\textsc{Splitn}_i$ is thus in $O\big(\hat{r}\,|\delta_{\mathrm{L}_{i+1}(B_i)}|\big)$. ∎

Lemma 3.29 AGGREGATE is in $O(\hat{r}m + n)$.

Proof The complexity of deriving the aggregated wta $(M/\mathcal{R}_t) = (Q', \Sigma, \delta', F')$ is a follows: The alphabet Σ is identical that in M, and he components Q' and F' are both given by the entries in $\mathcal{R}$-*blocks*; to determine if a block in $\mathcal{R}$-*blocks* is final, just follow the pointer to one of its constituent states and check if that state is final. This list can be read in time $O(n)$. To obtain δ' it suffices to iterate over δ, as each state is linked to its equivalence class, and this requires another $O(\hat{r}m)$ computation steps. ∎

Lemma 3.30 For each $q \in Q$, we have that $|\{B_i \mid i \in \mathbb{N} \text{ and } q \in B_i\}| \leq \log n$.

Proof Let B_i and B_j, where $i < j$, be two blocks that both include the state q. Since $\mathcal{R}_j$ is a refinement of $\mathcal{R}_i$, we have that B_j is a subset of B_i. We know then that $|B_j|$ is less or equal to $|B_i|/2$, or else B_j would violate the selection criteria for the B-blocks. If we order the B-blocks in which q occurs in descending order (with respect to their size), we have that each block in the list is at most half the size of its predecessor. The first block in which q occurs cannot be larger than n, and the last block cannot be smaller than a singleton. Hence, the q is included in at most $\log n$ distinct B-blocks. ∎

Theorem 3.31 The backward minimisation algorithm is in $O\big(\hat{r}^2\, m \log n\big)$.

Proof By Observation 3.22 and 3.24, together with Lemmata 3.23, 3.25, and 3.27 through 3.29, the time complexity of the algorithm can be written as

$$O\Big((\hat{r}m + n) + \sum\nolimits_{i \in [0,t-1]} (1 + |B_i| + \hat{r}\,|\delta_{\mathrm{L}_{i+1}(B_i)}| + \hat{r}\,|\delta_{\mathrm{L}_{i+1}(B_i)}|) + (\hat{r}m + n)\Big)\ .$$

Omitting the smaller terms and simplifying, we obtain $O\left(\hat{r} \sum_{i \in [0,t-1]} |\delta_{\mathrm{L}_{i+1}(B_i)}|\right)$. According to Lemma 3.30, no state occurs in more than $\log n$ distinct B-blocks, so no transition in δ will contribute by more than $\hat{r} \log n$ to the total sum. As there are m transitions, the overall time complexity of the algorithm is $O(\hat{r}^2 m \log n)$. ∎

We next compare the presented backward bisimulation to the bisimulation of [2].

Definition 3.32 (cf. [2, Section 5]) Let $\mathcal{P}$ be an equivalence relation on Q. We say that $\mathcal{P}$ is an *AKH-bisimulation on* M, if for every $(p, q) \in \mathcal{P}$ we have (i) $p \in F$ if and only if $q \in F$; and (ii) for every symbol f in $\Sigma_{(k)}$, index $i \in [1, k+1]$, and sequence $D_1, \ldots, D_{k+1}$ of blocks in $(Q/\mathcal{P})$

$$\bigvee_{\substack{p_1 \cdots p_{k+1} \in D_1 \cdots D_{k+1}, \\ p_i = p}} f(p_1, \ldots, p_k) \xrightarrow{\delta} p_{k+1} \iff \bigvee_{\substack{q_1 \cdots q_{k+1} \in D_1 \cdots D_{k+1}, \\ q_i = q}} f(q_1, \ldots, q_k) \xrightarrow{\delta} q_{k+1}\ . \qquad \square$$

Lemma 3.33 Every AKH-bisimulation on M is a backward bisimulation on M.

Proof The proof is straightforward if we consider the index $i = k + 1$ in Definition 3.32. ∎

Clearly, the coarsest backward bisimulation $\mathcal{R}$ on M must be coarser than the coarsest AKH-bisimulation $\mathcal{P}$ on M. Hence $(M/\mathcal{R})$ has at most as many states as $(M/\mathcal{P})$. Since our algorithm for minimisation via backward bisimulation is computationally as efficient as the algorithm of [2] (see Theorem 3.31 and [2, Section 3]), it supersedes the minimisation algorithm of [2].

4 Forward bisimulation

4.1 Foundation

In this section we consider a computationally simpler notion of bisimulation. Minimisation via forward bisimulation coincides with classical minimisation on deterministic nta (see Theorem 4.25). In addition, the two minimisation procedures greatly increase their potential when they are used together in an alternating fashion (see Section 5). We will follow the programme of Section 3.

First we introduce contexts. A *context* is a string that contains the special symbol $\Box$ exactly once. Into a context a symbol may be substituted; the symbol replaces the special symbol $\Box$ in the string.

Definition 4.1 Let Q be a set such that $\Box \notin Q$. The set $C^Q_{(k)}$ of *contexts (over Q)* is given by

$$\{w \in (Q \cup \{\Box\})^k \mid w \text{ contains } \Box \text{ exactly once}\} \ ,$$

and for every $c \in C^Q_{(k)}$ and $q \in Q$ we write $c[\![q]\!]$ to denote the word that is obtained from c by replacing the special symbol $\Box$ with q. ▫

Henceforth, we assume that the special symbol $\Box$ occurs in no set of states of any nta. By a simple renaming of states this property can easily be guaranteed.

Definition 4.2 (cf. [4, Definition 3.1]) Let $M = (Q, \Sigma, \delta, F)$ be a nta, and let $\mathcal{R}$ be an equivalence relation on Q. We say that $\mathcal{R}$ is a *forward bisimulation on M* if for every (p, q) in $\mathcal{R}$ we have (i) $p \in F$ if and only if $q \in F$; and (ii) for every symbol f in $\Sigma_{(k)}$, context $c \in C^Q_{(k)}$, and block D in $(Q/\mathcal{R})$

$$\bigvee_{r \in D} f(c[\![p]\!]) \xrightarrow{\delta} r \quad \text{if and only if} \quad \bigvee_{r \in D} f(c[\![q]\!]) \xrightarrow{\delta} r \ . \qquad ▫$$

Note that Condition (ii) in Definition 4.2 is automatically fulfilled for all nullary symbols. Let us continue Example 3.4 (the aggregated nta is defined in Definition 3.3).

Example 4.3 Recall the aggregated nta from Example 3.4. An isomorphic nta N is given by $([1, 4], \Sigma, \delta, \{3, 4\})$ with

$$a() \xrightarrow{\delta} 1 \qquad b() \xrightarrow{\delta} 2 \qquad f(1, 2) \xrightarrow{\delta} 3 \qquad f(1, 1) \xrightarrow{\delta} 4 \ .$$

We have seen in Example 3.12 that the identity is the only backward bisimulation on N. Let us consider $\mathcal{P} = \{1\}^2 \cup \{2\}^2 \cup \{3, 4\}^2$. We claim that $\mathcal{P}$ is a forward bisimulation on N. Clearly, Condition (i) of Definition 4.2 is met, and thus it remains to check Condition (ii) for

the symbol f. This is also trivial, since $(1,2) \notin \mathcal{P}$ and the states 3 and 4 only appear on the right hand side of $\xrightarrow{\delta}$. Thus $\mathcal{P}$ is a forward bisimulation.
The aggregated nta $(N/\mathcal{P})$ is $(Q', \Sigma, \delta', F')$ with $Q' = \{[1],[2],[3]\}$ and $F' = \{[3]\}$ and

$$a() \xrightarrow{\delta'} [1] \qquad b() \xrightarrow{\delta'} [2] \qquad f([1],[2]) \xrightarrow{\delta'} [3] \qquad f([1],[1]) \xrightarrow{\delta'} [3] \ . \qquad \square$$

For the rest of this section we let M be an arbitrary but fixed nta and $\mathcal{R}$ be a forward bisimulation on M. In the forward case, a collapsed state of $(M/\mathcal{R})$ functions like the combination of its constituents in M (cf. Section 3.1). In particular, bisimilar states need not recognise the same tree language. However, $(M/\mathcal{R})$ and M do recognise the same tree language.

Lemma 4.4 (cf. [4, Theorem 3.1]) $\mathcal{L}((M/\mathcal{R}))_{[q]} = \bigcup_{p \in [q]} \mathcal{L}(M)_p$ for every state q of Q.

Proof Let $(M/\mathcal{R}) = (Q', \Sigma, \delta', F')$. We prove the statement by induction for every tree t. Suppose that $t = f[t_1, \ldots, t_k]$ for some symbol $f \in \Sigma_{(k)}$, and sequence of subtrees $t_1, \ldots, t_k \in \mathrm{T}_\Sigma$.

$$\begin{aligned}
& f[t_1, \ldots, t_k] \in \mathcal{L}((M/\mathcal{R}))_{[q]} \\
\iff & [q] \in \delta'(f[t_1, \ldots, t_k]) \\
\iff & \bigvee_{q_1,\ldots,q_k \in Q} \Big(f([q_1], \ldots, [q_k]) \xrightarrow{\delta'} [q] \Big) \wedge \bigwedge_{i \in [1,k]} [q_i] \in \delta'(t_i) \\
\iff & \text{(by induction hypothesis applied } k \text{ times)} \\
& \bigvee_{\substack{q_1,\ldots,q_k \in Q, \\ p_1 \cdots p_k \in [q_1] \cdots [q_k]}} \Big(f([q_1], \ldots, [q_k]) \xrightarrow{\delta'} [q] \Big) \wedge \bigwedge_{i \in [1,k]} p_i \in \delta(t_i) \\
\iff & \text{(by Definitions 4.2 and 3.3)} \\
& \bigvee_{\substack{p \in [q], \\ q_1,\ldots,q_k \in Q}} \Big(f(q_1, \ldots, q_k) \xrightarrow{\delta} p \Big) \wedge \bigwedge_{i \in [1,k]} q_i \in \delta(t_i) \\
\iff & \bigvee_{p \in [q]} p \in \delta(f[t_1, \ldots, t_k]) \\
\iff & f[t_1, \ldots, t_k] \in \bigcup_{p \in [q]} \mathcal{L}(M)_p
\end{aligned}$$

■

Corollary 4.5 (cf. [4, Corollary 3.4]) $\mathcal{L}((M/\mathcal{R})) = \mathcal{L}(M)$.

Proof Let $t \in \mathrm{T}_\Sigma$ be arbitrary. We have $t \in \mathcal{L}((M/\mathcal{R}))$ if and only if there exists a state q of M such that $q \in F$ and $t \in \mathcal{L}((M/\mathcal{R}))_{[q]}$. By Definitions 4.2 and 3.3 and Lemma 4.4, the latter holds if and only if there exists a state p of M such that $p \in F$ and $t \in \mathcal{L}(M)_p$. Clearly, this is exactly the case when $t \in \mathcal{L}(M)$. ■

The coarsest of all forward bisimulations on M yields the smallest aggregated nta, and this nta cannot be reduced further by collapsing it with respect to to some forward bisimulation. We again first that the coarsest forward bisimulation on M exists.

Lemma 4.6 (cf. [4, Theorem 3.5]) Let $\mathcal{R}$ and $\mathcal{P}$ be forward bisimulations on M. Then there exists a forward bisimulation $\mathcal{R}'$ on M such that $\mathcal{R} \cup \mathcal{P} \subseteq \mathcal{R}'$.

Proof Let $\mathcal{R}'$ be the smallest equivalence containing $\mathcal{R} \cup \mathcal{P}$. We now show that $\mathcal{R}'$ is a forward bisimulation on M. Let $(p, q) \in \mathcal{R}'$. Thus there exist an integer $n \in \mathbb{N}$ and

$$(p_1, p_2), (p_2, p_3), \ldots, (p_{n-2}, p_{n-1}), (p_{n-1}, p_n) \in \mathcal{R} \cup \mathcal{P}$$

such that $p_1 = p$ and $p_n = q$. Thus $p \in F$ if and only if $q \in F$. Now to Condition (ii) of Definition 4.2. Let f be a symbol of $\Sigma_{(k)}$, D a block in $(Q/\mathcal{R}')$, and $c \in C^Q_{(k)}$ be a context. Clearly, the block D is a union of blocks of $(Q/\mathcal{R})$ as well as a union of blocks of $(Q/\mathcal{P})$. Thus Condition (ii) of Definition 4.2 is trivially met for all $(p_i, p_{i+1}) \in \mathcal{R} \cup \mathcal{P}$. Now we verify the condition for p and q.

$$\bigvee_{r \in D} f(c[\![p_1]\!]) \xrightarrow{\delta} r \iff \bigvee_{r \in D} f(c[\![p_2]\!]) \xrightarrow{\delta} r \iff \ldots \iff \bigvee_{r \in D} f(c[\![p_n]\!]) \xrightarrow{\delta} r \qquad \blacksquare$$

Theorem 4.7 (cf. [4, Theorem 3.2]) There exists a coarsest forward bisimulation $\mathcal{P}$ on M, and the identity is the only forward bisimulation on $(M/\mathcal{P})$.

Proof The existence of the coarsest forward bisimulation $\mathcal{P}$ follows from Lemma 4.6. The second part of the statement is proved by contradiction, so suppose that $(M/\mathcal{P}) = (Q', \Sigma, \delta', F')$ admits a non-identity forward bisimulation $\mathcal{R}'$. Whenever we write $[p]$ (with $p \in Q$) in the remaining proof, we mean $[p]_{\mathcal{P}}$. Let $\mathcal{P}' = \{(p, q) \mid ([p], [q]) \in \mathcal{R}'\}$. Obviously, $\mathcal{P}'$ is an equivalence relation on Q. We claim that $\mathcal{P}'$ is a forward bisimulation on M. This would contradict the assumption that $\mathcal{P}$ is the coarsest forward bisimulation on M because $\mathcal{P} \subseteq \mathcal{P}'$ and since $\mathcal{R}'$ is not the identity it follows that $\mathcal{P} \subset \mathcal{P}'$.

Let $(p, q) \in \mathcal{P}'$. We start with Condition (i) of Definition 4.2. Since $([p], [q]) \in \mathcal{R}'$ we have $[p] \in F'$ if and only if $[q] \in F'$. Moreover, by Definition 3.3 we also have

$$p \in F \iff [p] \in F' \iff [q] \in F' \iff q \in F \ .$$

It remains to validate Condition (ii). Let $f \in \Sigma_{(k)}$ be a symbol, $c = q_1 \cdots q_{i-1} \Box q_{i+1} \cdots q_k$ be a context of $C^Q_{(k)}$ for some $q_1, \ldots, q_k \in Q$ and $i \in [1, k]$, and $D \in (Q/\mathcal{P}')$ be a block. Clearly, the block D is a union $D_1 \cup \cdots \cup D_n$ of pairwise different blocks $D_1, \ldots, D_n \in (Q/\mathcal{P})$. Moreover, $D' = \{D_1, \ldots, D_n\}$ is a block in $(Q'/\mathcal{R}')$. Finally, let $c' = [q_1] \cdots [q_{i-1}] \Box [q_{i+1}] \cdots [q_k]$, which is a context of $C^{Q'}_{(k)}$.

$$\begin{aligned}
&\bigvee_{r \in D} f(c[\![p]\!]) \xrightarrow{\delta} r \iff \bigvee_{D'' \in D'} \Big(\bigvee_{r \in D''} f(c[\![p]\!]) \xrightarrow{\delta} r \Big) \\
\iff &\bigvee_{D'' \in D'} f(c'[\![[p]]\!]) \xrightarrow{\delta'} D'' && \text{(by Definition 3.3)} \\
\iff &\bigvee_{D'' \in D'} f(c'[\![[q]]\!]) \xrightarrow{\delta'} D'' && \text{(because } ([p], [q]) \in \mathcal{R}') \\
\iff &\bigvee_{D'' \in D'} \Big(\bigvee_{r \in D''} f(c[\![q]\!]) \xrightarrow{\delta} r \Big) \iff \bigvee_{r \in D} f(c[\![q]\!]) \xrightarrow{\delta} r && \text{(by Definition 3.3)}
\end{aligned}$$

Thus $\mathcal{P}'$ is a strictly coarser forward bisimulation on M than $\mathcal{P}$, which contradicts our assumption. Hence the identity is the only forward bisimulation on $(M/\mathcal{P})$. $\blacksquare$

4.2 Minimisation algorithm

We now modify the algorithm of Section 3.2 so as to minimise with respect to forward bisimulation. As in Section 3.2 this requires us to extend our notation. We denote by C_Q^k the of set of *contexts* over Q: the set of k-tuples over $Q \cup \{\Box\}$ that contain the special symbol $\Box$ exactly once. We denote by $c[\![q]\!]$, where $c \in C_Q^k$ and $q \in Q$, the tuple that is obtained by replacing the unique occurrence of $\Box$ in c by q.

Definition 4.8 For each state q in Q and $k \in \mathbb{N}$, the mapping $obsf_q^k \colon \Sigma_{(k)} \times C_Q^k \times \mathfrak{P}(Q) \to \mathbb{N}$ is defined for every symbol $f \in \Sigma_{(k)}$, context $c \in C_Q^k$, and set $D \subseteq Q$ of states by

$$obsf_q^k(f, c, D) = |\{q' \in D \mid f(c[\![q]\!]) \xrightarrow{\delta} q'\}| \ . \qquad \Box$$

Like obs_q^k, $obsf_q^k$ is a local observation of the properties of q. The difference here, is that $obsf_q^k(f, c, D)$ is the number of f-transitions that match the sequence $c[\![q]\!]$ and lead to a state of D. In contrast, obs_q^k looked from the other side of the rule.

Definition 4.9 Let D and D' be subsets of Q.

- We write $splitf(D)$ for the set of all pairs (q, q') in $Q \times Q$, for which there exist $f \in \Sigma_{(k)}$ and $c \in C_Q^k$ such that exactly one of $obsf_q^k(f, c, D)$ and $obsf_{q'}^k(f, c, D)$ is non-zero.
- Similarly, we write $splitfn(D, D')$ for the set of all pairs (q, q') in $Q \times Q$, for which there exist $f \in \Sigma_{(k)}$ and $c \in C_Q^k$ such that $obsf_p^k(f, c, D) = obsf_p^k(f, c, D')$ holds for either $p = q$ or $p = q'$ but not both. $\Box$

We can now construct a minimisation algorithm based on forward bisimulation by replacing the initialisation of $\mathcal{R}_0$ in Algorithm 1 with $\mathcal{R}_0 = ((Q \setminus F)^2 \cup F^2) \setminus splitf(Q)$ and the computation of $\mathcal{R}_{i+1}$ with $\mathcal{R}_{i+1} = \big(\mathcal{R}_i \setminus splitf(B_i)\big) \setminus splitfn(S_i, B_i)$.

Example 4.10 We show the execution of the minimisation algorithm on the nta N from Example 4.3. In the initialisation of $\mathcal{R}_0$, states 3 and 4 are separated from the others as they are accepting. State 1 is distinguished as only $obsf_1^2$ is non-zero on the symbol f, context $(\Box, 2) \in C_{[1,4]}^2$, and block Q in $\mathcal{P}_0$. When the initialisation phase is complete, we have the relations $\mathcal{P}_0 = Q \times Q$ and $\mathcal{R}_0 = \{1\}^2 \cup \{2\}^2 \cup \{3, 4\}^2$. As neither state 3 nor state 4 appear on a left-hand side of any transition, the algorithm will not separate them, so $\mathcal{R}_0$ is already equal the final $\mathcal{R}$ relation. The termination point is reached in the second iteration, when $\mathcal{P}_0$ has been refined to $\mathcal{R}_0$. $\Box$

We now verify that the minimisation algorithm is correct. Note that Lemmata 3.13 and 3.14 remain valid, i.e. the modified algorithm terminates in less than n iterations.

Observation 4.11 The following equality holds for all i.

$$obsf_q^k(f, c, S_i) = obsf_q^k(f, c, S_i \setminus B_i) + obsf_q^k(f, c, B_i)$$

Definition 4.12 Let $\mathcal{R}$ and $\mathcal{P}$ be two equivalence relations on Q, where $\mathcal{P}$ is coarser than $\mathcal{R}$. We say that $\mathcal{R}$ *is stable with respect to* $\mathcal{P}$ if for every pair (q, q') in $\mathcal{R}$ it holds that $obsf_q^k(f, c, D)$ is zero if and only if $obsf_{q'}^k(f, c, D)$ is zero, for every f in Σ, c in C_Q, and D in $(Q/\mathcal{P})$. We say that $\mathcal{R}$ is *stable* if it is stable with respect to itself. $\Box$

Lemma 4.13 The relation $\mathcal{R}_i$ is stable with respect to $\mathcal{P}_i$, for all $i \in \{0, \ldots, t\}$.

Proof By Lemma 3.13, the relation $\mathcal{P}_i$ is coarser than $\mathcal{R}_i$. The remaining proof is by induction on i. The base case follows from the definitions of $\mathcal{R}_0$ and $\mathcal{P}_0$. Now, let (q, q') be a pair in $\mathcal{R}_{i+1}$, let f be a symbol in Σ, and let c be a context in C_Q. We show that $obsf^k_q(f, c, D)$ is zero if and only if $obsf^k_{q'}(f, c, D)$ is zero, for each block D in $(Q/\mathcal{P}_{i+1})$. Now, depending on D, there are three cases:

In the first case, the intersection between D and S_i is empty. This implies that D is also a block of $(Q/\mathcal{P}_i)$. Furthermore, the fact that (q, q') is an element of $\mathcal{R}_{i+1}$ means that (q, q') is also an element of the coarser relation $\mathcal{R}_i$. Supporting ourselves on the induction hypothesis, we have that $obsf^k_q(f, c, D)$ is zero if and only if $obsf^k_{q'}(f, c, D)$ is zero.

In the second case, $D = B_i$, so the desired relation follows from the fact that that the pair (q, q') was not in $splitf(B_i)$ because it is in $\mathcal{R}_{i+1}$.

The third alternative is that $D = S_i \setminus B_i$, in which case we compute as follows.

$$
\begin{aligned}
& obsf^k_q(f, c, S_i \setminus B_i) = 0 \\
\iff\ & obsf^k_q(f, c, S_i) = obsf^k_q(f, c, B_i) && \text{(by Observation 4.11)} \\
\iff\ & obsf^k_{q'}(f, c, S_i) = obsf^k_{q'}(f, c, B_i) && \text{(since } (q, q') \text{ is in } \mathcal{R}_{i+1}\text{)} \\
\iff\ & obsf^k_{q'}(f, c, S_i \setminus B_i) = 0 && \text{(by Observation 4.11)}
\end{aligned}
$$

■

Lemma 4.14 Let $\mathcal{R}$ be a stable relation, and let D be union of one or more blocks in $(Q/\mathcal{R})$. For every $(q, q') \in \mathcal{R}$, $f \in \Sigma$, and $c \in C_Q$, we have that $obsf^k_q(f, c, D)$ is zero if and only if $obsf^k_{q'}(f, c, D)$ is zero.

Proof Let $(q, q') \in \mathcal{R}$, $f \in \Sigma$, and $c \in C_Q$. We compute as follows:

$$
\begin{aligned}
& obsf^k_q(f, c, D) = 0 \\
\iff\ & \text{(By definition of the } obsf^k_q \text{ function.)} \\
& \{p \in D \mid f(c[\![q]\!]) \xrightarrow{\delta} p\} = \emptyset \\
\iff\ & \text{(Since } \mathcal{R} \text{ is a refinement of } \mathcal{R}_i \text{ by induction hypothesis.)} \\
& \bigcup_{C \in (D/\mathcal{R})} \{p \in C \mid f(c[\![q]\!]) \xrightarrow{\delta} p\} = \emptyset \\
\iff\ & \text{(Because } (q, q') \text{ is in } \mathcal{R}\text{, and } \mathcal{R} \text{ is a stable relation.)} \\
& \bigcup_{C \in (D/\mathcal{R})} \{p \in C \mid f(c[\![q']\!]) \xrightarrow{\delta} p\} = \emptyset \\
\iff\ & \text{(Since } \mathcal{R} \text{ is a refinement of } \mathcal{R}_i \text{ by induction hypothesis.)} \\
& \{p \in D \mid f(c[\![q']\!]) \xrightarrow{\delta} p\} = \emptyset \\
\iff\ & \text{(By definition of the } obsf^k_{q'} \text{ function.)} \\
& obsf^k_{q'}(f, c, B_i) = 0
\end{aligned}
$$

■

Lemma 4.15 If an equivalence relation $\mathcal{R}$ is a stable refinement of $\mathcal{R}_0$, then $\mathcal{R}$ is also a refinement of $\mathcal{R}_i$ for every $i \in \{0, \ldots, t\}$.

Proof The proof is by induction on i, and the base case is trivial. To cover the induction step, we show that if (q, q') is in $\mathcal{R}$, then (q, q') is also in $\mathcal{R}_{i+1}$. This is done by examining how the minimisation algorithm obtains $\mathcal{R}_{i+1}$ from $\mathcal{R}_i$. It is required that (q, q') is in $\mathcal{R}_i$, and this is satisfied by the induction hypothesis. Moreover, the mappings $obsf^k_q(f, c, B_i)$ and $obsf^k_{q'}(f, c, B_i)$ must either both be zero, or both be non-zero, regardless of f and c. Since $\mathcal{R}$ is supposedly stable and B_i, being a block in the coarser $\mathcal{R}_i$, is the union of one or more blocks in $\mathcal{R}$, Lemma 4.14 assures that also this condition is met. Finally, it must hold that

$$obsf^k_q(f, c, B_i) = obsf^k_q(f, c, S_i) \iff obsf^k_{q'}(f, c, B_i) = obsf^k_{q'}(f, c, S_i) \ ,$$

for every $f \in \Sigma$ and $c \in C_Q$. We reason as follows:

$$\begin{aligned}
& obsf^k_q(f, c, B_i) = obsf^k_q(f, c, S_i) \\
\iff\ & obsf^k_q(f, c, S_i \setminus B_i) = 0 && \text{(by Observation 4.11)} \\
\iff\ & obsf^k_{q'}(f, c, S_i \setminus B_i) = 0 && \text{(by Lemma 4.14)} \\
\iff\ & obsf^k_{q'}(f, c, B_i) = obsf^k_{q'}(f, c, S_i) && \text{(by Observation 4.11)}
\end{aligned}$$

■

Theorem 4.16 $\mathcal{R}_t$ is the coarsest forward bisimulation on M.

Proof The relations $\mathcal{R}_i$ and $\mathcal{P}_i$ are such that, for every $i \in [0, t]$;

- $\mathcal{R}_i$ is stable with respect to $\mathcal{P}_i$ (Lemma 4.13), and
- if $\mathcal{R}$ is a stable refinement of $\mathcal{R}_0$, then $\mathcal{R}$ is also a refinement of $\mathcal{R}_i$ (Lemma 4.15).

The time complexity of the backward bisimulation algorithm is computed from the same assumptions and notations as in Section 3.2. In addition, given a block B_i, we denote by $\delta^f_{B_i}$ the part of of the transition table δ that contains transitions of the form $f(q_1 \cdots q_k) \to q$, where $q \in B_i$. Although the computations are quite similar, they differ in that when the backward algorithm would examine every transition in $\delta^b B_i$, the forward algorithm consider only those in $\delta^f_{B_i}$. Since the latter set is on average a factor $\hat{r}$ smaller, we are able to obtain a proportional speed-up of the algorithm.

Observation 4.17 The overall time complexity of the algorithm is

$$O\Big(\textsc{Init}^f + \sum_{i \in [0, t-1]} \Big(\textsc{Select}_i + \textsc{Cut}_i + \textsc{Split}^f_i + \textsc{Split}^{f,\varphi}_i\Big) + \textsc{Aggregate}^f\Big) \ ,$$

where $\textsc{Init}^f$, $\textsc{Select}_i$, $\textsc{Cut}_i$, $\textsc{Split}^f_i$, $\textsc{Split}^{f,\varphi}_i$, and $\textsc{Aggregate}^f$ are the complexity of: the initialisation phase; the choice of S_i and B_i; the computation of $\mathcal{P}_i \setminus cut(B_i)$; the computation of $\mathcal{R}_i \setminus splitf(B_i)$; the computation of $\mathcal{R}_i \setminus splitn(S_i, B_i)$; and the construction of the aggregated automaton $(M/\mathcal{R}_t)$; respectively.

Lemma 4.18 $\textsc{Init}^f$ is in $O(\hat{r}m + n)$.

Proof The single block of $\mathcal{P}_0$ can be initialised in $O(n)$ steps. By Lemma 4.20, we can calculate $splitf(Q)$, and thus initialise $\mathcal{R}_0$, in time $O(\hat{r}\,|\delta^f_Q|) = O(\hat{r}\,|\delta|) = O(\hat{r}m)$. ■

Lemma 4.19 The derivation of $splitf(B)$, where $B \subseteq Q$, is in $O(\hat{r}\,|\delta^f_B|)$.

Proof The computation is divided into two parts: for each entry transition $f(q_1, \ldots, q_k) \to q$ in δ^f_B and i in $[1, k]$, the value of $obsf^k_{q_i}(f, q_1 \cdots q_{i-1} \Box\, q_{i+1} \cdots q_k, B)$ is increased by one. Since this requires us to iterate over the sequence of states $q_1, \ldots, q_k$, the time required is in $O(\hat{r}\,|\delta^f_B|)$. The set $splitf(B)$ can thereafter be derived by hashing each state q in Q using $(obsf^k_q)_{k\in[0,\hat{r}]}$ on the domain

$$\bigcup\nolimits_{k\in[0,\hat{r}]}(\Sigma_{(k)} \times C^k_Q \times \{B\})$$

as key. Parallel to the hash process the lists $\mathcal{R}$-*blocks*, $\mathcal{P}$-*blocks*, and $\mathcal{P}$-*compound* are updated to reflect the changes in $\mathcal{R}$. We note that each entry $f(c[\![q]\!]) \to q'$ in δ^f_B can give rise to at most $\hat{r}$ elements in the support of $(obsf^k_q)_{q\in Q,k\in\mathbb{N}}$. It follows that the size of the support of $(obsf^k_q)_{q\in Q,k\in\mathbb{N}}$ does not exceed $O(\hat{r}\,|\delta^f_B|)$, so the time required to hash the states in Q is determined by the time it takes to compute their hash keys. ∎

Lemma 4.20 $\textsc{Split}^f_i$ is in $O(\hat{r}\,|\delta^f_{B_i}|)$.

Lemma 4.21 $\textsc{Split}^{f,\varphi}_i$ is in $O(\hat{r}\,|\delta^f_{B_i}|)$.

Proof To compute $\textsc{Split}^{f,\varphi}_i$, we must calculate $obsf^k_q(f, c, S_i)$ and $obsf^k_q(f, c, B_i)$ for every $q \in Q$, before taking differences. The latter task can be done in $O(\hat{r}\,|\delta^f_{B_i}|)$, and since S_i is equal to B_j or $S_j \setminus B_j$ for some $j < i$, we already know the value $obsf^k_q(f, c, S_i)$ for every $q \in Q$, so we have the desired result. ∎

Lemma 4.22 $\textsc{Aggregate}^f$ is in $O(\hat{r}m + n)$.

We omit the proof of Lemma 4.22 as it is quite similar to that of Lemma 3.29.

Theorem 4.23 The minimisation algorithm is in $O(\hat{r}m \log n)$.

Proof By Observations 3.24 and 4.17, together with Lemmata 3.25 and 4.18 through 4.22, the overall time complexity of the algorithm can be written as

$$O\Big((\hat{r}m + n) + \sum\nolimits_{i\in[0,t-1]} (1 + |B_i| + \hat{r}\,|\delta^f_{B_i}| + \hat{r}\,|\delta^f_{B_i}|) + (\hat{r}m + n)\Big)\ .$$

Omitting the smaller terms and simplifying, we obtain $O\left(\hat{r}\sum_{i\in[0,t-1]} |\delta^f_{B_i}|\right)$. By Lemma 3.30, no state occurs in more than $\log n$ distinct B-blocks, so no transition in δ will contribute by more than $\log n$ to the total sum. As there are m transitions, the overall time complexity of the algorithm is $O(\hat{r}m \log n)$. ∎

We next consider minimisation via forward bisimulation on deterministic (bottom-up) tree automata. We show that forward bisimulation minimisation coincides with classical minimisation and yields the minimal deterministic tree automaton.

Definition 4.24 We say that M is *deterministic* (respectively, *complete*), if for every symbol f in $\Sigma_{(k)}$, and sequence $(q_1, \ldots, q_k) \in Q^k$ of states there exists at most (respectively, at least) one state q in Q such that $f(q_1, \ldots, q_k) \to q$ is in δ. □

It is an important observation that $(M/\mathcal{R})$ is also deterministic and complete whenever M is so and $\mathcal{R}$ is a forward bisimulation on M. Moreover, it is known that there exists a unique (up to isomorphism) minimal complete and deterministic nta N that recognises $\mathcal{L}(M)$. The next theorem shows that N is isomorphic to $(M/\mathcal{R})$ if $\mathcal{R}$ is the coarsest forward bisimulation on M.

Theorem 4.25 Let M be a deterministic and complete nta without useless states. Then $(M/\mathcal{R}_t)$ is a minimal deterministic and complete nta recognising $\mathcal{L}(M)$.

Proof Let $M' = (Q', \Sigma, \delta', F')$ be the unique (up to isomorphism) minimal deterministic and complete nta such that $\mathcal{L}(M') = \mathcal{L}(M)$. We prove that there exists a forward bisimulation $\mathcal{R}$ on M such that $(M/\mathcal{R})$ and M' are isomorphic. By minimality of M' such a bisimulation must be the coarsest bisimulation $\mathcal{R}_t$ on M.

We define the relation $\imath = \{(q, q') \in Q \times Q' \mid \mathcal{L}(M)_q \cap \mathcal{L}(M')_{q'} \neq \emptyset\}$. Since M has no useless states we have that $\mathcal{L}(M)_q \neq \emptyset$ for every state $q \in Q$. Moreover, for every state $q \in Q$ there exists a state $q' \in Q'$ such that $\mathcal{L}(M)_q \subseteq \mathcal{L}(M')_{q'}$. Hence for every state $q \in Q$ there exists a state $q' \in Q'$ such that $(q, q') \in \imath$. Moreover, since M' is complete, for every tree $t \in \mathrm{T}_\Sigma$, there exists exactly one state $q' \in Q'$ such that $t \in \mathcal{L}(M')_{q'}$, and thus, for every state $q \in Q$ there exists at most one state $q' \in Q'$ such that $(q, q') \in \imath$. Thus $\imath \colon Q \to Q'$. Now suppose that there exists a state $q' \in Q'$ so that there exists no state $q \in Q$ with $\imath(q) = q'$. Clearly, $\mathcal{L}(M')_{q'} = \emptyset$ which contradicts to the minimality of M'. Thus $\imath$ is surjective.

Let $\mathcal{R} = \ker(\imath)$, which, by definition, is an equivalence relation. We first prove Condition (i) of Definition 4.2. Let $(p, q) \in \mathcal{R}$. We have to prove that $p \in F$ if and only if $q \in F$. Since M has no useless states, there exist trees t and u in T_Σ such that $t \in \mathcal{L}(M)_p$ and $u \in \mathcal{L}(M)_q$. Because $\mathcal{L}(M)_p \subseteq \mathcal{L}(M')_{\imath(p)}$ and $\mathcal{L}(M)_q \subseteq \mathcal{L}(M')_{\imath(q)}$ and $\mathcal{L}(M') = \mathcal{L}(M)$ we easily observe that

$$\begin{array}{llll} p \in F & \iff t \in \mathcal{L}(M) & \iff t \in \mathcal{L}(M') & \iff \imath(p) \in F' \\ \iff \imath(q) \in F' & \iff u \in \mathcal{L}(M')_{\imath(q)} & \iff u \in \mathcal{L}(M)_q & \iff q \in F \ . \end{array}$$

Now to Condition (ii). For every symbol f in $\Sigma_{(k)}$, context c in $C^Q_{(k)}$, there exist unique states p' and q' in Q such that $f(c[\![p]\!]) \xrightarrow{\delta} p'$ and $f(c[\![q]\!]) \xrightarrow{\delta} q'$ because M is deterministic and complete. Thus it only remains to show that $(p', q') \in \mathcal{R}$. We observe that $\imath(p') = \imath(q')$ if and only if there exists a state $r' \in Q'$ such that $\mathcal{L}(M)_{p'} \cap \mathcal{L}(M')_{r'}$ and $\mathcal{L}(M)_{q'} \cap \mathcal{L}(M')_{r'}$ are nonempty. By assumption, $(p, q) \in \mathcal{R}$ and thus there exists a state $r \in Q'$ and trees s and s' of T_Σ such that

$$s \in \mathcal{L}(M)_p \cap \mathcal{L}(M')_r \qquad\qquad s' \in \mathcal{L}(M)_q \cap \mathcal{L}(M')_r \ .$$

Let $c = q_1 \cdots q_{i-1} \Box q_{i+1} \cdots q_k$ for some $i \in [1, k]$ and $q_1, \ldots, q_k \in Q$. Since M has no useless states, for every $j \in [1, k]$ there exists a tree $s_j \in \mathrm{T}_\Sigma$ such that $s_j \in \mathcal{L}(M)_{q_j}$. Since M' is deterministic and complete, we obtain that there exists a state $r' \in Q'$ such that

$$\{f[s_1, \ldots, s_{i-1}, s, s_{i+1}, \ldots, s_k], f[s_1, \ldots, s_{i-1}, s', s_{i+1}, \ldots, s_k]\} \subseteq \mathcal{L}(M')_{r'} \ .$$

Clearly,

$$f[s_1, \ldots, s_{i-1}, s, s_{i+1}, \ldots, s_k] \in \mathcal{L}(M)_{p'} \quad \text{and} \quad f[s_1, \ldots, s_{i-1}, s', s_{i+1}, \ldots, s_k] \in \mathcal{L}(M)_{q'} \ .$$

This proves that the intersections $\mathcal{L}(M)_{p'} \cap \mathcal{L}(M')_{r'}$ and $\mathcal{L}(M)_{q'} \cap \mathcal{L}(M')_{r'}$ are nonempty, and thus $(p', q') \in \mathcal{R}$. Hence Condition (ii) of Definition 4.2 is fulfilled and $\mathcal{R}$ is a forward bisimulation on M.

It remains to prove that the aggregated nta $(M/\mathcal{R})$ is isomorphic to M'. Clearly, the nta $(M/\mathcal{R})$ is deterministic and complete, has $|Q'|$ states, and $\mathcal{L}((M/\mathcal{R})) = \mathcal{L}(M) = \mathcal{L}(M')$ by Theorem 4.5. Thus $(M/\mathcal{R})$ is a minimal complete dta recognising $\mathcal{L}(M)$. ∎

trees	original		backward		forward		b after f		f after b	
	states	rules	states	rules	states	rules	states	rules	states	rules
58	353	353	252	252	286	341	185	240	180	235
161	953	953	576	576	749	905	378	534	356	512
231	1373	1373	781	781	1075	1299	494	718	468	691
287	1726	1726	947	947	1358	1637	595	874	563	842

Table 1: Reduction of states and rules by using the forward and backward bisimulation minimisation algorithms

5 Implementation

In this section we present some experimental results that we obtained by applying a prototype implementation of Algorithm 1 to the problem of *language modelling* in the natural language processing domain [9].

A language model is a formalism for determining whether a given sentence is in a particular language. Language models are particularly useful in many applications of natural language and speech processing such as translation, transliteration, speech recognition, character recognition, etc., where transformation system output must be verified to be an appropriate sentence in the domain language. Recent research in natural language processing has focused on using tree-based models to capture syntactic dependencies in applications such as machine translation [6, 16]. Thus, the problem is elevated to determining whether a given syntactic tree is in a language. Language models are naturally representable as finite-state acceptors. For efficiency and data sparsity reasons, whole sentences are not typically stored, but rather a sliding window of partial sentences is verified. In the string domain this is known as *n-gram* language modelling. We instead model *n-subtrees*, fixed-size pieces of a syntactic tree.

We prepared a data set by collecting 3-subtrees, i.e. all subtrees of height 3, from sentences taken from the Penn Treebank corpus of syntactically bracketed English news text [11]. An initial nta was constructed by representing each 3-subtree in a single path. We then wrote an implementation of the forward and backward variants of Algorithm 1 in Perl and applied them to data sets of various sizes of 3-subtrees. To illustrate that the two algorithms perform different minimisations, we then ran the forward algorithm on the result from the backward algorithm, and vice-versa. As Table 1 shows, the combination of both algorithms reduces the automata nicely, to less than half the size (in the sum of rules and states) of the original.

Conclusion

We have introduced a general algorithm for bisimulation minimisation of tree automata and discussed its operation under forward and backward bisimulation. The algorithm has attractive runtime properties and is useful for applications that desire a compact representation of large non-deterministic tree automata. We plan to include a refined implementation of this algorithm in a future version of the tree automata toolkit described in [12].

References

[1] P. A. Abdulla, B. Jonsson, P. Mahata, and J. d'Orso. Regular tree model checking. In *Proc. 14th Int. Conf. Computer Aided Verification*, volume 2404 of *Lecture Notes in Computer Science*, pages 555–568. Springer, 2002.

[2] P. A. Abdulla, L. Kaati, and J. Högberg. Bisimulation minimization of tree automata. In *Proc. 11th Int. Conf. Implementation and Application of Automata*, volume 4094 of *LNCS*, pages 173–185. Springer Verlag, 2006.

[3] P. A. Abdulla, L. Kaati, and J. Högberg. Bisimulation minimization of tree automata. Technical Report UMINF 06.25, Umeå University, 2006.

[4] P. Buchholz. Bisimulation relations for weighted automata. unpublished, 2007.

[5] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata: Techniques and applications. Available on: http://www.grappa.univ-lille3.fr/tata, 1997.

[6] M. Galley, M. Hopkins, K. Knight, and D. Marcu. What's in a translation rule? In *Proc. 2004 Human Language Technology Conf. of the North American Chapter of the Association for Computational Linguistics*, pages 273–280, 2004.

[7] G. Gramlich and G. Schnitger. Minimizing nfas and regular expressions. In *Proc. 22nd Int. Symp. Theoretical Aspects of Computer Science*, volume 3404 of *LNCS*, pages 399–411. Springer Verlag, 2005.

[8] J. E. Hopcroft. An $n \log n$ algorithm for minimizing states in a finite automaton. In Z. Kohavi, editor, *Theory of Machines and Computations*. Academic Press, 1971.

[9] F. Jelinek. Continuous speech recognition by statistical methods. *Proc. IEEE*, 64(4):532–557, 1976.

[10] K. Knight and J. Graehl. An overview of probabilistic tree transducers for natural language processing. In *Proc. 6th Int. Conf. Computational Linguistics and Intelligent Text Processing*, volume 3406 of *LNCS*, pages 1–24. Springer Verlag, 2005.

[11] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of english: The Penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[12] J. May and K. Knight. Tiburon: A weighted tree automata toolkit. In *Proc. 11th Int. Conf. Implementation and Application of Automata*, volume 4094 of *LNCS*, pages 102–113. Springer Verlag, 2006.

[13] A. R. Meyer and L. J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proc. 13th Annual Symp. Foundations of Computer Science*, pages 125–129. IEEE Computer Society, 1972.

[14] R. Paige and R. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

[15] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[16] K. Yamada and K. Knight. A syntax-based statistical translation model. In *Proc. 39th Meeting of the Association for Computational Linguistics*, pages 523–530. Morgan Kaufmann, 2001.