Tighter Bounds for the Determinisation of Büchi Automata*

Sven Schewe

University of Liverpool
sven.schewe@liverpool.ac.uk

Abstract. The introduction of an efficient determinisation technique for Büchi automata by Safra has been a milestone in automata theory. To name only a few applications, efficient determinisation techniques for ω-word automata are the basis for several manipulations of ω-tree automata (most prominently the nondeterminisation of alternating tree automata) as well as for satisfiability checking and model synthesis for branching- and alternating-time logics. This paper proposes a determinisation technique that is simpler than the constructions of Safra, Piterman, and Muller and Schupp, because it separates the principle acceptance mechanism from the concrete acceptance condition. The principle mechanism intuitively uses a Rabin condition on the transitions; we show how to obtain an equivalent Rabin transition automaton with approximately $(1.65\,n)^n$ states from a nondeterministic Büchi automaton with n states. Having established this mechanism, it is simple to develop translations to automata with standard acceptance conditions. We can construct standard Rabin automata whose state-space is bilinear in the size of the input alphabet and the state-space of the Rabin transition automaton, or, for large input alphabets, contains approximately $(2.66\,n)^n$ states, respectively. We also provide a flexible translation to parity automata with $O(n!^2)$ states and $2n$ priorities based on a later introduction record, and hence connect the transformation of the acceptance condition to other record based transformations known from the literature.

1 Introduction

Automata over infinite words have been introduced by Büchi in his proof that the monadic second-order logic of one successor (S1S) is decidable [1]. Büchi automata are an adaptation of finite automata to languages over infinite sequences. They differ from finite automata only with respect to their acceptance condition: While finite runs of finite automata are accepting if a final state is visited at the end of the run, an infinite run of a Büchi automaton is accepting if a final state is visited infinitely many times. Unfortunately, this close relationship between finite and Büchi automata does not imply that automata manipulations

* This work was partly supported by the EPSRC through the grand EP/F033567/1 *Verifying Interoperability Requirements in Pervasive Systems.*

L. de Alfaro (Ed.): FOSSACS 2009, LNCS 5504, pp. 167–181, 2009.

for Büchi automata are equally simple as those for finite automata [2]. In particular, Büchi automata are not closed under determinisation: While a simple subset construction suffices to efficiently determinise finite automata [2], deterministic Büchi automata are strictly less expressive[1] than nondeterministic Büchi automata. Determinisation therefore requires automata with more involved acceptance mechanisms [3,4,5], such as automata with Muller's subset condition [6,7] or Rabin's [3,4] accepting pair condition. Also, an $n^{\Omega(n)}$ lower bound for the determinisation of Büchi automata has been established [8] even if we allow for Muller objectives, which implies that a simple subset construction cannot suffice.

The development of determinisation techniques for Büchi automata was inspired by the problem of synthesising reactive systems [9,10], a problem originally introduced by Church [9] in 1962: Given a relation $R \subseteq (2^I)^\omega \times (2^O)^\omega$ represented by a Büchi automaton (or an S1S or LTL formula), we want to find a function $p : (2^I)^\omega \to (2^O)^\omega$ such that $(\pi, p(\pi)) \in R$ satisfies the relation for all infinite sequences $\pi \in (2^I)^\omega$. Church's problem was solved independently by Rabin [11], and Büchi and Landweber [12,13] in 1969. Since their seminal works, the relation [14] between finite automata over infinite structures [11] and finite games of infinite duration [12,13] became apparent.

Determinisation is a key ingredient in these proofs. Rabin's extension of the correspondence between automata and monadic logic to the case of trees [11], for example, builds on McNaughton's determinisation theorem [7], and Muller and Schupp's [4] efficient nondeterminisation technique for alternating tree automata is closely linked to the determinisation of nondeterministic word automata. Indeed, the standard technique to nondeterminise an alternating automaton $\mathcal{A}$ with a memoryless acceptance conditions (such as a parity or Rabin automata [15]) is to enrich the input tree with a (guessed) memoryless strategy. Nondeterminising $\mathcal{A}$ can then be reduced to determinise the resulting universal automaton [4,14], and projecting away the strategy. Improved determinisation techniques thus have a considerable impact in automata theory and its application to module checking [16], satisfiability checking [1,11,17,18], and open synthesis [10].

Contribution. This paper contributes a determinisation technique for Büchi automata that simplifies the constructions of Safra [3] and Piterman [5] by separating the principle data structure of the algorithm — the history trees proposed in Section 3 — from the acceptance mechanism. It is my believe that this separations eases teaching and understanding the principles, but it also provides better bounds on the size of the resulting automata.

The central advancement of the proposed method over the previous leading determinisation techniques [3,4,5] is that we abandon the introduction of explicit names for the nodes. One positive effect of this decision is that it yields a leaner and simpler core data structure: The number $\mathit{hist}(n)$ of history trees for Büchi automata with n states is in $o\big((1.65\,n)^n\big)$. We use this observation to construct a deterministic Rabin automaton with only $\mathit{hist}(n)$ states whose pairs are defined

[1] Deterministic Büchi automata cannot, for example, recognise the simple ω-regular language that consists of all infinite words that contain only finitely many a's.

on the *transitions*. As Rabin tree automata have a memoryless accepting run if they accept a tree [15], this implies a $hist(n)$ bound on the size of a program that solves Church's problem as well as an $l \cdot hist(n)$ bound on the size of an ordinary deterministic Rabin automata on alphabets with l letters.

If we want the size of the Rabin automaton to be independent of the alphabet size, or if we want to construct a deterministic parity automaton because of the computational advantages attached to parity objectives, we have to add memory to the history trees. The required amount of memory depends on the acceptance mechanism. For Rabin automata, it suffices to store the acceptance information from the last transition, which only leads to a minor blow-up of the state-space to $o\big((2.66\,n)^n\big)$ states.

For parity automata, we turn to the proved method of keeping a record of the most recent relevant events in the tradition of later [19] and index appearance records [4]: We store (an abstraction of) the order in which the nodes of the current history tree have been introduced in a *later introduction record*.

The separation of concerns allows us to phrase our procedure as a nondeterministic determinisation technique: While the update rule for history trees is strict, the update rule for the later introduction record offers some leeway. This leeway is likely to reduce the size of a deterministic automaton in practice.

Waving this advantage, we still yield a determinisation procedure similar to Piterman's [5], but with an improved complexity analysis ($O(n!^2)$ vs. $O(n^n\,n!)$). However, a reviewer has pointed me to unpublished work of Liu and Wang [20], who independently[2] improved Piterman's complexity analysis to a similar bound.

Organisation of the Paper. In the following section, we recapitulate the different types of automata used in this paper. Section 3 then introduces history trees, which serve as the main data structure used in the proposed determinisation techniques, transitions between them, and a principle approach to exploit this data structure in an efficient determinisation technique. Finally, we use this blueprint of a determinisation technique in Sections 4 and 5 to devise different translations from nondeterministic Büchi tree automata to deterministic Rabin automata, and one to deterministic parity automata, respectively.

2 Preliminaries — Rabin, Parity and Büchi Automata

Nondeterministic Rabin automata are used to represent ω-regular languages $L \subseteq \Sigma^\omega = \omega \to \Sigma$ over a finite alphabet Σ. A nondeterministic Rabin automaton $\mathcal{A} = (\Sigma, Q, I, \delta, \{(A_i, R_i) \mid i \in J\})$ is a five tuple, consisting of a finite alphabet Σ, a finite set Q of states with a non-empty subset $I \subseteq Q$ of initial states, a transition function $\delta : Q \times \Sigma \to 2^Q$ that maps states and input letters to sets of successor states, and a finite family $\{(A_i, R_i) \in 2^Q \times 2^Q \mid i \in J\}$ of Rabin pairs.

[2] I was not aware of the unpublished work of Liu and Wang [20] when writing this paper. While their improvement of Piterman's analysis was submitted after the acceptance of this paper, I would like to point out that their work is older.

Nondeterministic Rabin automata are interpreted over infinite sequences $\alpha : \omega \to \Sigma$ of input letters. An infinite sequence $\rho : \omega \to Q$ of states of $\mathcal{A}$ is called a *run* of $\mathcal{A}$ on an input word α if the first letter $\rho(0) \in I$ of ρ is an initial state, and if, for all $i \in \omega$, $\rho(i+1) \in \delta\big(\rho(i), \alpha(i)\big)$ is an $\alpha(i)$-successor state of $\rho(i)$.

A run $\rho : \omega \to Q$ is *accepting* if, for some index $i \in J$, some state $q \in A_i$ in the acceptance set A_i of the Rabin pair (A_i, R_i), but no state $q' \in R_i$ from the rejecting set R_i of this Rabin pair appears infinitely often in ρ. ($\exists i \in J.\ \mathit{inf}(\rho) \cap A_i \neq \emptyset \wedge \mathit{inf}(\rho) \cap R_i = \emptyset$ for $\mathit{inf}(\rho) = \{q \in Q \mid \forall i \in \omega\ \exists j > i$ such that $\rho(j) = q\}$). A word $\alpha : \omega \to \Sigma$ is *accepted* by $\mathcal{A}$ if $\mathcal{A}$ has an accepting run on α, and the set $\mathcal{L}(\mathcal{A}) = \{\alpha \in \Sigma^\omega \mid \alpha$ is accepted by $\mathcal{A}\}$ of words accepted by $\mathcal{A}$ is called its *language*.

For technical convenience we also allow for finite runs $q_0 q_1 q_2 \dots q_n$ with $\delta\big(q_n, \alpha(n)\big) = \emptyset$. Naturally, no finite run satisfies the Rabin condition; finite runs are therefore rejecting, and have no influence on the language of an automaton.

Two particularly simple types of Rabin automata are of special interest: parity (or Rabin chain) and Büchi automata. We call a Rabin condition a *Rabin chain* condition if J is an initial sequence of the natural numbers ω, and if $R_i \subset A_i$ and $A_i \subset R_{i+1}$ holds for all indices. The Rabin chain condition is nowadays usually referred to by the term *parity* condition, because it can be represented by a priority function $\mathit{pri} : Q \to \omega$ that maps a state q to $2i+2$ (called the priority of q) if it appears in A_i but not in R_i, and to $2i+1$ if it appears in R_i but not in A_{i-1}. A run ρ of $\mathcal{A}$ then defines an infinite trace of priorities, and the parity of the lowest priority occurring infinitely often determines if ρ is accepting. That is, ρ is accepting if $\min(\mathit{inf}(\mathit{pri}(\rho)))$ is even. We denote parity automata $\mathcal{A} = (\Sigma, Q, I, \delta, \mathit{pri})$, using this priority function. Büchi automata are even simpler: they are Rabin automata with only one accepting pair $(F, \emptyset)$ that has an empty set of rejecting states (or, likewise, parity automata with a priority function pri whose codomain is $\{0, 1\}$. A Büchi automaton is denoted $\mathcal{A} = (\Sigma, Q, I, \delta, F)$, and the states in F are called *final* states.

A Rabin, parity, or Büchi automaton is called *deterministic*, if it has a single initial state and its transition function is deterministic. (That is, if $|\delta(q, \sigma)| \leq 1$ holds true for all states $q \in Q$ and all input letters $\sigma \in \Sigma$ of the automata $\mathcal{A}$.)

3 Büchi Determinisation

The determinisation technique discussed in this section is a variant of Safra's [3] determinisation technique, and the main data structure — the history trees proposed in the first subsection — can be viewed as a simplification of Safra trees [3].

3.1 History Trees

History trees are an abstraction of the possible initial sequences of runs of a Büchi automaton $\mathcal{A}$ on an input word α. They can be viewed as a simplification and abstraction of Safra trees [3]. The main difference between Safra trees and the simpler history trees introduced in this paper is the omission of explicit names for the nodes.

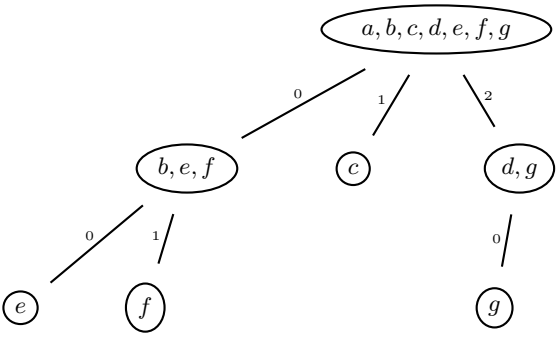

Fig. 1. Example History Tree. The labels of the children of every node are disjoint, and their union is a strict subset of their parent's label. The label of the root node contains the reachable states of the Büchi automaton $\mathcal{A}$ on the input seen so far.

An *ordered tree* $T \subseteq \omega^*$ is a finite prefix and order closed subset of finite sequences of natural numbers. That is, if a sequence $\tau = t_0, t_1, \dots t_n \in T$ is in T, then all sequences $s_0, s_1, \dots s_m$ with $m \leq n$ and, for all $i \leq m$, $s_i \leq t_i$, are also in T. For a node $\tau \in T$ of an ordered tree T, we call the number of children of τ its *degree*, denoted by $\deg_T(\tau) = |\{i \in \omega \mid \tau \cdot i \in T\}|$.

A *history tree* (cf. Figure 1) for a given nondeterministic Büchi automaton $\mathcal{A} = (\Sigma, Q, I, \delta, F)$ is a labelled tree $\langle T, l\rangle$, where T is an ordered tree, and $l : T \to 2^Q \setminus \{\emptyset\}$ is a labelling function that maps the nodes of T to non-empty subsets of Q, such that

- the label of each node is a proper superset of the union of the labels of its children, and
- the labels of different children of a node are disjoint.

We call a node τ the *host node* of a state q, if $q \in l(\tau)$ is in the label of τ, but not in the label of any child of τ.

Our estimation of the number of history trees for a given Büchi automaton draws from an estimation of Temme [21] (in the representation of Friedgut, Kupferman, and Vardi [22]) for the number of functions from a set with n elements onto a set with βn elements, where $\beta \in]0, 1[$ is a positive rational number smaller than 1: For the unique positive real number x that satisfies $\beta x = 1 - e^{-x}$, and for $a = -\ln x + \beta \ln(e^x - 1) - (1 - \beta) + (1 - \beta) \ln\left(\frac{1-\beta}{\beta}\right)$, the number of these functions is in $[(1 + o(1))M(\beta)n]^n$ for $M(\beta) = \left(\frac{\beta}{1-\beta}\right)^{1-\beta} e^{(a-\beta)}$. This simplifies to

$$m(x) = \frac{1}{ex}(e^x - 1)^{\beta(x)}$$

for $\beta(x) = \frac{1-e^{-x}}{x}$ and $m(x) = M\big(\beta(x)\big)$ when using $e^{a-\beta} = \frac{1}{ex}(e^x - 1)^{\beta}\left(\frac{1-\beta}{\beta}\right)^{1-\beta}$, where x can be any strictly positive real number.

To estimate the number $hist(n)$ of history trees for Büchi automata with n states, the number $order(m)$ of trees with m nodes can be estimated by 4^m. (More precisely, $order(m) = \frac{(2m-2)!}{m!(m-1)!}$ is the $(m-1)$-st Catalan number [5].) The

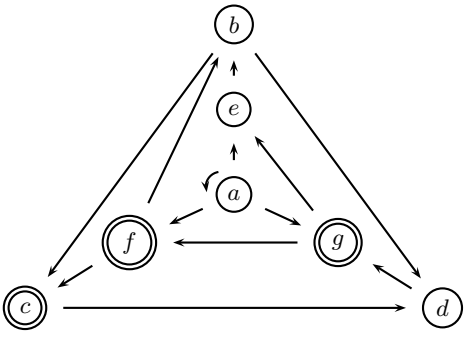

Fig. 2. Relevant Fragment of a Büchi Automaton. This figure captures all transitions for an input letter σ from the states in the history tree from Figure 1. The double lines indicate that the states c, f, and g are final states.

number of history trees with m nodes for a Büchi automaton with n states is the product of the number $\mathit{order}(m)$ of ordered trees with m nodes and functions from the set of n states onto the set of m nodes (if the root is mapped to all states of $\mathcal{A}$), plus the functions the automata states to a set with $(m+1)$ elements. Together with the estimation from above, we can numerically estimate

$$\mathit{hist}(n) \in \sup_{x>0} O\big(m(x) \cdot 4^{\beta(x)}\big) \subset o\big((1.65\,n)^n\big).$$

3.2 History Transitions

For a given nondeterministic Büchi automaton $\mathcal{A} = (\Sigma, Q, I, \delta, F)$, history tree $\langle T, l\rangle$, and input letter $\sigma \in \Sigma$, we construct the *σ-successor* $\langle \widehat{T}, \widehat{l}\rangle$ of $\langle T, l\rangle$ in four steps. (An example transition for the history tree shown in Figure 1 for the σ-transition of an automaton $\mathcal{A}$ shown in Figure 2 is described in Figures 3–6.)

In a first step (shown in Figure 3), we construct the labelled tree $\langle T', l' : T' \to 2^Q\rangle$ such that

- $\tau \in T' \supset T$ is a node of T' if, and only if, $\tau \in T$ is in T or $\tau = \tau' \cdot \deg_T(\tau')$ is formed by appending the degree $\deg_T(\tau')$ of a node $\tau' \in T$ in T to τ',
- the label $l'(\tau) = \delta(l(\tau), \sigma)$ of an old node $\tau \in T$ is the set $\delta(l(\tau), \sigma) = \bigcup_{q \in l(\tau)} \delta(q, \sigma)$ of σ-successors of the states in the label of τ, and
- the label $l'(\tau \cdot \deg_T(\tau')) = \delta(l(\tau), \sigma) \cap F$ of a new node $\tau \cdot \deg_T(\tau)$ is the set of *final* σ-successors of the states in the label of τ.

After this step, each old node is labelled with the σ-successors of the states in its old label, and every old node τ has spawned a new sibling $\tau' = \tau \cdot \deg(\tau)$, which is labelled with the final states $l'(\tau') = l'(\tau) \cap F$ in the label of its parent τ.

The new tree is not necessarily a history tree: (1) nodes may be labelled with an empty set (like node 000 of Figure 3), (2) the labels of siblings do not need to be disjoint (f and g are, for example, in the intersection of the labels of nodes 2 and 3 in Figure 3), and (3) the union of the children's labels do not need to form a proper subset of their parent's label (the union of the labels of node 20 and 21, for example, equals the label of node 2 in Figure 3).

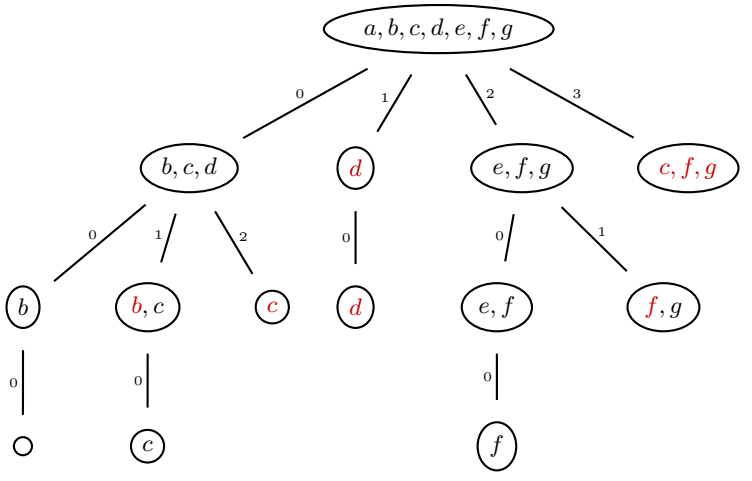

Fig. 3. First Step of the History Transition. This figure shows the tree resulting from the history tree of Figure 1 for the Büchi automaton and transition from Figure 2 alter the first step of the history transition. Every node of the tree from Figure 3 has spawned a new child, whose label may be empty (like the label of node 10) if no final state is reachable upon the read input letter from any state in the label of the parent node. (States printed in red are deleted from the respective label in the second step.)

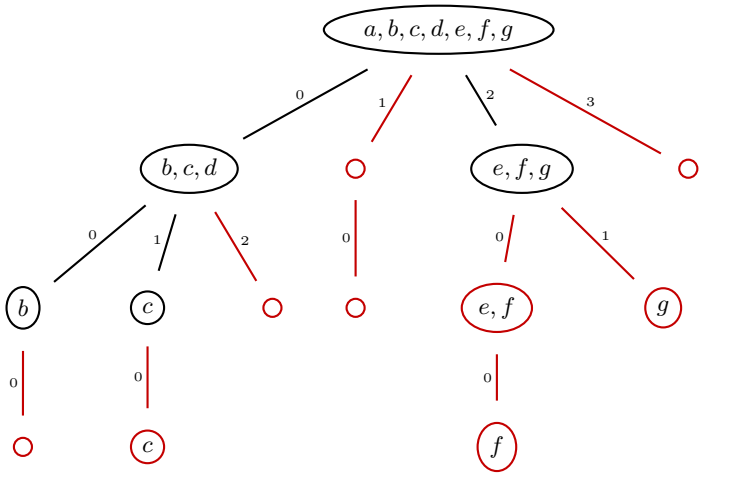

Fig. 4. Second Step of the History Transition. This figure shows the labelled tree that results from the second step of the history transition. the states from the labels of the tree shown in Figure 3 that also occur in the label of an older sibling (like the state f from the old label of the node 21) or in the label of an older sibling of an ancestor of the node (like the state d from the old label of the node 10) are deleted from the label. In this tree, the labels of the siblings are pairwise disjoint, but may be empty, and the union of the label of the children of a node are not required to form a *proper* subset of their parent's label. (The nodes colour coded red are deleted in the third step.)

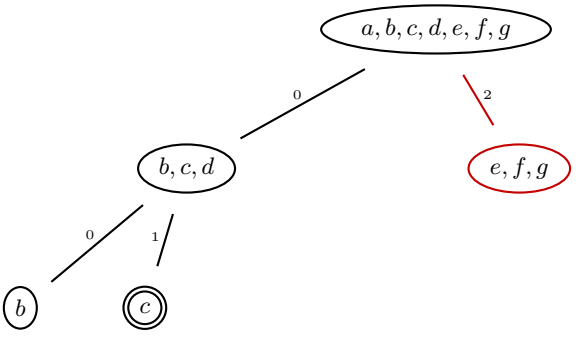

Fig. 5. Third Step of the History Transition. The nodes with (a) an empty label (nodes 000, 02, 1, 10 and 3 from the tree shown in Figure 4) and (b) the descendants of nodes whose children's labels decomposed their own label (nodes 010, 20, 200 and 21) have been deleted from the tree. The labels of the siblings are pairwise disjoint, and form a proper subset of their parent's label, but the tree is not order closed. The nodes that are renamed when establishing order closedness in the final step are depicted in red. Node 01 is the only accepting node (indicated by the double line): Its siblings have been removed due to (b), and, different to node 2, node 01 is stable.

In the second step, property (2) is re-established. We construct the tree $\langle T', l'' : T' \to 2^Q \rangle$, where l'' is inferred from l' by removing all states in the label of a node $\tau' = \tau \cdot i$ and all its descendants if it appears in the label $l'(\tau \cdot j)$ of an older sibling ($j < i$). In Figure 3, the states that are deleted by this rule are depicted in red, and the tree resulting from this deletion is shown in Figure 4.

Properties (1) and (3) are re-established in the third transformation step. In this step, we construct the tree $\langle T'', l'' : T'' \to 2^Q \rangle$ by (a) removing all nodes τ with an empty label $l''(\tau) = \emptyset$, and (b) removing all descendants of nodes whose label is disintegrated by the labels of its descendants from T'. (We use l'' in spite of the type mismatch, strictly speaking we should use its restriction to T''.) The part of the tree that is deleted during the third step is depicted in red in Figure 4, and the tree resulting from this transformation step is shown in Figure 5.

We call the greatest prefix and order closed subset of T'' the set of *stable* nodes and the stable nodes whose descendants have been deleted due to rule (b) *accepting*. In Figure 5, the unstable node 2 is depicted in red, and the accepting leaf 01 is marked by a double line. (Note that only leaves can be accepting.)

The tree resulting from this transformation satisfies the properties (1)–(3), but it is no longer order closed. The tree from Figure 5, for example, has a node 2, but no node 1. In order to obtain a proper history tree, the order closedness is re-established in the final step of the transformation. We construct the σ-successor $\langle \widehat{T}, \widehat{l} : \widehat{T} \to 2^Q \setminus \{\emptyset\} \rangle$ of $\langle T, l \rangle$ by "compressing" T'' to a an order closed tree, using the compression function $comp : T'' \to \omega^*$ that maps the empty word ε to ε, and $\tau \cdot i$ to $comp(\tau) \cdot j$, where $j = |\{k < i \mid \tau \cdot k \in T''\}|$ is the number of older siblings of $\tau \cdot i$. For this function $comp : T'' \to \omega^*$, we simply set $\widehat{T} = \{comp(\tau) \mid \tau \in T''\}$ and $\widehat{l}(comp(\tau)) = l''(\tau)$ for all $\tau \in T''$. The nodes

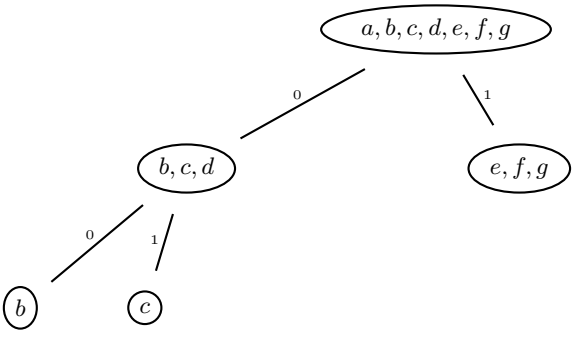

Fig. 6. Final Step of the History Transition. The history tree that results from the complete history transition, has the shape and labelling of the tree from Figure 5, but the former node 2 has been renamed to 1 in order to re-establishing order closedness.

that are renamed during this step are exactly those which are unstable. In our example transformation this is node 2 (depicted in red in Figure 5).

Figure 6 shows the σ-successor for the history tree of Figure 1 and an automaton with σ-transitions as shown in Figure 2.

3.3 Deterministic Acceptance Mechanism

For a nondeterministic Büchi automaton $\mathcal{A} = (\Sigma, Q, I, \delta, F)$, we call the history tree $\langle T_0, l_0\rangle = \langle\{\varepsilon\}, \varepsilon \mapsto I\rangle$ that contains only the empty word and maps it to the initial states I of $\mathcal{A}$ the initial history tree.

For an input word $\alpha : \omega \to \Sigma$ we call the sequence $\langle T_0, l_0\rangle, \langle T_1, l_1\rangle, \ldots$ of history trees that start with the initial history tree $\langle T_0, l_0\rangle$ and where, for every $i \in \omega$, $\langle T_i, l_i\rangle$ is followed by $\alpha(i)$-successor $\langle T_{i+1}, l_{i+1}\rangle$ the *history trace* or α. A node τ in the history tree $\langle T_{i+1}, l_{i+1}\rangle$ is called stable or accepting, respectively, if it is stable or accepting in the $\alpha(i)$-transition from $\langle T_i, l_i\rangle$ to $\langle T_{i+1}, l_{i+1}\rangle$.

Proposition 1. *An ω-word α is accepted by a nondeterministic Büchi automaton $\mathcal{A}$ if, and only if, there is a node $\tau \in \omega^*$ such that τ is eventually always stable and always eventually accepting in the history trace of α.*

Proof. For the "if" direction, let $\tau \in \omega^*$ be a node that is eventually always stable and always eventually accepting, and let $i_0 < i_1 < i_2 < \ldots$ be an ascending chain of indices such that τ is stable for the $\alpha(j)$-transitions from $\langle T_j, l_j\rangle$ to $\langle T_{j+1}, l_{j+1}\rangle$ for all $j \geq i_0$, and accepting for the $\alpha(i-1)$-transition from $\langle T_{i-1}, l_{i-1}\rangle$ to $\langle T_i, l_i\rangle$ for all indices i in the chain.

By definition of the σ-transitions, for every $j \in \omega$, the *finite* automaton $\mathcal{A}_j = (\Sigma, Q, l_{i_j}(\tau), \delta, F)$ has, for every state $q \in l_{i_{j+1}}(\tau)$, a run ρ_j^q on the finite word $\alpha(i_j)\alpha(i_{j+1})\alpha(i_{j+2})\ldots\alpha(i_{j+1}-1)$ that contains an accepting state and ends in q. Also, $\mathcal{A} = (\Sigma, Q, I, \delta, F)$ read as a finite automaton has, for every state $q \in l_{i_0}(\tau)$, a run ρ^q on the finite word $\alpha(0)\alpha(1)\alpha(2)\ldots\alpha(i_0-1)$ that ends in q. Let us fix such runs, and define a tree $T \subseteq Q^*$ that contains, besides the empty word and the initial states, a node iq_0 of length 2 if q_0 is in $l_{i_{j+1}}(\tau)$ and i is the

first letter of ρ^{q_0}, and a node $iq_0q_1q_2\dots q_kq_{k+1}$ of length $k+1>2$ if $iq_0q_1q_2\dots q_k$ is in T, q_{k+1} is in $l_{i_{k+1}}(n)$ and q_k is the first letter of $\rho_k^{q_{k+1}}$. By construction, T is an infinite tree with finite branching degree, and therefore contains an infinite path $iq_0q_1q_2\dots$ by König's Lemma. By construction, $\rho^{q_0}\rho_0^{q_1}\rho_1^{q_2}\dots$ is a run of $\mathcal{A}$ on α that visits some accepting state infinitely many times.

To demonstrate the "only if" direction, let us fix an accepting run, $\rho=q_0q_1\dots$ of $\mathcal{A}$ on an input word α. Then we can define the sequence $\vartheta=\tau_0\tau_1\dots$ of nodes such that, for the history trace $\langle T_0,l_0\rangle,\langle T_1,l_1\rangle,\dots$, τ_i is the host node of $q_i\in l_i(\tau_i)$ for the history tree $\langle T_i,l_i\rangle$. Let l be the shortest length $|\tau_i|$ of these nodes that occurs infinitely many times.

It is easy to see that the initial sequence of length l of the nodes in ϑ eventually stabilises: Let $i_0<i_1<i_2<\dots$ be an infinite ascending chain of indices such that the length $|\tau_j|\geq l$ of the j-th node is not smaller than l for all $j\geq i_0$, and equal to $l=|\tau_i|$ for all indices $i\in\{i_0,i_1,i_2,\dots\}$ in this chain. This implies that $\tau_{i_0},\tau_{i_1},\tau_{i_2},\dots$ is a descending chain when the single nodes τ_i are compared by lexicographic order, and hence almost all $\tau_i:=\pi$ are equal. This also implies that π is eventually always stable.

Let us assume that π is accepting only finitely many times. Then we can chose an index i from the chain $i_0<i_1<i_2<\dots$ such that $\tau_j=\pi$ holds for all indices $j\geq i$, and π is not accepting for any $j\geq i$. (Note that every time the length of τ_j is *reduced* to l, τ_j is unstable, which we excluded, or accepting, which violates the assumption.) But now we can pick an index $i'>i$ such that $q_{i'}\in F$ is a final state, which, together with $\tau_{i'}=\pi$, implies that π is accepting for $\big(\langle T_{i'-1},l_{i'-1}\rangle,\alpha(i'-1),\langle T_{i'},l_{i'}\rangle\big)$. (Note that $q_{i'}$ is in the label of $\pi\cdot\deg_{T_{i'-1}}(\pi)$ in the labelled tree $\langle T'_{i'-1},l'_{i'-1}\rangle$ resulting from the first step of the σ-transition of history trees.) $\lightning$ □

4 From Nondeterministic Büchi Automata to Deterministic Rabin Automata

In this section, we discuss three determinisation procedures for nondeterministic Büchi automata. First we observe that the acceptance mechanism from the previous section already describes a deterministic automaton with a Rabin condition, but the Rabin condition is on the *transitions*. This provides us with the first corollary:

Corollary 1. *For a given nondeterministic Büchi automaton with n states, we can construct a deterministic Rabin transition*[3] *automaton with $o\big((1.65\,n)^n\big)$ states and 2^n-1 accepting pairs that recognises the language $\mathcal{L}(\mathcal{A})$ of $\mathcal{A}$.* □

[3] A transition automaton records the history of transitions in addition to the history of states. For such a history of transitions, we can translate the acceptance condition $1:1$ by using the nodes as index set, and (A_τ,R_τ) where A_τ are the transitions where τ is accepting, and R_τ are the transitions where τ is unstable as Rabin pairs.

To see that the number of accepting pairs is bounded by $2^n - 1$, note that the labels of siblings are disjoint, and that the label of every node contains a state not in the label of any of its children. Thus, the number of ancestors and their older siblings of every node is strictly smaller than n. Thus, a node $i_0 i_1 i_2 \dots i_n$ can be represented by a sequence of i_0 0's followed by a 1, followed by i_1 0's and so on, such that every node that can be accepting is representable by a sequence of strictly less than n 0's and 1's.

There are two obvious ways to transform an automaton with a Rabin condition on the transitions to an automaton with Rabin conditions on the states. The first option is to "postpone" the transitions by one step. The new states are (with the exception of one dedicated initial state $\widehat{q_0}$) pairs, consisting of a state of the transition automaton and the input letter read in the previous round. Thus, if the deterministic Rabin transition automaton has the run ρ on an input word α, then the resulting ordinary deterministic Rabin automaton has the run $\rho' = \widehat{q_0}, \big(\rho(0), \alpha(0)\big), \big(\rho(1), \alpha(1)\big), \big(\rho(2), \alpha(2)\big), \dots$.

Corollary 2. *For a given nondeterministic Büchi automaton $\mathcal{A}$ with n states over an alphabet with l letters, we can construct a deterministic Rabin automaton with $l \cdot o\big((1.65\,n)^n\big)$ states and $2^n - 1$ accepting pairs that recognises the language $\mathcal{L}(\mathcal{A})$ of $\mathcal{A}$.* □

Given that the alphabets tend to be small in practice — in particular compared to $(1.65\,n)^n$ — a blow-up linear in the alphabet size is usually acceptable. However, an alphabet may, in principle, have up to 2^{n^2} distinguishable letters, and the imposed bound is not very good for extremely large alphabets. (Two letters σ_1 and σ_2 can be considered equivalent or indistinguishable for a Büchi automaton $\mathcal{A} = (\Sigma, Q, I, \delta, F)$ if $\delta(q, \sigma_1) = \delta(q, \sigma_2)$ holds true for all states $q \in Q$ of the automaton $\mathcal{A}$.) As an alternative to preserving one input letter in the state-space, we enrich the history trees with information about which node of the resulting tree was accepting or unstable in the third step of the transition.

To estimate the number of different enriched history trees with n nodes, we have to take into account that the unstable and accepting nodes are not arbitrarily distributed over the tree: Only leaves can be accepting, and if a node of the tree in unstable, then all of its descendants and all of its younger siblings are unstable, too. Furthermore, only stable nodes can be accepting and the root cannot be unstable. (An unstable root implies that the Büchi automaton has no run for this word. Instead of allowing for an unstable root, we use a partial transition function.)

The number $eOrder(n)$ of ordered trees enriched with this information can be recursively computed using the following case distinction: If the eldest child 0 of the root is unstable, then all nodes but the root are unstable. Hence, the number of trees of this form is $order(n) = \frac{(2n-2)!}{n!(n-1)!}$. For the case that the eldest child 0 of the root is stable, there are $eOrder(n-1)$ trees where the size of the sub-tree rooted in 0 is $n-1$, and $eOrder(i) \cdot eOrder(n-i)$ trees where the sub-tree rooted in 0 contains $i \in \{1, \dots, n-2\}$ nodes. (Every tree can be uniquely defined by the tree rooted in 0, and the remaining tree. The special treatment of the case that

0 has no younger siblings is due to the fact that the root cannot be accepting if it has a child.) Thus, we have $eOrder(1) = 2$ (as a leaf, the root can be accepting or stable but not accepting), and

$$eOrder(n) = eOrder(n-1) + order(n) + \sum_{i=1}^{n-2} eOrder(i)\,eOrder(n-1)$$

for $n \geq 2$. A numerical analysis[4] of this sequence shows that $eOrder(n) < 6.738^n$. This allows for an estimation of the number $eHist(n)$ of enriched history trees for a Büchi automaton with n states similar to the estimation of the number $hist(n)$ of history trees:

$$eHist(n) \in \sup_{x>0} O\big(m(x) \cdot 6.738^{\beta(x)}\big) \subset o\big((2.66\,n)^n\big).$$

Corollary 3. *Given a nondeterministic Büchi automaton $\mathcal{A}$ with n states, we can construct a deterministic Rabin automaton with $o\big((2.66\,n)^n\big)$ states and $2^n - 1$ accepting pairs that recognises the language $\mathcal{L}(\mathcal{A})$ of $\mathcal{A}$.* □

5 From Nondeterministic Büchi Automata to Deterministic Parity Automata

From a practical point of view, it is often preferable to trade state-space for simpler acceptance conditions. Algorithms that solve Rabin games, for example, are usually exponential in the index, while the index of the constructions discussed in the previous sections is exponential in the size to the Büchi automaton we want to determinise.

While a reasonable index has been a side product of previous determinisation techniques [3,4,5], the smaller state-spaces resulting from the determinisation techniques discussed in Sections 3 and 4 are partly paid for by a higher index.

Traditional techniques for the transformation of Muller and Rabin or Streett to parity acceptance conditions use later [19] and index appearance records [4], respectively. However, using index (or later) appearance records would result in an exponential blow-up of the state-space, and hence in a doubly exponential construction. We therefore introduce the later introduction record as a record tailored for ordered trees.

A *later introduction record* (LIR) stores the order in which the nodes of the ordered trees have been introduced. For an ordered tree T with m nodes, a later introduction record is a sequence $\tau_1, \tau_2, \ldots \tau_m$ that contains the nodes of T, such that every node appears after its parent and older siblings.

To analyse the effect of adding a later introduction record to a history tree on the state-space, we slightly change the representation: We represent the tree structure of a tree with m nodes *and* its later introduction record by a sequence

[4] $\frac{eOrder(n+1)}{eOrder(n)}$ is growing, and $\big(\frac{eOrder(n+1)}{eOrder(n)}\big)\big(1 + \frac{2}{n}\big)$ is falling for growing $n \geq 2$.

of $m-1$ integers $i_2, i_3, \dots i_m$, such that i_j points to the position $< j$ of the parent of node τ_j in the later introduction record $\tau_1, \tau_2, \dots \tau_m$. (The root τ_1 has no parent.) There are $(m-1)!$ such sequences.

The labelling function of a history tree $\langle T, l\rangle$ whose root is labelled with the complete set Q of states of the Büchi automaton can be represented by a function from Q onto $\{1, \dots, m\}$ that maps each state $q \in Q$ to the positions of its host node in the LIR. Let $t(n,m)$ denote the number of trees and later introduction record pairs for such history trees with m nodes and $n = |Q|$ states in the label of the root. First, $t(n,n) = (n-1)!n!$ holds: There are $(n-1)!$ ordered-tree / LIR pairs, and $n!$ functions from a set with n elements onto itself. For every $m \leq n$, a coarse estimation[5] provides $t(n, m-1) \leq \frac{1}{2}t(n,m)$. Hence, $\sum_{i=1}^{n} t(n,i) \leq 2(n-1)!n!$.

Likewise, the labelling function of a history tree $\langle T, l\rangle$ whose root is labelled with the complete set Q of states of the Büchi automaton can be represented by a function from Q onto $\{1, \dots, m\}$ that maps each state $q \in Q$ to the positions of its host node in the LIR, or to 0 if the state is not in the label of the root. Let $t'(n,m)$ denote the number of history tree / LIR pairs for such history trees with m nodes for a Büchi automaton with n states. We have $t'(n, n-1) = (n-2)!n!$ and, by an argument similar to the one used in the analysis of t, we also have $t'(n, m-1) \leq \frac{1}{2}t'(n,m)$ for every $m < n$, and hence $\sum_{i=1}^{n-1} t'(n,i) \leq 2(n-2)!n!$.

Proposition 2. *For a given nondeterministic Büchi automaton $\mathcal{A}$ with n states, we can build a deterministic parity automaton with $O(n!^2)$ states and $2n$ priorities that recognises the language $\mathcal{L}(\mathcal{A})$ of $\mathcal{A}$.*

Proof. We construct a deterministic parity automaton, whose states consist of the history tree / LIR pairs, and an explicitly represented priority. The priority is determined by the position i of the first node in the previous LIR that is either unstable or accepting in the σ-transition: If it is accepting, the priority is $2i$, if it is unstable, the priority is $2i-1$. If no node is unstable or accepting, the priority is $2n+1$. The automaton has at most the priorities $\{2, 3, \dots, 2n+1\}$ and $O(n!^2)$ states — $O\big((n-1)!n!\big)$ history tree / LIR pairs times $2n$ priorities.

Let α be a word in the language $\mathcal{L}(\mathcal{A})$ of $\mathcal{A}$. Then there is by Proposition 1 a node τ that is always eventually accepting and eventually always stable in the history tree, and will hence eventually always remain in the same position p in the LIR and be stable. (A stable node can only move further to the front of the LIR, which can only happen finitely many times.) From that time onward, no node with a smaller position $p' < p$ is deleted (this would result τ to move further to the front of the record), nor is the node τ on position p unstable. Hence, no odd number $< 2p$ occurs infinitely many times. Also from that time

[5] If we connect functions by letting a function g from Q onto $\{1, \dots, m-1\}$ be the successor of a function f from Q onto $\{1, \dots, m\}$ if there is an index $i \in \{1, \dots, m-1\}$ such that $g(q) = i$ if $f(q) = m$ and $g(q) = f(q)$ otherwise, then the functions onto m have $(m-1)$ successors, while every function onto $m-1$ has at least two predecessors. Hence, the number of labelling functions growth at most by a factor of $\frac{m-1}{2}$, while the number of ordered tree / LIR pairs is reduced by a factor of $m-1$.

onward, the node τ is accepting infinitely many times, which results in visiting a priority $\leq 2p$ by our prioritisation rule. Hence the smallest number occurring infinitely many times is even.

Let, on the other hand, $2i$ be the dominating priority of the run of our deterministic parity automaton. Then eventually no lower priority than $2i$ appears, which implies that all positions $\leq i$ remain unchanged in the LIR, and the respective nodes remain stable from that time onward. Also, the node that is from that time onward on position i is accepting infinitely many times, which implies by Proposition 1 that α is in the language $\mathcal{L}(\mathcal{A})$ of $\mathcal{A}$. □

While the separation of concerns does not generate the same theoretical benefit with respect to state-space reduction when we construct parity automata instead of Rabin automata, the practical advantage might be comparable. While the update rule for history trees is strict, the update rule for LIR's is much less so: The only property of LIR updates used in the proof of Proposition 2 is that the position of accepting positions is reduced, and strictly reduced if there was an unstable node on a smaller position of the previous LIR. This leaves much leeway for updating the LIR — any update that satisfies this constraint will do.

Usually only a fragment of the state-space is reachable, and determinisation algorithms tend to construct the state-space of the automaton on the fly. The simplest way to exploit the leeway in the update rule for LIR's is to check if there is a suitable LIR such that a state with an appropriate history tree / LIR pair has already been constructed. If this is the case, then we can, depending on the priority of that state, turn to this state or construct a new state that differs only in the priority, which allows us to ignore the new state in the further expansion of the state-space. It is my belief that such a nondeterministic determinisation procedure will result in a significant state-space reduction compared to any deterministic rule.

References

1. Büchi, J.R.: On a decision method in restricted second order arithmetic. In: Proceedings of the International Congress on Logic, Methodology, and Philosophy of Science, 1960, Berkeley, California, USA, pp. 1–11. Stanford University Press (1962)
2. Rabin, M.O., Scott, D.S.: Finite automata and their decision problems. IBM Journal of Research and Development 3, 115–125 (1959)
3. Safra, S.: On the complexity of ω-automata. In: Proceedings of the 29th Annual Symposium on Foundations of Computer Science (FOCS 1988), White Plains, New York, USA, pp. 319–327. IEEE Computer Society Press, Los Alamitos (1988)
4. Muller, D.E., Schupp, P.E.: Simulating alternating tree automata by nondeterministic automata: new results and new proofs of the theorems of Rabin, McNaughton and Safra. Theoretical Computer Science 141, 69–107 (1995)
5. Piterman, N.: From nondeterministic Büchi and Streett automata to deterministic parity automata. Journal of Logical Methods in Computer Science 3 (2007)
6. Muller, D.E.: Infinite sequences and finite machines. In: Proceedings of the 4th Annual Symposium on Switching Circuit Theory and Logical Design (FOCS 1963), Chicago, Chicago, Illinois, USA, pp. 3–16. IEEE Computer Society Press, Los Alamitos (1963)

7. McNaughton, R.: Testing and generating infinite sequences by a finite automaton. Information and Control 9, 521–530 (1966)
8. Yan, Q.: Lower bounds for complementation of *omega*-automata via the full automata technique. Journal of Logical Methods in Computer Science 4 (2008)
9. Church, A.: Logic, arithmetic and automata. In: Proceedings of the International Congress of Mathematicians, Institut Mittag-Leffler, Djursholm, Sweden, 1962 (Stockholm 1963), 15–22 August, pp. 23–35 (1962)
10. Pnueli, A., Rosner, R.: On the synthesis of a reactive module. In: Proceedings of the 16th Annual ACM Symposium on Principles of Programming Languages (POPL 1989), Austin, Texas, USA, pp. 179–190. ACM Press, New York (1989)
11. Rabin, M.O.: Decidability of second order theories and automata on infinite trees. Transaction of the American Mathematical Society 141, 1–35 (1969)
12. Büchi, J.R., Landweber, L.H.: Solving sequential conditions by finite-state strategies. Transactions of the American Mathematical Society 138, 295–311 (1969)
13. Büchi, J.R., Landweber, L.H.: Definability in the monadic second-order theory of successor. Journal of Symbolic Logic 34, 166–170 (1969)
14. Wilke, T.: Alternating tree automata, parity games, and modal μ-calculus. Bulletin of the Belgian Mathematical Society 8 (2001)
15. Emerson, E.A.: Automata, tableaux and temporal logics. In: Parikh, R. (ed.) Logic of Programs 1985. LNCS, vol. 193, pp. 79–88. Springer, Heidelberg (1985)
16. Kupferman, O., Vardi, M.: Module checking revisited. In: Grumberg, O. (ed.) CAV 1997. LNCS, vol. 1254, pp. 36–47. Springer, Heidelberg (1997)
17. Emerson, E.A., Jutla, C.S.: Tree automata, μ-calculus and determinacy. In: Proceedings of the 32nd Annual Symposium on Foundations of Computer Science (FOCS 1991), San Juan, Puerto Rico, pp. 368–377. IEEE Computer Society Press, Los Alamitos (1991)
18. Schewe, S., Finkbeiner, B.: Satisfiability and finite model property for the alternating-time μ-calculus. In: Ésik, Z. (ed.) CSL 2006. LNCS, vol. 4207, pp. 591–605. Springer, Heidelberg (2006)
19. Gurevich, Y., Harrington, L.: Trees, automata, and games. In: Proceedings of the 14th Annual ACM Symposium on Theory of Computing (STOC 1982), San Francisco, California, USA, pp. 60–65. ACM Press, New York (1982)
20. Liu, W., Wang, J.: A tigher analysis of Piterman's Büchi determinization. Information Processing Letters (submitted, 2009)
21. Temme, N.M.: Asymptotic estimates of Stirling numbers. Studies in Applied Mathematics 89, 233–243 (1993)
22. Friedgut, E., Kupferman, O., Vardi, M.Y.: Büchi complementation made tighter. International Journal of Foundations of Computer Science 17, 851–867 (2006)