Unifying Büchi Complementation Constructions

Seth Fogarty[1], Orna Kupferman[2], Moshe Y. Vardi[1], and Thomas Wilke[3]

1 **Department of Computer Science, Rice University**
2 **School of Computer Science and Engineering, Hebrew University of Jerusalem**
3 **Institut für Informatik, Christian-Albrechts-Universität zu Kie**

Abstract

Complementation of Büchi automata, required for checking automata containment, is of major theoretical and practical interest in formal verification. We consider two recent approaches to complementation. The first is the *rank-based approach* of Kupferman and Vardi, which operates over a DAG that embodies all runs of the automaton. This approach is based on the observation that the vertices of this DAG can be ranked in a certain way, termed an *odd ranking*, iff all the runs are rejecting. The second is the *slice-based approach* of Kähler and Wilke. This approach is based on tracking levels of "split trees" – run trees in which only essential information about the history of each run is maintained. While the slice-based construction is conceptually simple, the complementing automata it generates are exponentially larger than those of the recent rank-based construction of Schewe, and it suffers from the difficulty of symbolically encoding levels of split trees.

In this work we reformulate the slice-based approach in terms of run DAGs and preorders over states. In doing so, we begin to draw parallels between the rank-based and slice-based approaches. Through deeper analysis of the slice-based approach, we strongly restrict the nondeterminism it generates. We are then able to employ the slice-based approach to provide a new odd ranking, called a *retrospective ranking*, that is different from the one provided by Kupferman and Vardi. This new ranking allows us to construct an deterministic-in-the-limit rank-based automaton with a highly restricted transition function. Further, by phrasing the slice-based approach in terms of ranks, our approach affords a simple symbolic encoding and achieve the tight bound of Schewe's construction.

1 Introduction

The complementation problem for nondeterministic automata is central to the automata-theoretic approach to formal verification [24]. To test that the language of an automaton $\mathcal{A}$ is contained in the language of a second automaton $\mathcal{B}$, check that the intersection of $\mathcal{A}$ with an automaton that complements $\mathcal{B}$ is empty. In model checking, the automaton $\mathcal{A}$ corresponds to the system, and the automaton $\mathcal{B}$ corresponds to a property [25]. While it is easy to complement properties given as temporal logic formulas, complementation of properties given as automata is not simple. Indeed, a word w is rejected by a nondeterministic automaton $\mathcal{A}$ if *all* runs of $\mathcal{A}$ on w reject the word. Thus, the complementary automaton has to consider all possible runs, and complementation has the flavor of determinization. Representing liveness, fairness, or termination properties requires automata that recognize languages of infinite words. Most commonly considered are nondeterministic Büchi automata, in which some of the states are designated as accepting, and a run is accepting if it visits accepting states infinitely often [2]. For automata on finite words, determinization, and hence also complementation, is done via the subset construction [15]. For Büchi automata the subset construction is not sufficient, and optimal complementation constructions are more complicated [22].

Efforts to develop simple complementation constructions for Büchi automata started early in the 60s, motivated by decision problems of second-order logics. Büchi suggested a complementation construction for nondeterministic Büchi automata that involved a Ramsey-based combinatorial argument and a doubly-exponential blow-up in the state space [2]. Thus, complementing an automaton with n states resulted in an automaton with $2^{2^{O(n)}}$ states. In [19], Sistla et al. suggested an improved implementation of Büchi's construction, with only $2^{O(n^2)}$ states, which is still not optimal. Only

in [16] Safra introduced a determinization construction, based on *Safra trees*, which also enabled a $2^{O(n \log n)}$ complementation construction, matching a lower bound described by Michel [11]. A careful analysis of the exact blow-up in Safra's and Michel's bounds, however, reveals an exponential gap in the constants hiding in the $O()$ notations: while the upper bound on the number of states in the complementary automaton constructed by Safra is n^{2n}, Michel's lower bound involves only an $n!$ blow up, which is roughly $(n/e)^n$. In addition, Safra's construction has been resistant to optimal implementations [1, 21], which has to do with the complicated combinatorial structure of its states and transitions, which can not be encoded symbolically.

The use of complementation in practice has led to a resurgent interest in the exact blow-up that complementation involves and the feasibility of a symbolic complementation construction. In 2001, Kupferman and Vardi suggested a new analysis of runs of Büchi automata that led to a simpler complementation construction [10]. In this new analysis, one considers the DAG that embodies all the runs of an automaton $\mathcal{A}$ on a given word w. It is shown in [10] that the nodes of such a DAG can be mapped to ranks, where the rank of a node essentially indicates the progress made towards reaching a suffix of the run in which no accepting states are visited. Further, all the runs of $\mathcal{A}$ on w are rejecting iff there is a *bounded odd ranking* of the DAG: one in which the maximal rank is bounded, ranks along paths do not increase, paths become trapped in odd ranks, and nodes associated with accepting states are not assigned an odd rank. Consequently, complementation can circumvent Safra's determinization construction along with the complicated data structure of Safra trees, and can instead be based on an automaton that guesses an odd ranking. The state space of such an automaton is based on annotating states in subsets with the guessed ranks. Beyond the fact that the *rank-based construction* can be implemented symbolically [20], it gave rise to a sequence of works improving both the blow-up it involves and its implementation in practice. The most notable improvements are the introduction of tight rankings [5] and Schewe's improved cut-point construction [17]. These improvements tightened the $(6n)^n$ upper bound of [10] to $(0.76n)^n$. Together with recent work on a tighter lower bound [26], the gap between the upper and lower bound is now a quadratic term. Addressing practical concerns, Doyen and Raskin have introduced a useful subsumption technique for the rank-based approach [4].

In an effort to unify Büchi complementation with other operations on automata, Kähler and Wilke introduced yet another analysis of runs of nondeterministic Büchi automata [7]. The analysis is based on *reduced split trees*, which are related to the Müller-Schupp trees used for determinization [13]. A reduced split tree is a binary tree whose nodes are sets of states as follows: the root is the set of initial states; and given a node associated with a set of states, its left child is the set of successors that are accepting, while the right child is the set of successors that are not accepting. In addition, each state of the automaton appears at most once in each level of the binary tree: if it would appear in more than one set, it occurs only in the leftmost one. The construction that follows from the analysis, termed the *slice-based construction*, is simpler than Safra's determinization, but its implementation suffers from similar difficulties: the need to refer to leftmost children requires encoding of a preorder, and working with reduced split trees trees makes the transition relation between states awkward. Thus, as has been the case with Safra's construction, it is not clear how the slice-based approach can be implemented symbolically. This is unfortunate, as the slice-based approach does offer a very clean and intuitive analysis, suggesting that a better construction is hidden in it.

In this paper we reveal such a hidden, elegant, construction, and we do so by unifying the rank-based and the slice-based approaches. Before we turn to describe our construction, let us point to a key conceptual difference between the two approaches. This difference has made their relation of special interest and challenge. In the rank-based approach, the ranks assigned to a node bound the visits to accepting states yet to come. Thus, the ranks refer to the *future* of the run, making the rank-based approach inherently nondeterministic. In contrast, in the slice-based approach, the partition

of the states of the automaton to the different sets in the tree is based on previous visits to accepting states. Thus, the partition refers to the *past* of the run, and does not depend on its future.

In order to draw parallels between the two approaches, we present a formulation of the slice-based approach in terms of run DAGs. A careful analysis of the slice-based approach then enables us to reduce the nondeterminism in the construction. We can then employ this improved slice-based approach in order to define a particular odd ranking of rejecting run DAGs, called a *retrospective ranking*. In addition to revealing the theoretical connections between the two seemingly different approaches, the new ranks lead to a complementation construction with a transition function that is smaller and deterministic in the limit: every accepting run of the automaton is eventually deterministic. This presents the first deterministic-in-the-limit complementation construction that does not use determinization. Determinism in the limit is central to verification in probabilistic settings [3] and has proven useful in experimental results [18]. Phrasing slice-based complementation as an odd ranking also immediately affords us the improved cut-point of Schewe, the subsumption operation of Doyen and Raskin, and provides an easy symbolic encoding.

2 Preliminaries

A *nondeterministic Büchi automaton on infinite words* (NBW) is a tuple $\mathcal{A} = \langle \Sigma, Q, Q^{in}, \rho, F \rangle$, where Σ is a finite alphabet, Q a finite set of states, $Q^{in} \subseteq Q$ a set of initial states, $F \subseteq Q$ a set of accepting states, and $\rho : Q \times \Sigma \to 2^Q$ a nondeterministic transition relation. A state $q \in Q$ is *deterministic* if for every $\sigma \in \Sigma$ it holds that $|\rho(q, \sigma)| \leq 1$. We lift the function ρ to sets R of states in the usual fashion: $\rho(R, \sigma) = \bigcup_{q \in R} \rho(q, \sigma)$.

A *run* of an NBW $\mathcal{A}$ on a word $w = \sigma_0 \sigma_1 \cdots \in \Sigma^\omega$ is an infinite sequence of states $p_0, p_1, \ldots \in Q^\omega$ such that $p_0 \in Q^{in}$ and, for every $i \geq 0$, we have $p_{i+1} \in \rho(p_i, \sigma_i)$. A run is *accepting* iff $p_i \in F$ for infinitely many $i \in \mathbb{N}$. A word $w \in \Sigma^\omega$ is accepted by $\mathcal{A}$ if there is an accepting run of $\mathcal{A}$ on w. The words accepted by $\mathcal{A}$ form the *language* of $\mathcal{A}$, denoted by $L(\mathcal{A})$. The complement of $L(\mathcal{A})$, denoted $\overline{L(\mathcal{A})}$, is $\Sigma^\omega \setminus L(\mathcal{A})$. We say an automaton is *deterministic in the limit* if every state reachable from an accepting state is deterministic. Converting $\mathcal{A}$ to an equivalent deterministic in the limit automaton involves an exponential blowup [3, 16]. One can simultaneously complement and determinize in the limit, via co-determinization into a parity automaton [14], and then converting that parity automaton to a deterministic-in-the-limit Büchi automaton, with a cost of $(n^2/e)^n$.

Consider an NBW $\mathcal{A}$ and an infinite word $w = \sigma_0 \sigma_1 \cdots$. The runs of $\mathcal{A}$ on w can be arranged in an infinite DAG (directed acyclic graph) $G = \langle V, E \rangle$, where

- $V \subseteq Q \times \mathbb{N}$ is such that $\langle q, i \rangle \in V$ iff some run p of $\mathcal{A}$ on w has $p_i = q$.
- $E \subseteq \bigcup_{i \geq 0} (Q \times \{i\}) \times (Q \times \{i+1\})$ is s.t. $E(\langle q, i \rangle, \langle q', i+1 \rangle)$ iff $\langle q, i \rangle \in V$ and $q' \in \rho(q, \sigma_i)$.

The DAG G, called the *run* DAG *of* $\mathcal{A}$ *on* w, embodies all possible runs of $\mathcal{A}$ on w. We are primarily concerned with *initial paths* in G: paths that start in $Q^{in} \times \{0\}$. Define a node $\langle q, i \rangle$ to be an F-node when $q \in F$, and a path in G to be *accepting* when it is both initial and contains infinitely many F-nodes. An accepting path in G corresponds to an accepting run of $\mathcal{A}$ on w. When G contains an accepting path, call G an accepting run DAG, otherwise call it a rejecting run DAG. We often consider DAGs H that are subgraphs of G. A node u is a *descendant* of v in H when u is reachable from v in H. A node v is *finite* in H if it has only finitely many descendants in H. A node v is *F-free* in H if it is not an F-node, and has no descendants in H that are F-nodes. We say a node *splits* when it has at least two children, and conversely that two nodes *join* when they share a common child.

Example 1. In Figure 1 we describe an NBW $\mathcal{A}$ that accepts words with finitely many b's. On the right is a prefix of the rejecting run DAG of $\mathcal{A}$ on $w = babaabaaabaaaa \cdots$.

If an NBW $\mathcal{A}$ does not accept a word w, then every run of $\mathcal{A}$ on w must eventually cease visiting accepting states. The notion of *ranking*s, foreshadowed in [9] and introduced in [10], uses natural

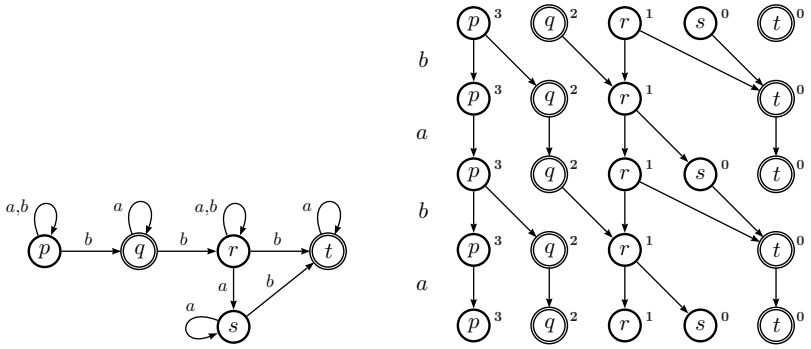

Figure 1 Left, the NBW $\mathcal{A}$, in which all states are initial. Right, the rejecting run DAG G of $\mathcal{A}$ on $w = babaabaaabaaaa\cdots$. Nodes are superscripted with the prospective ranks of Section 2.

numbers to track the progress of each run in the DAG towards this point. A ranking for a DAG $G = \langle V, E\rangle$ is a mapping from V to $\mathbb{N}$, in which no F-node is given an odd rank, and in which the ranks along all paths do not increase. Formally, a ranking is a function $\mathbf{r} : V \to \mathbb{N}$ such that if $u \in V$ is an F-node then $\mathbf{r}(u)$ is even; and for every $u, v \in V$, if $(u, v) \in E$ then $\mathbf{r}(u) \geq \mathbf{r}(v)$. Since each path starts at a finite rank and ranks cannot increase, every path eventually becomes trapped in a rank. A ranking is called an *odd ranking* if every path becomes trapped in an odd rank. Since F-nodes cannot have odd ranks, if there is an odd ranking $\mathbf{r}$, then every path in G must stop visiting accepting nodes when it becomes trapped in its final, odd, rank, and G must be a rejecting DAG.

▶ **Lemma 1.** [10] *If a run* DAG G *has an odd ranking, then* G *is rejecting.*

A ranking is *bounded by* l when its range is $\{0..l\}$, and an NBW $\mathcal{A}$ is of rank l when for every $w \notin L(\mathcal{A})$, the rejecting DAG G has an odd ranking bounded by l. If we can prove that an NBW $\mathcal{A}$ is of rank l, we can use the notion of odd rankings to construct a complementary automaton. This complementary NBW, denoted $\mathcal{A}^l_R$, tracks the levels of the run DAG and attempts to guess an odd ranking bounded by l. An *l-bounded level ranking* for an NBW $\mathcal{A}$ is a function $f : Q \to \{0, \ldots, l, \bot\}$, such that if $q \in F$ then $f(q)$ is even or $\bot$. Let $\mathcal{R}^l$ be the set of all l-bounded level rankings. The state space of $\mathcal{A}^l_R$ is based on the set of l-bounded level rankings for $\mathcal{A}$. To define transitions of $\mathcal{A}^l_R$, we need the following notion: for $\sigma \in \Sigma$ and $f, f' \in \mathcal{R}^l$, say that *f' follows f under σ* when for every $q \in Q$ and $q' \in \rho(q, \sigma)$, if $f(q) \neq \bot$ then $f'(q') \neq \bot$ and $f'(q') \leq f(q)$: i.e. no transition between f and f' on σ increases in rank. Finally, to ensure that the guessed ranking is an odd ranking, we employ the cut-point construction of Miyano and Hayashi, which maintains an obligation set of nodes along paths obliged to visit an odd rank [12]. For a level ranking f, let $even(f) = \{q \mid f(q) \text{ is even}\}$ and $odd(f) = \{q \mid f(q) \text{ is odd}\}$.

▶ **Definition 2.** For an NBW $\mathcal{A} = \langle \Sigma, Q, Q^{in}, \rho, F\rangle$ and $l \in \mathbb{N}$, define $\mathcal{A}^l_R$ to be the NBW $\langle \Sigma, \mathcal{R}^l \times 2^Q, \langle f^{in}, \emptyset\rangle, \rho_R, \mathcal{R}^l \times \{\emptyset\}\rangle$, where

- $f^{in}(q) = l$ for each $q \in Q^{in}$, $\bot$ otherwise.
- $\rho_R(\langle f, O\rangle, \sigma) = \begin{cases} \{\langle f',\ \rho(O, \sigma) \setminus odd(f')\rangle \mid f' \text{ follows } f \text{ under } \sigma\} & \text{if } O \neq \emptyset, \\ \{\langle f',\ even(f')\rangle \mid f' \text{ follows } f \text{ under } \sigma\} & \text{if } O = \emptyset. \end{cases}$

By [10], for every $l \in N$, the NBW $\mathcal{A}^l_R$ accepts only words rejected by $\mathcal{A}$ – exactly all words for which there exists an odd ranking with maximal rank l. In addition, [10] proves that for every rejecting run DAG there exists a bounded odd ranking. Below we sketch the derivation of this ranking. Given a rejecting run DAG G, we inductively define a sequence of subgraphs by eliminating nodes

that cannot be part of accepting runs. At odd steps we remove finite nodes, while in even steps we remove nodes that are F-free. Formally, define a sequence of subgraphs as follows:

- $G_0 = G$.
- $G_{2i+1} = G_{2i} \setminus \{v \mid v \text{ is finite in } G_{2i}\}$.
- $G_{2i+2} = G_{2i+1} \setminus \{v \mid v \text{ is } F\text{-free in } G_{2i+1}\}$.

It is shown in [6, 10] that only $m = 2|Q \setminus F|$ steps are necessary to remove all nodes from a rejecting run DAG: G_m is empty. Nodes can be ranked by the last graph in which they appear: for every node $u \in G$, the *prospective rank* of u is the index i such that $u \in G_i$ but $u \notin G_{i+1}$. The *prospective ranking* of G assigns every node its prospective rank. Paths through G cannot increase in prospective rank, and no F-node can be given an odd rank: thus the prospective ranking abides by the requirements for rankings. We call these rankings prospective because the rank of a node depends solely on its descendants. By [10], if G is a rejecting run DAG, then the prospective ranking of G is an odd ranking bounded by m. By the above, we thus have the following.

▶ **Theorem 3.** [10] *For every NBW $\mathcal{A}$, it holds that $L(\mathcal{A}_R^m) = \overline{L(\mathcal{A})}$.*

Example 2. In Figure 1, nodes for states s and t are finite in G_0. With these nodes removed, r-nodes are F-free in G_1. Without r-nodes, q-nodes are finite in G_2. Finally, p-nodes are F-free in G_3.

Karmarkar and Chakraborty have derived both theoretical and practical benefits from exploiting properties of this prospective ranking: they demonstrated an unambiguous complementary automaton that, for certain classes of problems, is exponentially smaller than $\mathcal{A}_R^m$ [8].

Tight Rankings: For an odd ranking $\mathbf{r}$ and $l \in N$, let $max\_rank(\mathbf{r}, l)$ be the maximum rank that $\mathbf{r}$ assigns a vertex on level l. We say that $\mathbf{r}$ is *tight* [1] if there exists an $i \in N$ such that, for every level $l \geq i$, all odd ranks below $max\_rank(\mathbf{r}, l)$ appear on level l. It is shown in [5] that the retrospective ranking is tight. This observation suggests two improvements to $\mathcal{A}_R^m$. First, we can postpone, in an unbounded manner, the level in which it starts to guess the level ranking. Until this point, $\mathcal{A}_R^m$ may use sets of states to deterministically track only the levels of the run DAG, with no attempt to guess the ranks. Second, after this point, $\mathcal{A}_R^m$ can restrict attention to *tight level rankings* – ones in which all the odd ranks below the maximal rank appear. Formally, say a level ranking f with a maximum rank $mr = \max\{f(q) \mid q \in Q,\ f(q) \neq \bot\}$ is tight when, for every odd $i \leq mr$, there exists a $q \in Q$ such that $f(q) = i$. Let $\mathcal{R}_T^m$ be the subset of $\mathcal{R}^m$ that contains only tight level rankings. The size of $\mathcal{R}_T^m$ is at most $(0.76n)^n$ [5]. Including the cost of the cut-point construction, this reduces the state space of $\mathcal{A}_R^m$ to $(0.96n)^n$.

3 Analyzing DAGs With Profiles

In this section we present an alternate formulation of the slice-based complementation construction of Kähler and Wilke [7]. Whereas Kähler and Wilke approached the problem through reduced split trees, we derive the slice-based construction directly from an analysis of the run DAG. This analysis proceeds by pruning G in two steps: the first removes edges, and the second removes vertices.

3.1 Profiles

Consider a run DAG $G = \langle V, E \rangle$. Let $l : V \to \{0, 1\}$ be such that $l(\langle q, i \rangle) = 1$ if $q \in F$ and $l(\langle q, i \rangle) = 0$ otherwise. Thus, l labels F-nodes by 1 and all other nodes by 0. The *profile* of a path in G is the sequence of labels of nodes in the path. The profile of a node is then

[1] This definition of tightness for an odd ranking is weaker that of [5], but does not affect the resulting bounds.

the lexicographically maximal profile of all initial paths to that node. Formally, let $\leq$ be the lexicographic ordering on $\{0,1\}^* \cup \{0,1\}^\omega$. The profile of a finite path $b = v_0, v_1, \ldots, v_n$ in G, written h_b, is $l(v_0)l(v_1)\cdots l(v_n)$, and the profile of an infinite path $b = v_0, v_1, \ldots$ is $h_b = l(v_0)l(v_1)\cdots$. Finally, the profile of a node v, written h_v, is the lexicographically maximal element of $\{h_b \mid b \text{ is an initial path to } v\}$. The lexicographic order of profiles induces a preorder over nodes.

We define the sequence of preorders $\preceq_i$ over the nodes on each level of the run DAG as follows. For every two nodes u and v on a level i, we have that $u \prec_i v$ if $h_u < h_v$, and $u \approx_i v$ if $h_u = h_v$. For convenience, we conflate nodes on the ith level of the run DAG with their states when employing this preorder, and say $q \preceq_i r$ when $\langle q, i\rangle \preceq_i \langle r, i\rangle$. Note that $\approx_i$ is an equivalence relation. Since the final element of a node's profile is 1 iff the node is an F-node, all nodes in an equivalence class must agree on membership in F. We call an equivalence class an F-class when all its members are F-nodes, and a non-F-class when none of its members is an F-node. We now use profiles in order to remove from G edges that are not on lexicographically maximal paths. Let G' be the subgraph of G obtained by removing all edges $\langle u, v\rangle$ for which there is another edge $\langle u', v\rangle$ such that $u \prec_{|u|} u'$. Formally, $G' = \langle V, E'\rangle$ where $E' = E \setminus \{\langle u, v\rangle \mid \text{there exists } u' \in V \text{ such that } \langle u', v\rangle \in E \text{ and } u \prec_{|u|} u'\}$.

▶ **Lemma 4.** *For every two nodes u and v, if $(u, v) \in E'$, then $h_v \in \{h_u 0, h_u 1\}$.*

Proof. Assume by way of contradiction that $h_v \notin \{h_u 0, h_u 1\}$. Recall that h_v is the lexicographically maximal element of $\{h_b \mid b \text{ is an initial path to } v\}$. Thus our assumption entails an initial path b to v so that $h_b > h_u 1$. Let u' be $b_{|u|}$: the node on the same level of G as u. Since b is a path to v, it holds that $(u', v) \in E$. Further, since $h_b > h_u 1$, it must be that $h_{u'} > h_u$. By definition of E', the presence of (u', v) where $h_{u'} > h_u$ precludes the edge (u, v) from being in E': a contradiction. □

Note that while in it is possible for two nodes with different profiles to share a child in G, Lemma 4 precludes this possibility in G'. If two nodes join in G', they must have the same profile and be in the same equivalence class. We can thus conflate nodes and equivalence classes, and for every edge $(u, v) \in E'$, consider $[v]$ to be the child of $[u]$. Lemma 4 then entails that the class $[u]$ can have at most two children: the class of F-nodes with profile $h_u 1$, and the class of non-F-nodes with profile $h_u 0$. We call the first class the F-child of $[u]$, and the second class the non-F-child of $[u]$.

By using lexicographic ordering we can derive the preorder for each level $i+1$ of the run DAG solely from the preorder for the previous level i. To determine the relation between two nodes, we need only know the relation between the parents of those nodes, and whether the nodes are F-nodes. Formally, we have the following.

▶ **Lemma 5.** *For all nodes u, v on level i, and nodes u', v' where $E'(u, u')$ and $E'(v, v')$:*
- *If $u \prec_i v$, then $u' \prec_{i+1} v'$.*
- *If $u \approx_i v$ and either both u' and v' are F-nodes, or neither are F-nodes, then $u' \approx_{i+1} v'$.*
- *If $u \approx_i v$ and v' is an F-node while u' is not, then $u' \prec_{i+1} v'$.*

Proof. If $u \prec_i v$, then $h_u < h_v$, and by Lemma 4 $h_{u'} \in \{h_u 0, h_u 1\}$ must be smaller than $h_{v'} \in \{h_v 0, h_v 1\}$, implying that $u' \prec_{i+1} v'$. If $u \approx_i v$, we have three sub-cases. If it is the sub-case that v' is an F-node and u' is not, then $h_{u'} = h_u 0 = h_v 0 < h_v 1 = h_{v'}$, and $u' \prec_{i+1} v'$. If it is the sub-case that both u' and v' are F-nodes, then $h_{u'} = h_u 1 = h_v 1 = h_{v'}$, implying that $u' \approx_i v'$. Finally, if neither u' nor v' are F-nodes, then $h_{u'} = h_u 0 = h_v 0 = h_{v'}$ and $u' \approx_i v'$. □

We now demonstrate that by keeping only edges associated with lexicographically maximal profiles, we capture an accepting path in G'.

▶ **Lemma 6.** *G' has an accepting path iff G has an accepting path.*

Proof. In one direction, if G' has an accepting path, then its superset G has the same path.

In the other direction, assume G has an accepting path. Consider the set P of accepting paths in G. We prove that there is a lexicographically maximal element $\pi \in P$. To begin, we construct an infinite sequence, $P_0, P_1, \ldots$, of subsets of P such that the elements of P_i are lexicographically maximal in the first $i+1$ positions. If P contains paths starting in an F-node, then $P_0 = \{b \mid b \in P,\ b_0 \text{ is an } F\text{-node}\}$ is all elements beginning in F-nodes . Otherwise $P_0 = P$. Inductively, if P_i contains an element b such that b_{i+1} is an F-node, then $P_{i+1} = \{b \mid b \in P_i,\ b_{i+1} \text{ is an } F\text{-node}\}$. Otherwise $P_{i+1} = P_i$. For convenience, define the predecessor of P_i to be P if $i = 0$, and P_{i-1} otherwise. Note that since G has an accepting path, P is non-empty. Further, every set P_i is not equal to its predecessor P' only when there is a path in P' with an F-node in the ith position. In this case, that path is in P_i. Thus every P_i is non-empty.

First, we prove that there is a path $\pi \in \bigcap_{i \geq 0} P_i$. Consider the sequence $U_0, U_1, U_2, \ldots$ where U_i is the set of nodes that occur at position i in runs in P_i. Formally, $U_i = \{u \mid u \in G,\ b \in P_i,\ u = b_i\}$. Each node in U_{i+1} has a parent in U_i, although it may not have a child in U_{i+2}. We can thus connect the nodes in $\bigcup_{i>0} U_i$ to their parents, forming a sub-DAG of G. As every P_i is non-empty, every U_i is non-empty, and this DAG has infinitely many nodes. Since each node has at most n children, by Koñig's Lemma there is an initial path π through this DAG, and thus through G. We now show by induction that $\pi \in P_i$ for every i. As a base case, $\pi \in P$. Inductively, assume π is in the predecessor P' of P_i. The set P_i is either P', in which case $\pi \in P_i$, or the set $\{b \mid b \in P',\ b_i \text{ is an } F\text{-node}\}$. In this latter case, as U_i consists only of F-nodes, the node π_i must be an F-node. and $\pi \in P_i$.

Second, having established that there must be an element $\pi \in \bigcap_{i \geq 0} P_i$, we prove π is lexicographically maximal in P. Assume by way of contradiction that there exists an accepting path π' so that $h_{\pi'} > h_\pi$. Let k be the first point where $h_{\pi'}$ differs from h_π. At this point, it must be that π_k is not an F node, while π'_k is an F node. However, π' is an accepting path that shares a profile with π up until this point. As π is in the predecessor P' of P_k, it must also be that π' is in P'. By definition, P_k then would be $\{b \mid b \in P',\ b_k \text{ is an } F\text{-node}\}$. This would imply $\pi \notin P_k$, a contradiction.

Finally, we demonstrate that every edge in π occurs in G'. Assume by way of contradiction that some edge (π_i, π_{i+1}) is in E but not in E'. This implies there is a node u on level i such that (u, π_{i+1}) is in E and $\pi_i \prec_i u$. Since $u \in G$, there is an initial path b to u. Thus, the path $b, u, \pi_{i+1}, \pi_{i+2} \ldots$ is an accepting path in G. This path would be lexicographically larger than π, contradicting the second claim above. Hence, we conclude π is an accepting path in G'. □

In the next stage, we remove from G' finite nodes. Let $G'' = G' \setminus \{v \mid v \text{ is finite in } G'\}$. Note there may be nodes that are not finite G but are finite in G'. It is not hard to see that G may have infinitely many F-nodes and still not contain a path with infinitely many F-nodes. Indeed, G may have infinitely many paths each with finitely many F-nodes. We now show that the transition from G via G' to G'' removes this possibility, and the presence of infinitely many F-nodes in G'' does imply the existence of a path with infinitely many F-nodes.

▶ **Lemma 7.** *G has an accepting path iff G'' has infinitely many F-nodes.*

Proof. Assume first that G has an accepting path. Then, by Lemma 6, the DAG G' contains an accepting path. Every node in this path is infinite in G', and thus this path is preserved in G''. This path contains infinitely many F-nodes, and thus G'' contains infinitely many F-nodes.

In the other direction, we consider the DAG over equivalence classes induced by G''. Given a node u in G'', recall that its equivalence class in G'' contains all states v such that $v \in G''$ and $h_u = h_v$. Given two equivalence classes U and V, recall that V is a child of U when there are $u \in U$, $v \in V$, and $E''(u, v)$. As mentioned above, once we have pruned edges not in G', two nodes of different classes cannot join. Thus this DAG is a tree. Further, as every node u in G'' is infinite and has a child, its equivalence class must also have a child. Thus the DAG of classes in G'' is a

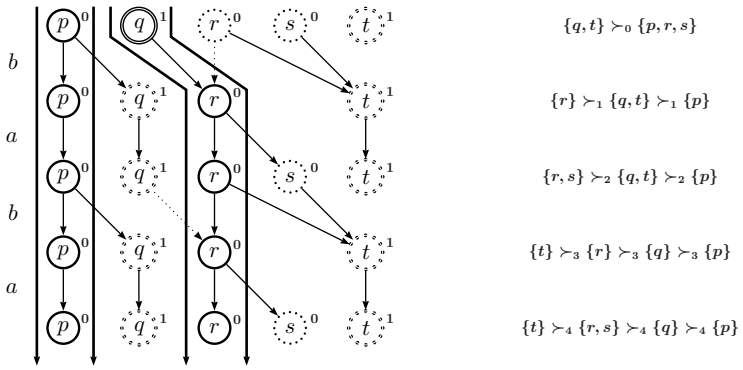

Figure 2 The run DAG G'', where dotted edges were removed from G and dotted states were removed from G'. Nodes are superscripted with their l-labels. Bold lines denote the pipes of G''. The lexicographic order of equivalence classes for each level of G' is to the right.

leafless tree. The width of this tree must monotonically increase and is bounded by n. It follows that at some level j the tree reaches a stable width. We call this level j the *stabilization level* of G.

After the stabilization level, each class U has exactly one child: as noted above, U cannot have zero children, and if U had two children the width of the tree would increase. Therefore, we identify each equivalence class on level j of G'' with its unique branch of children in G'', which we term its *pipe*. These pipes form a partition of nodes in G'' past j. Every node in these pipes has an ancestor, or it would not be in the DAG, and has a child, or it would not be infinite and in G''. Therefore each node is part of an infinite path in this pipe. Thus, the pipe with infinitely many F-classes contains only accepting paths. These paths are accepting in G, which subsumes G''. □

In the proof above we demonstrated there is a *stabilization level* j at which the number of equivalence classes in G'' stabilized, and discussed the *pipes* of G'': the single chain of descendants from each equivalence class on the stabilization level j of G''.

Example 3. Figure 2 displays G'' for the example of Figure 1. Edges removed from G' are dotted: at levels 1 and 3. When both r and s transition to t, they have the same profile and both edges remain. The removed edges render all but the first q-node finite in G'. The stabilization level is 0.

3.2 Complementing With Profiles

We now complement $\mathcal{A}$ by constructing an NBW, $\mathcal{A}_S$, that employs Lemma 7 to determine if a word is in $L(\mathcal{A})$. This construction is a reformulation of the slice-based approach of [7] in the framework of run DAGs; see Appendix C. The NBW $\mathcal{A}_S$ tracks the levels of G' and guesses which nodes are finite in G' and therefore do not occur in G''. To track G', the automaton $\mathcal{A}_S$ stores at each point in time a set S of states that occurs on each level. The sets S are labeled with a guess of which nodes are finite and which are infinite. States that are guessed to be infinite, and thus correspond to nodes in G'', are labeled $\top$, and states that are guessed to be finite, and thus omitted from G'', are labeled $\bot$. In order to track the edges of G', and thus maintain this labeling, $\mathcal{A}_S$ needs to know the lexicographic order of nodes. Thus $\mathcal{A}_S$ also maintains the preorder $\preceq_i$ over states on the corresponding level of the run DAG. To enforce that states labeled $\bot$ are indeed finite, $\mathcal{A}_S$ employs the cut-point construction of Miyano and Hayashi [12], keeping an "obligation set" of states currently being verified as finite. Finally, to ensure the word is rejected, $\mathcal{A}_S$ must enforce that there are finitely many F-nodes in G''. To do so, S_A uses a bit b to guess the level from which no more F-nodes appear in G''. At this point, it enforces that all F-nodes are labeled $\bot$.

Before we define $\mathcal{A}_S$, we formalize *preordered subsets* and operations over them. For a set Q of states, define $\mathbf{Q} = \{\langle S, \preceq \rangle \mid S \subseteq Q \text{ and } \preceq \text{ is a preorder over } S\}$ to be the set of preordered

subsets of Q. Let $\langle S, \preceq\rangle$ be an element in $\mathbf{Q}$. When considering the successors of a state, we want to consider edges that remain in G'. For every state $q \in S$ and $\sigma \in \Sigma$, define $\rho_{\langle S,\preceq\rangle}(q,\sigma) = \{r \in \rho(q,\sigma) \mid \text{for every } q' \in S, \text{ if } r \in \rho(q',\sigma) \text{ then } q' \preceq q\}$. Now define the *$\sigma$-successor of* $\langle S, \preceq\rangle$ as the tuple $\langle \rho(S,\sigma), \preceq'\rangle$, where for every $q, r \in S$, $q' \in \rho_{\langle S,\preceq\rangle}(q,\sigma)$, and $r' \in \rho_{\langle S,\preceq\rangle}(r,\sigma)$:

- If $q \prec r$, then $q' \prec' r'$
- If $q \approx r$ and either both $r' \in F$ and $q' \in F$, or both $r' \notin F$ and $q' \notin F$, then $q' \approx' r'$.
- If $q \approx r$ and one of q' and r', say r', is in F while the other, q', is not, then $q' \prec' r'$.

We now define $\mathcal{A}_S$. The states of $\mathcal{A}_S$ are tuples $\langle S, \preceq, \lambda, O, b\rangle$ where: $\langle S, \preceq\rangle \in \mathbf{Q}$ is preordered subset of Q; $\lambda : S \to \{\top, \bot\}$ is a labeling indicating which states are guessed to be finite ($\bot$) or infinite ($\top$), $O \subseteq S$ is the obligation set, and $b \in \{0, 1\}$ is a bit indicating whether we have seen the last F-node in G''. To transition between states of $\mathcal{A}_s$, say that $\boldsymbol{t}' = \langle S', \preceq', \lambda', O', b'\rangle$ *follows* $\boldsymbol{t} = \langle S, \preceq, \lambda, O, b\rangle$ *under* σ when:

(1) $\langle S', \preceq'\rangle$ is the σ-successor of $\langle S, \preceq\rangle$.

(2) λ' is such that for every $q \in S$:
- If $\lambda(q) = \top$, then there exists $r \in \rho_{\langle S,\preceq\rangle}(q,\sigma)$ such that $\lambda'(r) = \top$,
- If $\lambda(q) = \bot$, then for every $r \in \rho_{\langle S,\preceq\rangle}(q,\sigma)$, it holds that $\lambda'(r) = \bot$.

(3) $O' = \begin{cases} \bigcup_{q\in O} \rho_{\langle S,\preceq\rangle}(q,\sigma) & O \neq \emptyset, \\ \{q \mid q \in S' \text{ and } \lambda'(q) = \bot\} & O = \emptyset. \end{cases}$

(4) $b' \geq b$.

We want to ensure that runs of A_S reach a suffix where all F-nodes are labeled finite. To this end, given a state of $\mathcal{A}_S$ $\langle S, \preceq, \lambda, O, b\rangle$, we say that λ is *F-free* if for every $q \in S \cap F$ we have $\lambda(q) = \bot$.

▶ **Definition 8.** For an NBW $\mathcal{A} = \langle \Sigma, Q, Q^{in}, \rho, F\rangle$, let $\mathcal{A}_S$ be the NBW $\langle \Sigma, Q_S, Q_S^{in}, \rho_S, F_S\rangle$, where:

- $Q_S = \{\langle S, \preceq, \lambda, O, b\rangle \mid \text{if } b = 1 \text{ then } \lambda \text{ is } F\text{-free}\}$,
- $Q_S^{in} = \{\langle Q^{in}, \preceq, \lambda, \emptyset, 0\rangle \mid \text{for all } q, r \in Q^{in},\ q \preceq r \text{ iff } q \notin F \text{ or } r \in F\}$,
- $\rho_S(\mathbf{t}, \sigma) = \{\mathbf{t}' \mid \mathbf{t}' \text{ follows } \mathbf{t} \text{ under } \sigma\}$, and
- $F_S = \{\langle S, \preceq, \lambda, \emptyset, 1\rangle\}$.

▶ **Theorem 9.** *For every NBW $\mathcal{A}$, it holds that $L(\mathcal{A}_S) = \overline{L(\mathcal{A})}$.*

The proof of correctness for Theorem 9 (see Appendix A) is straightforward and based on correlating runs of $\mathcal{A}_S$ with G and its subgraphs. If $n = |Q|$, the number of preordered subsets is roughly $(0.53n)^n$ [23]. As there are 2^n labelings, and a further 2^n obligation sets, the state space of $\mathcal{A}_s$ is at most $(2n)^n$. The slice-based automaton obtained in [7] coincides with $\mathcal{A}_S$, modulo the details of labeling states and the cut-point construction (see Appendix C). The correctness proof in [7], however, is given by means of reduced split trees, whereas here we proceed directly on the run DAG.

4 Retrospection

Consider an NBW $\mathcal{A}$. So far, we presented two complementation constructions for $\mathcal{A}$, generating the NBWs $\mathcal{A}_R^m$ and $\mathcal{A}_S$. In this section we present a third construction, generating an NBW that combines the benefits of the two constructions above. Both constructions refer to the run DAG of $\mathcal{A}$. In the rank-based approach applied in $\mathcal{A}_R^m$, the ranks assigned to a node bound the visits in accepting states yet to come. Thus, the ranks refer to the future, making $\mathcal{A}_R^m$ inherently nondeterministic. On the other hand, the NBW $\mathcal{A}_S$ refers to both the past, using profiles to prune edges from G, as well as to the future, by keeping in G'' only nodes that are infinite in G'. Guessing which nodes are infinite and labeling them $\top$ inherently introduces nondeterminism into the automaton.

Our first goal in the combined construction is to reduce this latter nondeterminism. Recall that a labeling is F-free if all the states in F are labeled $\bot$. Observe that the fewer labels of $\bot$ (finite

nodes) we have, the more difficult it is for a labeling to be F-free and, consequently, the more difficult it is for a run of $\mathcal{A}_S$ to proceed to the F-free suffix in which $b = 1$. It is therefore safe for $\mathcal{A}_S$ to underestimate which nodes to label $\bot$, as long as the requirement to reach an F-free suffix is maintained. We use this observation in order to introduce a purely retrospective construction.

For a run DAG G, say that a level k is an *F-finite level* of G when all F-nodes *after* level k (i.e. on a level k' where $k' > k$) are finite in G'. Recall that, by Lemma 7, G is rejecting iff there is a level after which G'' has no F-nodes. Since finite nodes in G' are removed from G'', we have:

▶ **Corollary 10.** *A run* DAG G *is rejecting iff it has an F-finite level.*

4.1 Retrospective Labeling

The labeling function λ used in the construction of $\mathcal{A}_S$ labels nodes by $\{\top, \bot\}$, with $\bot$ standing for "finite" and $\top$ standing for "infinite". In this section we introduce a variant of λ that again maps nodes to $\{\top, \bot\}$ except that now $\top$ stands for "unrestricted", allowing us to underestimate which nodes to label $\bot$. To capture the relaxed requirements on labelings, say that a labeling λ is *legal* when every $\bot$-labeled node is finite in G'. This enables the automaton to track the labeling and its effect on F-nodes only after it guesses that an F-finite level k has been reached: all nodes *at or before* level k (i.e. on a level k' where $k' \leq k$) are unrestricted, whereas F-nodes after level k and their descendants are required to be finite. The only nondeterminism in the automaton lies in guessing when the F-finite level has been reached. This reduces the branching degree of the automaton to 2, and renders it deterministic in the limit.

The suggested new labeling is parametrized by the F-finite level k. The labeling λ^k is defined inductively over the levels of G. Let S_i be the set of nodes on level i of G. For $i \geq 0$, the function $\lambda^k : S_i \to \{\top, \bot\}$ is defined as follows:

- If $i \leq k$, then for every $u \in S_i$ we define $\lambda^k(u) = \top$.
- If $i > k$, then for every $u \in S_i$:
  - If u is an F-node, then $\lambda^k(u) = \bot$.
  - Otherwise, $\lambda^k(u) = \lambda^k(v)$, for a node v where $E'(v, u)$.

For λ^k to be well defined when $i > k$ and u is not an F-node, we need to show that $\lambda^k(u)$ does not depend on the choice of the node v where $E'(v, u)$. By Lemma 4, all parents of a node in G' belong to the same equivalence class. Therefore, it suffices to prove that all nodes in the same class share a label: for all nodes u and u', if $u' \approx_{|u|} u$ then $\lambda^k(u) = \lambda^k(u')$. The proof proceeds by an induction on $i = |u|$. Consider two nodes u and u' on level i where $u' \approx_i u$. As a base case, if $i \leq k$, then u and u' are labeled $\top$. For $i > k$, if u is an F-node, then u' is also an F-node and $\lambda^k(u) = \lambda^k(u') = \bot$. Finally, if u and u' are both non-F-nodes, recall that all parents of u are in the same equivalence class V. As $u \approx_i u'$, Lemma 4 implies that all parents of u' are also in V By the induction hypothesis, all nodes in V share a label, and thus $\lambda^k(u) = \lambda^k(u')$.

▶ **Lemma 11.** *For a run* DAG G *and $k \in \mathbb{N}$, the labeling λ^k is legal iff k is an F-finite level for G.*

Proof. If λ^k is legal, then every $\bot$-labeled node is finite in G'. Every F-node after level k is labeled $\bot$, and thus k is an F-finite level for G. If λ^k is not legal, then there is a $\bot$-labeled node u that is infinite in G'. Note that every ancestor of u is also infinite. Let u' be the earliest ancestor of u (possibly u itself) so that $\lambda^k(u') = \bot$. Observe that only nodes after level k can be $\bot$-labeled, and so u' is on a level $i > k$. It must be that u' is an F-node: otherwise it would inherit the label of its parent, and by assumption the parents of u' are $\top$-labeled. Thus, u' is an F-node on a level $i > k$ that is infinite in G', and k is not an F-finite level for G. □

▶ **Corollary 12.** *A run* DAG G *is rejecting iff, for some k, the labeling λ^k is legal.*

4.2 From Labelings to Rankings

In this section we derive an odd ranking for G from the function λ^k, thus unifying the retrospective analysis behind λ^k with the rank-based analysis of [10]. Consider again the DAG G' and the function λ^k. Recall that every equivalence class U has at most two child equivalence classes, one F-class and one non-F-class. Past the F-finite level k, only non-F-classes can be labeled $\top$. Hence, past level k, every $\top$-labeled equivalence class U can only have a one child that is $\top$-labeled. For every class U on level k, we consider this possibly infinite sequence of $\top$-labeled non-F-children. The odd ranking we are going to define, termed the *retrospective ranking*, gives these sequences of $\top$-labeled children odd ranks. The $\bot$-labeled classes, which lie between these sequences of $\top$-labeled classes, are assigned even ranks. The ranks increase in inverse lexicographic order, i.e. the maximal $\top$-labeled class in a level is given rank 1. As with λ^k, the retrospective ranking is parametrized by k. The primary insight that allows this ranking is that there is no need to distinguish between two adjacent $\bot$-labeled classes. Formally, we have the following.

▶ **Definition 13** (k-retrospective ranking)**.** Consider a run DAG G, $k \in \mathbb{N}$, and a labeling $\lambda^k : G \to \{\top, \bot\}$. Let $m = 2|Q \setminus F|$. For a node u on level i of G, let $\alpha(u)$ be the number of $\top$-labeled classes larger than u; $\alpha(u) = |\{[v] \mid \lambda^k(v) = \top \text{ and } u \prec_i v\}|$. The *$k$-retrospective ranking* of G' is the function $\mathbf{r}^k : V \to \{0..m\}$ defined for every node u on level i as follows.

$$\mathbf{r}^k(u) = \begin{cases} m & \text{if } i \leq k, \\ 2\alpha(u) & \text{if } i > k \text{ and } \lambda^k(u) = \bot, \\ 2\alpha(u) + 1 & \text{if } i > k \text{ and } \lambda^k(u) = \top. \end{cases}$$

Note that $\mathbf{r}^k$ is tight. As defined in Section 2, a ranking is tight if there exists an $i \in N$ such that, for every level $l \geq i$, all odd ranks below $max\_rank(\mathbf{r}, l)$ appear on level l. For $\mathbf{r}^k$ this level is $k + 1$, after which each $\top$-labeled class is given the odd rank greater by two than the rank of the next lexicographically larger $\top$-labeled class.

▶ **Lemma 14.** *For every $k \in \mathbb{N}$, the following hold:*
(1) If $u \prec_{|u|} u'$ then $\mathbf{r}^k(u) \geq \mathbf{r}^k(u')$.
(2) If $(u, v) \in E'$, then $\mathbf{r}^k(u) \geq \mathbf{r}^k(v)$.

Proof. As both claims are trivial when u is at or before level k, assume u is on level $i > k$. To prove the first claim, note that $\alpha(u) \geq \alpha(u')$: every class, $\top$-labeled or not, that is larger than u' must also be larger than u. If $\alpha(u) > \alpha(u')$, then (1) follows immediately. Otherwise $\alpha(u) = \alpha(u')$, which implies that $\lambda^k(u') = \bot$: otherwise $[u']$ would be a $\top$-labeled equivalence class larger than u, but not larger than itself. Thus $\mathbf{r}^k(u') = 2\alpha(u)$, and $\mathbf{r}^k(u) \in \{2\alpha(u),\ 2\alpha(u)+1\}$ is at least $\mathbf{r}^k(u')$.

As a step towards proving the second claim, we show that $\alpha(u) \geq \alpha(v)$. Consider every $\top$-labeled class $[v']$ where $v \prec_{i+1} v'$. The class $[v']$ must have a $\top$-labeled parent $[u']$. Since $v \prec_{i+1} v'$, the contrapositive of Lemma 5, part 1, entails that $u \preceq_i u'$. By the definition of λ^k, the class $[u']$ can only have one $\top$-labeled child class: $[v']$. We have thus established that for every $\top$-labeled class larger than v, there is a unique $\top$-labeled class larger than u, and can conclude that $\alpha(u) \geq \alpha(v)$. We now show by contradiction that $\mathbf{r}^k(u) \geq \mathbf{r}^k(v)$. For $\mathbf{r}^k(u) < \mathbf{r}^k(v)$, it must be that $\alpha(u) = \alpha(v)$, that $\mathbf{r}^k(u) = 2\alpha(u)$ and that $\mathbf{r}^k(v) = 2\alpha(u) + 1$. In this case, $\lambda^k(u) = \bot$ and $\lambda^k(v) = \top$. Since a $\bot$-labeled node cannot have a $\top$-labeled child in G', this is impossible. □

When k is an F-finite level of G, the k-retrospective ranking is an m-bounded odd ranking.

▶ **Lemma 15.** *For a run DAG G and $k \in \mathbb{N}$, the function $\mathbf{r}^k$ is a ranking bounded by m. Further, if the labeling λ^k is legal then $\mathbf{r}^k$ is an odd ranking.*

Proof. There are three requirements for $\mathbf{r}^k$ to be a ranking bounded by m:

(1) *Every F-node must have an even rank.* At or before level k, every node has rank m, which is even. After k only $\top$-labeled nodes are given odd ranks, while every F-node is labeled $\bot$.

(2) *For every $(u, v) \in E$, it must hold that $\mathbf{r}^k(u) \geq \mathbf{r}^k(v)$.* If u is at or before level k, then it has the maximal rank of m. If u is after level k, we consider two cases: edges in E', and edges in $E \setminus E'$. For edges in E', this follows from Lemma 14 (2). For edges $(u, v) \in E \setminus E'$, we know there exists a u' where $u \prec_{|u|} u'$ and $(u', v) \in E'$. By Lemma 14, $\mathbf{r}^k(u) \geq \mathbf{r}^k(u') \geq \mathbf{r}^k(v)$.

(3) *The rank is bounded by m.* No F-node can be $\top$-labeled. Thus the maximum number of $\top$-labeled classes on every level is $|Q \setminus F|$. The largest possible rank is given to a node smaller than all $\top$-labeled classes, which must be be a F-node and $\bot$-labeled. Since the number of $\top$-labeled classes is at most $|Q \setminus F|$, this node is given a rank of at most $m = 2|Q \setminus F|$.

It remains to show that if λ^k is legal, then $\mathbf{r}^k$ is an odd ranking. Consider an infinite path $u_0, u_1, \ldots$ in G. We demonstrate that for every $i > k$ such that $\mathbf{r}^k(u_i)$ is an even rank e, there exists $i' > i$ such that $\mathbf{r}^k(u_{i'}) \neq e$. Since a path cannot increase in rank, this implies $\mathbf{r}^k(u_{i'}) < e$. To do so, define the sequence $U_i, U_{i+1}, \ldots$, of sets of nodes inductively as follows. Let $U_i = \{v \mid \mathbf{r}^k(v) = e\}$. For every $j \geq i$, let $U_{j+1} = \{v \mid v' \in U_j,\ (v', v) \in E'\}$. As $\mathbf{r}^k(v)$ is even only when $\lambda^k(v) = \bot$, if λ^k is legal then every node given an even rank (such as e) must be finite in G'. Therefore every element of U_i is finite in G', and thus at some $i' > i$, the set $U_{i'}$ is empty. Since $U_{i'}$ is empty, to establish that $\mathbf{r}^k(u_{i'}) \neq e$, it is sufficient to prove that for every j, if $\mathbf{r}^k(u_j) = e$, then $u_j \in U_j$.

To show that $\mathbf{r}^k(u_j) = e$ entails $u_j \in U_j$, we prove a stronger claim: for every $j \geq i$ and v on level j, if $u_j \preceq_j v$ and $\mathbf{r}^k(v) = e$, then $v \in U_j$. We proceed by induction over j. For the base case of $j = i$, this follows from the definition of U_i. For the inductive step, take a node v on level $j+1$ where $\mathbf{r}^k(v) = e$ and $u_{j+1} \preceq_{j+1} v$. We consider two cases. If $\mathbf{r}^k(u_{j+1}) \neq e$ then the path from u_i to u_{j+1} entails that $\mathbf{r}^k(u_{j+1}) < e$, and this case of the subclaim follows from Lemma 14 (1). Otherwise, it holds that $\mathbf{r}^k(u_{j+1}) = e$, and thus $\mathbf{r}^k(u_j) = e$. Let u' and v' be nodes on level j so that $(u', u_{j+1}) \in E'$ and $(v', v) \in E'$. As $u_{j+1} \preceq_{j+1} v$, the contrapositive of Lemma 5, part 1, entails that $u' \preceq_j v'$. Further, since $(u', u_{j+1}) \in E'$ and $(u_j, u_{j+1}) \in E$, we know $u_j \preceq_j u'$. By transitivity we can thus conclude that $u_j \preceq_j v'$, which along with Lemma 14 (1) entails $\mathbf{r}^k(u') = e \geq \mathbf{r}^k(v')$. As $(v', v) \in E$, Lemma 14 (2) entails that $\mathbf{r}^k(v') \geq \mathbf{r}^k(v) = e$. Thus $\mathbf{r}^k(v') = e$, and by the inductive hypothesis $v' \in U_j$. As $E'(v', v)$, by definition $v \in U_{j+1}$, and our subclaim is proven. □

The ranking of Definition 13 is termed *retrospective* as it relies on the relative lexicographic order of equivalence classes; this order is determined purely by the history of nodes in the run DAG, not by looking forward to see which descendants are infinite or F-free in some subgraph of G.

Example 4. Figure 3 displays λ^0 and the 0-retrospective ranking of our running example. In the prospective ranking (see Figure 2), the nodes for state t on levels 1 and 2 are given rank 0, like other t-nodes. In the absence of a path forcing this rank, the retrospective ranking gives them rank 2.

We are now ready to define a new construction, generating an NBW $\mathcal{A}_L$, which combines the benefits of the previous two constructions. The automaton $\mathcal{A}_L$ guesses the F-finite level k, and uses level rankings to check if the k-retrospective ranking is an odd ranking. We partition the operation of $\mathcal{A}_L$ into two stages. Until the level k, the NBW $\mathcal{A}_L$ is in the first stage, where it deterministically tracks preordered subsets. After level k, the NBW $\mathcal{A}_L$ moves to the second stage, where it tracks ranks. This stage is also deterministic. Consequently, the only nondeterminism in $\mathcal{A}_L$ is indeed the guess of k. Before defining $\mathcal{A}_L$, we need some definitions and notations.

Recall that $\mathbf{Q}$ denotes the set of preordered subsets of Q, and $\mathcal{R}_T^m$ the set of tight level rankings bounded by m. We distinguish between three types of transitions of $\mathcal{A}_L$: transitions within the first stage, transitions from the first stage to the second, and transitions within the second stage. The first type of transition is similar to the one taken in $\mathcal{A}_S$, by means of the σ-successor relation between

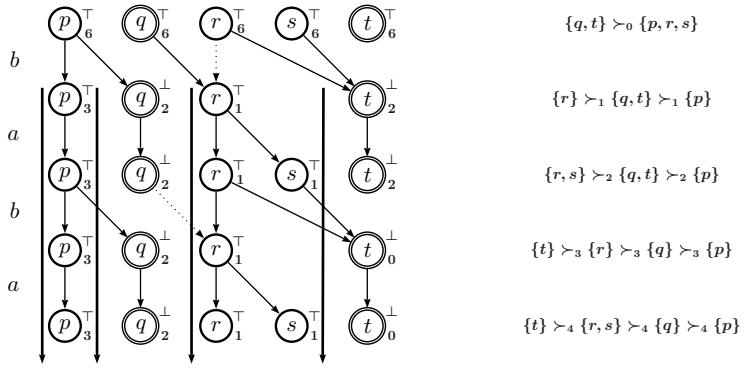

Figure 3 The run DAG G', where 0 is an F-finite level. The labels of λ^0 and ranks in $\mathbf{r}^0$ are displayed as superscripts and subscripts, respectively. The bold lines display the sequences of $\top$-labeled classes in G'.

preordered subsets. Below we explain in detail the other two types of transitions. Recall that in the retrospective ranking $\mathbf{r}^k$, each class in G' labeled $\top$ by λ^k is given a unique odd rank. Thus the rank of a node u depends on the number of $\top$-labeled classes larger than it, denoted $\alpha(u)$.

We begin with transitions where $\mathcal{A}_L$ moves between the stages: from a preordered subset $\langle S, \preceq \rangle$ to a level ranking. On level $k+1$, a node is labeled $\top$ iff it is an non-F-node. Thus for every $q \in S$, let $\beta(q) = |\{[v] \mid v \in S \setminus F, u \prec v\}|$ be the number of non-F-classes larger than q. We now define $\texttt{torank} : \mathbf{Q} \to \mathcal{R}^m_T$. Let $\texttt{torank}(\langle S, \preceq \rangle)$ be the tight level ranking f where for every q:

$$f(q) = \begin{cases} \bot & \text{if } q \notin S, \\ 2\beta(q) & \text{if } q \in S \cap F, \\ 2\beta(q)+1 & \text{if } q \in S \setminus F. \end{cases}$$

We now turn to transitions within the second stage, between level rankings. The rank of a node v is inherited from its predecessor u in G'. However, λ^k may label a finite class $\top$. If a $\top$-labeled class larger than u has no children, then $\alpha(u) \geq \alpha(v)$. In this case the rank of v decreases. Given a level ranking f, for every $q \in Q$ where $f(q) \neq \bot$, let $\gamma(q) = |\{f(q') \mid q' \in Q,\ f(q') \text{ is odd}, f(q') < f(q)\}|$ be the number of odd ranks in the range of f lower than $f(q)$. We define the function $\texttt{tighten} : \mathcal{R}^m \to \mathcal{R}^m_T$. Let $\texttt{tighten}(f)$ be the tight level ranking f' where for every q:

$$f'(q') = \begin{cases} \bot & \text{if } f(q) = \bot, \\ 2\gamma(q) & \text{if } f(q) \neq \bot \text{ and } q \in F, \\ 2\gamma(q)+1 & \text{if } f(q) \neq \bot \text{ and } q \notin F. \end{cases}$$

Note that if f is tight, then $f' = f$, and that while $\texttt{tighten}$ may merge two even ranks, it cannot merge two odd ranks.

For a level ranking f, letter $\sigma \in \Sigma$, and $q' \in Q$, let $\texttt{pred}(q', \sigma, f) = \{q \mid f(q) \neq \bot,\ q' \in \rho(q, \sigma)\}$ be the predecessors of q' given a non-$\bot$ rank by f. The predecessor in this set with the lowest rank corresponds to the predecessor in G with the maximal profile. With two exceptions, q' will inherit this lowest rank. First, $\texttt{tighten}$ might shift the rank down. Second, if q' is in F, it cannot be given an odd rank. For $n \in N$, let $\lfloor n \rfloor_{even}$ be: n when n is even; and $n-1$ when n is odd. Define the *σ-successor of f* to be $\texttt{tighten}(f')$ where for every $q' \in Q$:

$$f'(q') = \begin{cases} \bot & \text{if } \texttt{pred}(q', \sigma, f) = \emptyset, \\ \lfloor \texttt{min}(\{f(q) \mid q \in \texttt{pred}(q', \sigma, f)\}) \rfloor_{even} & \text{if } \texttt{pred}(q', \sigma, f) \neq \emptyset \text{ and } q' \in F, \\ \texttt{min}(\{f(q) \mid q \in \texttt{pred}(q', \sigma, f)\}) & \text{if } \texttt{pred}(q', \sigma, f) \neq \emptyset \text{ and } q' \notin F. \end{cases}$$

▶ **Definition 16.** For an NBW $\mathcal{A} = \langle \Sigma, Q, Q^{in}, \rho, F \rangle$, let $\mathcal{A}_L$ be the NBW $\langle \Sigma, \mathbf{Q} \cup (\mathcal{R}_T^m \times 2^Q), Q_L^{in}, \rho_L, \mathcal{R}_T^m \times \{\emptyset\} \rangle$, where

- $Q_L^{in} = \{\langle Q^{in}, \preceq^{in} \rangle\}$ where $\preceq^{in}$ is such that for all $q, r \in Q^{in}$, $q \preceq r$ iff $q \notin F$ or $r \in F$.
- $\rho_L(\mathcal{S}, \sigma) = \{\mathcal{S}'\} \cup \{\langle \texttt{torank}(\mathcal{S}'), \emptyset \rangle\}$, where $\mathcal{S}'$ is the σ-successor of $\mathcal{S}$.
- $\rho_L(\langle f, O \rangle, \sigma) = \{\langle f', O' \rangle\}$ where f' is the σ-successor of f

$$\text{and } O' = \begin{cases} \rho(O, \sigma) \setminus \mathit{odd}(f') & \text{if } O \neq \emptyset, \\ \mathit{even}(f') & \text{if } O = \emptyset. \end{cases}$$

Theorem 17, proven in Appendix B, follows from Lemmas 1 and 15 and Corollary 12.

▶ **Theorem 17.** *For every NBW $\mathcal{A}$, it holds that $L(\mathcal{A}_L) = \overline{L(\mathcal{A})}$.*

Analysis: Like the tight-ranking construction in Section 2, the automaton $\mathcal{A}_L$ operates in two stages. In both, the second stage is the set of tight level rankings and obligation sets. The tight-ranking construction uses sets of states in the first stage, and is bounded by the size of the second stage: $(0.96n)^n$ [5]. The automaton $\mathcal{A}_L$ replaces the first stage with preordered subsets. As the number of preordered subsets is $O((\frac{n}{e \ln 2})^n) \approx (0.53n)^n$ [23], the size of $\mathcal{A}_L$ remains bounded by $(0.96n)^n$. This can be improved to $(0.76n)^n$: see below. Further, $\mathcal{A}_L$ has a very restricted transition relation: states in the first stage only guess whether to remain in the first stage or move to the second, and have nondeterminism of degree 2. States in the second stage are deterministic. Thus the transition relation is linear in the number of states and size of the alphabet, and $\mathcal{A}_L$ is deterministic in the limit.

5 Discussion

We have unified the slice-based and rank-based approaches by phrasing the former in the language of run DAGs. This enables us to define and exploit a retrospective ranking, providing a deterministic-in-the-limit complementation construction that does not employ determinization. Experiments show that the more deterministic automata are, the better they perform in practice [18]. By avoiding determinization, we reduce the cost of such a construction from $(n^2/e)^n$ to $(0.76n)^n$ [14].

In addition, our transition generates a transition relation that is linear in the number of states and size of the alphabet. Schewe demonstrated how to achieve a similar linear bound on the transition relation, but the resulting relation is larger and is not deterministic in the limit [17].

The use of level rankings affords several improvements from existing research on the rank-based approach. First, the cut-point construction of Miyano and Hayashi [12] can be improved. Schewe's construction only checks one even rank at a time, reducing the size of the state space to $(0.76n)^n$, only an n^2 factor from the lower bound [17]. As Schewe's approach does not alter the progression of the level rankings, it could be applied directly to the second stage of Definition 16: a detailed presentation is in Appendix D. The resulting construction inherits the asymptotic state-space complexity of [17]. Second, symbolically encoding a preorder is complicated. In contrast, ranks are easily encoded, and the transition between ranks is nearly trivial to implement in SMV [20]. By changing the states in first stage of $\mathcal{A}_L$ from preordered subsets to simple subsets, and guessing the appropriate transition to the second stage, we obtain an symbolic representation while maintaining determinism in the limit: a detailed presentation can be found in Appendix D. This approach does sacrifice the linear-sized transition relation, but this is less important in a symbolic encoding. Finally, the subsumption relations of Doyen and Raskin [4] could be applied to the second stage of the automaton, while it is unclear if it could be applied at all to the slice-based construction.

From a broader perspective, we find it very interesting that the prospective and retrospective approaches are so strongly related. Odd rankings seem to be inherently "prospective," depending on the descendants of nodes in the run DAG. By investigating the slice-based approach, we are able to pinpoint the dependency on the future to a single component: the F-free level. This suggests it may

be possible to use odd rankings for determinization, automata with other accepting conditions, and automata on infinite trees.

Acknowledgment The authors are grateful to Yoad Lustig for his extensive help in analyzing the original slice-based construction.

References

1. C.S. Althoff, W. Thomas, and N. Wallmaier. Observations on determinization of Büchi automata. In *TCS*, 2006.
2. J.R. Büchi. On a decision method in restricted second order arithmetic. In *ICLMPS*, 1962.
3. C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *J. ACM*, 1995.
4. L. Doyen and J.-F. Raskin. Antichains for the automata-based approach to model-checking. In *LMCS*, 2009.
5. E. Friedgut, O. Kupferman, and M.Y. Vardi. Büchi complementation made tighter. In *FCS*, 2006.
6. S. Gurumurthy, O. Kupferman, F. Somenzi, and M.Y. Vardi. On complementing nondeterministic Büchi automata. In *CHARME*, 2003.
7. D. Kähler and T. Wilke. Complementation, disambiguation, and determinization of Büchi automata unified. In *ICALP*, 2008.
8. H. Karmarkar and S. Chakraborty. On minimal odd rankings for Büchi complementation. In *ATVA*, 2009.
9. N. Klarlund. *Progress Measures and finite arguments for infinite computations*. PhD thesis, Cornell University, 1990.
10. O. Kupferman and M.Y. Vardi. Weak alternating automata are not that weak. In *TOCL*, 2001.
11. M. Michel. Complementation is more difficult with automata on infinite words. *CNET*, Paris, 1988.
12. S. Miyano and T. Hayashi. Alternating finite automata on ω-words. In *TCS*, 1984.
13. D.E. Muller and P.E. Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of theorems of Rabin, McNaughton and Safra. In *TCS*, 1995.
14. N. Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. In *LICS*, 2006.
15. M.O. Rabin and D. Scott. Finite automata and their decision problems. In *IBM JRD*, 1959.
16. S. Safra. On the complexity of ω-automata. In *FOCS*, 1988.
17. S. Schewe. Büchi complementation made tight. In *STACS*, 2009.
18. R. Sebastiani and S. Tonetta. “More deterministic” vs. “smaller" Büchi automata for efficient LTL model checking. In *CHARME*, 2003.
19. A.P. Sistla, M.Y. Vardi, and P. Wolper. The complementation problem for Büchi automata with applications to temporal logic. In *TCS*, 1987.
20. D. Tabakov and M.Y. Vardi. Model checking Büchi specifications. In *LATA*, 2007.
21. S. Tasiran, R. Hojati, and R.K. Brayton. Language containment using non-deterministic omega-automata. In *CHARME*, 1995.
22. M. Y. Vardi. The Büchi complementation saga. In *STACS*, 2007.
23. M.Y. Vardi. Expected properties of set partitions. Research report, The Weizmann Institute of Science, 1980.
24. M.Y. Vardi. Automata-theoretic model checking revisited. In *VMCAI*, 2007.
25. M.Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *LICS*, 1986.
26. Q. Yan. Lower bounds for complementation of ω-automata via the full automata technique. In *ICALP*, 2006.

A Proof of Theorem 9

We divide runs of $\mathcal{A}_S$ into two parts. The *prefix* of a run is the initial sequence of states in which b_i is 0, and the *suffix* is the remaining sequence states, in which b_i is 1. A run without a suffix, where b stays 0 for the entire run, has no accepting states.

▶ **Theorem 16.** *For every NBW $\mathcal{A}$, it holds that $L(\mathcal{A}_S) = \overline{L(\mathcal{A})}$.*

Proof. Consider a word $w \in \Sigma^\omega$ and the run DAG G. We first make the following claims about every infinite run $\mathbf{t}_0, \mathbf{t}_1, \ldots$, where $\mathbf{t}_i = \langle S_i, \preceq_i, \lambda_i, O_i, b_i \rangle$. For convenience, define $\mathcal{S}_i = \langle S_i, \preceq_i \rangle$.

(1) *The states in S_i are precisely $\{q \mid \langle q, i\rangle \in G\}$.*
We exploit this claim to conflate a state q in the ith state with the node $\langle q, i\rangle$, and speak of states in S_i being in, being finite in, and being infinite in a graph G.

(2) *The preorder $\preceq_i$ is the projection of $\preceq$ onto states occurring at level i.*
This follows from Lemma 5 and the definition of one state following another.

(3) *For every $p \in S_i$, $q \in S_{i+1}$, it holds that $q \in \rho_{\mathcal{S}_i}(p, \sigma_i)$ iff $E'(\langle p, i\rangle, \langle q, i+1\rangle)$.*
This follows from the definitions of E' and $\rho_{\mathcal{S}}$.

(4) *O_i is empty for infinitely many i's iff every state labeled $\bot$ is not in G''.*
This follows from the cut-point construction of Miyano and Hayashi. [12].

(5) *Every state labeled $\top$ is in G''.*
This follows from the definition of transitions between states: every $\top$-labeled state must have a $\top$-labeled child, and thus is infinite in G' and in G''.

We can now prove the theorem. In one direction, assume there is an accepting run $\mathbf{t}_0, \mathbf{t}_1, \ldots$. As this run is accepting, infinitely often $O_i = \emptyset$. By (4) and (5), this implies the states in S_i are correctly labeled $\top$ when and only when they occur in G''. Further, for this run to be accepting we must be able to divide into a prefix, where $b_i = 0$ and suffix, where b_i must increase to 1. In the suffix no state in F can be labeled $\top$, and no F-nodes occur in G'' past this point. As only finitely many F-nodes can occur before this point, by Lemma 6 G does not have an accepting path and $w \notin L(\mathcal{A})$.

In the other direction, assume $w \notin L(\mathcal{A})$. This implies there are finitely many F-nodes in G'', and thus a level j where the last F-node occurs. We construct an accepting run $\mathbf{t}_0, \mathbf{t}_1, \ldots$, demonstrating along the way that we satisfy the requirements for $\mathbf{t}_{i+1}$ to be in $\rho_S(\mathbf{t}_i, \sigma_i)$. Given w, the sequence $\langle S_0, \preceq_0\rangle, \langle S_1, \preceq_1\rangle, \ldots$ of preordered subsets is uniquely defined by ρ_S. There are many possible labelings λ. For every i, select λ_i so that a state $q \in S_i$ is labeled with $\top$ when $\langle q, i\rangle \in G''$, and $\bot$ when it is not. Since every node in G'' has a child, by (3), for every $p \in S_i$ where $\lambda_i(p) = \top$, there exist a $q \in \rho_{\mathcal{S}_i}(p, \sigma_i)$ so that $\lambda_{i+1}(q) = \top$. Further, every node labeled $\bot$ has only finitely many descendants, and so for every $p \in S_i$ where $\lambda_i(p) = \bot$ and $q \in \rho_{\mathcal{S}_i}(p, \sigma_i)$, it holds that $\lambda_{i+1}(q) = \bot$. Therefore the transition from λ_i to λ_{i+1} satisfies the requirements of ρ_S. The set $O_0 = \emptyset$, and given the sets S_i and labelings λ_i, the sets O_{i+1}, $i \geq 0$ are again uniquely defined by ρ_S. Finally, we choose $b_i = 0$ when $i < j$, and $b_i = 1$ for $i \geq j$. Since there are no F-nodes in G'' past j, no F-node will be labeled $\top$ and all states past j will be F-free. We have satisfied the last requirement for the transitions from every $\mathbf{t}_i$ to $\mathbf{t}_{i+1}$ to be valid, rendering this sequence a run. By (4), infinitely often $O_i = \emptyset$, including infinitely often after j, thus there infinitely many states $\mathbf{t}_i$ where $b_i = 1$ and $O_i = \emptyset$, and this run is accepting. □

B Proof of Theorem 17

▶ **Theorem 17.** *For every NBW $\mathcal{A}$, it holds that $L(\mathcal{A}_L) = \overline{L(\mathcal{A})}$.*

Proof. Consider a word $w \in \Sigma^\omega$ and the run DAG G. We first make the following claims about every infinite run $\langle S_0, \preceq_0\rangle, \ldots, \langle S_k, \preceq_k\rangle, \langle f_{k+1}, O_{k+1}\rangle, \langle f_{k+2}, O_{k+2}\rangle, \ldots$. For $i > k$, define $S_i = \{q \mid f_i(q) \neq \bot\}$.

(1) *The states in S_i are precisely $\{q \mid \langle q, i\rangle \in G\}$.*
This follows by the definitions of σ-successors of preordered subsets and σ-successors of level rankings.
(2) *The preorder $\preceq_i$ is the projection of $\preceq$ onto states occurring at level i.*
This follows from Lemma 5 and the definition of σ-successors.
(3) *For every $i \leq k$, state $q \in S_i$, and $s \in S_{i+1}$, it holds that $s \in \rho_{\langle S_i, \preceq_i\rangle}(q, \sigma_i)$ iff $E'(\langle q, i\rangle, \langle s, i+1\rangle)$.*
This follows from the definitions of E' and $\rho_{\langle S_i, \preceq_i\rangle}$.
(4) *For every $i > k$ and $q, s \in S_i$, if $f_i(q) > f_i(s)$, then $\langle q, i\rangle \prec_i \langle s, i\rangle$.*
(5) *For every $i > k$ and $q, s \in S_i$, if $f_i(s)$ is odd and $\langle q, i\rangle \prec_i \langle s, i\rangle$, then $f_i(q) > f_i(s)$.*
This and (4) are proven below.
(6) *For every $i \geq k$ and $q \in S_i$, it holds that $f_i(q)$ is even iff $\lambda^k(\langle q, i\rangle) = \bot$.*
This follows from the definition of λ^k, which assigns $\bot$ to F-nodes and their descendants in G', and f_i, which assigns even ranks to states in F. By (4), the parent of a node in G' will be the parent with the lowest rank. Thus the descendants of F-nodes in G' will inherit the even rank of their parent.

We simultaneously prove (4) and (5) by induction. As a base case, both hold from the definition of `torank`. As the inductive step, assume both hold for level i. To prove step (4), take two states $q, s \in S_{i+1}$ where $f_{i+1}(q) > f_{i+1}(s)$. Each state has a parent in G', i.e. a q' and s' so that $E'(q', q)$ and $E'(s', s)$. By the inductive hypothesis, this implies $f_i(q') = \texttt{min}(\{f_i(q') \mid q \in \rho(q', \sigma_i)\})$ and $f_i(s') = \texttt{min}(\{f_i(s') \mid s \in \rho(s', \sigma_i)\})$. We analyze two cases. When $f_i(q') > f_i(s')$, by the inductive hypothesis we have $\langle q', i\rangle \prec_i \langle s', i\rangle$. Since $E'(q', q)$ and $E'(s', s)$, by Lemma 4 this implies $\langle q, i+1\rangle \prec_{i+1} \langle s, i+1\rangle$. Alternately, when $f_i(q') = f_i(s')$, then for $f_{i+1}(q) > f_{i+1}(s)$ to hold, it must be that $f_i(q')$ is odd, $s \in F$, and $q \notin F$. Since $f_i(q') = f_i(s')$ is odd, by the inductive hypothesis we have that $\langle q', i\rangle \equiv \langle s', i\rangle$. By Lemma 4 we then have $h_{\langle q, i+1\rangle} = h_{\langle q', i\rangle}0 < h_{\langle s, i+1\rangle} = h_{\langle q', i\rangle}1$.

To prove step (5), consider when $f_{i+1}(s)$ is odd and $\langle q, i+1\rangle \prec \langle s, i+1\rangle$. This implies that $h_{\langle s, i+1\rangle} = h_{\langle s', i\rangle}0$. Thus in order for $\langle q, i+1\rangle \prec_{i+1} \langle s, i+1\rangle$ to hold, $\langle q', i\rangle \prec_i \langle s', i\rangle$ must hold. By the inductive hypothesis, this implies $f_i(q') > f_i(s')$. Before the `tighten` function reduces ranks, since $f_{i+1}(q) = \lfloor f_i(q')\rfloor_{even}$, and $f_{i+1}(s)$ is odd, it must be that $f_{i+1}(q) > f_{i+1}(s)$. The `tighten` function can shift $f_{i+1}(q)$ down more than $f_{i+1}(s)$ only when an odd rank between $f_{i+1}(s)$ and $f_{i+1}(q)$ becomes empty. Since this odd rank must be two greater than $f_{i+1}(s)$, reducing $f_{i+1}(q)$ by 2 cannot change that $f_{i+1}(q) > f_{i+1}(s)$.

We now proceed with the proof of Theorem 17. In one direction, assume the run $\langle S_0, \preceq_0\rangle, \ldots, \langle S_k, \preceq_k\rangle, \langle f_{k+1}, O_{k+1}\rangle, \langle f_{k+2}, O_{k+2}\rangle, \ldots$ is accepting. We construct a ranking $\mathbf{r}$ of G_w as follows. For all nodes u on level $i \leq k$, $\mathbf{r}(u) = m$. For all nodes $\langle q, i\rangle$ where $i > k$, $\mathbf{r}(\langle q, i\rangle) = f_i(q)$. We note that each state is given at most the minimum rank of all its parents, and that no state in F is given an odd rank, thus $\mathbf{r}$ is in fact a ranking. That $\mathbf{r}$ is an odd ranking follows from the cut-point construction.

In the other direction, assume G is a rejecting run DAG. By Lemma 15 there exists a k so that $\mathbf{r}^k$ is an odd ranking. We construct a run $\mathcal{S}_0, \ldots, \mathcal{S}_k, \langle f_{k+1}, O_{k+1}\rangle, \langle f_{k+2}, O_{k+2}\rangle, \ldots$, which is uniquely defined by the transition relation of Definition 16. Further, the transition relation of Definition 16 is total, so this run is infinite. To demonstrate that this run is accepting, we will prove below that for every $i > k$ and $q \in S_i$, it holds that $f_i(q) = \mathbf{r}^k(\langle q, i\rangle)$. Since $\mathbf{r}^k$ is an odd ranking and the cut-point construction is identical to that of Definition 2, this is sufficient to show the run is accepting.

Recall that if $\lambda(\langle q, i\rangle) = \bot$, then $\mathbf{r}^k(\langle q, i\rangle) = 2\alpha(\langle q, i\rangle)$, and otherwise $\mathbf{r}^k(\langle q, i\rangle) = 2\alpha(\langle s, i\rangle)+1$. We can thus use (6) to simplify our claim. It suffices to show that for every $i > k$ and $q \in S_i$, we have $\alpha(\langle q, i\rangle) = \lfloor f_i(q)/2\rfloor$. We proceed by induction over $i > k$. As the base case, consider a node $\langle q, k\rangle$. Recall that $\alpha(\langle q, k\rangle) = |\{[v] \mid \lambda^k(v) = \top, \langle q, k\rangle \prec_k v\}|$. By the definition of λ^k,

a node on level k is labeled $\bot$ only when it is an F-node. All other nodes inherit the label of their parents, and every node on level k is $\top$-labeled. From (2), we then have that $\alpha(\langle q, k+1\rangle) = |\{[v] \mid v \in S \setminus F, u \prec v\}|$, which is the definition of $\beta(q) = \lfloor f_i(q)/2 \rfloor$.

Inductively, assume the claim holds for every $q \in S_i$. We show for every $s \in S_{i+1}$, it holds that $\alpha(\langle s, i+1\rangle) = \lfloor f_{i+1}(s)/2 \rfloor$. Let q be s's parent in G', i.e. $E'(q, s)$. Take the set $P = \{[v] \mid \lambda^k(v) = \top, \langle q, i\rangle \prec_i v\}$. of $\top$-labeled equivalence classes greater than Q, By the inductive hypothesis, $\lfloor f_i(q)/2 \rfloor = \alpha(\langle q, i\rangle) = |P|$. By the definition of $\mathbf{r}^k$, each $[v] \in P$ has a unique odd rank assigned to each of its elements. By (5), for each $[v]$ this odd rank is smaller than $f_i(q)$. Consider P's subset $P_s = \{[v] \mid [v] \in P,\ [v]$ has $\top$-labeled child class on level $i+1\}$. Define $P_e = P \setminus P_s$ to be the complementary set: pipes that die on level i. By (5), before the `tighten` operation is applied, every element of P_e has a corresponding odd rank that is unoccupied on level $i+1$. Since q is clearly not in an element of P_e, this odd rank must be less than $\lfloor f_i(q) \rfloor_{even}$. Thus the final rank assigned to s, after `tighten`, is either $f_i(q) - 2|P_e|$ or $\lfloor f_i(q) - 2|P_e| \rfloor_{even}$. In both cases $\lfloor f_{i+1}(s)/2 \rfloor = \lfloor f_i(q)/2 \rfloor - |P_e|$. By the inductive hypothesis this is equivalent to $\alpha(\langle q, i\rangle) - |P_e| = |P| - |P_e|$. By the definition of P_s and P_e, $|P| - |P_e| = |P_s|$. By Lemma 4, every $\top$-labeled child of a class in P_s is lexicographically larger than $\langle s, i+1\rangle$. As every $\top$-labeled child must have a unique parent in P_s, we conclude that $|P_s| = \alpha(\langle s, i+1\rangle)$. □

C Slices

The paper of Kähler et al. introduces the notion of the split tree, reduced split tree, and skeleton of an automaton $\mathcal{A}$ and word w. The *split tree*, written T^{sp}, is the 2^Q-labeled tree defined inductively as follows. As a base case, the root is labeled with the set of initial states. Inductively, given a node v on level i labeled with a set of states P, the node v's left child is labeled with the states in $\rho(P, \sigma_i) \cap F$, and v's right child is labeled with $\rho(P, \sigma_i) \setminus F$. If either of these labels are empty, the corresponding child is omitted from T^{sp}. As argued in [7], branches in T^{sp} correspond to runs of $\mathcal{A}$ on w. We gloss over this discussion and simply state that $w \in L(\mathcal{A})$ iff $T^{sp}(\mathcal{A}, w)$ has a branch that goes left infinitely often.

The *reduced split tree*, written T^{rs}, keeps only the leftmost instance of each state at each level of the tree. This bounds the width of T^{rs} to n. The reduced split tree is a representation of G'. Each path p to node $\langle q, i\rangle \in G'$ corresponds to a node v on level i of T^{sp} that contains q in its label. The profile of this path, once the first bit is removed, is the path to v in T^{sp}: when we follow a left branch to level i of T^{rs}, the path travels through an F-node, and the $l(p_i)$ is 1. When we follow a right branch, the path travels through a non-F-node, $l(p_i)$ is 0. The profile of $\langle q, i\rangle$ is the profile of the lexicographically maximal path to $\langle q, i\rangle$ in G'. This profile is then the left-most path in T^{sp} to an instance of q. This is the left-most instance of q, and the only instance that remains in T^{rs}.

Finally, the *skeleton* T^{sp} is obtained by removing from the reduced split tree all nodes that are finite. Since the reduced split tree is a representation of G', the skeleton is a representation of G''. The slice automaton of Kähler and Wilke proceeds by tracking the levels of T^{rs} and guessing which nodes occur in T^{sp}. Each level i of T^{rs} is encoded as a *slice*, a sequence $\langle P_0, \ldots, P_m\rangle$ of pairwise disjoint subsets of Q. This slice is precisely the sequence of equivalence classes in level i of G', indexed by their relative lexicographic ordering (see Figure 2). The automaton of Kähler and Wilke differs from Definition 8 only in the details of labeling states and the cut-point construction.

D Variations on $\mathcal{A}_L$

Schewe's construction alters the cut-point of the rank-based construction to check only one even rank at a time. Doing so drastically reduces the size of the cut-point: intuitively, we can avoid carrying the obligation set explicitly. Instead we could carry the current rank i we are checking, and add to

the domain of our ranking function a single extra symbol c that indicates the state is currently being checked, and thus is of rank i. For an analysis of the resulting state space, please see [17]. For clarity , we do not remove the obligation set from the construction. Instead, states in this variant of the automaton carry with them the index i, and in a state $\langle f, O, i\rangle$, it holds that $O \subseteq \{q \mid f(q) = i\}$. For a level ranking f, let $\mathtt{mr}(f)$ be the largest rank in f. Note that $\mathtt{mr}(f)$, for a tight ranking, is always odd.

▶ **Definition 18.** For an NBW $\mathcal{A} = \langle \Sigma, Q, Q^{in}, \rho, F\rangle$, let $\mathcal{A}_{Schewe}$ be the NBW $\langle \Sigma, \mathbf{Q} \cup (\mathcal{R}^m \times 2^Q \times N), Q^{in}_L, \rho_{Sch}, F_{Sch}\rangle$, where

- $\rho_{Sch}(\mathcal{S}, \sigma) = \{\langle \mathtt{torank}(\mathcal{S}'), \emptyset, 0\rangle\} \cup \{\mathcal{S}'\}$, where $\mathcal{S}'$ is the σ-successor of $\mathcal{S}$.
- $\rho_{Sch}(\langle f, O, i\rangle, \sigma) = \{\langle f', O', i'\rangle\}$ where

$$f' \text{ is the } \sigma\text{-successor of } f$$

$$i' = \begin{cases} i & \text{if } O \neq \emptyset, \\ (i+2) \bmod (\mathtt{mr}(f')+1) & \text{if } O = \emptyset, \end{cases}$$

$$\text{and } O' = \begin{cases} \rho(O, \sigma) \setminus \mathit{odd}(f') & \text{if } O \neq \emptyset, \\ \{q \mid f'(i') = q\} & \text{if } O = \emptyset. \end{cases}$$

- $F_{Sch} = \mathcal{R}^m \times \{\emptyset\} \times \{0\}$

To symbolically encode a deterministic-in-the-limit automaton, we avoid storing the preorders. To encode the preorder in a BDD as a relation would require a quadratic number of variables, increasing the size unacceptably. Alternately, we could associate each state with its index in the preorder. Unfortunately, calculating the index of each state in the succeeding preorder would require a global compacting step, to remove indices that had become empty. To handle this difficulty, we simply store only the subset in the first stage, and transition to an arbitrary level ranking when we move to the second stage. This maintains determinism in the limit, and cannot result in false accepting run: we can always construct an odd ranking from the sequence of level rankings. The construction and a small example encoding are provided below.

▶ **Definition 19.** For an NBW $\mathcal{A} = \langle \Sigma, Q, Q^{in}, \rho, F\rangle$, let $\mathcal{A}_{Symb}$ be the NBW $\langle \Sigma, 2^Q \cup (\mathcal{R}^m \times 2^Q), Q^{in}, \rho_{Symb}, \mathcal{R}^m \times \{\emptyset\}\rangle$, where

- $\rho_{Symb}(S, \sigma) = \{\langle f, \emptyset\rangle \mid f \in \mathcal{R}^m \text{ and for all } q \in Q,\ q \in S \text{ iff } f(q) \neq \bot\} \cup \{\rho(S, \sigma)\}$
- $\rho_{Symb}(\langle f, O\rangle, \sigma) = \rho_L(\langle f, O\rangle, \sigma)$

As an example, this is the SMV encoding of the automaton of Figure 1, with state s removed.

```
typedef STATE 0..3;
/* Size for complemented automaton: 4, maximum allowed rank = 4*/
module main() {
 /* The transition letter */
 letter: {a,b};

 /* If we have transitioned out of subset construction.  */
 phase : 0..1;

 /* The ranking function. The value 5 represents _|_ (bottom) */
 rank: array STATE of 0..5;

 /* The obligation set vector */
 subset: array STATE of boolean;
```

```
/* The initial ranking assigns 4 to initial states and 5 to others.  */
init(rank) := [4,4,4,4];

/* The obligation set is initially rejecting */
init(subset) := [1,1,1,1];

init(phase) := 0;
next(phase) := {i : i=0..1, i >= phase};

/* Define the rank of states in the next time step */
/* state 0 has transition from 0 on a, and from 0 on b */
next(rank[0]) := case {
    rank[0] = 5 : 5;
    phase = 0 & next(phase) = 0 : 4;
    phase = 0 & next(phase) = 1 : {i : i=0..4, i <= rank[0]};
    phase = 1 : rank[0];
};

/* 1 has transition from 1 on a and from 0 on b. 1 is accepting */
next(rank[1]) := case {
   letter = a & rank[1] = 5  : 5;
   letter = a & phase = 0 & next(phase)=0 : 4;
   letter = a & phase = 0 & next(phase)=1 :
           {i : i=0..4, i <= rank[1] & i in {0,2,4}};
   letter = a & phase = 1 :
           {i : i=0..4, i in {rank[1], rank[1]-1} & i in {0,2,4}};
   letter = b & rank[0] = 5 : 5;
   letter = b & phase = 0 &  next(phase)=0 : 4;
   letter = b & phase = 0 &  next(phase)=1 :
           {i : i=0..4, i <= rank[0] & i in {0,2,4}};
   letter = b & phase = 1 :
           {i : i=0..4, i in {rank[0], rank[0]-1} & i in {0,2,4}};
};

/* 2 has incoming transition from 2 on a and from 1 and 2 on b */
next(rank[2]) := case {
   letter = a & rank[2] = 5 : 5;
   letter = a & phase = 0 &  next(phase)=0 : 4;
   letter = a & phase = 0 &  next(phase)=1 : {i : i=0..4, i <= rank[0]};
   letter = a & phase = 1 : rank[2];
   letter = b & rank[1] = 5  & rank[2] = 5 : 5;
   letter = b & phase = 0 & next(phase)=0 : 4;
   letter = b & phase = 0 & next(phase)=1 :
           {i : i=0..4, i <= rank[1] & i <= rank[2]};
   letter = b & phase = 1 :
           {i : i=0..4, i in {rank[1],rank[2]} & i<=rank[1] & i<=rank[2]};
};
```

```
 /* 3 has incoming transition from 3 on a and 2 on b. 3 is accepting */
 next(rank[3]) := case {
     letter = a & rank[3] = 5 : 5;
     letter = a & phase = 0 & next(phase) = 0 : 4;
     letter = a & phase = 0 & next(phase) = 1 :
            {i : i=0..4, i <= rank[3] & i in {0,2,4}};
     letter = a & phase = 1 : rank[3];
     letter = b & rank[2] = 5 : 5;
     letter = b & phase = 0 & next(phase) = 0 : 4;
     letter = b & phase = 0 & next(phase) = 1 :
            {i : i=0..4, i <= rank[2] & i in {0,2,4}};
     letter = b & phase = 1 :
            {i : i=0..4, i in {rank[2], rank[2]-1} & i in {0,2,4}};
 };

 /* Defining the transitions of the P-set */
 if (next(phase) = 0)  {
     forall (i in STATE)
        next(subset[i]) := 1;
 } else {
   if (subset = [0,0,0,0]) { /* The P-set is empty */
     forall (i in STATE)
       next(subset[i]) := next(rank[i]) in {0,2,4};
   } else { /* The P-set is non-empty */
     if (letter = a) {
       next(subset[0]) := (subset[0]) & next(rank[0]) in {0,2,4};
       next(subset[1]) := (subset[1]) & next(rank[1]) in {0,2,4};
       next(subset[2]) := (subset[2]) & next(rank[2]) in {0,2,4};
       next(subset[3]) := (subset[3]) & next(rank[3]) in {0,2,4};
     } else { /* letter = b */
       next(subset[0]) := (subset[0]) & next(rank[0]) in {0,2,4};
       next(subset[1]) := (subset[0]) & next(rank[1]) in {0,2,4};
       next(subset[2]) := (subset[1]|subset[2]) & next(rank[2]) in {0,2,4};
       next(subset[3]) := (subset[2]) & next(rank[3]) in {0,2,4};
 }}} /* Done with P-set */

 SPEC 0;
 FAIRNESS subset = [0,0,0,0];
}
```