Proof Systems for the Modal μ-Calculus Obtained by Determinizing Automata

Maurice Dekker, Johannes Kloibhofer, Johannes Marti(✉), and Yde Venema

ILLC, University of Amsterdam, Amsterdam, Netherlands
pmauricedekker@gmail.com, {j.kloibhofer,y.venema}@uva.nl,
johannes.marti@gmail.com

Abstract. Automata operating on infinite objects feature prominently in the theory of the modal μ-calculus. One such application concerns the tableau games introduced by Niwiński & Walukiewicz, of which the winning condition for infinite plays can be naturally checked by a nondeterministic parity stream automaton. Inspired by work of Jungteerapanich and Stirling we show how determinization constructions of this automaton may be used to directly obtain proof systems for the μ-calculus. More concretely, we introduce a binary tree construction for determinizing nondeterministic parity stream automata. Using this construction we define the annotated cyclic proof system BT, where formulas are annotated by tuples of binary strings. Soundness and Completeness of this system follow almost immediately from the correctness of the determinization method.

Keywords: modal mu-calculus · derivation system · determinisation of Büchi and parity automata · non-wellfounded and cyclic proofs

1 Introduction

The Modal μ-calculus. The modal μ-calculus is a natural extension of basic modal logic with explicit least and greatest fixpoint operators. Allowing the formulation of various recursive phenomena, this extension raises the expressive power of the language (at least when it comes to bisimulation-invariant properties of transition systems) to that of monadic second-order logic [12]. The μ-calculus is generally regarded as a universal specification language, since it embeds most other logics that are used for this purpose, such as LTL, CTL, CTL* and PDL. Despite its expressive power the μ-calculus has still reasonable computational properties; its model checking problem is in quasi-polynomial time [4] and its satisfiability problem is EXPTIME-complete [7]. Another interesting feature of the theory of the modal μ-calculus lies in its connections with other fields, in particular the theory of finite automata operating on infinite objects, and that of infinite games.

R. Ramanayake and J. Urban (Eds.): TABLEAUX 2023, LNAI 14278, pp. 242–259, 2023.
https://doi.org/10.1007/978-3-031-43513-3_14

Derivation Systems. Given the importance of the modal μ-calculus, there is a natural interest in the development and study of derivation systems for its validities. And indeed, already in [15] Kozen proposed an axiomatization. Despite the naturality of this axiom system, he only established a partial completeness result, and it took a substantial amount of time before Walukiewicz [25] managed to prove soundness and completeness for the full language.

Kozen's axiomatization amounts to a Hilbert-style derivation system, making it less attractive for proof search. If one is interested in a cut-free system, a good starting point is the two-player tableau-style game introduced by Niwiński & Walukiewicz [19]. Here we will present their system in the shape of a derivation system NW (this change of perspective can be justified by identifying winning strategies for one of the players in the game with NW-proofs). NW is a one-sided sequent system which allows for infinite proofs: although its proof rules are completely standard (and finitary), due to the unfolding rules for the fixpoint operators, derivations may have infinite branches. A crucial aspect of the NW-system is that one has to keep track of the *traces* of individual formulas along the infinite branches. A derivation will only count as a proper proof if each of its infinite branches is *successful*, in the sense that it carries a so-called ν-trace: a trace which is dominated by a *greatest* fixpoint operator.

This condition is easy to formulate but not so nice to work with. One could describe the subsequent developments in the proof theory for the modal μ-calculus as a series of modifications of the system NW which aim to get a grip on the complexities and intricacies of the above-mentioned traces, and in particular, to use the resulting "trace management" for the introduction of finitary, cyclic proof systems. Landmark results were obtained by Jungteerapanich [13] and Stirling [23], who introduced cyclic proof systems for the μ-calculus, two calculi that we will identify here under the name JS.

Automata and Derivation Systems. Applications of automata theory are ubiquitous in the theory of the modal μ-calculus, and the area of proof theory is no exception. In particular, Niwiński & Walukiewicz [19] observed that infinite matches of their game, corresponding to infinite branches in an NW-derivation, can be seen as infinite words or *streams* over some finite alphabet. It follows that *stream automata* (automata operating on infinite words) can be used to determine whether such a match/branch carries a ν-trace. Niwiński & Walukiewicz used this perspective to link their results to the exponential-time complexity of the satisfiability problem for the μ-calculus.

A key contribution of Jungteerapanich and Stirling [13,23] was to bring automata *inside* the proof system. The basic idea would be to decorate each sequent in a derivation with a state of the stream automaton which recognizes whether an infinite branch is successful or not; starting from the root, the successive states decorating the sequents on a given branch simply correspond to a run of the automaton on this branch. For this idea to work one needs the stream automaton to be *deterministic*. To see this, observe that two successful but distinct branches in a derivation would generally require two distinct runs,

and in the case of a nondeterministic automaton, these two runs might already diverge before the two branches split.

Interestingly, there is a natural stream automaton recognizing the successful branches of an NW-derivation: One may simply take the states of such an automaton to be the formulas in the (Fischer-Ladner) *closure* of the root sequent. But given the *nondeterministic* format of this automaton, before it can be used in a proof system, we need to transform it into an equivalent deterministic one. This explains the relevance of constructions for determinizing stream automata to the proof theory of the modal μ-calculus.

Determinization of Stream Automata. Using the ideas we just sketched, one may obtain sound and complete derivation systems for the modal μ-calculus in an easy way. For any deterministic automaton $\mathbb{A}$ that recognizes the successful branches in NW-derivations, one could simply introduce new-style sequents consisting of an NW-sequent decorated with a state of $\mathbb{A}$, and adapt the proof rules of NW incorporating the transition map of $\mathbb{A}$. This could be done in such a way that the stream of decorations of an infinite branch corresponds to the run of $\mathbb{A}$ on the stream of sequents of the same branch. The trace condition of NW-derivations could then be replaced by the acceptance condition of $\mathbb{A}$ (which is generally much simpler, since it does not refer to traces).

More interesting is to use specific determinization constructions, in order to design more attractive proof systems or to prove results *about* the derivation system (and thus, potentially, about the μ-calculus). In particular, some determinization constructions are based on a *power construction*, meaning that the states of the deterministic automaton consist of *macrostates* (*subsets* of the nondeterministic original) with some additional structure. Such constructions allow for proof calculi where this additional structure is incorporated into the sequents. For instance, the derivation system JS is based on the well-known Safra construction [20], in which the states of the deterministic automaton consist of macrostates of the original automaton that are organised by means of so-called *Safra trees*. Concretely, the (augmented) sequents in JS consist of a set of *annotated formulas*, with the annotations indicating the position of the formula in the Safra tree and a so-called *control* which provides additional information on the Safra tree.

Our Contribution. Our overall goal is to explicitize the role of automata theory in the design of derivation systems for the modal μ-calculus (and other fixpoint logics). Our point is that distinct determinization constructions lead to distinct sequent system, and that we may look for alternatives to the Safra construction. Concretely the contribution of this paper is threefold:

1. We provide a new determinization construction for both Büchi and parity stream automata which is based on binary trees. Our construction is similar to constructions related to so-called profile trees [8,16].
2. We apply our construction to obtain a new derivation system BT for the modal μ-calculus. While our system is similar in spirit to the system JS, a

key difference is that our sequents consist of annotated formulas, and nothing else.
3. We establish the soundness and completeness of BT. A distinguishing feature of our approach is that (up to some optimizations) this result is a *direct* consequence of the soundness and completeness of NW and the adequacy of our determinization construction.

Related Work. There is an extensive literature on applications of automata theory in the theory of the modal μ-calculus, among others [6,11,12,26]. Jungteerapanich and Stirling [13,23] were the first to obtain an annotated proof system inspired by the determinization of automata. The proof system Focus for the alternation-free μ-calculus designed by Marti & Venema [18] originates with a rather simple determinization construction for so-called weak automata. In [17], Leigh & Wehr also take a rather general approach towards the use of determinization constructions in the design of derivation systems, but they confine attention to the Safra construction.

Overview of Paper. In the next section we provide the necessary background material on binary trees, on ω-automata, on the modal μ-calculus and the proof system NW; doing so we fix our notation. In Sect. 3 we introduce a new determinization method for nondeterministic Büchi and parity automata. We will use this construction to prove the soundness and completeness of the proof system BT, which we introduce in Sect. 4. All missing proofs can be found in the extended version of this paper [5].

2 Preliminaries

Binary Trees. We let 2^* denote the set of *binary strings*; we write $<$ for the lexicographical order of 2^*, and $\sqsubseteq$ for the (initial) substring relation given by $s \sqsubseteq t$ if $sr = t$ for some r. *Substitution* for binary strings is defined in the following way: Let $s, t, \tilde{s}, r \in 2^*$ be such that $s = t\tilde{s}$, then $s[t \backslash r]$ denotes the binary string $r\tilde{s}$. A *binary tree* is a finite set of binary strings $T \subseteq 2^*$ such that $s0 \in T \Rightarrow s \in T$ and $s0 \in T \Leftrightarrow s1 \in T$. Here we let $\mathrm{leaves}(T) = \{s \in T \mid s0 \notin T\}$ denote its set of *leaves*, and $\mathrm{minL}(T)$ its *minimal* leaf of T, i.e. the unique leaf of the form $0 \cdots 0$. A set of binary strings L is a *set of leaves of a binary trees* if for all $s \neq t \in L$ we have $s \not\sqsubseteq t$ and $\mathrm{tree}(L) = \{s \in 2^* \mid \exists t \in L : s \sqsubseteq t\}$ is a binary tree.

Stream Automata. A *non-deterministic automaton* over a finite alphabet Σ is a quadruple $\mathbb{A} = \langle A, \Delta, a_I, \mathrm{Acc} \rangle$, where A is a finite set, $\Delta : A \times \Sigma \to \mathcal{P}(A)$ is the transition function of $\mathbb{A}$, $a_I \in A$ its initial state and $\mathrm{Acc} \subseteq A^\omega$ its acceptance condition. An automaton is called *deterministic* if $|\Delta(a, y)| = 1$ for all pairs $(a, y) \in A \times \Sigma$. A *run* of an automaton $\mathbb{A}$ on a stream $w = y_0 y_1 y_2 ... \in \Sigma^\omega$ is a stream $a_0 a_1 a_2 ... \in A^\omega$ such that $a_0 = a_I$ and $a_{i+1} \in \Delta(a_i, y_i)$ for all $i \in \omega$. A stream w is *accepted* by $\mathbb{A}$ if there is a run of $\mathbb{A}$ on w, which is in Acc; we define $\mathcal{L}(\mathbb{A})$ to be the set of all accepting streams of $\mathbb{A}$.

The acceptance condition can be given in different ways: A *Büchi* condition is given as a subset $F \subseteq A$. The corresponding acceptance condition is the set of runs, which contain infinitely many states in F. A *parity* condition is given as a map $\Omega : A \to \omega$. The corresponding acceptance condition is the set of runs α such that $\min\{\Omega(a) \mid a \text{ occurs infinitely often in } \alpha\}$ is even. A *Rabin* condition is given as a set $R = ((G_i, B_i))_{i \in I}$ of pairs of subsets of A. The corresponding acceptance condition is the set of runs α for which there exists $i \in I$ such that α contains infinitely many states in G_i and finitely many in B_i. Automata with these acceptance conditions are called *Büchi*, *parity* and *Rabin automata*, respectively.

Modal μ-calculus: Syntax. The set $\mathcal{L}_\mu$ of *formulas* of the modal μ-calculus is generated by the grammar

$$\varphi \ ::= \ p \mid \overline{p} \mid \bot \mid \top \mid (\varphi \vee \varphi) \mid (\varphi \wedge \varphi) \mid \Diamond\varphi \mid \Box\varphi \mid \mu x.\varphi \mid \nu x.\varphi,$$

where p and x are taken from a fixed set Prop of propositional variables and in formulas of the form $\mu x.\varphi$ and $\nu x.\varphi$ there are no occurrences of $\overline{x}$ in φ.

Formulas of the form $\mu x.\varphi$ ($\nu x.\varphi$) are called *μ-formulas* (*ν-formulas*, respectively); formulas of either kind are called *fixpoint formulas*. We write $\eta, \lambda \in \{\mu, \nu\}$ to denote an arbitrary fixpoint operator. We use standard terminology and notation for the binding of variables by the fixpoint operators and for substitutions, and make sure only to apply substitution in situations where no variable capture will occur. An important use of the substitution operation concerns the *unfolding* $\chi[\xi/x]$ of a fixpoint formula $\xi = \eta x.\chi$.

Given two formulas $\varphi, \psi \in \mathcal{L}_\mu$ we write $\varphi \to_C \psi$ if ψ is either a direct boolean or modal subformula of φ, or else φ is a fixpoint formula and ψ is its unfolding. The *closure* $\mathsf{Clos}(\Phi) \subseteq \mathcal{L}_\mu$ of $\Phi \subseteq \mathcal{L}_\mu$ is the least superset of Φ that is closed under this relation. It is well known that $\mathsf{Clos}(\Phi)$ is finite iff Φ is finite. A *trace* is a sequence $(\varphi_n)_{n<\kappa}$, with $\kappa \leq \omega$, such that $\varphi_n \to_C \varphi_{n+1}$, for all $n + 1 < \kappa$.

We define a *dependence order* on the fixpoint formulas occurring in Φ, written $\mathsf{Fix}(\Phi)$, by setting $\eta x.\varphi <_\Phi \lambda y.\psi$ (where smaller in $<_\Phi$ means being of higher priority) if $\mathsf{Clos}(\eta x.\varphi) = \mathsf{Clos}(\lambda y.\psi)$ and $\eta x.\varphi$ is a subformula of $\lambda y.\psi$. One may define a *parity function* $\Omega : \mathsf{Fix}(\Phi) \to \omega$, which respects this order (i.e., $\Omega(\eta x.\varphi) < \Omega(\lambda y.\psi)$ if $\eta x.\varphi <_\Phi \lambda y \psi$) and satisfies $\Omega(\eta x.\varphi)$ is even iff $\eta = \nu$. Let $\max_\Omega(\Phi) = \max\{\Omega(\nu x.\varphi) \mid \nu x.\varphi \in \mathsf{Fix}(\Phi)\}$.

It is well known that any infinite trace $\tau = (\varphi_n)_{n<\kappa}$ features a unique formula φ that occurs infinitely often on τ and is a subformula of φ_n for cofinitely many n. This formula is always a fixpoint formula, and where it is of the form $\eta x.\psi$ we call τ an *η-trace*.

Since the semantics of the modal μ-calculus only plays an indirect role in our paper, we refrain from giving the definition.

Non-wellfounded Proofs. A sequent Γ is a finite set of μ-calculus formulas, possibly with additional structure such as annotations. Rules have the following form, possibly with additional side conditions:

$$R\colon \frac{\Gamma_1 \quad \cdots \quad \Gamma_n}{\Gamma} \qquad\qquad \mathsf{D}^{\mathsf{x}}\colon \frac{\begin{array}{c}[\Gamma]^{\mathsf{x}}\\ \vdots \\ \Gamma\end{array}}{\Gamma}$$

A rule R, where $n = 0$, is called an axiom. The rules D^{x} are called *discharge* rules. Each discharge rule is marked by a unique *discharge token* taken from a fixed infinite set $\mathcal{D} = \{\mathsf{x}, \mathsf{y}, ...\}$.

Definition 1. *A* derivation system $\mathcal{P}$ *is a set of rules. A* $\mathcal{P}$ derivation $\pi = (T, P, \mathsf{S}, \mathsf{R}, \mathsf{f})$ *is a quintuple such that* (T, P) *is a, possibly infinite, tree with nodes* T *and parent relation* P*;* S *is a function that maps every node* $u \in T$ *to a non-empty sequent* Σ_u*;* R *is a function that maps every node* $u \in T$ *to its* label $\mathsf{R}(u)$*, which is either (i) the name of a rule in* $\mathcal{P}$ *or (ii) a discharge token; and* f *is a partial function that maps some nodes* $u \in T$ *to its* principal formula $\mathsf{f}(u) \in \mathsf{S}(u)$*. If a specific formula* φ *in the conclusion of a rule is designated, then* $\mathsf{f}(u) = \varphi$ *and otherwise* $\mathsf{f}(u)$ *is undefined. To qualify as a derivation, such a quintuple is required to satisfy the following conditions:*

1. *If a node is labeled with the name of a rule then it has as many children as the rule has premises, and the annotated sequents at the node and its children match the specification of the rules.*
2. *If a node is labeled with a discharge token then it is a leaf. For every leaf* l *that is labeled with a discharge token* $\mathsf{x} \in \mathcal{D}$ *there is exactly one node* $u \in T$ *that is labeled with* D^{x}*. This node* u *and its child are proper ancestors of* l*. In this situation we call* l *a* discharged leaf*, and* u *its* companion*; we write* c *for the function that maps a discharged leaf* l *to its companion* $c(l)$ *and write* $p(l)$ *for the path in* T *from* $c(l)$ *to* l*.*

A derivation $\pi' = (T', P', \mathsf{S}', \mathsf{R}', \mathsf{f}')$ is a *subderivation* of $\pi = (T, P, \mathsf{S}, \mathsf{R}, \mathsf{f})$ if (T', P') is a subtree of (T, P) and $\mathsf{S}', \mathsf{R}', \mathsf{f}'$ and $\mathsf{S}, \mathsf{R}, \mathsf{f}$ are equal on (T', P'). A derivation π is called *regular* if it has finitely many distinct subderivations.

Definition 2. *Let* $\pi = (T, P, \mathsf{S}, \mathsf{R}, \mathsf{f})$ *be a derivation. We define two graphs we are interested in: (i) The usual* proof tree $\mathcal{T}_\pi = (T, P)$ *and (ii) the* proof tree with back edges $\mathcal{T}_\pi^C = (T, P^C)$*, where* $P^C = P \cup \{(l, c(l)) \mid l$ *is a discharged leaf*$\}$ *is the parent relation plus back-edges for every discharged leaf.*

A strongly connected subgraph *in* $\mathcal{T}_\pi^C$ *is a set* S *of nodes, such that for every* $u, v \in S$ *there is a* P^C*-path from* u *to* v*.*

The NW *Proof System.* The rules of the derivation system NW, which is directly based on the tableau games introduced by Niwiński & Walukiewicz [19], are given in Fig. 1.

In order to decide whether an NW derivation qualifies as a proper *proof*, one has to keep track of the development of individual formulas along infinite branches of the proofs.

$$\mathsf{Ax1}\ \frac{}{p, \bar{p}, \Gamma} \qquad \mathsf{Ax2}\ \frac{}{\top, \Gamma} \qquad \mathsf{R}_\vee\ \frac{\varphi, \psi, \Gamma}{\varphi \vee \psi, \Gamma} \qquad \mathsf{R}_\wedge\ \frac{\varphi, \Gamma \quad \psi, \Gamma}{\varphi \wedge \psi, \Gamma}$$

$$\mathsf{R}_\Box\ \frac{\varphi, \Gamma}{\Box\varphi, \Diamond\Gamma, \Delta} \qquad \mathsf{R}_\mu\ \frac{\varphi[\mu x.\varphi/x], \Gamma}{\mu x.\varphi, \Gamma} \qquad \mathsf{R}_\nu\ \frac{\varphi[\nu x.\varphi/x], \Gamma}{\nu x.\varphi, \Gamma}$$

Fig. 1. Rules of NW

Definition 3. *Let Γ, Γ' be sequents, ξ be a formula such that Γ is the conclusion and Γ' is a premise of a rule in Fig. 1 with principal formula ξ. We define the* active *and* passive trail relation $\mathsf{A}_{\Gamma,\xi,\Gamma'}, \mathsf{P}_{\Gamma,\xi,\Gamma'} \subseteq \Gamma \times \Gamma'$. *Both relations are defined via a case distinction on ξ:*

Case $\xi = \Box\varphi$: *Then $\Gamma = \Box\varphi, \Diamond\Lambda, \Delta$ and $\Gamma' = \varphi, \Lambda$. We define $\mathsf{A}_{\Gamma,\xi,\Gamma'} = \{(\Box\varphi, \varphi)\} \cup \{(\Diamond\chi, \chi) \mid \chi \in \Lambda\}$ and $\mathsf{P}_{\Gamma,\xi,\Gamma'} = \varnothing$.*

Case $\xi = \varphi \vee \psi$: *Then $\Gamma = \varphi \vee \psi, \Lambda$ and $\Gamma' = \varphi, \psi, \Lambda$. We define $\mathsf{A}_{\Gamma,\xi,\Gamma'} = \{(\varphi \vee \psi, \varphi), (\varphi \vee \psi, \psi)\}$ and $\mathsf{P}_{\Gamma,\xi,\Gamma'} = \{(\chi, \chi) \mid \chi \in \Lambda\}$.*

The relations for the remaining rules are defined analogously.

The trail relation $\mathsf{T}_{\Gamma,\xi,\Gamma'} \subseteq \Gamma \times \Gamma'$ *is defined as $\mathsf{A}_{\Gamma,\xi,\Gamma'} \cup \mathsf{P}_{\Gamma,\xi,\Gamma'}$. Finally, for nodes u, v in an* NW *proof π, such that $P(u, v)$, we define $\mathsf{T}_{u,v} = \mathsf{T}_{\mathsf{S}(u),\mathsf{f}(u),\mathsf{S}(v)}$*

Note that for any two nodes u, v with $P(u, v)$ and $(\varphi, \psi) \in \mathsf{T}_{u,v}$, we have either $(\varphi, \psi) \in \mathsf{A}_{u,v}$ and $\varphi \to_C \psi$, or else $(\varphi, \psi) \in \mathsf{P}_{u,v}$ and $\varphi = \psi$. The idea is that A connects the active formulas in the premise and conclusion, whereas P connects the side formulas.

Definition 4. *Let $\pi = (T, P, \mathsf{S}, \mathsf{R}, \mathsf{f})$ be an* NW *derivation. A* branch *of π is simply a (finite or infinite) branch of the underlying tree (T, P) of π. A* trail *on a branch $\alpha = (v_n)_{n<\kappa}$ is a sequence $\tau = (\varphi_n)_{n<\kappa}$ of formulas such that $(\varphi_i, \varphi_{i+1}) \in \mathsf{T}_{v_i,v_{i+1}}$, whenever $i + 1 < \kappa$. We obtain the* tightening $\widehat{\tau}$ *of such a τ by erasing all φ_{i+1} from τ for which $(\varphi_i, \varphi_{i+1})$ belongs to the passive trail relation $\mathsf{P}_{v_i,v_{i+1}}$. We call τ a* ν-trail *if its tightening $\widehat{\tau}$ is a ν-trace (and so, in particular, it is infinite).*

Definition 5. *An* NW proof π *is an* NW *derivation such that on every infinite branch of π there is a ν-trail. We write $\mathsf{NW} \vdash \Gamma$ if there is an* NW *proof of Γ, i.e., an* NW *proof, where Γ is the sequent at the root of the proof.*

Soundness and Completeness of NW for guarded formulas, (i.e., where in every subformula $\eta x.\psi$ all free occurrences of x in ψ are in the scope of a modality) follows from the results by Niwiński & Walukiewicz [19]. As pointed out in [2], it follows from [24] and [10] that the result in fact holds for arbitrary formulas. By Theorem 6.3 in [19], NW-proofs can be assumed to be regular, and this observation applies to unguarded formulas as well.

Theorem 1 (Soundness & Completeness). *Let Γ be a sequent, then $\bigvee \Gamma$ is valid iff $\mathsf{NW} \vdash \Gamma$ iff Γ has a regular* NW*-proof.*

3 Determinization of Automata with Binary Trees

3.1 Büchi automata

Let Σ be an alphabet and $\mathbb{B} = \langle B, \Delta, b_I, F\rangle$ a nondeterministic Büchi automaton over Σ. We want to present an equivalent deterministic Rabin automaton.

The *run tree* of $\mathbb{B}$ on a word $w = (w_i)_{i\in\omega}$ is a pair $\mathsf{R} = (R, l)$, where R is the full infinite binary tree and l labels every node s with $B_s \subseteq B$, such that $l(\epsilon) = \{b_I\}$ and for $|s| = i$: $l(s1) = \Delta(B_s, w_i) \cap F$ and $l(s0) = \Delta(B_s, w_i) \cap \overline{F}$, where we define $\Delta(B_s, y) = \bigcup_{b\in B_s} \Delta(b, y)$. It describes all possible runs of $\mathbb{B}$ on w, using the 1 s to keep track of visited states in F.

The *profile tree*, introduced in [9], is a pruned version of the run tree, where 1) at each level all but the (lexicographically) greatest occurrence of a state b are removed and 2) nodes labelled by the empty set are deleted. This results in a tree of bounded width, where every node has 0,1 or 2 children. It can be proved that $\mathbb{B}$ accepts a stream w iff the corresponding profile tree has a branch with infinitely many 1 s.

In [8] a determinization method was defined, where macrostates encode levels of the profile tree. In our approach macrostates encode a compressed version of the whole profile tree up to some level: Nodes u, v are identified (iteratively), if v is the unique child of u. This results in finite binary trees, where leaves are labelled by subsets of B. In every step of the transition function we add one level of the run tree and then prune and compress the tree to obtain a binary tree again. Whenever a 1 is compressed (in the sense of a node being identified with its right child) we know that a state in F has been visited and mark the node green. A run of the deterministic automaton is accepted if there is a node, which never gets removed and is marked green infinitely often. Figure 2 contains an example of this determinization construction.

Formally we define the deterministic Rabin automaton $\mathbb{B}^D = \langle B^D, \delta, b'_I, R\rangle$ as follows: An element S in the carrier B^D of $\mathbb{B}^D$ is called a *macrostate* and consists of

- a subset B_S of B,
- a map $f : B_S \to 2^*$, such that[1] $\mathrm{ran}(f)$ is a set of leaves of a binary tree and
- a colouring map $c : \mathrm{tree}(\mathrm{ran}(f)) \to \{\mathrm{green}, \mathrm{red}, \mathrm{white}\}$.

We define T^S to be the binary tree $\mathrm{tree}(\mathrm{ran}(f))$, that has $\mathrm{ran}(f)$ as its leaves and say that a binary string s is *in play* if $s \in T^S$. If it is clear from the context we occasionally abbreviate T^S by T. We will sometimes denote a macrostate by a set of pairs (b, s), usually written as b^s, where $b \in B_S$ and $s = f(b)$ and deal with the colouring c implicitly.

The initial macrostate b'_I consists of the singleton $\{b_I^\epsilon\}$, where $c(\epsilon) = \mathrm{white}$. To define the transition function δ let S be in B^D and $y \in \Sigma$. We define $\delta(S, y) = S'$, where starting from the empty set we build up S' in the following steps:

1. Move: For every $a^s \in S$ and $b \in \Delta(a, y)$, add b^s to S'.

[1] Here $\mathrm{ran}(f)$ denotes the co-domain of f.

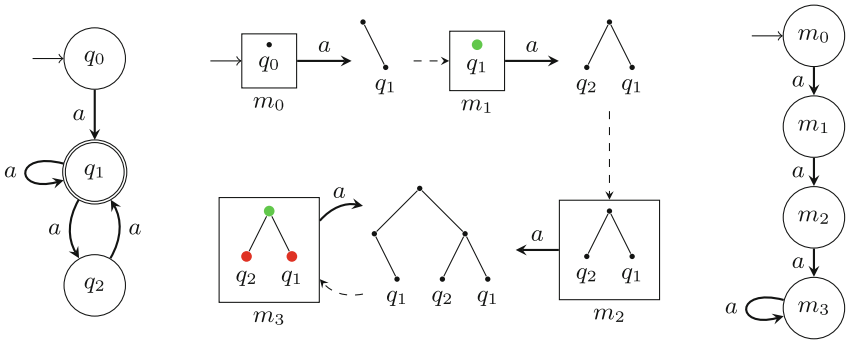

Fig. 2. A nondeterministic Büchi automaton $\mathbb{B}$ on the left and its determinization $\mathbb{B}^D$ on the right. The diagram in the middle shows the internal structure of the macrostates m_0, m_1, m_2 and m_3 of $\mathbb{B}^D$. Binary trees are represented in the obvious way (i.e., the root is the string ϵ, and for every node the left child appends 0 and the right child appends 1). The transitions of $\mathbb{B}^D$ are split in two parts: In the first part one level of the run tree is added, corresponding to the steps 1 and 2 in the definition of the transition function. In the second part (the dashed arrows) the tree is pruned and compressed, corresponding to the steps 3 and 4. The acceptance condition of $\mathbb{B}^D$ is such that the word a^ω is accepted by $\mathbb{B}^D$ because the string ϵ is always in play, marked green infinitely often and never red.

2. Append: For every $a^s \in S'$, where $a \notin F$, change a^s to a^{s0}. For every $a^s \in S'$, where $a \in F$, change a^s to a^{s1}.
3. Resolve: If a^s and a^t are in S', where $s < t$, delete a^s.
4. Compress/Colour: Let $c(t) =$ white for every $t \in T^{S'}$. Now we compress and colour T in the following way, until there exists no *witness* $t \in T$, such that (a) or (b) is applicable:[2]
   (a) For any $t \in T$, such that $t0 \in T$ and $t1 \notin T$, change every $a^s \in S'$, where $t0 \sqsubseteq s$, to $a^{s[t0\backslash t]}$. For any $s \in T$, where $t \sqsubset s$, let $c(s) =$ red.
   (b) For any $t \in T$, such that $t0 \notin T$ and $t1 \in T$, change every $a^s \in S'$, where $t1 \sqsubseteq s$, to $a^{s[t1\backslash t]}$. For any $s \in T$ such that $t = s0 \cdots 0$, let $c(s) =$ green, if $c(s) \neq$ red. In particular let $c(t) =$ green if $c(t) \neq$ red. For any $s \in T$, where $t \sqsubset s$, let $c(s) =$ red.

We define B^D as the set of macrostates that can be reached from b'_I in this way.

A run of $\mathbb{B}^D$ is accepting if there is a binary string s, which is in play cofinitely often such that $c(s)$ is green infinitely often and red only finitely often.

Theorem 2. *$\mathbb{B}$ accepts a word w iff $\mathbb{B}^D$ accepts w.*

Remark 1. For a Büchi automaton of n states, our construction yields a deterministic automaton $\mathbb{B}^D$ with $n^{\mathcal{O}(n)}$ states and a Rabin condition of $\mathcal{O}(2^n)$ pairs,

[2] As shown in Proposition 1 of [5] this procedure does not depend on the order in which witnesses are chosen, and thus produces a unique binary tree.

see Lemma 5 of [5]. With some adaptations we could also match the optimal Rabin condition, which is known to be linear-size [20].

This can be achieved by adding an labelling function as follows: Let $L = \{1, \ldots, 2n-1\}$ be the set of potential labels. Macrostates are defined as before, where an additional injective function $l : T^S \to L$ is added. For the initial state we let $l(\epsilon) = 1$. The steps 1–4 in the transition function remain the same, where we add a final step 5 in which we define the new labeling function l': We let $l'(s) = l(s)$ for all s that already occurred in T^S and for all $s \in T^{S'} \setminus T^S$ we let $c(s) =$ red and choose new, distinct labels in L, i.e. ones which do not occur in $\mathrm{ran}(l)$. The binary tree $T^{S'}$ has at most n leaves, hence it has at most $2n-1$ many nodes and this is always possible.

The new acceptance condition has the following form: A run of the automaton is accepting if there is a label $k \in L$, such that $c(l^{-1}(k))$ is green infinitely often and red only finitely often. Here $c(l^{-1}(k))$ is defined to be red if $k \notin \mathrm{ran}(l)$. This is a Rabin condition with $\mathcal{O}(n)$ pairs. Notably we still have $n^{\mathcal{O}(n)}$ macrostates, thus the determination method is optimal.

3.2 Parity Automata

We now extend the approach to parity automata. Let Σ be an alphabet and $\mathbb{A} = \langle A, \Delta_A, a_I, \Omega \rangle$ be a nondeterministic parity automaton.

In order to present the intuitive idea behind the construction we first transform $\mathbb{A}$ into an equivalent nondeterministic Büchi automaton $\mathbb{B}$. Let m be the maximal even priority of Ω. For even $k = 0, 2, ...m$ we define $\mathbb{A}_0, \mathbb{A}_2, ..., \mathbb{A}_n$ as copies of $\mathbb{A}$ without the states of priority smaller than k, i.e. $\mathbb{A}_k = \langle A_k, \Delta_k, F_k \rangle$ with $A_k = \{a_k \mid a \in A \wedge \Omega(a) \geq k\}$, $\Delta_k = \Delta_A|_{A_k}$ and $F_k = \{a_k \in A_k \mid \Omega(a) = k\}$. Now we define the nondeterministic Büchi automaton $\mathbb{B} = \langle B, \Delta_B, b_I, F \rangle$:[3]

$$B = A \cup \bigcup_{\substack{k=0 \\ k \text{ even}}}^{m} A_k, \qquad b_I = a_I, \qquad F = \bigcup_{\substack{k=0 \\ k \text{ even}}}^{m} F_k,$$

$$\Delta_B = \Delta_A \cup \bigcup_{\substack{k=0 \\ k \text{ even}}}^{m} \Delta_k \cup \{(a, y, b_k) \in A \times \Sigma \times A_k \mid b \in \Delta_A(a, y), k = 0, 2, ..., m\}.$$

Although $\mathbb{A}_k$ is not an automaton, as it does not have an initial state, we can define the Büchi automaton $\mathbb{A} \cup \mathbb{A}_k = \langle A \cup A_k, \Delta_B|_{A \cup A_k}, a_I, F_k \rangle$ for $k = 0, ..., m$.

The intuition behind the determinization of the parity automaton $\mathbb{A}$ is the following: We apply the binary tree construction to every automaton $\mathbb{A} \cup \mathbb{A}_k$ for $k = 0, 2, ..., m$, which is possible as there are no paths from A_k to A_j if $k \neq j$ and none of the accepting states of $\mathbb{B}$ are in the set A. The annotation of a state $a \in \mathbb{A}$ will then be the tuple $(s_0, s_2, ..., s_m)$, where s_k is the annotation at the state $a_k \in \mathbb{A} \cup \mathbb{A}_k$. Note that the automaton $\mathbb{A}^D$ will be different from the automaton obtained from the binary tree construction on the whole $\mathbb{B}$.

[3] For easier notation we represent the transition function $B \times \Sigma \to \mathcal{P}(B)$ by its corresponding relation (i.e., subset of $B \times \Sigma \times B$).

To make that formal we need some definitions. A *treetop* L is a set of leaves of a binary tree, where potentially the minimal leaf is missing, i.e. L is a finite set of binary strings such that for all $s \neq t \in L$ it holds $s \not\sqsubseteq t$ and $\text{tree}(L) = \{s \in 2^* \mid \exists t \in L : s \sqsubseteq t\} \cup \{s0 \mid s = 0 \cdots 0 \text{ and } s1 \in L\}$ is a binary tree.

For even m let $\text{TSeq}(m) = \{(s_0, s_2, ..., s_m) \mid s_0, s_2, ..., s_m \in 2^*\}$ be the set of sequences of length $\frac{m}{2} + 1$, where $s_0, ..., s_m$ are binary strings. Let π_k be the projection function, which maps $\sigma = (s_0, ..., s_m)$ to s_k for $k = 0, 2, ..., m$. We define a partial order $<$ on $\text{TSeq}(m)$: Let $(s_0, ..., s_m) < (t_0, ..., t_m)$ if there exists $l \in \{0, ..., m\}$ such that $s_l < t_l$ and $s_j = t_j$ for $j = 0, ..., l-2$.

We now define the deterministic Rabin automaton $\mathbb{A}^D = \langle A^D, \delta_A, a'_I, R_A \rangle$. Let m be the maximal even priority of Ω in $\mathbb{A}$. An element S in the carrier A^D of $\mathbb{A}^D$ consists of a tuple $(A_S, f, c_0, ..., c_m)$, where

- A_S is a subset of A,
- $f : A_S \to \text{TSeq}(m)$, such that $\text{ran}(\pi_k \circ f)$ is a treetop for $k = 0, ..., m$ and
- c_k is a colouring map from $\text{tree}(\text{ran}(\pi_k \circ f)) \to \{\text{green}, \text{red}, \text{white}\}$ for $k = 0, 2, ..., m$.

We define T_k^S to be the binary tree $\text{tree}(\text{ran}(\pi_k \circ f))$ for $k = 0, 2, ..., m$ and say a binary string s is *in play at position* k if $s \in T_k^S$. If the context is clear we will abbreviate T_k^S with T_k. Again we sometimes denote a macrostate by a set of pairs (a, σ), usually written as a^σ, where $a \in A_S$ and $\sigma = f(a)$ and deal with the colourings c_k implicitly.

The initial macrostate a'_I consists of the singleton $\{a_I^{(\epsilon,...,\epsilon)}\}$. To define the transition function δ_A let S be in A^D and $y \in \Sigma$. We define $\delta_A(S, y) = S'$, where S' is constructed in the following steps:

1. (a) Move: For every $a^\sigma \in S$ and $b \in \Delta_A(a, y)$, add b^σ to S'.
   (b) Reduce: For every $a^\sigma \in S'$, change a^σ to $a^{\sigma'}$, where σ' is obtained from $\sigma = (s_0, ..., s_m)$ by replacing every s_j with $j > \Omega(a)$ by $\text{minL}(T_j)$.
2. Append: For every $a^\sigma \in S'$ and $\sigma = (s_0, ..., s_m)$, change a^σ to $a^{\sigma'}$, where $\sigma' = (s_0 0, ..., s_{k-2} 0, s_k 1, s_{k+2} 0, ..., s_m 0)$ if $\Omega(a) = k$ is even, and $\sigma' = (s_0 0, ..., s_m 0)$ if $\Omega(a) = k$ is odd.
3. Resolve: If a^σ and a^τ are in S' and $\sigma < \tau$, delete a^σ.
4. Compress/Colour: Do for every $k = 0, 2, ..., m$: Let $c_k(t) = \text{white}$ for any $t \in T_k$. Now we compress and colour T_k inductively in the following way, until there exists no *witness* $t \in T_k$, such that (a) or (b) is applicable:
   (a) For any $t \in T_k$, such that $t0 \in T_k$ and $t1 \notin T_k$, change every $a^\sigma \in S'$, where $\sigma = (s_0, ..., s_m)$, and $t0 \sqsubseteq s_k$, to $a^{\sigma'}$, where $\sigma' = (s_0, ..., s_k[t0 \backslash t], ..., s_m)$. For any $s \in T_k$, where $t \sqsubset s$, let $c_k(s) = \text{red}$.
   (b) For any $t \in T_k$, such that $t0 \notin T_k$, $t1 \in T_k$ and $t \neq 0 \cdots 0$, change every $a^\sigma \in S'$, where $\sigma = (s_0, ...s_m)$, and $t1 \sqsubseteq s_k$, to $a^{\sigma'}$, where $\sigma' = (s_0, ..., s_k[t1 \backslash t], ..., s_m)$. For any $s \in T_k$ such that $t = s0 \cdots 0$, let $c_k(s) = \text{green}$, if $c_k(s) \neq \text{red}$. In particular let $c_k(t) = \text{green}$ if $c_k(t) \neq \text{red}$. For any $s \in T_k$, where $t \sqsubset s$, let $c_k(s) = \text{red}$.

A run of $\mathbb{A}^D$ is accepting if there is $k \in \{0, 2, ..., m\}$ and a binary string s, which is in play at position k cofinitely often such that $c_k(s)$ is green infinitely often and red only finitely often.

Theorem 3. *Let $\mathbb{A}$ be a parity automaton and $\mathbb{A}^D$ the deterministic Rabin automaton defined from $\mathbb{A}$. Then $L(\mathbb{A}) = L(\mathbb{A}^D)$.*

Remark 2. For a parity automaton $\mathbb{A}$ of size n with highest even priority m, our construction produces a deterministic Rabin automaton with $n^{\mathcal{O}(m \cdot n)}$ macrostates and $\mathcal{O}(m \cdot 2^n)$ Rabin pairs, see Lemma 6 of [5].

4 BT Proofs

4.1 Proof Systems

We present two non-wellfounded proof systems for the modal μ-calculus, namely BT and BT^∞. The idea is that annotated sequents in the BT system correspond to macrostates of $\mathbb{A}^D$, where $\mathbb{A}$ is a nondeterministic parity automaton checking the trace condition in an NW proof. The rules of BT resemble the transition function of $\mathbb{A}^D$.

Let Φ be a set of formulas, the sequent we want to prove, and let $m = \max_\Omega(\Phi)$ be the maximal even priority of Ω. *Annotated sequents* are sets of pairs (φ, σ), usually written as φ^σ, where $\varphi \in \mathsf{Clos}(\Phi)$ and $\sigma \in \mathrm{TSeq}(m)$. For an annotated sequent Γ we let Γ^N be the set of annotations occurring in Γ, i.e. $\Gamma^N = \{\sigma \in \mathrm{TSeq}(m) \mid \exists \varphi \text{ s.t. } \varphi^\sigma \in \Gamma\}$. We let Γ^N_k be the set of binary strings occurring at the k-th position of the annotations in Γ, i.e., $\Gamma^N_k = \pi_k[\Gamma^N]$. We say that a string s *occurs in* Γ^N_k if there exists $t \in \Gamma^N_k$ such that $s \sqsubseteq t$.

For $\sigma = (s_0, ..., s_m) \in \mathrm{TSeq}(m)$ we define $\sigma \cdot 1_k = (s_0, ..., s_k 1, ..., s_m)$ and $\sigma \cdot 0_k = (s_0, ..., s_k 0, ..., s_m)$. For an annotated sequent Γ we let $\Gamma^{\cdot 0_k}$ denote the annotated sequent $\{\varphi^{\sigma \cdot 0_k} \mid \varphi^\sigma \in \Gamma\}$.

Let Γ be an annotated sequent and $\varphi^\sigma \in \Gamma$. We define $\sigma \restriction k^\Gamma$ to be the tuple of binary strings obtained from $\sigma = (s_0, ..., s_m)$ by replacing every s_j with $j > k$ by $\mathrm{minL}(\mathrm{tree}(\Gamma^N_j)$. If the context Γ is clear we write $\sigma \restriction k$ instead of $\sigma \restriction k^\Gamma$.

The rules $\mathsf{Compress}^{s0}_k$ and $\mathsf{Compress}^{s1}_k$ are schemata for $k = 0, 2, ..., m$ and $s \in 2^*$. In these rules the notation $\varphi_i^{(..., st_i, ...)}$ is to be read such that st_i is the binary string in the k-th position of the annotation. We will write $\mathsf{Compress}$ for any of those rules and write $\mathsf{Compress}^s_k$ for either $\mathsf{Compress}^{s0}_k$ or $\mathsf{Compress}^{s1}_k$.

Note that, if one ignores the annotations, the rules $\mathsf{Ax1}$, $\mathsf{Ax2}$, $\mathsf{R}_\vee$, $\mathsf{R}_\wedge$, R_μ, R_ν and $\mathsf{R}_\Box$ in Fig. 3 are the same as the rules of NW. As mentioned above annotated sequents in the BT system correspond to macrostates of $\mathbb{A}^D$, where $\mathbb{A}$ is a nondeterministic parity automaton checking the trace condition in an NW proof. The rules of BT correspond to the transition function δ_A of $\mathbb{A}^D$, where the transformations of δ_A are distributed over multiple rules: Step 1(a) of δ_A is carried out in every rule and step 1(b) and step 2 correspond to the modification of the annotations in the rules R_μ and R_ν. Notably, we do not add zeros to the annotations if the zeros would get deleted anyway in step 4 of the transition function. The rules $\mathsf{Resolve}$ and $\mathsf{Compress}$ are additional and correspond to steps 3 and 4 of δ_A.

$$\mathsf{Ax1}\colon \frac{}{p^\sigma, \bar{p}^\tau, \Gamma} \qquad \mathsf{Ax2}\colon \frac{}{\top^\sigma, \Gamma} \qquad \mathsf{R}_\vee\colon \frac{\varphi^\sigma, \psi^\sigma, \Gamma}{(\varphi \vee \psi)^\sigma, \Gamma} \qquad \mathsf{R}_\wedge\colon \frac{\varphi^\sigma, \Gamma \quad \psi^\sigma, \Gamma}{(\varphi \wedge \psi)^\sigma, \Gamma}$$

$$\mathsf{R}_\mu\colon \frac{\varphi[x \backslash \mu x.\varphi]^{\sigma \restriction \Omega(\mu x.\varphi)}, \Gamma}{\mu x.\varphi^\sigma, \Gamma} \qquad \mathsf{R}_\nu\colon \frac{\varphi[x \backslash \nu x.\varphi]^{\sigma \restriction k \cdot 1_k}, \Gamma^{\cdot 0_k}}{\nu x.\varphi^\sigma, \Gamma} \quad \text{where } k = \Omega(\nu x.\varphi)$$

$$\mathsf{R}_\Box\colon \frac{\varphi^\sigma, \Gamma}{\Box\varphi^\sigma, \Diamond\Gamma, \Delta} \qquad \mathsf{Resolve}\colon \frac{\varphi^\sigma, \Gamma}{\varphi^\sigma, \varphi^\tau, \Gamma} \quad \text{where } \sigma > \tau \qquad \mathsf{D}^{\mathsf{x}}\colon \frac{\begin{array}{c}[\Gamma]^{\mathsf{x}} \\ \vdots \\ \Gamma\end{array}}{\Gamma}$$

$$\mathsf{Compress}_k^{s0}\colon \frac{\varphi_1^{(\ldots,st_1,\ldots)}, \ldots, \varphi_n^{(\ldots,st_n,\ldots)}, \Gamma}{\varphi_1^{(\ldots,s0t_1,\ldots)}, \ldots, \varphi_n^{(\ldots,s0t_n,\ldots)}, \Gamma} \quad \text{where } s \text{ does not occur in } \Gamma_k^N$$

$$\mathsf{Compress}_k^{s1}\colon \frac{\varphi_1^{(\ldots,st_1,\ldots)}, \ldots, \varphi_n^{(\ldots,st_n,\ldots)}, \Gamma}{\varphi_1^{(\ldots,s1t_1,\ldots)}, \ldots, \varphi_n^{(\ldots,s1t_n,\ldots)}, \Gamma} \quad \text{where } s \text{ does not occur in } \Gamma_k^N \text{ and } s \neq 0 \cdots 0$$

Fig. 3. Rules of BT

Definition 6. *A* BT derivation π *is a derivation defined from the rules in Fig. 3, such that the rules are applied with the following priority: first* Resolve*, then* Compress*, and then all other rules.*

Just as annotated sequents correspond to macrostates of the deterministic automaton $\mathbb{A}^D$, the soundness condition of BT^∞ and BT correspond to the acceptance condition of $\mathbb{A}^D$: We say that a pair (k, s) is preserved at a node, if s is in play at position k at the corresponding macrostate and not marked red; and progresses if it is marked green.

Definition 7. *Let* π *be a* BT *derivation of* Φ, $m = \max_\Omega(\Phi)$ *and* S *be a set of nodes in* π*. Let* $k \in \{0, 2, ..., m\}$ *and* $s \in 2^*$*. We say that the pair* (k, s)

- *is* preserved *on* S *if*
  - s *occurs in* $\mathsf{S}(v)_k^N$ *for every* v *in* S *and*
  - *if* $\mathsf{R}(v) = \mathsf{Compress}_k^t$ *for a node* v *in* S*, then* $t \not\sqsubset s$*,*
- progresses *(infinitely often) on* S *if there is* $s' = s0 \cdots 0$ *such that* $\mathsf{R}(v) = \mathsf{Compress}_k^{s'1}$ *for some* v *in* S *(for infinitely many* $v \in S$*).*

Definition 8. *Let* π *be a* BT *derivation. An infinite branch* $\alpha = (u_i)_{i \in \omega}$ *in* π *is* successful *if there are* N *and* (k, s) *such that* (k, s) *is preserved and progresses infinitely often on* $\{u_i \mid i \geq N\}$*. A* BT^∞ proof *is a* BT *derivation without occurrences of* D^{x} *and such that all infinite branches are successful. A* BT proof *is a finite* BT *derivation such that for each* strongly connected subgraph S *in* $\mathcal{T}_\pi^C$ *there exists* (k, s) *that is preserved and progresses on* S*.*

We write $\mathsf{BT} \vdash \Gamma$ *(*$\mathsf{BT}^\infty \vdash \Gamma$*) if there is a* BT *(*$\mathsf{BT}^\infty$*) proof of* Γ*, i.e., a proof, where* Γ *is the sequent at the root of the proof.*

Remark 3. In the proof system JS introduced by Jungteerapanich and Stirling [13,23] annotated sequents are of the form $\theta \vdash \varphi_1^{a_1}, ..., \varphi_n^{a_n}$, where $a_1, ..., a_n$ are sequences of names and the so-called *control* θ is a linear order on all names occurring in the sequent. In contrast to JS our sequents consist of formulas with annotations and nothing else, that is, no control. On the other hand the soundness condition of BT is less local: It speaks about strongly connected subgraphs, whereas in JS only paths between leafs and its companions have to be checked. We see that the control in JS gives information on the structure of the cyclic proof tree. Interestingly, we could also add a control to our sequents and obtain a soundness condition that talks about paths, if desired. Similarly, in [1] a control was added to a cyclic system for the first-order μ-calculus introduced by [22] to obtain a path-based system.

4.2 Soundness and Completeness

The intuitive idea behind the BT^∞ proof system is the following: Starting with an NW proof, we can define a nondeterministic parity automaton $\mathbb{A}$, that checks if an infinite branch carries a ν-trail. Using the determinization method from Sect. 3 we simulate macrostates of $\mathbb{A}^D$ by annotated formulas in the proof system. Thus an infinite branch in BT^∞ resembles an infinite run of $\mathbb{A}^D$. This will be formalised in the Soundness and Completeness proofs.

Tracking Automaton. Let Φ be a sequent of formulas, $\eta x_1.\psi_1, ..., \eta x_n.\psi_n$ the fixpoint formulas in $\mathsf{Fix}(\Phi)$ and Ω the parity function on $\mathsf{Fix}(\Phi)$.

We define a nondeterministic parity automaton that checks if there is a ν-trail on an infinite branch of some NW proof of Φ. The alphabet Σ consists of all triples (Γ, ξ, Γ'), where $\Gamma \subseteq \mathsf{Clos}(\Phi)$ is the conclusion and $\Gamma' \subseteq \mathsf{Clos}(\Phi)$ is the premise of a rule in Fig. 1 with principal formula ξ. We define the following nondeterministic parity automaton $\mathbb{A} = (A, \Delta, a_I, \Omega_A)$:

- $A = a_I \cup \mathsf{Clos}(\Phi) \cup \{\eta x.\psi^* \mid \eta x.\psi \in \mathsf{Clos}(\Phi)\}$,
- For each $\gamma \in A$ and $(\Gamma, \xi, \Gamma') \in \Sigma$:
  1. if $\gamma = a_I$, then $\Delta(\gamma, (\Gamma, \xi, \Gamma')) = \Phi$,
  2. if $\gamma = \xi = \eta x.\psi$ then $\Delta(\gamma, (\Gamma, \xi, \Gamma')) = \{\eta x.\psi^*\}$,
  3. if $\gamma = \eta x.\psi^*$, then $\Delta(\gamma, (\Gamma, \xi, \Gamma')) = \{\gamma' \mid (\psi[x \backslash \eta x.\psi], \gamma') \in \mathsf{T}_{\Gamma,\xi,\Gamma'}\}$ and
  4. else $\Delta(\gamma, (\Gamma, \xi, \Gamma')) = \{\gamma' \mid (\gamma, \gamma') \in \mathsf{T}_{\Gamma,\xi,\Gamma'}\}$.
- For all states $\eta x.\psi^*$ let $\Omega_A(\eta x.\psi^*) = \Omega(\eta x.\psi)$. For all other states a let $\Omega_A(a) = \max_\Omega(\Phi)$ if $\max_\Omega(\Phi)$ is odd and $\Omega_A(a) = \max_\Omega(\Phi) + 1$ else.

Let $\alpha = (v_n)_{n \in \omega}$ be an infinite branch in an NW-proof π. We define $w(\alpha) \in \Sigma^\omega$ to be the infinite word $(\mathsf{S}(v_0), \mathsf{f}(v_0), \mathsf{S}(v_0))(\mathsf{S}(v_0), \mathsf{f}(v_0), \mathsf{S}(v_1))(\mathsf{S}(v_1), \mathsf{f}(v_1), \mathsf{S}(v_2))....$

Lemma 1. *Let α be an infinite branch in an NW proof. Then α carries a ν-trail iff $w(\alpha) \in \mathcal{L}(\mathbb{A})$.*

Combining Lemma 1 and Theorem 3 from Sect. 3 we get

Lemma 2. *Let π be an* NW *derivation. Then π is an* NW *proof iff for every infinite branch α in π it holds $w(\alpha) \in \mathcal{L}(\mathbb{A}^D)$.*

Lemma 3. *Let Γ be a sequent. Then* $\mathsf{NW} \vdash \Gamma$ *iff* $\mathsf{BT} \vdash \Gamma^\epsilon$.

Proof (Sketch). Let π be an NW proof of a sequent Γ. Inductively we translate every node v in π to a node v' plus some additional nodes, such that v' is labeled by the same sequent as v plus annotations. This can be achieved by replacing every rule in NW by its corresponding rule in BT and adding the rules Resolve and Compress whenever applicable. This yields a BT derivation ρ. It remains to show that every infinite branch $\alpha = (v_i)_{i\in\omega}$ in ρ is successful. Let $\hat{\alpha}$ be the corresponding infinite branch in π. Due to Lemma 2 it holds that $\hat{\alpha} \in \mathcal{L}(\mathbb{A}^D)$. Thus there is (k, s) such that s is in play at position k cofinitely often and $c_k(s)$ is green infinitely often and red only finitely often. As the annotations in α resemble the annotations in the run of $\mathbb{A}^D$ on $\hat{\alpha}$ it follows that there is some $N \in \omega$ such that (k, s) is preserved and progresses infinitely often on $\{v_i \mid i \geq N\}$.

Conversely let ρ be a BT proof of Γ^ϵ. We let π be the NW derivation defined from ρ by omitting the rules Resolve and Compress and reducing the other rules to the corresponding NW rules. We have to show that every infinite branch α in π is successful. Let $\alpha' = (v_i)_{i\in\omega}$ be the corresponding infinite branch in ρ. Because ρ is a BT proof there is N,(k, s) such that (k, s) is preserved and progresses infinitely often on $\{v_i \mid i \geq N\}$. Again the annotations in α' resemble the annotations in the run of $\mathbb{A}^D$ on α, thus (k, s) witnesses the acceptance of the run of $\mathcal{L}(\mathbb{A}^D)$ on α and Lemma 2 concludes the proof.

Theorem 4 (Soundness and Completeness). *Let Γ be a sequent. Then there is a* BT^∞*-proof of Γ^ϵ iff $\bigvee \Gamma$ is valid.*

Proof. This follows from Lemma 3 and Theorem 1.

4.3 Cyclic BT Proofs

As NW proofs can be assumed to be regular and annotations are added deterministically we can also assume BT^∞ proofs to be regular. A standard argument then transforms regular BT^∞ proofs into BT proofs and vice versa.

Lemma 4. *An annotated sequent is provable in* BT *iff it is provable in* BT^∞.

Theorem 5 (Soundness and Completeness). *Let Γ be a sequent. Then there is a* BT*-proof of Γ^ϵ iff $\bigvee \Gamma$ is valid..*

Remark 4. The number of distinct subtrees in a regular BT^∞ proof can be bounded by the number of distinct annotated sequents. This follows because the same statement holds for NW proofs [19] and because in the proof of Lemma 3 annotations and extra rules are added deterministically to sequents in NW proofs.

Let Φ be a sequent, $n = |\mathsf{Clos}(\Phi)|$ and $m = \max_\Omega(\Phi)$. There are at most $n^{\mathcal{O}(m\cdot n)}$ many distinct annotated sequents occurring in a proof of Φ, because

annotated sequents resemble macrostates in $\mathbb{A}^D$ and as seen in Remark 2 there are at most $n^{\mathcal{O}(m \cdot n)}$ distinct macrostates in $\mathbb{A}^D$.

Combining these two observations with the proof of Lemma 4 yields that the height of a BT proof of a sequent Φ can be bound by $n^{\mathcal{O}(m \cdot n)}$. This is the same complexity as in JS [13].

Remark 5. Given a BT derivation π, we can check if π is a BT proof in coNP. We can give the following algorithm in NP, that checks if π is not a BT proof: Choose non-deterministically a strongly connected subgraph S and check if there exists (k, s) that is preserved and progresses on S, the latter can be done in polynomial time. The complexity of proof checking can be compared to linear time in JS and PSPACE in NW. Note that, if we add a control to the BT proof system, the soundness condition boils down to checking paths between leafs and its companions. In that case proof checking could also be done in linear time.

5 Conclusions and Future Work

We hope that this paper contributes to the theory of non-wellfounded and cyclic proof systems by discussing applications of automata theory in the field. We have argued for the relevance of the notion of determinizing stream automata in the design of proof systems for the modal μ-calculus. More concretely, we have introduced a determinization construction based on binary trees and used this to obtain a new derivation system BT which is cyclic, cutfree, and sound and complete for the collection of valid $\mathcal{L}_\mu$-formulas. In the remainder of this concluding section we point out some directions for future research.

First of all, our approach is not restricted to the modal μ-calculus, but will apply to non-wellfounded and cyclic derivation systems for many other logics as well. For instance, in the proof systems LKID$^\omega$ [3] for first-order logic with inductive definitions, cyclic arithmetic CA [21] and similar systems the trace condition is of the form that on every infinite branch there is a term/variable which progresses infinitely often. This condition can be checked by a nondeterministic Büchi automaton and thus our method would yield an annotated proof system, where the annotations are binary strings, which label the terms/variables.

Second, in Remark 3 we discussed some relative advantages and disadvantages of the systems JS and BT. It would be interesting to either design a system that combines the advantages of both systems (i.e. sequents consisting of annotated formulas only as in BT, and a local condition for proof checking as in JS), or prove that such a system cannot exist.

Finally, it would be interesting (and in fact, it was one of the original aims of our work), to connect annotation-based sequent calculi such as JS and BT to Kozen's Hilbert-style proof system and to see whether a more structured automata-theoretic approach would yield an alternative proof of Walukiewicz' completeness result. Note that this was also the goal of Afshari & Leigh [2]; unfortunately, it was recently shown by the second author [14] that the system Clo, a key system in Afshari & Leigh's approach linking JS to Kozen's axiomatization, is in fact incomplete.

References

1. Afshari, B., Enqvist, S., Leigh, G.E.: Cyclic proofs for the first-order μ-calculus. Logic J. IGPL (2022). https://doi.org/10.1093/jigpal/jzac053
2. Afshari, B., Leigh, G.E.: Cut-free completeness for modal μ-calculus. In: Proceedings of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavík, Iceland. IEEE Press (2017)
3. Brotherston, J.: Sequent calculus proof systems for inductive definitions. Ph.D. thesis (2006). https://era.ed.ac.uk/handle/1842/1458
4. Calude, C., Jain, S., Khoussainov, B., Li, W., Stephan, F.: Deciding parity games in quasipolynomial time. In: Hatami, H., McKenzie, P., King, V. (eds.) Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, (STOC 2017), pp. 252–263 (2017)
5. Dekker, M., Kloibhofer, J., Marti, J., Venema, Y.: Proof systems for the modal μ-calculus obtained by determinizing automata (2023). https://doi.org/10.48550/arXiv.2307.06897
6. Doumane, A.: Constructive completeness for the linear-time μ-calculus. In: 2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), pp. 1–12 (2017). https://doi.org/10.1109/LICS.2017.8005075
7. Emerson, E., Jutla, C.: The complexity of tree automata and logics of programs. SIAM J. Comput. **29**(1), 132–158 (1999)
8. Fogarty, S., Kupferman, O., Vardi, M.Y., Wilke, T.: Profile trees for Büchi word automata, with application to determinization. Inf. Comput. **245**, 136–151 (2015)
9. Fogarty, S., Kupferman, O., Wilke, T., Vardi, M.: Unifying Büchi complementation constructions. Log. Methods Comput. Sci. **9**(1) (2013). https://doi.org/10.2168%2Flmcs-9%281%3A13%292013
10. Friedmann, O., Lange, M.: Deciding the unguarded modal μ-calculus. J. Appl. Non-Class. Logics **23**(4), 353–371 (2013). https://doi.org/10.1080/11663081.2013.861181
11. Janin, D., Walukiewicz, I.: Automata for the modal μ-calculus and related results. In: Wiedermann, J., Hájek, P. (eds.) MFCS 1995. LNCS, vol. 969, pp. 552–562. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-60246-1_160
12. Janin, D., Walukiewicz, I.: On the expressive completeness of the propositional μ-calculus with respect to monadic second order logic. In: Montanari, U., Sassone, V. (eds.) CONCUR 1996. LNCS, vol. 1119, pp. 263–277. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61604-7_60
13. Jungteerapanich, N.: Tableau systems for the modal μ-calculus. Ph.D. thesis, School of Informatics; The University of Edinburgh (2010). http://hdl.handle.net/1842/4208
14. Kloibhofer, J.: A note on the incompleteness of Afshari & Leigh's system Clo (2023). https://doi.org/10.48550/arXiv.2307.06846
15. Kozen, D.: Results on the propositional μ-calculus. Theoret. Comput. Sci. **27**, 333–354 (1983)
16. Löding, C., Pirogov, A.: Determinization of Büchi Automata: Unifying the Approaches of Safra and Muller-Schupp. Schloss Dagstuhl - Leibniz-Zentrum für Informatik GmbH, Wadern/Saarbruecken, Germany (2019). http://drops.dagstuhl.de/opus/volltexte/2019/10696/
17. Leigh, G.E., Wehr, D.: From GTC to reset: generating reset proof systems from cyclic proof systems. Technical report (2023). https://doi.org/10.48550/arXiv.2301.07544. http://arxiv.org/abs/2301.07544

18. Marti, J., Venema, Y.: A focus system for the alternation-free μ-calculus. In: Das, A., Negri, S. (eds.) TABLEAUX 2021. LNCS (LNAI), vol. 12842, pp. 371–388. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86059-2_22
19. Niwinski, D., Walukiewicz, I.: Games for the μ-Calculus. Theor. Comput. Sci. **163**(1&2), 99–116 (1996). https://doi.org/10.1016/0304-3975(95)00136-0
20. Safra, S.: On the complexity of ω-automata. In: Proceedings of the 29th Symposium on the Foundations of Computer Science, pp. 319–327. IEEE Computer Society Press (1988)
21. Simpson, A.: Cyclic arithmetic is equivalent to peano arithmetic. In: Esparza, J., Murawski, A.S. (eds.) FoSSaCS 2017. LNCS, vol. 10203, pp. 283–300. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54458-7_17
22. Sprenger, C., Dam, M.: On the structure of inductive reasoning: circular and tree-shaped proofs in the μcalculus. In: Gordon, A.D. (ed.) FoSSaCS 2003. LNCS, vol. 2620, pp. 425–440. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36576-1_27
23. Stirling, C.: A tableau proof system with names for modal μ-calculus. In: Voronkov, A., Korovina, M. (eds.) HOWARD-60. A Festschrift on the Occasion of Howard Barringer's 60th Birthday. EPiC Series in Computing, vol. 42, pp. 306–318. EasyChair (2014). https://doi.org/10.29007/lwqm
24. Studer, T.: On the proof theory of the modal μ-calculus. Studia Logica **89**(3), 343–363 (2008). https://doi.org/10.1007/s11225-008-9133-6
25. Walukiewicz, I.: Completeness of Kozen's axiomatisation of the propositional μ-calculus. Inf. Comput. **157**, 142–182 (2000)
26. Wilke, T.: Alternating tree automata, parity games, and modal μ-calculus. Bull. Belgian Math. Soc. **8**, 359–391 (2001)