The $\mu$-calculus is an expressive specification language in which modal logic is extended with fixpoint operators, subsuming many dynamic, temporal, and description logics. Formulas of $\mu$-calculus are classified according to their alternation depth, which is the maximal length of a chain of nested alternating least and greatest fixpoint operators. Alternation depth is the major factor in the complexity of $\mu$-calculus model-checking algorithms. A refined classification of $\mu$-calculus formulas distinguishes between formulas in which the outermost fixpoint operator in the nested chain is a least fixpoint operator ($\Sigma_i$ formulas, where $i$ is the alternation depth) and formulas where it is a greatest fixpoint operator ($\Pi_i$ formulas). The alternating-free $\mu$-calculus (AFMC) consists of $\mu$-calculus formulas with no alternation between least and greatest fixpoint operators. Thus, AFMC is a natural closure of both $\Sigma_i \cup \Pi_1$, which is contained in both $\Sigma_2$ and $\Pi_2$. In this work we show that $\Sigma_2 \cap \Pi_2 \equiv$ AFMC. In other words, if we can express a property both as a least fixpoint nested inside a greatest fixpoint and as a greatest fixpoint nested inside a least fixpoint, then we can express also with no alternation between greatest and least fixpoints. Our result refers to $\mu$-calculus over arbitrary Kripke structures. A similar result, for directed $\mu$-calculus formulas interpreted over trees with a fixed finite branching degree, follows from results by Arnold and Niwinski. Their proofs there cannot be easily extended to Kripke structures, and our extension involves symmetric nondeterministic Büchi tree automata, and new constructions for them.

1 Introduction

The $\mu$-calculus is an expressive specification language in which formulas are built from Boolean operators, existential (\(\diamond\)) and universal (\(\Box\)) next-time modalities, and least (\(\mu\)) and greatest (\(\nu\)) fixpoint operators [Koz83]. The discovery and use of symbolic model-checking methods [McM93] for verification of large systems has made the $\mu$-calculus important also from a practical point of view: symbolic model-checking tools proceed by computing fixpoint expressions over the model’s set of states. For example, to find the set of states from which a state satisfying some predicate \(p\) is reachable, the model checker starts with the set \(S\) of states in which \(p\) holds, and repeatedly add to \(S\) the set \(\diamond S\) of states that have a successor in \(S\). Formally, the model checker calculates the set of states that satisfy the $\mu$-calculus formula \(\mu y.p \lor \diamond y\).

Formulas of $\mu$-calculus are classified according to their alternation depth, which is the maximal length of a chain of nested alternating least and greatest fixpoint operators. From a practical point of view, the classification is important, as the alternation depth is the major factor in the complexity of $\mu$-calculus model-checking algorithms: the original algorithm for model checking a structure of size \(m\) with respect to a formula of length \(n\) and alternation depth \(d\) requires time \(O(nm)^d\) [EL86], and more sophisticated algorithms can do the job in time roughly \(O(nm)^{d+1}\) [Jur00]. From a theoretical point of view, the classification naturally raises questions about the expressive power of the classes. In particular, the question whether the expressiveness hierarchy for the $\mu$-calculus collapses (i.e., whether there is some \(d \geq 1\) such
that all $\mu$-calculus formulas can be translated to formulas of alternation depth $d$) has been answered to the negative [Bra98]. The alternation-depth hierarchy of $\mu$-calculus and the model-checking problem for the various classes in the hierarchy are strongly related to the index hierarchy in parity games and to the problem of deciding such games [Jur00].

A more refined classification of $\mu$-calculus formulas distinguishes between formulas in which the outermost fixpoint operator in the nested chain is a least fixpoint operator ($\Sigma_i$ formulas, where $i$ is the alternation depth) and formulas where it is a greatest fixpoint operator ($\Pi_i$ formulas). For example, the formula $\nu y.p \lor \Diamond y$ is a $\Sigma_1$ formula, as it has alternating depth 1 and its outermost fixpoint operator is $\nu$. Similarly, the formula $\mu y.\mu z.\Box (p \land y) \lor z$ is a $\Pi_2$ formula. By duality of the least and greatest fixpoint operators, the classes $\Pi_i$ and $\Sigma_i$ are complementary, in the sense that a formula $\psi$ is in $\Pi_i$ iff the formula $\neg \psi$ (in positive normal form, where negation is applied to atomic propositions only) is in $\Sigma_i$.

Some fragments of $\mu$-calculus are of special interest in computer science: Modal Logic (ML) consists of $\mu$-calculus formulas with no fixpoint operators (that is, ML = $\Sigma_0 \cup \Pi_0$). It is actually more correct to say that $\mu$-calculus is the extension of ML with fixpoint operators. Extending ML with fixpoint operators still retain some of its basic semantic properties, in particular the property of being invariant under bisimulation [Ben91]. The alternation-free $\mu$-calculus (AFMC) consists of $\mu$-calculus formulas with no alternation between least and greatest fixpoint operators. Thus, AFMC is a natural closure of $\Sigma_1 \cup \Pi_1$, which is contained in both $\Sigma_2$ and $\Pi_2$. AFMC subsumes the branching temporal logic CTL and the dynamic logic PDL [FL79].

Formulas of AFMC can be symbolically evaluated in time linear in the structure [CS91,KVW00]. While designers may prefer to use higher-level logics to specify properties, model-checking tools often proceed by evaluating the corresponding AFMC formulas [BRS99]. Finally, it is hard to produce an understandable formula with more than one alternation. Thus, $\Pi_2 \cup \Sigma_2$ subsumes almost all formulas one may wish to specify in practice. Formally, $\Pi_2 \cup \Sigma_2$ subsumes the branching temporal logic CTL*, and in fact, until [Bra98], the strictness of the expressiveness hierarchy of $\mu$-calculus was known only for $\Pi_i$ and $\Sigma_i$ with $i \leq 2$ [AN90]. Also, the symbolic evaluation of linear properties is reduced to calculating a $\Pi_2$ formula [VW86,EL85].

For several hierarchies in computer science, even strict ones, it is possible to show local coalescence, where membership in some class of the hierarchy and in its complementary class implies membership in a lower class. For example, RE $\cap$ co-RE = Rec describes coalescence at the bottom of the arithmetical hierarchy [Rog67]. On the other hand, the analogous coalescence for the polynomial hierarchy is not known; it is a major open question whether NP $\cap$ co-NP = P [GJ79]. In [KV01], we showed that the bottom levels of the $\mu$-calculus expressiveness hierarchy coalesce: $\Sigma_1 \cap \Pi_1 \equiv ML$. In other words, if we can express a property $\xi$ both as a least fixpoint and as a greatest fixpoint, then we can express $\xi$ without fixpoints. The proof uses the fact that $\mu$-calculus formulas in $\Sigma_1 \cap \Pi_1$ correspond to languages that are both safety and co-safety. Consequently, for every property $\xi \in \Sigma_1 \cap \Pi_1$, we can construct two nondeterministic looping tree automata $U$ and $U'$ such that $U$ and $U'$ accept exactly all the trees that satisfy $\xi$ and its complement, respectively (the fact that $U$ and $U'$ are looping means that they have trivial acceptance conditions – every infinite run is accepting). We showed in [KV01] how $U$ and $U'$ can be combined to a cycle-free automaton and then translated to an ML formula expressing $\xi$.

In this paper we show coalescence in higher classes of the hierarchy, namely $\Sigma_2 \cap \Pi_2 \equiv AFMC$. In other words, if we can specify a property $\xi$ both as a least fixpoint nested inside a greatest fixpoint and as a greatest fixpoint nested inside a least fixpoint, then we can express $\xi$ also with no alternation between greatest and least fixpoints. Unfortunately, the technique of [KV01] is too weak to be helpful here. Indeed, formulas in $\Pi_2$ cannot be expressed by looping automata. As we explain below, the known automata-theoretic characterizations of $\Sigma_2$ and $\Pi_2$, and their relation to AFMC, cannot help us either.

---

1 An exact definition of the classes $\Sigma_i$ and $\Pi_i$ refers to the scope of the fixpoint operators. As we discuss in Section 4, several different definitions are studied in the literature, and we follow here the definition of [Niw86].

4 The analogous complexity-theoretic result would be $\Sigma_2^p \cap \Pi_2^p = P^{NP}$, where $\Sigma_2^p$ and $\Pi_2^p$ form the second level of the polynomial hierarchy and $P^{NP}$ is the polynomial closure of NP [GJ79].
One such known characterization [Niw86,AN92] refers to the expressive power of the \( \mu \)-calculus over trees with fixed finite branching degrees. Over such trees, the existential next-time modality of the \( \mu \)-calculus can be parameterized with directions. A modality parameterized with direction \( d \) means that the corresponding existential requirement should be satisfied in the \( d \)-th child of the current state. For example, for a binary tree in which each node has a left child and a right child, the formula \( \Diamond_1 p \) means that the left child of the root satisfies \( p \), and the formula \( \mu y.p \lor \Diamond_2 y \) means that some node in the rightmost path of the tree satisfies \( p \). The ability of directed \( \mu \)-calculus to distinguish between the various children of a node makes it convenient to translate formulas to tree automata and vice versa. In particular, it is known that directed-\( \Pi_2 \) is as expressive as nondeterministic B"uchi tree automata [AN90,Kai95]. Our interest in this paper is in the expressive power of the \( \mu \)-calculus over arbitrary Kripke structures, possibly with an infinite branching degree, which means that we cannot restrict attention to trees of fixed branching degrees.

An automata-theoretic framework for \( \mu \)-calculus without directions is suggested in [JW95], by means of \( \mu \)-automata, which are essentially symmetric alternating tree automata in a certain normal form. A related approach, in which alternation is more explicit, is presented in [Wil99]. Alternation allows the automaton to send several requirements to the same child. Symmetry means that the automaton does not distinguish between the different children of a node, and it sends copies to child nodes only in either a universal or an existential manner. It also means that the automaton can handle trees with a variable and even infinite branching degree. Formulas of \( \mu \)-calculus in \( \Pi_1 \) and \( \Sigma_1 \) can be linearly translated to symmetric alternating parity/co-parity automata of index \( i \). While it is possible to translate \( \mu \)-calculus formulas to symmetric alternating automata, it is not immediately clear how such a translation can help in a translating of \( \Sigma_2 \cap \Pi_2 \) into the AFMC. By [AN92,KV99], formulas that are members of both directed-\( \Pi_2 \) and directed-\( \Sigma_2 \) can be translated to directed-AFMC. The proofs in [AN92,KV99] shows that given a formula \( \psi \in \Sigma_2 \cap \Pi_2 \), we can construct two nondeterministic B"uchi tree automata \( \Uu \) and \( \Uu' \), for \( \psi \) and \( \neg \psi \), and then combine the automata to a weak alternating automaton equivalent to \( \psi \). The combination of \( \Uu \) and \( \Uu' \), however, crucially depends on the fact that the automata are nondeterministic (rather than alternating) and the fact that the automata can refer to particular directions in the tree.

The key to the results in [KV01] and here is a development of a theory of symmetric nondeterministic tree automata. In [KV01], we defined symmetric nondeterministic \textit{looping} automata, and showed how to construct such automata for formulas in \( \Pi_1 \). In order to handle \( \Sigma_2 \) and \( \Pi_2 \), we define here symmetric nondeterministic B"uchi automata, and translate \( \Pi_2 \) formulas to such automata. From a technical point of view, symmetric nondeterministic tree automata are essentially symmetric alternating automata with transitions in disjunctive normal form. Our main contribution is the development of various constructions for symmetric nondeterministic tree automata and their application to the study of the expressive power of the \( \mu \)-calculus. Since removal of alternation in B"uchi automata should take into an account the acceptance condition of the automaton and keep track of the states visited in each path of the run tree, the symmetry of the automaton poses real technical challenges. We then extend the construction in [KV99] to symmetric automata and combine the symmetric nondeterministic B"uchi tree automata for \( \psi \) and \( \neg \psi \) to a symmetric weak alternating automaton for \( \psi \). Again, symmetry poses real technical challenges. (In fact, while the construction in [KV99] for the directed case is quadratic, here we end up with quadratically many states but exponentially many transitions.) Once we have a weak symmetric alternating automaton for \( \psi \), it is possible to generate from it an equivalent AFMC formula [KV98].

2 Preliminaries

For a set \( D \subseteq \mathbb{N} \) of directions, a \( D \)-tree is a nonempty set \( T \subseteq D^* \), where for every \( x \cdot d \in T \) with \( x \in D^* \) and \( d \in D \), we have \( x \in T \). The elements of \( T \) are called \textit{nodes}, and the empty word \( \epsilon \) is the \textit{root} of \( T \). For every \( x \in T \), the nodes \( x \cdot d \), for \( d \in D \), are the \textit{children} of \( x \). A node with no children is a \textit{leaf}. The \textit{degree} of a node \( x \) is the number of children \( x \) has. Note that the degree of \( x \) is bounded by \( |D| \). For
technical convenience, we assume that the set \( D \) is finite. A \( D \)-tree is leafless if it has no leaves. Note that a leafless tree is infinite. A path \( \pi \) of a tree \( T \) is a set \( \pi \subseteq T \) such that \( \varepsilon \in \pi \) and for every \( x \in \pi \), either \( x \) is a leaf or exactly one child of \( x \) is in \( \pi \). For two nodes \( x_1 \) and \( x_2 \) of \( T \), we say that \( x_1 \leq x_2 \) iff \( x_1 \) is a prefix of \( x_2 \); i.e., there exists \( z \in D^* \) such that \( x_2 = x_1 \cdot z \). We say that \( x_1 < x_2 \) iff \( x_1 \leq x_2 \) and \( x_1 \neq x_2 \).

A frontier of a leafless tree is a set \( E \subseteq T \) of nodes such that for every path \( \pi \subseteq T \), we have \( |\pi \cap E| = 1 \). For example, the set \( E = \{ 0, 100, 101, 11 \} \) is a frontier of the \( \{0,1\} \)-tree \( \{0,1\}^* \). For two frontiers \( E_1 \) and \( E_2 \), we say that \( E_1 \leq E_2 \) iff for every node \( x_2 \in E_2 \), there exists a node \( x_1 \in E_1 \) such that \( x_1 \leq x_2 \). We say that \( E_1 < E_2 \) iff for every node \( x_2 \in E_2 \), there exists a node \( x_1 \in E_1 \) such that \( x_1 < x_2 \). Note that while \( E_1 < E_2 \) implies that \( E_1 \leq E_2 \) and \( E_1 \neq E_2 \), the other direction does not necessarily hold. Given an alphabet \( \Sigma \), a \( \Sigma \)-labeled \( D \)-tree is a pair \( (T, V) \) where \( T \) is a \( D \)-tree and \( V : T \to \Sigma \) maps each node of \( T \) to a letter in \( \Sigma \). We extend \( V \) to paths in a straightforward way. For a \( \Sigma \)-labeled \( D \)-tree \( T \), and a set \( A \subseteq \Sigma \), we say that \( E \) is an A-frontier iff \( E \) is an frontier and for every node \( x \in E \), we have \( V(x) \in A \). We denote by \( \text{trees}(D, \Sigma) \) the set of all \( \Sigma \)-labeled \( D \)-trees, and by \( \text{trees}(\Sigma) \) the set of all \( \Sigma \)-labeled \( D \)-trees, for some \( D \). For a set \( T \subseteq \text{trees}(\Sigma) \), we denote by \( \text{comp}(T) \) the set of \( \Sigma \)-labeled trees that are not in \( T \); thus \( \text{comp}(T) = \text{trees}(\Sigma) \setminus T \).

Automata on infinite trees (tree automata, for short) run on leafless \( \Sigma \)-labeled trees. Alternating tree automata generalize nondeterministic tree automata and were first introduced in [MS87]. Symmetric alternating tree automata [ JW95, Wil99] are capable of reading trees with variable branching degrees. When a symmetric automaton reads a node of the input tree it sends copies to all successors of that node or to some successor. Formally, for a given set \( X \), let \( B^+(X) \) be the set of positive Boolean formulas over \( X \). For a set \( Y \subseteq X \) and a formula \( \theta \in B^+(X) \), we say that \( Y \) satisfies \( \theta \) iff assigning \( \text{true} \) to elements in \( Y \) and assigning \( \text{false} \) to elements in \( X \setminus Y \) satisfies \( \theta \). A symmetric alternating Büchi tree automaton (symmetric ABT, for short) is a tuple \( A = (\Sigma, Q, \delta, q_0, F) \) where \( \Sigma \) is the input alphabet, \( Q \) is a finite set of states, \( \delta : Q \times \Sigma \to B^+(\{\Box, \Diamond\} \times Q) \) is a transition function, \( q_0 \in Q \) is an initial state, and \( F \subseteq Q \) is a Büchi acceptance condition. Intuitively, an atom \( \langle \Box, q \rangle \) in \( \delta(q, \sigma) \) denotes a universal requirement to send a copy of the automaton in state \( q \) to all the children of the current node. An atom \( \langle \Diamond, q \rangle \) denotes an existential requirement to send a copy of the automaton in state \( q \) to some child of the current node. When, for instance, the automaton is in state \( q \), reads a node \( x \) with \( k \) children \( x \cdot 1, \ldots, x \cdot k \), and \( \delta(q, V(x)) = (\Box, q_1) \land (\Diamond, q_2) \lor (\Diamond, q_3) \land (\Diamond, q_4) \), it can either send \( k \) copies in state \( q_1 \) to the nodes \( x \cdot 1, \ldots, x \cdot k \) and send a copy in state \( q_2 \) to some node in \( x \cdot 1, \ldots, x \cdot k \) or send one copy in state \( q_3 \) to some node in \( x \cdot 1, \ldots, x \cdot k \) and send one copy in state \( q_4 \) to some node in \( x \cdot 1, \ldots, x \cdot k \). So, while nondeterministic tree automata send exactly one copy to each child, symmetric alternating automata can send several copies to the same child. On the other hand, symmetric alternating automata cannot distinguish between the different successors and can send copies to child nodes only in either a universal or an existential manner. Formally, a run of \( A \) on an input \( \Sigma \)-labeled \( D \)-tree \( (T, V) \), for some set \( D \) of directions, is an \( (D^* \times Q) \)-labeled \( \text{N} \)-tree \( (T_r, r) \) such that \( \varepsilon \in T_r \) and \( r(\varepsilon) = (\varepsilon, q_0) \), and for all \( y \in T_r \), with \( r(y) = (x, q) \) and \( \delta(q, V(x)) = \theta \), there is a (possibly empty) set \( S \subseteq \{\Box, \Diamond\} \times Q \), such that \( S \) satisfies \( \theta \), and for all \( (e, s) \in S \), the following hold: (1) If \( e = \Box \), then for each \( d \in D \), there is \( j \in \text{N} \) such that \( y \cdot j \in T_r \) and \( r(y \cdot j) = (x \cdot d, s) \). (2) If \( e = \Diamond \), then for some \( d \in D \), there is \( j \in \text{N} \) such that \( y \cdot j \in T_r \) and \( r(y \cdot j) = (x \cdot d, s) \). Note that if \( \theta = \text{true} \), then \( y \) need not have children. This is the reason why \( T_r \) may have leaves. Also, since there exists no set \( S \) as required for \( \theta = \text{false} \), we cannot have a run that takes a transition with \( \theta = \text{false} \). For a run \( (T_r, r) \) and an infinite path \( \pi \subseteq T_r \), we define \( \text{in} f(\pi) \) to be the set of states that are visited infinitely often in \( \pi \), thus \( q \in \text{in} f(\pi) \) if and only if there are infinitely many \( y \in \pi \) for which \( r(y) \in T \times \{q\} \). A run \( (T_r, r) \) is accepting if all its infinite paths satisfy the Büchi acceptance condition; thus \( \text{in} f(\pi) \cap F \neq \emptyset \). A tree \( (T, V) \) is accepted by \( A \) iff there exists an accepting run of \( A \) on

\footnote{As we detail in the proof of Theorem 6, due to the bounded-tree-model property for \( \mu \)-calculus, this technical assumption does not prevent us from proving our main result also for general structures with an infinite branching degree.}
\(\langle T, V \rangle\), in which case \(\langle T, V \rangle\) belongs to \(L(A)\). A tree \(\langle T, V \rangle\) is accepted by \(U\) iff there exists an accepting run of \(A\) on \(\langle T, V \rangle\), in which case \(\langle T, V \rangle\) belongs to the language, \(L(A)\), of \(A\).

The transition function of an ABT \(A\) induces a graph \(G_A = \langle Q, E \rangle\) where \(E(q, q')\) if there is \(\sigma \in \Sigma\) such that \((\sqcup, q')\) or \((\sqcap, q')\) appears in \(\delta(q, \sigma)\). An ABT is a weak alternating tree automaton (AWT, for short) if for each strongly connected component \(C \subseteq Q\) of \(G_A\), either \(C \subseteq F\) or \(C \cap F = \emptyset\) [MSS86]. Note that every infinite path of a run of an AWT ultimately gets “trapped” within some strongly connected component \(C\) of \(G_A\). The path then satisfies the acceptance condition if and only if \(C \subseteq F\).

The symmetry condition can also be applied to nondeterministic tree automata. In a symmetric nondeterministic Büchi tree automaton (symmetric NBT, for short) \(U = \langle \Sigma, Q, \delta, q_0, F \rangle\), the state space is \(Q = 2^S\) for some set \(S\) of micro-states, and the transition function \(\delta : Q \times \Sigma \to 2^{S \times 2^S}\) maps a state and a letter to sets of pairs \((U, E)\) of subsets of \(S\). The set \(U \subseteq S\) is the universal set and it describes the micro-states that should be members in all the child states. The set \(E \subseteq S\) is the existential set and it describes micro-states each of which has to be a member in at least one child state. Formally, given \(k \geq 1\), a \(k\)-tuple \(\langle S_1, \ldots, S_k \rangle\) is induced by \(\delta(q, \sigma)\) if there is \((U, E)\) in \(\delta(q, \sigma)\) such that for all \(1 \leq i \leq k\) we have \(U \subseteq S_i\), and for all \(s \in E\) there is \(1 \leq i \leq k\) such that \(s \in S_i\). Intuitively, when the automaton reads a node \(q\) labeled \(\sigma\) with \(k\) children, it proceeds from the state \(q\); it has to take two choices. First, the automaton chooses a pair \((U, E)\) \(\in \delta(q, \sigma)\). Then, it chooses a way to deliver \(E\) among the \(k\) children. Thus, we can describe the two choices of the automaton by a pair \((U, \{E_1, \ldots, E_k\})\), where \((U, \cup_{1 \leq i \leq k} E_i) \in \delta(q, \sigma)\). Note that \(E_i\) may be empty. We denote by \(\delta(q, \sigma)\) the set of such pairs. A run of \(U\) on an input tree \(\langle T, V \rangle\) is a \(Q\)-labeled tree \(\langle T, r \rangle\), such that \(r(\varepsilon) = q_0\), and for every \(x \in T\) with \(r(x) = q\), there exists \(\langle q_1, \ldots, q_k \rangle \in \delta_k(q, V(x))\) such that for all \(1 \leq i \leq k\), we have \(r(x \cdot i) = q_i\). Note that each node of the input tree corresponds to exactly one node in the run tree. A run \(\langle T, r \rangle\) is accepting if all its paths satisfy the Büchi acceptance condition. Thus, for all paths \(\pi\), we have \(\inf f(\pi) \cap F = \emptyset\), where \(q \in \inf f(\pi)\) if and only if there are infinitely many \(x \in \pi\) for which \(r(x) = q\). Equivalently, \(\langle T, r \rangle\) is accepting iff \((T, r)\) contains infinitely many \(F\)-frontiers \(G_0 < G_1 < \ldots\). For a state \(q \in Q\), let \(U^q\) be \(U\) with initial state \(q\). We say that a symmetric NBT is monotonic if for every two states \(q\) and \(p\) such that \(q \subseteq p\), we have that \(L(U^q) \subseteq L(U^p)\), and \(p \in F\) implies \(q \in F\). In other words, the smaller the state is, the easier it is to accept from it. Note that symmetric nondeterministic tree automata are essentially symmetric alternating automata with transitions in disjunctive normal form (DNF); if we write the transition functions in DNF, then each disjunct is a conjunction of universal and existential requirements, corresponding to a pair \((U, E)\).

3 From symmetric NBT and co-NBT to symmetric AWT

Let \(U = \langle \Sigma, D, Q, q_0, M, F \rangle\) and \(U' = \langle \Sigma, D, Q', q'_0, M', F' \rangle\) be two NBT, and let \(|Q| = |Q'| = m\). In [Rab70], Rabin studies the joint behavior of a run of \(U\) with a run of \(U'\). Recall that an accepting run of \(U\) contains infinitely many \(F\)-frontiers \(G_0 < G_1 < \ldots\), and an accepting run of \(U'\) contains infinitely many \(F'\)-frontiers \(G'_0 < G'_1 < \ldots\). It follows that for every labeled tree \(\langle T, V \rangle \in L(U) \cap L(U')\) and accepting runs \(\langle T, r \rangle\) and \(\langle T, r' \rangle\) of \(U\) and \(U'\) on \(\langle T, V \rangle\), the joint behavior of \(\langle T, r \rangle\) and \(\langle T, r' \rangle\) contains infinitely many frontiers \(E_i \subset T\), with \(E_i < E_{i+1}\), such that \(\langle T, r \rangle\) reaches an \(F\)-frontier and \(\langle T, r' \rangle\) reaches an \(F'\)-frontier between \(E_i\) and \(E_{i+1}\). Rabin shows that the existence of \(m\) such frontiers, in the joint behavior of some runs of \(U\) and \(U'\), is sufficient to imply that the intersection \(L(U) \cap L(U')\) is not empty. We now extend Rabin’s result to symmetric automata.

Assume that \(U\) and \(U'\) above are symmetric NBT. We say that a sequence \(E_0, \ldots, E_m\) of frontiers of \(T\) is a trap for \(U\) and \(U'\) iff \(E_0 = \{\varepsilon\}\) and there exists a tree \(\langle T, V \rangle\) and (not necessarily accepting) runs \(\langle T, r \rangle\) and \(\langle T, r' \rangle\) of \(U\) and \(U'\) on \(\langle T, V \rangle\), such that for every \(0 \leq i \leq m - 1\), we have that \(\langle T, r \rangle\) contains an \(F\)-frontier \(G_i\) such that \(E_i < G_i < E_{i+1}\), and \(\langle T, r' \rangle\) contains an \(F'\)-frontier \(G'_i\) such that \(E_i < G'_i < E_{i+1}\). We say that \(\langle T, r \rangle\) and \(\langle T, r' \rangle\) witness the trap for \(U\) and \(U'\).

**Theorem 1.** Consider two symmetric nondeterministic Büchi tree automata \(U\) and \(U'\). If there exists a trap for \(U\) and \(U'\), then \(L(U) \cap L(U')\) is not empty.
Proof. The proof follows the same line of reasoning as in [Rab70]. For a state \( q \in Q \), let \( \mathcal{U} \) be \( \mathcal{U} \) with initial state \( q \), and similarly for \( q' \in Q' \) and \( \mathcal{U}' \). We define a sequence of relations over \( Q \times Q' \). Let \( H_0 = Q \times Q' \). Then, \( \langle q, q' \rangle \in H_{i+1} \) iff \( \langle q, q' \rangle \in H_i \) and there is a nonempty \( \Sigma \)-labeled D-tree \( \langle T, V \rangle \), a frontier \( E \subseteq T \), and runs \( \langle T, r \rangle \) and \( \langle T, r' \rangle \) of \( \mathcal{U} \) and \( \mathcal{U}' \) on \( \langle T, V \rangle \), such that there is an \( F \)-frontier \( G < E \) and an \( F' \)-frontier \( G' < E \), such that for all \( x \in E \), we have \( \langle r(x), r'(x) \rangle \in H_i \). It is easy to see that \( H_0 \subseteq H_1 \subseteq H_2 \subseteq \ldots \) Also, if \( H_i = H_{i+1} \), then \( H_i = H_{i+k} \) for all \( k \geq 0 \). In particular, since \( |Q| \times |Q'| = m \), it must be that \( H_m = H_{m+k} \) for all \( k \geq 0 \). As in [Rab70], it can now be shown that \( \mathcal{L}(\mathcal{U}) \cap \mathcal{L}(\mathcal{U}') = \emptyset \) iff \( H_m(q_0, q'_0) \), and the result follows.

Theorem 1 is the key to the construction described in Theorem 2 below.

**Theorem 2.** Let \( \mathcal{U} \) and \( \mathcal{U}' \) be two symmetric monotonic NBT with \( \mathcal{L}(\mathcal{U}') = \text{comp}(\mathcal{L}(\mathcal{U})) \). There exists a symmetric AWT \( A \) such that \( \mathcal{L}(A) = \mathcal{L}(\mathcal{U}) \).

**Proof.** Let \( \mathcal{U} = \langle \Sigma, Q, q_0, M, F \rangle \) and \( \mathcal{U}' = \langle \Sigma, Q', q'_0, M', F' \rangle \), and let \( |Q| \times |Q'| = m \). Also, let \( S \) and \( S' \) be the micro-states of \( \mathcal{U} \) and \( \mathcal{U}' \), respectively, such that \( Q = 2^S \) and \( Q' = 2^{S'} \). We define the symmetric AWT \( A = \langle \Sigma, P, p_0, \delta, \alpha \rangle \) as follows.

- \( P = Q \times Q \times \{0, \ldots, 2m - 1\} \) and \( p_0 = \langle q_0, q'_0, 0 \rangle \). Intuitively, a copy of \( A \) that visits the state \( \langle q, q', i \rangle \) as it reads the node \( x \) of the input tree corresponds to runs \( r \) and \( r' \) of \( \mathcal{U} \) and \( \mathcal{U}' \) that visit the states \( q \) and \( q' \), respectively, as they read the node \( x \) of the input tree. Let \( \rho = y_0, y_1, \ldots, y_{|\rho|} \) be the path from \( \varepsilon \) to \( x \). Consider the joint behavior of \( r \) and \( r' \) on \( \rho \). We can represent this behavior by a sequence \( \tau_\rho = \langle t_0, t'_0, \langle t_1, t'_1 \rangle, \ldots, \langle t_{|\rho|}, t'_{|\rho|} \rangle \rangle \) of pairs in \( Q \times Q \) where \( t_j = r(y_j) \) and \( t'_j = r'(y_j) \). We say that a pair \( \langle t, t' \rangle \in Q \times Q \) is an \( F \)-pair iff \( t \in F \) and is an \( F' \)-pair iff \( t' \in F' \). We can partition the sequence \( \tau_\rho \) to blocks \( \beta_0, \beta_1, \ldots, \beta_k \) such that \( \beta_0 \) is the open block \( \beta_{k+1} \) whenever we reach the first \( F' \)-pair that is preceded by an \( F \)-pair in \( \beta_0 \). In other words, whenever we open a block, we first look for an \( F \)-pair, ignoring \( F' \)-pairiness. Once an \( F \)-pair is detected, we look for an \( F' \)-pair, ignoring \( F' \)-pairiness. Once an \( F' \)-pair is detected, we close the current block and open a new block. Note that a block may contain a single pair that is both an \( F \)-pair and an \( F' \)-pair. The third element of a state keeps track of the visits to blocks. When we visit \( \langle q, q', i \rangle \), the index of the last block in \( \tau_\rho \) is \( \lceil \frac{i}{2} \rceil \), and this block already contains an \( F \)-pair iff \( i \) is odd. We refer to \( i \) as the status of the state \( \langle q, q', i \rangle \). For a status \( i \in \{0, \ldots, 2m - 1\} \), let \( P_i = Q \times Q \times \{i\} \) be the set of states with status \( i \).

- In order to define the transition function \( \delta \), we first define a function \( \text{next} : P \to \{0, \ldots, 2m - 1\} \) that updates the status of states. For that, we first define the function \( \text{next}' : P \to \{0, \ldots, 2m \} \) as follows.

\[
\text{next}'(\langle q, q', i \rangle) = \begin{cases} 
    i & \text{if } (i \text{ is even and } q \not\in F) \text{ or } (i \text{ is odd and } q' \not\in F') \\
    i + 1 & \text{if } (i \text{ is even and } q \in F) \text{ and } q' \not\in F' \text{ or } (i \text{ is odd and } q \not\in F) \\
    i + 2 & \text{if } i \text{ is even and } q \in F \text{ and } q' \in F'.
\end{cases}
\]

Now, \( \text{next}(\langle q, q', i \rangle) = \min\{\text{next}'(\langle q, q', i \rangle), 2m - 1\} \).

Intuitively, \( \text{next} \) updates the status of states by recording and tracking of blocks. Recall that the status \( i \) indicates in which block we are and whether an \( F \)-pair in the current block has already been detected. The conditions for not changing \( i \) or for increasing it to \( i + 1 \) and \( i + 2 \) follow directly from the definition of the status. For example, the new status stays \( i \) if the current \( i \) is even and \( \langle q, q' \rangle \) is not an \( F \)-block, or if \( i \) is odd and \( \langle q, q' \rangle \) is not an \( F' \)-block. When \( i \) reaches or exceeds \( 2m - 1 \), we no longer increase it, even if \( q' \in F' \).

The automaton \( A \) proceeds as follows. Essentially, for every run \( \langle T, r \rangle \) of \( \mathcal{U} \), the automaton \( A \) guesses a run \( \langle T', r' \rangle \) of \( \mathcal{U}' \) such that for every path \( \rho \) of \( T \), the run \( \langle T, r \rangle \) visits \( F \) along \( \rho \) at least as many times as \( \langle T', r' \rangle \) visits \( F' \) along \( \rho \). Thus, when we record blocks along \( \rho \), we do not want to get stuck in an even status. Since \( \mathcal{L}(\mathcal{U}) \cap \mathcal{L}(\mathcal{U}') = \emptyset \), then, by Theorem 1, no run \( \langle T, r \rangle \) can witness \( \langle T', r' \rangle \) a trap for \( \mathcal{U} \) and \( \mathcal{U}' \). Consequently, recording of visits to \( F \) and \( F' \) along \( \rho \) can be completed once \( A \) detects that \( \tau_\rho \) contains \( m \) blocks as above.
Recall that \( Q = 2^S \) and \( Q' = 2^S' \). For a set \( E \subseteq S \), a partition of \( E \) is a set \( \{E_1, \ldots, E_k\} \) with \( E_i \subseteq E \) such that \( E = \bigcup_{1 \leq i < j \leq k} E_i \), and for all \( 1 \leq i \neq j \leq n \), we have \( E_i \cap E_j = \emptyset \). Let \( \text{par}(E) \) be the set of partitions of \( E \). Consider a set \( E' \subseteq S' \) and a partition \( \gamma' \in \text{par}(E') \). For a set \( E \subseteq S \), we say that a partition \( \eta \) of \( E \cup E' \) agrees with \( \gamma' \) if for all \( s' \) and \( s'' \) in \( E' \), we have that \( s' \) and \( s'' \) are in the same set in \( \eta \) iff they are in the same set in \( \gamma' \). Let \( \text{agree}(E, 
abla') \) be the set of partitions of \( E \cup E' \) that agree with \( \gamma' \). For example, if \( E = \{s_1\} \) and \( E' = \{s_2, s_3\} \), then the two possible partitions of \( E' \) are \( \gamma'_1 = \{s_2, s_3\} \) and \( \gamma'_2 = \{s_2\}, \{s_3\} \). Then, \( \text{agree}(E, \gamma'_1) \) contains the two partitions \( \{s_1, s_2, s_3\} \) and \( \{s_1\}, \{s_2, s_3\} \), and \( \text{agree}(E, \gamma'_2) \) contains the three partitions \( \{s_1, s_2\}, \{s_3\} \), \( \{s_1, s_3\}, \{s_2\} \), and \( \{s_1\}, \{s_2\}, \{s_3\} \).

Now, let \( p = (q, q', i) \) be a state in \( P \) such that \( M(q, \sigma) = \{(U_1, E_1), \ldots, (U_n, E_n)\} \) and \( M'(q', \sigma) = \{(U'_1, E'_1), \ldots, (U'_{n'}, E'_{n'})\} \). We distinguish between two cases.

- If \( i < 2m - 1 \) or \( q \notin F \), then

\[
\delta(p, \sigma) = \bigwedge_{1 \leq j' \leq n'} \bigwedge_{\eta \in \text{agree}(E, \gamma')} \left( \bigvee_{1 \leq j \leq n} \text{go}(j, j', \eta, \text{next}(p)) \right),
\]

where

\[
\text{go}(j, j', \eta, l) = \square(U_j, U'_j, l) \land \bigwedge_{X \in \eta} (U_j \cup (X \cap E_j), U'_j \cup (X \cap E'_j), l).
\]

That is, for every choice of \( U' \) for a \( 1 \leq j' \leq n' \) and for the way the existential requirements in \( E'_j \) are partitioned, there is a choice of \( U \) for a \( 1 \leq j \leq n \) and for the way the existential requirements in \( E_j \) are partitioned and combined with these in \( E'_j \) to a partition of \( E_j \cup E'_j \), such that the universal requirements in \( U_j \) and \( U'_j \) are sent to all directions, and existential requirements that are in the same set in the joint partition of \( E_j \cup E'_j \) are sent to the same direction. Note that the sets \( U_j \) and \( U'_j \) are sent along with the existential requirements. This guarantees that the states that are sent in the existential mode correspond to the states that \( U \) and \( U' \) visit, and not to subsets of such states.

- If \( i = 2m - 1 \) and \( q \in F \), then \( \delta(p, \sigma) = \text{true} \).

Note that \( \text{par}(E') \) is exponential in \( |E'| \), and the number of possible \( \eta \in \text{agree}(E, \gamma') \) is exponential in \( E \cup E' \). Thus, the size of \( \delta \) is exponential in the sizes of \( M \) and \( M' \).

- \( \alpha = Q \times Q' \times \{i : i \text{ is odd}\} \). Thus, \( \alpha \) makes sure that infinite paths of the run visits infinitely many states in which the status is odd, thus states in which we are in the second phase of blocks. moshe2.

The automaton \( A \) is indeed an AWT. Each status \( 1 \leq i \leq 2m - 1 \) induces a strongly connected component in the partition of the state space into sets. Clearly, each set \( P_i \) is either contained in \( \alpha \) or is disjoint from \( \alpha \). Note that, by the definition of \( \alpha \), a run is accepting iff no path of it gets trapped in a set of the form \( P_i \), for an even \( i \), namely a set in which \( A \) is waiting for a visit of \( U \) in a state in \( F \). The number of states of \( A \) is \( O(m^2) \). We prove that \( \mathcal{L}(U) = \mathcal{L}(A) \). We first prove that \( \mathcal{L}(U) \subseteq \mathcal{L}(A) \). Consider a \( D \)-tree \( (T, V) \). With every run \( \langle T, r \rangle \) of \( U \) on \( (T, V) \) we can associate a run \( (T_R, R) \) of \( A \) on \( (T, V) \).

Intuitively, the run \( (T_R, R) \) directs \( (T_R, R) \) in the nondeterminism in \( \delta \) (that is, the choices of \( 1 \leq j \leq n \) and \( \eta \in \text{agree}(E_j, \gamma') \)). Formally, recall that a run of an \( A \) on a \( D \)-tree \( (T, V) \) is a \( (T \times P) \)-labeled tree \( (T_R, R) \), where a node \( y \in T_R \) with \( R(y) = \langle x, p \rangle \) corresponds to a \( A \) that reads the node \( x \in T \) and visits the state \( p \). We define \( (T_R, R) \) as follows.

- \( \varepsilon \in T_R \) and \( R(\varepsilon) = (\varepsilon, (q_0, q_0, 0)). \)
- Consider a node \( y \in T_R \) with \( R(y) = (x, (q, q', i)) \). By the definition of \( (T_R, R) \) so far, we have \( r(x) = t \) for \( q \leq t \). Consider first the case that \( t = q \). Let \( \{x, 1, \ldots, x, k\} \) be the children of \( x \in T \), and let \( \{U_1, E_1, \ldots, E_k\} \in M_k(q, V(x)) \) describe the choice \( U \) makes when it proceeds from the node \( x \). Thus, for each \( 1 \leq z \leq k \), we have \( r(x, z) = U \cup E_z \). Set \( j = \text{next}(q, q', i) \). Consider the set

\[
Y = \bigcup_{(q', U', j') \in M'(q', V(x))} \{(1, (U, j'), (1, (U \cup E_1, U' \cup E'_1), j'), \ldots, (k, (U, j'), (k, (U \cup E_k, U' \cup E'_k), j'))\}. 
\]
By the definition of δ, the set Y satisfies δ((q, q', i), V(x)) 6. Let l = |M_k(q', V(x))|, and let (U^w, E^w_1, ..., E^w_m), for 1 ≤ w ≤ l, be the w-th pair in M_k(q', V(x)). For all 1 ≤ w ≤ l and 1 ≤ z ≤ k, we have \{y \in (2k(w-1)+z-1), y \in (2k(w-1)+z) \} \in T_R, with R(y) = (2k(w-1)+z-1) = (x \cdot z, \{U, U^w, j\}) and R(y) = (2k(w-1)+z) = (x \cdot z, \{U \cup E_z, U^w \cup E^w_1, j\}). Note that the invariant that for all y \in T_R \text{ with } R(y) = (x \cdot (q, q', i)), we have r(x) = t for q \subseteq t, is maintained. If fact, we know that all the nodes y \in T_R that correspond to copies of A that satisfy an existential requirement have q = t, and node y \in T_R that correspond to copies of A that satisfy a universal requirement have q \supset t if the run r sends no existential requirement to the corresponding direction.

Consider now the case where q \subset t. Since \mathcal{U} is monotonic, there is an accepting run \langle T^x, r^x \rangle of \mathcal{U} on the subtree of T with root x. We can proceed exactly as above, with \langle T^x, r^x \rangle instead of \langle T, r \rangle.

Consider a tree \langle T, V \rangle \in \mathcal{L}(\mathcal{U}). Let \langle T, r \rangle be an accepting run of \mathcal{U} on \langle T, V \rangle, and let \langle T_R, R \rangle be the run of A on \langle T, V \rangle induced by \langle T, r \rangle (and the “subtree runs”, like \langle T^x, r^x \rangle above). It can be shown that \langle T_R, R \rangle is a legal accepting run. Indeed, since \langle T, r \rangle and the subtree runs contains infinitely many F-frontiers, and since (by the definition of monotonic automaton) we do not lose visits to F when we switch to subset runs), no infinite paths of \langle T_R, R \rangle can get trapped in a set \mathcal{P}_i for an even i.

It is left to prove that \mathcal{L}(A) \subseteq \mathcal{L}(\mathcal{U}). For that, we prove that \mathcal{L}(A) \cap \mathcal{L}(\mathcal{U}) = \emptyset. Since \mathcal{L}(\mathcal{U}) = \text{comp}(\mathcal{L}(\mathcal{U}')), it follows that every tree that is accepted by A is also accepted by \mathcal{U}. Consider a tree \langle T, V \rangle. With each run \langle T_R, R \rangle of A on \langle T, V \rangle and run \langle T, r' \rangle of \mathcal{U} on \langle T, V \rangle, we associate a run \langle T, r' \rangle of \mathcal{U} on \langle T, V \rangle. Intuitively, \langle T, r \rangle makes the choices that \langle T_R, R \rangle has made in its copies that correspond to the run \langle T, r' \rangle. Formally, we define \langle T, r' \rangle as follows.

1. \text{r}(x) = q_0.
2. Consider a node x \in T with r(x) = q. Let \{x \cdot 1, ..., x \cdot k\} be the children of x in T, and let r'(x) = q'.

The run \langle T, r' \rangle selects a pair \langle U^x, (E_{x, 1}, ..., E_{x, k}) \rangle \in M_k(q', V(x)) that \mathcal{U} proceeds with when it reads the node x. Formally, for all 1 ≤ z ≤ k, we have r'(x \cdot z) = U^x \cup E_{x, z}. 7 By the definition of r(x) so far, the run \langle T_R, R \rangle contains a node y \in T_R with \text{R}(y) = (x \cdot (q, q', i)) for some status i. If \delta((q, q', i), V(x)) = \text{true}, we define the reminder of \langle T, r \rangle arbitrarily. Otherwise, let 1 ≤ j' ≤ n' and \gamma' \in \text{par(E')}. be such that \langle U^x, (E_{x, 1}, ..., E_{x, k}) \rangle corresponds to j' and \gamma'. By the definition of \delta, there are 1 ≤ j ≤ n and \eta \in \text{agree(E_{j, j}, \gamma') such that go(j, j', \eta, next((q, q', i))) is satisfied and R proceeds according to j and \eta. Thus, if \text{E_{j, 1}, ..., E_{j, k}} is the partition of E_j that corresponds to \eta, then T_R contains at least k nodes y \cdot c_z, for 1 ≤ z ≤ k, such that R(y \cdot c_z) = (x \cdot z, \{U_j \cup E_{j, z}, U^x \cup E_{x, z}, next((q, q', i))\}). For all 1 ≤ z ≤ k, we define r(x \cdot z) = U_j \cup E_{j, z}. Note that the invariant about the runs \langle T, r \rangle and \langle T_R, R \rangle is maintained. Note also that if E_{j, 1} \cup E_{j, k} = \emptyset, then the existence of a node y \cdot c_z as above is guaranteed from universal part of \delta, and if E_{j, 1} \cup E_{j, k} \neq \emptyset, its existence is guaranteed from the existential part (in which case it is crucial that we sent the universal requirements along with the existentials).

We can now prove that \mathcal{L}(A) \cap \mathcal{L}(\mathcal{U}) = \emptyset. Assume, by way of contradiction, that there exists a tree \langle T, V \rangle such that \langle T, V \rangle is accepted by both A and \mathcal{U}'. Let \langle T_R, R \rangle and \langle T', r' \rangle be the accepting runs of A and \mathcal{U}' on \langle T, V \rangle, respectively, and let \langle T, r \rangle be the run of \mathcal{U} on \langle T, V \rangle induced by \langle T_R, R \rangle and \langle T', r' \rangle. We claim that then, \langle T, r \rangle and \langle T', r' \rangle witness a trap for \mathcal{U} and \mathcal{U}'. Since, however, \mathcal{L}(\mathcal{U}) \cap \mathcal{L}(\mathcal{U}') = \emptyset, it follows from Theorem 1, that no such trap exists, and we reach a contradiction. To see that \langle T, r \rangle and \langle T', r' \rangle indeed witness a trap, define E_0 = \{\varepsilon\}, and define, for 0 ≤ i ≤ m - 1, the set E_{i+1} to contain exactly all nodes x for which there exists y \in T_R such that either R(y) = (x \cdot q, \langle r(x), r'(x), 2i + 1 \rangle) and r'(x) \in F' or R(y) = (x \cdot q, \langle r(x), r'(x), 2i \rangle) and r(x) \in F and r'(x) \in F'. That is, for every path p of

---

6 Note that \delta((q, q', i), V(x)) is a formula in \mathcal{B}^{+}(\{\varepsilon, \cup\} \times P), whereas Y \subseteq \{1, ..., k\} \times P, but the extension of the satisfaction relation to this setting is straightforward: an atom \langle \cup, p \rangle is satisfied in Y if there is 1 ≤ z ≤ k with (z, p) \in Y, and an atom \langle (\varepsilon, p) \rangle is satisfied in Y if for all 1 ≤ z ≤ k, we have (z, p) \in Y.

7 For a monotonic NBT, we assume that \mathcal{U} runs satisfy the requirements in transition function in an optimal way; thus when \mathcal{A} chooses to proceed with \langle U^x, (E_{x, 1}, ..., E_{x, k}) \rangle \in M_k(q', V(x)), it is indeed the case that r'(x \cdot z) = U^x \cup E_{x, z}. If r'(x \cdot z) \supset U^x \cup E_{x, z}, we can replace r' with a run for which the equation holds.
exists next) and Boolean connectives, the temporal operators version defined in [Niw86]:

\[ \exists r \lor \forall r \] occurrences of \( r \) being a combination of formulas in which there are at most \( m \).

In other words, to form \( \exists r \lor \forall r \), we substitute \( r \) into \( y \) by a formula \( \mu r \cdot \varphi(y, r) \), where \( y \) is a free variable.

We classify formulas to classes \( \Sigma_i \) and \( \Pi_i \) according to the nesting of fixpoint operators in them. Several versions to such a classification can be found in the literature [EL86,Niw86,Bra98]. We describe here the version defined in [Niw86]:

- A formula is in \( \Sigma_0 = \Pi_0 \) if it contains no fixpoint operators.
- A formula is in \( \Sigma_i+1 \) if it is one of the following \( \theta_i, \theta_i \land \theta_i', \theta_i \lor \theta_i', \Box \theta_i, \Diamond \theta_i, \mu \theta_i \cdot \varphi_i+1(y), \varphi_i+1(Y) \) where \( \theta_i \) and \( \theta_i' \) are \( \Sigma_i \) formulas, \( \varphi_i+1 \) and \( \varphi_i+1 \) are \( \Sigma_i+1 \) formulas, \( Y \subseteq APV, y \in Y \), and no free variable of \( \varphi_i+1 \) is in \( Y \). In other words, to form \( \Sigma_i+1 \), we take \( \Sigma_i \) and close under Boolean and modal operations, \( \mu \theta_i \cdot \varphi_i(y) \) for \( y \in \Sigma_i+1 \), and substitution of a free variable of \( \varphi_i \) in \( \Sigma_i+1 \) by a formula \( \varphi' \in \Sigma_i+1 \) provided that no free variable of \( \varphi' \) is captured by \( \varphi_i \).
- A formula is in \( \Pi_i+1 \) if it is one of the following \( \theta_i, \theta_i \land \theta_i', \theta_i \lor \theta_i', \Box \theta_i, \Diamond \theta_i, \nu y \cdot \psi_i+1(y), \psi_i+1(Y) \) where \( \theta_i \) and \( \theta_i' \) are \( \Pi_i \) formulas, \( \psi_i+1 \) and \( \psi_i+1 \) are \( \Pi_i+1 \) formulas, \( Y \subseteq APV, y \in Y \), and no free variable of \( \psi_i+1 \) is in \( Y \).

Note that the “substitution step” suggests that the formula \( \psi = \nu y \cdot (\Diamond(y \lor (\mu z.p \lor \Diamond z))) \) is in both \( \Pi_2 \) and \( \Sigma_2 \). To see that \( \psi \) is in \( \Sigma_2 \) (it is easy to see that \( \psi \in \Pi_2 \)), note that \( \mu z.p \lor \Diamond z \) is in \( \Sigma_1 \), and hence also in \( \Sigma_2 \). In addition, the formula \( \nu y \cdot (\Diamond(y \land x)) \), for \( x \in APV \), is in \( \Pi_1 \), and hence also in \( \Sigma_2 \). The formula \( \mu z.p \lor \Diamond z \) has no free variables. Then, we can substitute \( x \) by it, get \( \psi \), and stay in \( \Sigma_2 \). Note that for formulas that do not allow such a substitution, the formula \( \psi \) is not in \( \Sigma_2 \). Note also that \( \psi \) is neither in \( \Pi_1 \) nor \( \Sigma_1 \).

Finally, we say that a formula is in \( \Delta_i \) if it is one of the following \( \theta_i, \theta_i \land \theta_i', \theta_i \lor \theta_i', \Box \theta_i, \Diamond \theta_i, \theta(Y)[y \leftarrow \theta_i^'] \), where \( \theta_i \) and \( \theta_i' \) are \( \Delta_i \) formulas, \( Y \subseteq APV, y \in Y \), and no variable of \( \theta_i^' \) is in \( Y \). In other words, to form \( \Delta_i \), we take \( \Sigma_i \) and close under Boolean and modal operations, and under substitution that does not increase the alternation depth. Note that \( \Delta_0 \) is ML and \( \Delta_1 \) is AFMC.

Essentially, \( \Sigma_i \) contains all Boolean and modal combinations of formulas in which there are at most \( i-1 \) alternations of \( \mu \) and \( \nu \), with the external fixpoint being a \( \mu \). Similarly, \( \Pi_i \) contains all Boolean and modal combinations of formulas in which there are at most \( i \) alternations of \( \mu \) and \( \nu \), with the external fixpoint being a \( \nu \). A \( \mu \)-calculus formula is alternation free if, for all atomic propositional variables \( y \), there are no occurrences of \( \nu (\mu y) \) on any syntactic path from an occurrence of \( \mu y \) (\( \nu y \), respectively) to an occurrence of \( y \). For example, the formula \( \mu x.(p \lor \mu y.(x \lor EX y)) \) is alternation free (and is in \( \Sigma_1 \)) and the formula \( \nu x.(p \lor x) \lor EX y \) is not alternation free (and is in \( \Pi_2 \)). The alternation-free \( \mu \)-calculus is a subset of \( \mu \)-calculus containing only alternation-free formulas. The alternation-free \( \mu \)-calculus is a strict syntactic
Theorem 5. [KV98] A set $T \subseteq \text{trees}(\Sigma)$ can be expressed in AFMC iff $T$ can be recognized by a symmetric weak alternating automaton.

In [Kai95], Kaivola considered $\mu$-calculus formulas in which the $\Diamond$ modality is parameterized with directions and translates $\Pi_2$ formulas to NBT. In order to apply Theorem 2, we should translate $\Pi_2$ formulas to symmetric monotonic NBT. For that, we first use a known translation of alternating B"uchi word automata to equivalent nondeterministic B"uchi word automata [MH84]. Mostowski extended the translation to tree automata [Mos84], and we extend it further to symmetric tree automata. Since the nondeterministic automaton needs to keep track of the states visited in each path of the run tree of the alternating automaton, the symmetry of the automaton poses real technical challenges.

Theorem 5. Let $A$ be a symmetric alternating B"uchi tree automaton. There is a symmetric monotonic nondeterministic B"uchi tree automaton $A'$, with exponentially many states, such that $L(A') = L(A)$.

Proof. Let $A = \langle \Sigma, S, s_{in}, \delta, \alpha \rangle$. Then $A' = \langle \Sigma, Q, \{\{s_{in}, 2\}\}, \delta', \alpha' \rangle$, where

- $Q = 2^S \times \{1, 2\}$. For a state $q \in Q$, let $q[1] = \{s : (s, 1) \in q\}$ and $q[2] = \{s : (s, 2) \in q\}$. Intuitively, the automaton $A'$ guesses a run of $A$. At a given node $x$ of a run of $A'$, it keeps in its memory the set of all the states of $A$ that visit $x$ in the guessed run. As it reads the next input letter, it guesses the way in which an accepting run of $A$ proceeds from all of these states. This guess induces the states that the run of $A'$ visit in the children of $x$. In order to make sure that every infinite path visits states in $\alpha$ infinitely often, the states are tagged by 1 or 2. States tagged by 1 correspond to copies that have already visit $\alpha$, and states tagged by 2 correspond to copies that owe a visit to $\alpha$. When all the copies visit $\alpha$ (that is, all the states are tagged by 1), we change the tag of all states to 2.
- Given $S' \subseteq S$, $\sigma \in \Sigma$, and a pair $\langle U, E \rangle$ of subsets of $S$, we say that $\langle U, E \rangle$ covers $S'$ and $\sigma$ if the set $\{\Box s : s \in U\} \cup \{\Diamond s : s \in E\}$ satisfies $\bigwedge_{s' \in S'} \delta(s', \sigma).

Now, $\delta' : Q \times S \rightarrow 2^{Q \times Q}$ is defined, for all $q \in Q$ and $\sigma \in \Sigma$, as follows.

- If $q[2] \neq \emptyset$, then $\delta'(q, \sigma)$ contains all pairs $\langle U, E \rangle$ such that there is $\langle U_1, E_1 \rangle$ that covers $q[1]$ and $\sigma$, and there is $\langle U_2, E_2 \rangle$ that covers $q[2]$ and $\sigma$, and the following hold.
  * $U = \{\langle s, 1 \rangle : s \in U_1 \cup (U_2 \cap \alpha)\} \cup \{\langle s, 2 \rangle : s \in U_2 \setminus \alpha\}$.
  * $E = \{\langle s, 1 \rangle : s \in E_1 \cup (E_2 \cap \alpha)\} \cup \{\langle s, 2 \rangle : s \in E_2 \setminus \alpha\}$.
- If $q[2] = \emptyset$, then $\delta'(q, \sigma)$ contains all pairs $\langle U, E \rangle$ such that there is $\langle U_1, E_1 \rangle$ that covers $q[1]$ and $\sigma$ and the following hold.
  * $U = \{\langle s, 1 \rangle : s \in U_1 \cap \alpha\} \cup \{\langle s, 2 \rangle : s \in U_1 \setminus \alpha\}$.
  * $E = \{\langle s, 1 \rangle : s \in E_1 \cap \alpha\} \cup \{\langle s, 2 \rangle : s \in E_1 \setminus \alpha\}$.
- $\alpha' = \{q : q[2] = \emptyset\}$. Note that a sequence of states of $A$, which corresponds to the behavior of a copy of $A$, changes the tag of its states from 2 to 1 when the copy visits a state in $\alpha$. Also, once all the sequences change the tag of their states to 1, the attribution is changed back to 2. Thus, $\alpha'$ guarantees that all sequences visit $\alpha$ infinitely often.
It is easy to see that $A$ is monotonic. Indeed, if $q \subseteq q'$, then $q[1] \subseteq q'[1]$ and $q[2] \subseteq q'[2]$. Thus, if a pair $(U, E)$ covers $q^1[1]$ and $q^2$, then $(U, E)$ also covers $q'[1]$ and $q'$, and similarly for $q'[2]$ and $q[2]$. Hence, given an accepting run of $A'$, we can make it an accepting run of $A^*$ by changing the labels of the root from $(\varepsilon, q')$ to $(\varepsilon, q)$. In addition, if $q'[2]$ is empty, so is $q^2$.

**Remark 1.** A related approach for translating $\mu$-calculus formulas into symmetric automata is taken in [JW95] (see also [AN01]). First, $\mu$-calculus formulas are transformed into a disjunctive form. The removal of conjunctions described there is similar to the removal of universal branches in alternating tree automata (and indeed it involves the same determinization construction that is present in the automata-theoretic approach [MS87]). It is then shown that disjunctive $\mu$-calculus formulas correspond to $\mu$-automata. Our focus here is on the translation of $\Pi_2$ formulas to symmetric monotonic nondeterministic Büchi tree automata. It is possible to recast our proof in an extension of the framework of $\mu$-automata [Wal03], but we find our notion of symmetric nondeterministic automata more transparent.

**Theorem 6.** $\Pi_2 \cap \Sigma_2 \equiv \text{AFMC}$.  

**Proof.** Since AFMC is a syntactic fragment of $\Pi_2 \cap \Sigma_2$, one direction is trivial. Let $\xi$ be a property expressible in $\Pi_2 \cap \Sigma_2$. Given $\theta \in \Pi_2$ expressing $\xi$, we can construct, by Theorems 4 and 5, a symmetric monotonic NBT $U_0$ that accepts exactly all trees that satisfy $\theta$. Also, $\xi \in \Sigma_2$ implies that there is $\psi \in \Pi_2$ that is equivalent to $\neg \theta$, so we can also construct a symmetric monotonic NBT $U_0$ that accepts exactly all trees that do not satisfy $\theta$. Clearly, $L(U_0) = \text{comp}(L(U_0))$. Hence, by Theorem 2, there is a symmetric alternating weak automaton $A_\theta$ that is equivalent to $U_0$. By Theorem 3, the automaton $A_\theta$ can be translated to a formula $\varphi$ in AFMC such that a tree satisfies $\varphi$ if it is accepted by $U_0$ iff it is not accepted by $U_0$. We claim that $\varphi$ is logically equivalent to $\theta$ over arbitrary structures (in particular, structures with an infinite branching degree). To see this, assume, by way of contradiction, that $\varphi$ is not logically equivalent to $\theta$. Then, either $\theta \land \neg \varphi$ or $\varphi \land \psi$ is satisfiable in some general structure. But then, either $\theta \land \neg \varphi$ or $\varphi \land \psi$ is satisfiable by a tree model [SE84] of a finite branching degree, contradicting the fact that a tree satisfies $\varphi$ iff it is accepted by $U_0$ iff it is not accepted by $U_0$.

**Remark 2.** Since it is also known that the $\mu$-calculus has the finite-model property [KP84], it follows that Theorem 6 can also be relativized to finite Kripke structures.

## 5 Concluding Remarks

We showed that $\Sigma_2 \cap \Pi_2 \equiv \text{AFMC}$. In other words, if we can specify a property $\psi$ both as a least fixpoint nested inside a greatest fixpoint and as a greatest fixpoint nested inside a least fixpoint, we should be able to specify $\psi$ also with no alternation between greatest and least fixpoints. This offers an elegant characterization of alternation freedom. The key to our results is a development of a theory of symmetric nondeterministic Büchi tree automata. A technical outcome of this theory is that the blow-up of our construction, i.e., going from formulas in $\Sigma_2 \cap \Pi_2$ to equivalent formulas in AFMC is doubly exponential. It would be interesting to try to improve this complexity or to prove its optimality.

Combining our result here with the result in [KV01] ($\Sigma_1 \cap \Pi_1 \equiv \text{ML}$) suggests the possibility of a general coalescence result for the $\mu$-calculus hierarchy. Recall the definition of $\Delta_i$ as the closure of $\Sigma_i \cap \Pi_i$ under Boolean and modal operations and under alternation-preserving substitutions. Then we have that $\Sigma_i \cap \Pi_i \equiv \Delta_{i-1}$ for $i = 1, 2$. It is tempting to conjecture that this holds for all $i > 0$, in analogy for such coalescence for the quantifier alternation hierarchy of first-order logic (cf. [Add62]). As is shown, however, in [AS03], this is not the case for $i > 3$.

**Acknowledgements:** We are grateful to J.W. Addison for valuable discussions regarding the first-order quantifier-alternation hierarchy and to I. Walukiewicz for valuable discussions regarding $\mu$-automata.
References


