The topic of this paper is the determinization problem of $\omega$-automata under the transition-based Emerson-Lei acceptance (called TELA), which generalizes all standard acceptance conditions and is defined using positive Boolean formulas. Such automata can be determined by first constructing an equivalent generalized Büchi automaton (GBA), which is later determinized. The problem of constructing an equivalent GBA is considered in detail, and three new approaches of solving it are proposed. Furthermore, a new determinization construction is introduced which determines several GBA separately and combines them using a product construction. An experimental evaluation shows that the product approach is competitive when compared with state-of-the-art determinization procedures. The second part of the paper studies limit-determinization of TELA and we show that this can be done with a single-exponential blow-up, in contrast to the known double-exponential lower-bound for determinization. Finally, one version of the limit-determinization procedure yields good-for-MDP automata which can be used for quantitative probabilistic model checking.

Keywords Emerson-Lei automata · Determinization algorithms · Limit-deterministic automata · Good-for-MDP automata · Probabilistic model checking

1 Introduction

Many standard algorithms in the fields of verification and synthesis of reactive systems [15,37,41,44] are based on the theory of $\omega$-automata, which are automata accepting infinite words. For logical specification languages such as linear temporal logic (LTL), many verification systems, including SPIN [5] or PRISM [30], use logic-to-automata translations internally to verify a given system against the specification.

As classical automata-based solutions to reactive synthesis and probabilistic verification use deterministic automata [37,44], the question of whether and how $\omega$-automata can be determinized efficiently is a major research area [31,36,38,40,42]. The first single-exponential and asymptotically optimal determinization for Büchi automata was presented in [40]. It produces Rabin automata, whose acceptance condition is a generalization of the Büchi condition.

The high state-complexity of determinization and most logic-to-automata translations have raised the question of more succinct representations of $\omega$-automata. Using generalized acceptance conditions (e.g. generalized Büchi [16] or generalized Rabin [10,12]) and transition-based [22], rather than state-based, conditions are common techniques in this direction. An even more general approach has led to the HOA-format [1], which represents the acceptance condition as a positive Boolean formula over standard Büchi (Inf) and co-Büchi (Fin) conditions, also called Emerson-Lei conditions [20,41]. This standardized format has proven to be very useful: together with a vast body of work on heuristics and dedicated procedures it has led to practically
usable and mature tools and libraries such as SPOT [18] and OWL [29] which support a wide range of operations on \(\omega\)-automata.

While determinization algorithms have made large steps towards practical feasibility, another line of work has considered automata with restricted forms of nondeterminism, which, while being nondeterministic, nevertheless share some of the desired properties of deterministic automata. In this spirit, the classes of good-for-MDP [24] and good-for-games [28] automata can be used for quantitative probabilistic model checking of Markov decision processes [23, 43], while limit-deterministic Büchi automata can be used for qualitative model-checking [15]. Another important class in this context are unambiguous automata, which also share some good algorithmic properties of deterministic automata [13], and are useful for probabilistic verification of Markov chains [4]. Dedicated translations from LTL directly to (limit-) deterministic [21] and unambiguous [25] automata have been studied.

This paper considers the problems of determining and limit-determinizing transition based Emerson-Lei automata (TELA). In contrast to limit-determinization, the theoretical complexity of determinization is well understood (a tight, double-exponential, bound was given in [40, 41]). However, it has not been studied yet from a practical point of view, which is the aim of this paper. All presented algorithms have been implemented, and are openly accessible together with the experimental evaluation [27].

1.1 Contribution

Three new translations from TELA to GBA are proposed, and an example is given in which they perform exponentially better than state-of-the-art implementations. We demonstrate how they can be used to build novel determinization approaches for TELA. The determinization procedures are based on a product construction which combines multiple GBA. Our experiments show that this approach often outperforms approaches based on determinizing a single GBA. A simple adaptation of the product construction produces limit-deterministic TELA of single-exponential size (in contrast to the double-exponential worst-case complexity of full determinization). We show that deciding whether \(Pr^\text{max}_M(L(A)) > 0\) holds is NP-complete if the TELA \(A\) is limit-deterministic, and in P if the acceptance of \(A\) is fin-less (Proposition 5.3). Finally, a limit-determinization construction for TELA based on the breakpoint-construction is presented. We show that a version of this procedure yields good-for-MDP Büchi automata (Definition 5.4). Thereby \(Pr^\text{max}_M(L(A))\) is computable in single-exponential time for any MDP \(M\) and TELA \(A\) (Theorem 5.3). The experimental evaluation compares the performance of the different constructions using three benchmark sets.

This article is an extended version of the conference paper [26] which provides full proofs (including some technical lemmas which were omitted in the conference version) and further examples. The experiments are also extended to an additional benchmark set.

1.2 Related work

The upper-bound for determinization of TELA [40, 41] relies on a translation to GBA which first transforms the acceptance formula into disjunctive normal form (DNF). We refine this idea by discussing and comparing several ways of performing this transformation. In particular, we present different strategies to remove Fin-atoms from the acceptance formula, which is a necessary ingredient of the approach. The removal of Fin-atoms is also discussed in [17, Sect. 4.5], together with a transformation into Büchi automata which relies on computing the conjunctive normal form of the acceptance formula after removing Fin-atoms. Our fin-removal strategies build on the ideas of [17], but our transformation to GBA avoids the CNF conversion and therefore sometimes produces exponentially smaller automata. Furthermore, we extend it to produce limit-deterministic automata. Translations from LTL to TELA have been proposed in [9, 32, 35], and all of them use product constructions to combine automata for subformulas.

The emptiness-check for \(\omega\)-automata under different types of acceptance conditions has been studied in [2, 10, 12, 19], where [2] covers the general case of Emerson-Lei conditions and also considers qualitative probabilistic model checking using deterministic TELA. The generalized Rabin condition from [10, 12] is equivalent to the DNF that we use and it is a special case of the hyper-Rabin condition for which the emptiness problem is in P [11, 20]. Probabilistic model checking for deterministic automata under this condition is considered in [12], while [10] is concerned with standard emptiness while allowing nondeterminism. In contrast to the above works, we consider probabilistic model checking for nondeterministic automata under general Emerson-Lei conditions. A procedure to transform TELA into parity automata is presented in [39], but it does not guarantee the output to be deterministic.

The first limit-determinization procedure for nondeterministic Büchi automata was presented in [14, 15], and it relies on the idea of combining a copy of the automaton with its breakpoint automaton. This idea was also used in [7, 23]. We lift it to Emerson-Lei automata by using multiple different breakpoint automata, corresponding to different parts of the Emerson-Lei condition.
1.3 Outline

After the preliminaries in Section 2, we present the new translation approaches to construct GBA in Sect. 3. In Sect. 4, we show how to use them to determine TELA and introduce the determinization based on a product construction. The translations to limit-deterministic TELA and good-for-MDP automata are contained in Sect. 5. Finally, we compare the different approaches in our experimental evaluation in Sect. 6 before we conclude the paper in Sect. 7.

2 Preliminaries

The infinite words over a finite alphabet Σ are denoted by Σω. If w = w0w1w2... ∈ Σω, we define w[j..m] = wjwj+1... and w[jm] = wjwj+1...wm, for any j, m ∈ N such that j ≤ m.

2.1 Automata on infinite words

A transition-based Emerson-Lei automaton (TELA) A is a tuple (Q, Σ, δ, I, α), where Q is a finite set of states, Σ is a finite alphabet, δ ⊆ Q × Σ × Q is the transition relation, I ⊆ Q is the set of initial states and α is a symbolic acceptance condition over δ, defined by:

α := π | ff | Inf(T) | Fin(T) | (α ∨ α) | (α ∧ α),
with T ⊆ δ

If α is π, ff, Inf(T) or Fin(T), then it is called atomic. We denote by |α| the number of atomic conditions contained in α, where multiple occurrences of the same atomic condition are counted multiple times. Symbolic acceptance conditions describe sets of transitions T ⊆ δ. Their semantics is defined recursively as follows:

T ⊨ π T ⊨ Inf(T)′ iff T ∩ T′ ≠ ∅
T ⊭ ff T ⊨ Fin(T)′ iff T ∩ T′ = ∅
T ⊨ α1 ∨ α2 iff T ⊨ α1 or T ⊨ α2
T ⊨ α1 ∧ α2 iff T ⊨ α1 and T ⊨ α2

Two acceptance conditions α and β are δ-equivalent (α ≡δ β) if for all T ⊆ δ we have T ⊨ α ⇔ T ⊨ β. A run of A for an infinite word u = u0u1... ∈ Σω is an infinite sequence of transitions ρ = (q0, u0, q1)(q1, u1, q2)... ∈ δω that starts with an initial state q0 ∈ I. The set of transitions that appear infinitely often in ρ are denoted by Inf(ρ). A run ρ is accepting (ρ ⊨ α) iff Inf(ρ) ⊨ α. The language of A, denoted by L(A), is the set of all words for which there exists an accepting run of A. The sets of infinite words which are the language of some TELA are called ω-regular. A TELA A is deterministic if the set of initial states contains exactly one state and the transition relation is a function δ : Q × Σ → Q. It is complete, if for all (q, a) ∈ Q × Σ: δ ∩ {(q, a, q′) | q′ ∈ Q} ≠ ∅. A Büchi condition is an acceptance condition of the form Inf(T) and a generalized Büchi condition is a condition of the form \( \bigwedge_{1≤i≤k} \text{Fin}(T_i) \). We call the sets \( T_i \) appearing in a generalized Büchi condition its acceptance sets. Rabin (resp. Street) conditions are of the form \( \bigwedge_{1≤i≤k} (\text{Fin}(F_i) \land \text{Inf}(T_i)) \) (resp. \( \bigwedge_{1≤i≤k} (\text{Fin}(F_i) \lor \text{Inf}(T_i)) \)).

2.2 Probabilistic systems

A labeled Markov decision process (MDP) M is a tuple (S, S0, Act, P, Σ, L) where S is a finite set of states, s0 ∈ S is the initial state, Act is a finite set of actions, P : S × Act × S → [0, 1] defines the transition probabilities with \( \sum_{s′} P(s, a, s′) ∈ [0, 1] \) for all (s, a) ∈ S × Act and L : S → Σ is a labeling function of the states into a finite alphabet Σ. Action α ∈ Act is enabled in s if \( \sum_{a, s′} P(s, a, s′) = 1 \), and Act(s) = {α | α is enabled in s}.

A path of M is an infinite sequence s0a0s1a1... ∈ S × Actω such that P(s1, a1, s1+1) > 0 for all i ≥ 0. The set of all paths of M is denoted by Paths(M) and Pathsfin(M) denotes the finite paths. Given a path π = s0a0s1a1..., we let L(π) = L(s0)L(s1)... ∈ Σω.

A Markov chain is an MDP with |Act(s)| ≤ 1 for all states s. A scheduler of M is a function \( \hat{S} : (S × Act)^* → Act \) such that \( \hat{S}(s_0a_0S_1...s_n) ∈ Act(s_n) \). It induces a Markov chain M\( \vec{S} \) and thereby a probability measure over Paths(M). The probability of a set of paths Π starting in s0 under this measure is \( \Pr_M^{s0}(\Pi) = \sup_{\hat{S} ∈ \hat{S}_s} \Pr_M^{s0}(\{ |π | ∈ Paths(M) \land L(π) ∈ \Pi \}) \).

We present four different constructions to transform TELA into equivalent GBA. One of them is the construction that is implemented by SPOT, the other ones are new. First, we define some operations on TELA, which we combine later to define our constructions.

3 From TELA to generalized Büchi automata

3.1 Operations on Emerson-Lei automata

The first operator splits a TELA along the top-level disjuncts of its acceptance condition. Let A = (Q, Σ, δ, I, α) be a
TELA where $\alpha = \sqrt[1 \leq i \leq m]{\alpha_i}$ and the $\alpha_i$ are arbitrary acceptance conditions. We define split($A$) := ($A_1, \ldots, A_m$) with $A_i = (Q, \Sigma, \delta, I, \alpha_i)$ for $1 \leq i \leq m$, and split($A$)[i] := $A_i$.

**Lemma 3.1** If $A$ is a TELA, then

$$\mathcal{L}(A) = \bigcup_{1 \leq i \leq m} \mathcal{L}(\text{split}(A)[i]).$$

**Proof** “⊆”: Let $\rho$ be a run of $A$ for $w$ such that $\rho \models \alpha$. Then, it follows that $\rho \models \alpha_i$ for some $1 \leq i \leq m$. But then $\rho$ is also an accepting run of split($A$)[i] and hence $w \in \mathcal{L}(\text{split}(A)[i])$.

“⊇”: Let $\rho$ be an accepting run of split($A$)[i] for some $1 \leq i \leq m$. Then $\rho \models \alpha_i$, and hence $\rho \models \alpha$. It follows that $\rho$ is an accepting run of $A$. □

The analogous statement does not hold for conjunction and intersection (see Example 3.1). In other words, the intersection of languages of automata one gets by splitting along a top-level conjunction is not necessarily the language of the original automaton. This is because a conjunctive acceptance formula requires the existence of a run satisfying all conjuncts.

**Example 3.1** Consider the automaton $A$ with acceptance condition $\alpha$ as shown in Fig. 1, and observe that we have $\mathcal{L}(A) = \varnothing$. Splitting the acceptance condition of $A$ leads to the acceptance conditions $\alpha_1$ and $\alpha_2$. We call the corresponding automata that use these two acceptance conditions $A_1$ and $A_2$. Note, that $A_1$ and $A_2$ use the same transition system as $A$. The automata $A_1$ and $A_2$ both accept the word $a^m$, i.e. $\mathcal{L}(A_1) \cap \mathcal{L}(A_2) = \{a^m\}$. Hence, we have $\mathcal{L}(A) \neq \mathcal{L}(A_1) \cap \mathcal{L}(A_2)$.

We also need constructions to realize the union of a sequence of automata. This can either be done using the sum (i.e. disjoint union) or the disjunctive product of the state spaces. We define a general sum (simply called sum) operation and one that preserves GBA acceptance (called GBA-specific sum). The disjunctive product construction for TELA is mentioned in [17] and similar constructions are used in [32,35]. While the sum operations yield smaller automata in general, only the product construction preserves determinism.

**Definition 3.1** Let $A_i = (Q_i, \Sigma, \delta_i, I_i, \alpha_i)$, with $i \in \{0, 1\}$, be two complete TELA with disjoint state-spaces. The sum $A_0 \oplus A_1$ is defined as follows:

$$A_0 \oplus A_1 = (Q_0 \cup Q_1, \Sigma, \delta_0 \cup \delta_1, I_0 \cup I_1, \alpha_0 \oplus \alpha_1)$$

with $\alpha_0 \oplus \alpha_1 = (\alpha_0 \land \inf(\delta_0)) \lor (\alpha_1 \land \inf(\delta_1))$.

If $\alpha_i = \inf(T_i^0) \land \ldots \land \inf(T_i^k)$, with $i \in \{0, 1\}$, (i.e. both automata are GBA), then we can use the GBA-specific sum:

$$A_0 \oplus_{\text{GBA}} A_1 = (Q_0 \cup Q_1, \Sigma, \delta_0 \cup \delta_1, I_0 \cup I_1, \alpha_{\text{GBA}})$$

with $\alpha_{\text{GBA}} = (\inf(T_0^0 \cup T_1^0) \land \ldots \land \inf(T_0^k \cup T_1^k))$.

The disjunctive product is defined as follows:

$$A_0 \otimes A_1 = (Q_0 \times Q_1, \Sigma, \delta_0 \times \delta_1, I_0 \times I_1, (\uparrow(\alpha_0) \lor \uparrow(\alpha_1)))$$

with

$$\delta_0 = \{(q_0, q_1), a, (q_0', q_1') \mid (q_0, a, q_0') \in \delta_0, (q_1, a, q_1') \in \delta_1\}$$

and $\uparrow(\alpha_i)$ is constructed by replacing every occurring set of transitions $T$ in $\alpha_i$ by

$$\{(q_0, q_1), u, (q_0', q_1') \in \delta_0 \mid (q_i, u, q_i') \in T\}.$$

**Proposition 3.1** Let $A_i = (Q_i, \Sigma, \delta_i, I_i, \alpha_i)$, with $i \in \{0, 1\}$, be two complete TELA with disjoint state-spaces. The following statements hold:

1. $\mathcal{L}(A_0) \cup \mathcal{L}(A_1) = \mathcal{L}(A_0 \oplus A_1)$
2. Assume that $\alpha_i = \inf(T_i^0) \land \ldots \land \inf(T_i^k)$, for $i \in \{0, 1\}$. Then we have: $\mathcal{L}(A_0) \cup \mathcal{L}(A_1) = \mathcal{L}(A_0 \oplus_{\text{GBA}} A_1)$
3. $\mathcal{L}(A_0) \cup \mathcal{L}(A_1) = \mathcal{L}(A_0 \otimes A_1)$

**Proof** 1.) This is clear as any accepting run of $A_i$ (for $i \in \{0, 1\}$) can be mapped directly to an accepting run of $A_0 \oplus A_1$, and an accepting run $A_0 \oplus A_1$ corresponds to an accepting run of one of the automata $A_0, A_1$.

2.) “⊆”: Let $\rho$ be an accepting run of $A_0$ (w.l.o.g.). Then $\inf(\rho) \cap T_i^0 \neq \varnothing$ holds for all $1 \leq j \leq k$, and hence also $\inf(\rho) \cap (T_0^0 \cup T_1^0) \neq \varnothing$. But then $\rho$ is also an accepting run of $A_0 \oplus_{\text{GBA}} A_1$.

“⊇”: Let $\rho$ be an accepting run of $A_0 \oplus_{\text{GBA}} A_1$. Then $\inf(\rho) \cap (T_0^0 \cup T_1^0)$ holds for all $1 \leq j \leq k$. Recall that we have $\delta_0 \cap \delta_1 = \varnothing$. We have either $\inf(\rho) \subseteq \delta_0$ or $\inf(\rho) \subseteq \delta_1$ and $\delta_j \subseteq \delta_i$ for all $1 \leq j \leq k$ and $i \in \{0, 1\}$. As a consequence, there exists $i \in \{0, 1\}$ such that $\inf(\rho) \cap T_j^i$ for all $1 \leq j \leq k$. But then $\rho$ is an accepting run of $A_i$.

3.) “⊆”: Let $\rho = q_0 q_1 \ldots$ be an accepting run of $A_0$ (w.l.o.g.) for $w$. As $A_1$ is complete, we find a run $\rho^*$ of $A_0 \otimes A_1$ for $w$ such that $\rho^* = (q_0, q_0', q_1, q_1', \ldots)$, where $q_0 q_1' \ldots$ is a run of $A_1$ for $w$. As $\rho \models \alpha_0$, it follows that $\rho^* \models \uparrow(\alpha_0)$ and hence $\rho^*$ is an accepting run of $A_0 \otimes A_1$. 

\[\text{Fig. 1} \quad \text{An example showing that the analogue of Lemma 3.1 for conjunction and intersection is not correct.}\]
Definition 3.2 (DNF for TELA) Let $\alpha = \text{Fin}(0) \lor \text{Fin}(1)$.

Fig. 2 An example showing why the conditions $\text{Inf}(\delta_0)$ and $\text{Inf}(\delta_1)$ are added to the acceptance condition of $A_0 \oplus A_1$.

“$\supseteq$” Let $\rho^* = (q_0, q_0'(q_1, q_1') \ldots$ be an accepting run of $A_0 \otimes A_1$ for $w$ and assume, w.l.o.g., that $\rho^* \models (\alpha_0)$. Then $q_0q_1 \ldots$, as a run for $w$, models $\alpha_0$ and hence it is an accepting run of $A_0$.

Note, that the acceptance $\alpha_{GBA}$ contains the same number of atomic acceptance conditions as $A_0$ and $A_1$, i.e., the acceptance condition does not increase when $\oplus_{GBA}$ is used to combine two GBA.

The additional $\text{Inf}(\delta_0)$ and $\text{Inf}(\delta_1)$ atoms in the acceptance condition of $A_0 \oplus A_1$ are essential (see Example 3.2).

Example 3.2 Consider the TELA $A_0$ and $A_1$ in Fig. 2 and Fig. 2, which both accept the empty language. If we unify the state spaces and disjunct the acceptance conditions we get the TELA in Fig. 2. However, this automaton accepts the language $\alpha'' \neq L(A_0) \cup L(A_1)$.

We can apply the GBA-specific sum to any two GBA by adding $\text{Inf}(\delta_i)$ atoms until the acceptance conditions are of equal length. Many of our constructions will require the acceptance condition of the TELA to be in DNF. We will use the following normal form throughout the paper (also called generalized Rabin in [10,12]).

Definition 3.2 (DNF for TELA) Let $A = (Q, \Sigma, \delta, I, \alpha)$ be a TELA. We say that $A$ is in DNF if $\alpha$ is of the form $\alpha = \bigvee_{1 \leq i \leq m} \alpha_i$, with $\alpha_i = \text{Fin}(T_0^i) \land \bigwedge_{1 \leq j \leq k_i} \text{Inf}(T_j^i)$ and such that all $k_i \geq 1$.

A single Fin atom in each disjunct is enough because $\text{Fin}(T_1) \land \text{Fin}(T_2) \equiv \text{Fin}(T_1 \cup T_2)$ holds for all $T_1, T_2, \delta$. Taking $k_i \geq 1$ is also no restriction, as we can add “$\land \text{Inf}(\delta)$” to any disjunct. Using standard Boolean operations one can transform a TELA with acceptance $\beta$ into DNF by just translating the acceptance formula into a formula $\alpha$ of the above form, with $|\alpha| \leq 2|\beta|$ and each disjunct contains at most $|\beta|$ atomic acceptance conditions.

Fig. 3 Example of applying removeFin and removeFinGBA (Definition 3.3) to the automaton in (a). The result is the automaton in (b) with acceptance $\alpha'$ (removeFin), respectively $\alpha''$ (removeFinGBA).

3.1.1 Fin-less acceptance

To transform a TELA in DNF (see Definition 3.2) into an equivalent one without Fin-atoms we use the idea of [10,17]: a main copy of $A$ is connected to one additional copy for each disjunct $\alpha_i$ of the acceptance condition, in which transitions from $T_0^i$ are removed. The acceptance condition ensures that every accepting run leaves the main copy eventually. Figure 3 shows an example.

Definition 3.3 Let $F_i = (Q_i, \Sigma, \delta_i, I_i, \phi_i)$, defined using $Q_i = \{q^{(i)} \mid q \in Q\}$, $\delta_i = \{(q^{(i)}, a, q^{(i)'}), (q, a, q') \in \delta \land T_0^i \}$ and $\phi_i = \bigwedge_{1 \leq j \leq k_i} \text{Inf}(U_j^i)$, where $U_j^i = \{(q^{(i)}, a, q^{(i)'}) \mid (q, a, q') \in (T_j^i \setminus T_0^i)\}$. Let

$\text{removeFin}(A) = (Q', \Sigma, \delta', I, \alpha')$ and $\text{removeFinGBA}(A) = (Q', \Sigma, \delta', I, \alpha'')$ where

- $Q' = Q \cup \bigcup_{1 \leq i \leq m} Q_i$
- $\delta' = \delta \cup \bigcup_{1 \leq i \leq m} (\delta_i \cup \{(q, a, q^{(i)}) \mid (q, a, q') \in \delta\})$
- $\alpha' = \bigvee_{1 \leq i \leq m} \phi_i$
• \( \alpha'' = \bigwedge_{1 \leq j \leq k} \text{Inf}(U^1_j \cup \ldots \cup U^m_j) \), with \( k = \max_i k_i \) and \( U^j_i = \delta_i \) if \( k_i < j \leq k \).

The construction \text{removeFin}(A)\) already appears in [10, 17]. The automaton \text{removeFin}_{\text{GBA}}(A)\) uses the same transition system but a different acceptance condition.

**Lemma 3.2** If \( A \) is a TELA:
\[
\mathcal{L}(A) = \mathcal{L}(\text{removeFin}(A)) = \mathcal{L}(\text{removeFin}_{\text{GBA}}(A)).
\]

**Proof** We first prove that \( \mathcal{L}(A) = \mathcal{L}(\text{removeFin}(A)) \).

"\( \subseteq \)" : Let \( \rho \) be an accepting run of \( A \) for \( u \), and assume that \( \rho \models \text{Fin}(T_0^i) \land \bigwedge_{1 \leq j \leq k} \text{Inf}(T^j_i) \) (the \( i \)-th disjunct of acceptance condition \( \alpha_i \), which is in DNF). It follows that there exists a position \( K \) after which \( \rho \) sees no transitions in \( T_0^i \). We construct an accepting run \( \rho' \) of \text{removeFin}(A)\) for \( u \): for the first \( K \) positions, it copies \( \rho \) in the main copy \( Q \). Then, it moves to \( Q_i \), and continues to simulate the moves of \( \rho \). From that fact that \( \rho \) sees infinitely many transitions in each set \( T^j_i \), it follows that \( \rho' \) models \( \phi_i \) and hence is accepting.

"\( \supseteq \)" : Let \( \rho' \) be an accepting run of \text{removeFin}(A)\) for \( u \), i.e. \( \rho' \models \phi_i \) for some \( 1 \leq i \leq m \). It follows that \( \rho' \) eventually moves to \( Q_i \), because \( \phi_i \) contains at least one \( \text{Inf} \)-atom whose all transitions are fully included in \( Q_i \). As no other copy is reachable from \( Q_i \), \( \rho' \) stays in \( Q_i \) thereafter. Recall that copy \( Q_i \) does not contain any transition in \( T_0^i \). Projecting the part of \( \rho' \) which is in \( Q_i \) onto the corresponding transitions in \( Q \) yields a run \( \rho \) for \( u \) in \( A \). From the definition of \( \phi_i \) and the fact that \( \rho' \) models \( \phi_i \) it follows directly that \( \rho \) models \( \alpha_i \), and hence \( \rho \) is accepting.

We show \( \mathcal{L}(\text{removeFin}(A)) = \mathcal{L}(\text{removeFin}_{\text{GBA}}(A)) \) by proving that for all runs \( \rho \) in \((\delta')^\omega\) we have
\[
\rho \models \alpha' \iff \rho \models \alpha''.
\]

First, assume that \( \rho \models \alpha' \), which implies that there exists an \( 1 \leq i \leq m \) such that \( \rho \models \text{Inf}(U^i_j) \) for all \( 1 \leq j \leq k_i \). It follows that \( \inf(\rho) \subseteq \delta_i \) and hence also \( \rho \models \text{Inf}(U^i_j) \) for all \( k_i < j \leq k \). But then clearly \( \rho \models \text{Inf}(U^1_j \cup \ldots \cup U^m_j) \) for all \( 1 \leq j \leq k \) and hence \( \rho \models \alpha'' \).

Now assume that \( \rho \models \alpha'' \). Then, in particular we have \( \rho \models \text{Inf}(U^1_j \cup \ldots \cup U^m_j) \), which implies that \( \rho \models \text{Inf}(U^i_j) \) for some \( 1 \leq i \leq m \). By construction of \text{removeFin}(A)\) it follows that \( \inf(\rho) \subseteq \delta_i \) and \( \inf(\rho) \cap \delta_i = \emptyset \), for \( i' \neq i \). As a consequence we have \( \inf(\rho) \cap \text{Inf}(U^i_j) = \emptyset \) for all \( 1 \leq j \leq k_i \) and \( i' \neq i \). The only possibility for \( \rho \) to satisfy all conjuncts in \( \alpha'' \) is to satisfy \( \inf(\rho) \cap \text{Inf}(U^j_i) \neq \emptyset \) for all \( 1 \leq j \leq k \). But then it follows that \( \rho \models \phi_i \) and hence \( \rho \models \alpha' \). \( \square \)

Observe that, in contrast to \text{removeFin}, \text{removeFin}_{\text{GBA}}\) always produces GBA. Both constructions consist of \( m + 1 \) copies of \( A \) (with \text{Fin}-transitions removed). Each disjunct of the acceptance condition of \text{removeFin}(A)\) contains at most as many atomic conditions than the corresponding disjunct of the acceptance condition of \( A \).

### 3.2 Construction of generalized Büchi automata

#### 3.2.1 Construction of spot

The transformation from TELA to GBA from [17] is implemented in SPOT[18]. It transforms the automaton into DNF and then applies (an optimized version of) \text{removeFin}. The resulting fin-less acceptance condition is translated into conjunctive normal form (CNF). As \( \text{Inf}(T_1) \lor \text{Inf}(T_2) \equiv_4 \text{Inf}(T_1 \cup T_2) \) holds for all \( \delta \), one can rewrite any fin-less condition in CNF into a conjunction of \text{Inf}-atoms, which is a generalized Büchi condition. When starting with a TELA \( B \) with acceptance \( \beta \) and \( N \) states, one gets a GBA with \( N \cdot 2^{O(|\beta|)} \) states and \( 2^{O(|\beta|)} \) acceptance sets, as the fin-removal may introduce exponentially (in \(|\beta|\)) many copies, and the CNF may also be exponential in \(|\beta|\).

Transforming a fin-less automaton into a GBA by computing the CNF has the advantage of only changing the acceptance condition, and in some cases it produces simple conditions directly. For example, SPOT’s TELA to GBA construction transforms a Rabin into a Büchi automaton, and a Streett automaton with \( m \) acceptance pairs into a GBA with \( m \) accepting sets. However, computing the CNF may also incur an exponential blow-up (Fig. 4 shows such an example).

#### 3.2.2 Copy-based approaches

We now describe three approaches (\text{remFin} \rightarrow \text{splitF}, \text{splitF} \rightarrow \text{remFin} \rightarrow \text{rewriteF})\), which construct GBA with at most \(|\beta| \) acceptance sets. On the other hand, they generally produce automata with more states than the approach that is implemented in SPOT. They are based on [41] which first translates copies of \( A \) (corresponding to the disjuncts of the acceptance condition) to GBA, and then takes their sum. However, it is not specified in [41] how exactly \text{Fin}-atoms should be removed (they were concerned only with the
We evaluate the effect of the translations to GBA introduced in the previous chapter in the context of determinization in 4.1 Determinization via single GBA

The standard way of determinizing TELA is to first construct a GBA, which is then determinized. Dedicated determinization procedures for GBA with \( N \) states and \( K \) acceptance sets produce deterministic Rabin automata with \( 2^{O(N \log N + \log K)} \) states [42]. For a TELA \( B \) with \( n \) states and acceptance \( \beta \), the above translations yield GBA with \( N = n 2^{O(\beta)} \) and \( K = 2^{O(\beta)} \) (Spot’s construction) or \( N = n 2^{O(\beta)} \) and \( K = O(|\beta|) \) (copy-based approaches). We evaluate the effect of the translations to GBA introduced in the previous chapter in the context of determinization in Sect. 6.

4.2 Determinization via a product construction

Another way to determinize a TELA \( \mathcal{A} \) in DNF is to determinize the automata \( \text{split}(\mathcal{A})[i] \) one by one and then combining them with the disjunctive product construction of Definition 3.1:

\[
\bigotimes_{1 \leq i \leq m} \text{split}(\mathcal{A})[i] \rightarrow \text{det}(\text{removeFin}(\mathcal{A}))
\]

where “det” is any GBA-determinization procedure. Let \( B \) be a TELA with acceptance \( \beta \) and \( n \) states, and let \( \alpha \) be an equivalent condition in DNF with \( m \) disjuncts. Then, for all the automata \( \text{removeFin}(\text{split}(\mathcal{A})[i]) \), the number of states is bounded by \( O(n) \) and the number of acceptance sets of the GBA conditions is bounded by \( O(\beta) \). Assuming an optimal GBA-determinization procedure, the product combines \( m \) automata with \( 2^{O(n \log n + \log |\beta|)} \) states and hence the result has \( (2^{O(n \log n + \log |\beta|)})^m = 2^{O(2|\beta| n \log n + \log |\beta|)} \) states.

5 Limit-deterministic TELA

Limit-determinism has been studied mainly for Büchi automata [15,43,44], and we define it here for general TELA.

Definition 5.1 A TELA \( \mathcal{A} = (Q, \Sigma, \delta, I, \alpha) \) is called limit-deterministic if there exists a partition \( Q_N, Q_D \) of \( Q \) such that

1. \( \delta \cap (Q_D \times \Sigma \times Q_N) = \emptyset \),
2. for all \( (q, a) \in Q_D \times \Sigma \) there exists at most one \( q' \) such that \( (q, a, q') \in \delta \), and
3. \( \inf(\rho) \cap (Q_N \times \Sigma \times Q_N) = \emptyset \) holds for every accepting run \( \rho \) of \( \mathcal{A} \).

This is a semantic definition and as checking emptiness of deterministic TELA is already coNP-complete, checking whether a TELA is limit-deterministic is as well. To establish this result, we first show that a canonical partition into non-deterministic and deterministic parts of the state space exists. Let \( \mathcal{A} = (Q, I, \Sigma, \delta, \alpha) \) be a fixed TELA. We say that a state \( q \in Q \) is deterministic if it has at most one successor in \( \delta \) for each symbol, and define:

\[
Q_D = \{ q \in Q \mid \text{all states } q' \text{ reachable from } q \text{ are deterministic} \}
\]

and \( Q_N = Q \setminus Q_D \). The sets \( Q_D \) and \( Q_N \) can be computed in polynomial time using an SCC analysis of the underlying graph of \( \mathcal{A} \).

Lemma 5.1 A TELA \( \mathcal{A} \) is limit-deterministic iff the partition \( Q_D, Q_N \) satisfies conditions 1-3 of Definition 5.1.
Proof The direction from right to left is immediate. We now show that if there exists a partition $Q_D, Q_N$ which satisfies 1-3, then so does $Q_D^e, Q_N^e$. By construction, $Q_D^e, Q_N^e$ satisfy conditions 1 and 2. As every state in $Q_D$ is only allowed to reach deterministic states, it follows that $Q_D \subseteq Q_D^e$ and hence $Q_N^e \subseteq Q_N$. But then it follows from the fact that every accepting run of $A$ eventually leaves $Q_N$ forever, that it also eventually leaves $Q_N^e$ forever. This shows that the partition $Q_D^e, Q_N^e$ satisfies condition 3. \qed

Proposition 5.1 Checking limit-determinism for TELA is coNP-complete.

Proof As we have seen above, checking whether $A$ is limit-deterministic amounts to checking whether $Q_D^e, Q_N^e$ satisfies conditions 1-3 of Definition 5.1. Conditions 1-2 hold by construction. To check condition 3 one can construct an automaton $N$ which contains all states and transitions of $Q_N^e$ and replaces all transitions out of $Q_N^e$ by a transition to a rejecting trap state (this may need a simple rewrite of the acceptance condition). Then, condition 3 holds if and only if $N$ accepts the empty language, which can be checked in coNP for TELA (this follows from NP-completeness of the non-emptiness problem [20, Thm. 4.7]).

For coNP-hardness, we reduce from the emptiness problem of TELA, which is coNP-hard. First observe that an automaton $A$ in which $Q_D^e$ is empty is limit-deterministic if and only if $L(A) = \emptyset$. So it suffices to translate an arbitrary TELA $B$ into a TELA $A$ in which $Q_D^e$ is empty, and such that $L(B) = \emptyset$ iff $L(A) = \emptyset$. To do this we add a nondeterministic SCC to $B$, a transition from each original state in $B$ to that SCC, and make sure using the acceptance condition that no run which gets trapped in the new SCC is accepting. \qed

A syntactic definition for TELA in DNF, which implies limit-determinism, is provided in Definition 5.2.

Definition 5.2 A TELA $A = (Q, \Sigma, \delta, (q_0), \alpha)$ in DNF, with $\alpha = \bigwedge_{1 \leq i \leq m} \alpha_i, \alpha_i = \text{Fin}(T_0^i) \land \bigwedge_{1 \leq j \leq k_i} \text{Inf}(T_j^i)$ and $k_i \geq 1$ for all $i$, is syntactically limit-deterministic if there exists a partition $Q_N, Q_D$ of $Q$ satisfying 1. and 2. of Definition 5.1 and additionally $T_j^i \subseteq Q_D \times \Sigma \times Q_D$ for all $i \leq m$ and $1 \leq j \leq k_i$.

This property implies limit-determinism because each disjunction of the acceptance formula contains at least one atomic $\text{inf}(T_i^j)$ formula (for $1 \leq i \leq m$ and $1 \leq j \leq k_i$), and all transitions in $T_i^j$ are included in the deterministic part of the automaton.

5.1 Limit-determinization

Let $A = (Q, \Sigma, \delta, I, \bigvee_{1 \leq i \leq m} \alpha_i)$ be a fixed TELA in DNF for the entire section, where $\alpha_i = \text{Fin}(T_0^i) \land \bigwedge_{1 \leq j \leq k_i} \text{Inf}(T_j^i)$ for $1 \leq i \leq m$. Additionally, define $A_i = \text{split}(A)[i]$. Replacing the product by a sum in the product-based determination of the last section yields single exponential limit-deterministic automata (in contrast to the double-exponential lower-bound for determination).

Proposition 5.2 The automaton $\bigoplus_{1 \leq i \leq m} \text{det(removem}(A_i))$ is limit-deterministic and has $m \cdot 2^{O(n (\log n + \log k))}$ states, where $k = \max\{k_i | 1 \leq i \leq m\}$, $n = |Q|$ and “det” is an optimal, i.e. single-exponential, determination procedure for GBA.

Proof The resulting automaton is limit-deterministic as the only nondeterminism in the transition system consists in the choice of the initial component. Its state space is of size $\sum_{1 \leq i \leq m} |\text{det}(A_i)|$ and hence in $m \cdot 2^{O(n (\log n + \log k))}$, as GBA can be determined with a blow up of $2^{O(n (\log n + \log k))}$ states [42].

If “det” is instantiated by a GBA-determination that produces Rabin automata, then the result is in DNF and syntactically limit-deterministic. Indeed, in this case the only nondeterminism is the choice of the initial state. But “det” can, in principle, also be replaced by any limit-determinization procedure for GBA.

We now extend the limit-determinization constructions of [15] (for Büchi automata) and [6,7,23] (for GBA) to Emerson-Lei conditions in DNF. These constructions use an initial component and an accepting breakpoint component [33] for $A$, which is deterministic. The following construction differs in two ways: there is one accepting component per disjunct of the acceptance condition, and the accepting components are constructed from $A$ without considering the Fin-transitions of that disjunct. We will use the subset transition function $\theta$ associated with $\delta$, defined by: $\theta(P, a) = \bigcup_{q \in P} \{q' | (q, a, q') \in \delta\}$ for $(P, a) \in 2^Q \times \Sigma$, and additionally we define $\theta|_T(P, a) = \bigcup_{q \in P} \{q' | (q, a, q') \in \delta \cap T\}$. These functions are extended to finite words in the standard way. The following definitions refer to the fixed automaton $A$ as defined in the beginning of the section.

Definition 5.3 Let $\theta_i = \theta|_{\delta_i \cap T_i}$. The $i$th accepting breakpoint component of $A$ is $B_i = (Q_i, \Sigma, \delta_i, (p_0), \text{Inf}(\delta_i^{\text{break}}))$, with $p_0 = (I, \emptyset, 1)$, $Q_i = \{(R, B, l) \in 2^Q \times 2^Q | B \subseteq R\}$, and

$$
\delta_{i}^{\text{main}} = \{((R_1, B_1, l), a, (R_2, B_2, l)) \mid R_2 = \theta_i(R_1, a), \quad B_2 = \theta_i(B_1, a) \cup \theta_i|_{T_i}(R_1, a)\}
$$

$$
\delta_{i}^{\text{break}} = \{((R_1, B_1, l), a, (R_2, B_2, l)) \in \delta_{i}^{\text{main}} \mid (R_2, a, l') \in \delta_i^{\text{break}}\}
$$

$$
\delta_i = \{(R_1, B_1, l), a, (R_2, B_2, l) \in \delta_{i}^{\text{main}} \mid R_2 \neq B_2\} \cup \delta_{i}^{\text{break}}
$$

In a state $(R, B, l)$ of the above construction, intuitively $R$ is the set of states reachable for the word seen so far in $A$. 

© Springer
without using transitions from $T_0^i$, while $B$ are the states in $R$ which have seen a transition in $T_0^i$ since the last “breakpoint”. The breakpoint-transitions are $δ_{\text{break}}^i$ which occur when all states in $R$ have seen an accepting transition since the last breakpoint (namely if $R = B$). The standard breakpoint construction underapproximates the language of a given GBA, in general.

We define two limit-deterministic Büchi automata (LDBA) $G_A^\text{LD}$ and $G_A^\text{GFM}$ where $G_A^\text{GFM}$ is additionally good-for-MDP (GFM) [24]. This means that $G_A^\text{GFM}$ can be used to solve certain quantitative probabilistic model checking problems (see Sect. 5.2). Both use the accepting breakpoint components defined above. While $G_A^\text{LD}$ simply uses a copy of $A$ as initial component, $G_A^\text{GFM}$ uses the deterministic subset-automaton of $A$ (it resembles the cut-deterministic automata of [6]). Furthermore, to ensure the GFM property, there are more transitions between initial and accepting copies in $G_A^\text{GFM}$. The construction of $G_A^\text{GFM}$ extends the approach for GBA in [23] (also used for probabilistic model checking) to TELA. We will distinguish elements from sets $Q_i$ for different $i$ from Definition 5.3 by using subscripts (e.g. $(R, P, l_i)$) and assume that these sets are pairwise disjoint.

**Definition 5.4** Let $Q_{\text{acc}} = \bigcup_{1 \leq i \leq m} Q_i$, $δ_{\text{acc}} = \bigcup_{1 \leq i \leq m} δ_i$ and $α_{\text{acc}} = \text{Inf}(\bigcup_{1 \leq i \leq m} δ_{\text{acc}})$.

\[
G_A^\text{LD} = (Q \cup Q_{\text{acc}}, \Sigma, δ_{\text{LD}}, I, α_{\text{acc}})
\]

\[
G_A^\text{GFM} = (\Sigma \cup Q_{\text{acc}}, \Sigma, δ_{\text{GFM}}, I, α_{\text{acc}}),
\]

where $δ_{\text{LD}} = δ \cup δ_{\text{bridge}} \cup δ_{\text{acc}}$ and $δ_{\text{GFM}} = \theta \cup δ_{\text{bridge}} \cup δ_{\text{acc}}$

\[
δ_{\text{bridge}} = \left\{ (q, a, ((q'), \varnothing, 0)) \mid (q, a, q') \in δ \text{ and } 1 \leq i \leq m \right\}
\]

\[
δ_{\text{acc}} = \left\{ (P, a, (P', \varnothing, 0)) \mid P' \subseteq \theta(P, a) \text{ and } 1 \leq i \leq m \right\}
\]

As $δ_{\text{break}}^i \subseteq δ_{\text{acc}}$ for all $i$, both $G_A^\text{LD}$ and $G_A^\text{GFM}$ are syntactically limit-deterministic. The proofs of correctness are similar to ones of the corresponding constructions for GBA [6, Thm. 7.6]. We show later in Theorem 5.2 that $G_A^\text{GFM}$ is GFM.

**Theorem 5.1** The automata $G_A^\text{LD}$ and $G_A^\text{GFM}$ are syntactically limit-deterministic and satisfy $\mathcal{L}(G_A^\text{LD}) = \mathcal{L}(G_A^\text{GFM}) = \mathcal{L}(A)$. Their number of states is bounded by $(n + 3^m m k)$ for $G_A^\text{LD}$ and $(2^n + 3^n m k) \in O(|\alpha|^3 \cdot 3^n)$ for $G_A^\text{GFM}$, with $k = \max\{|k_i| \mid 1 \leq i \leq m\}$.

Before giving a proof of Theorem 5.1 we prove two lemmas related to the breakpoint construction, which will be necessary for the correctness part of Theorem 5.1. The functions $θ$ and $θ_i$ are defined as in Definition 5.3.

The proofs of the lemmas follow known arguments for the correctness of limit-determinization for Büchi and generalized Büchi automata (see [15, Sect. 4.2] and [6, Sects. 7.4 and 7.6]). Still, the extension to Emerson-Lei requires additional arguments and hence we give the proofs here in our notation for completeness. The first lemma extends [6, Lemma 7.1].

**Lemma 5.2** For every accepting run $ρ = q_0 q_1 \ldots$ of $A$ for $w = w_0 w_1 \ldots \in \Sigma^ω$ there exists an $1 \leq i \leq m$ such that $ρ \models α_i$, and a $K \geq 0$ such that:

- for all $l \geq K$: $(q_l, w_i, q_{l+1}) \notin T_0^i$,
- for all $l \geq K$ there exists $u > l$ such that $θ_i((q_l), w_{[l..u]}) = θ_i((q_K), w_{[K..u]})$.

**Proof** Let $ρ$ is accepting there must exist an $1 \leq i \leq m$ such that $ρ \models α_i$, which implies $ρ \models \text{Fin}(T_0^i)$. Fix such an $i$.

To ensure the GFM property, there are more transitions between initial and accepting copies in $G_A^\text{GFM}$. The construction of $G_A^\text{GFM}$ extends the approach for GBA in [23] (also used for probabilistic model checking) to TELA. We will distinguish elements from sets $Q_i$ for different $i$ from Definition 5.3 by using subscripts (e.g. $(R, P, l_i)$) and assume that these sets are pairwise disjoint.

**Definition 5.4** Let $Q_{\text{acc}} = \bigcup_{1 \leq i \leq m} Q_i$, $δ_{\text{acc}} = \bigcup_{1 \leq i \leq m} δ_i$ and $α_{\text{acc}} = \text{Inf}(\bigcup_{1 \leq i \leq m} δ_{\text{acc}})$. Define $\mathcal{L}(G_A^\text{GFM}) \subseteq \mathcal{L}(A)$ and $\mathcal{L}(G_A^\text{LD}) \subseteq \mathcal{L}(A)$.

**Proof** We give the argument only for $G_A^\text{GFM}$, as it is the same for $G_A^\text{LD}$ with simple modifications. Any accepting run of $G_A^\text{GFM}$ for any $w_0 w_1 \ldots \in \Sigma^ω$ has the form

\[I \rightarrow Q_1 \rightarrow \ldots \rightarrow Q_k \rightarrow (P_0, \varnothing, 0)_i \rightarrow (P_1, B_1, h_1)_i \rightarrow \ldots\]

where the first part from $I$ to $Q_k$ is a path through the initial component of $G_A^\text{GFM}$, and the subsequent part is a path through some accepting breakpoint component $B_i$, with $1 \leq i \leq m$. As $B_0$ is the standard breakpoint automaton for $A$ under acceptance $α_i$ and after removing transitions in $T_0^i$, it follows from the soundness of the known construction [6, Lemma 7.3] that there exists a run $ρ$ through $A$ from some state $q \in P_0$ for the word $w[k+1..]$ such that $ρ \models α_i$ and $ρ$ sees no transition in $T_0^i$. Furthermore, clearly any state in $P_0$ is reachable from $I$ with a path labeled by $w[0..k]$. Concatenating such a path with the run $ρ$ yields an accepting run of $A$ for $w_0 w_1 \ldots$. $\square$
Now we are in a position to prove Theorem 5.1. The last part of the proof follows the proof of [6, Lemma 7.2].

**Proof of Theorem 5.1** As \( \delta_i^{break} \subseteq \delta_i \) holds for all \( i \), and the breakpoint components are deterministic, both \( G_A^{GFM} \) and \( G_A^{LD} \) are syntactically limit-deterministic. The bound \( (2^n + 3^m n k) \) on the states of \( G_A^{GFM} \) follows as the state-space of \( \mathcal{B}_i \) is bounded by \( 3^n k \) (this uses that \( B \subseteq R \) holds for states \( (R, B, l) \) of \( \mathcal{B}_i \), and \( l \leq k \) for all \( i \), and there are \( m \) breakpoint components. Additionally, the initial component is a direct subset construction of \( A \), adding another \( 2^n \) states. For \( G_A^{LD} \), the initial component is of size \( n \), hence the number of states is bounded by \( (n + 3^m n k) \).

It remains to show that \( \mathcal{L}(G_A^{LD}) = \mathcal{L}(G_A^{GFM}) = \mathcal{L}(A) \). The inclusions in \( \mathcal{L}(A) \) follow by Lemma 5.3. For the other direction, let \( \rho = q_0 \sigma_1 \ldots \) be an accepting run of \( A \) for \( w = w_0 w_1 \ldots \in \Sigma^\omega \). By Lemma 5.2 there exist \( 1 \leq i \leq m \) and \( K \geq 0 \) such that:

- \( \rho \models \alpha_i \)
- for all \( l \geq K \) : \( (q_l, w_1, q_{l+1}) \notin T_i^l \),
- for all \( l \geq K \) there exists \( u > l \) such that
  
  \[ \theta_i((q_l), w[l..u]) = \theta_i((q_K), w[K..u]). \]

We define a run of \( G_A^{GFM} \) for \( w \) as follows. For the first \( K-1 \) steps, it remains inside the initial component. It will reach a state \( s \subseteq Q \) with \( q_{K-1} \in P \). Then it takes the transition \( (P, w[K-1], (q_K, \emptyset, 0)), \) and continues deterministically thereafter in component \( \mathcal{B}_i \). This transition is in \( \delta_i^{GFM} \) as \( (q_{K-1}, w[K-1], q_K) \in \delta_i \), and hence \( \{q_K\} \subseteq \theta_i(P, w[K-1]) \). We claim that the remaining run

\[ ((q_K), \emptyset, 0)(R_{K+1}, B_{K+1}, h_{K+1}) \ldots \]

of \( \mathcal{B}_i \) sees infinitely many transitions in \( \delta_i^{break} \). For contradiction, suppose that this is not the case. Then, there exists \( N > K \) such that for all \( j \geq N \) we have \( B_j \subseteq R_j \) and \( h_j = h_{j+1} \). As \( \rho \models \alpha_i \) holds, there must exist \( N_1 > N \) satisfying \( (q_{N_1}, w[N_1], q_{N_1+1}) \in T_i^{h_j} \). It follows that \( q_{N_1+1} \in B_{N_1+1} \). By the third property above, however, there exists \( u > N_1+1 \geq K \) such that

\[ R_u = \theta_i((q_K), w[K..u]) = \theta_i((q_{N_1+1}, w[N_1+1..u])) \subseteq B_u. \]

This contradicts \( B_u \subseteq R_u \). Hence, the constructed run is an accepting run of \( G_A^{GFM} \) for \( w \), which finishes the proof of \( \mathcal{L}(A) = \mathcal{L}(G_A^{GFM}) \).

A run for \( G_A^{LD} \) can be constructed in the same way, with the only exception that the initial part of the run is set to be \( q_0 \ldots q_{K-1} \) (rather than the unique path in the subset-construction for prefix \( w[0..K-2] \)).

As a direct corollary of Theorem 5.1 we get that TELA can be transformed into LDIA of single-exponential size.

**Corollary 5.1** Given TELA \( B \) (not necessarily in DNF) with acceptance condition \( \beta \) and \( N \) states, there exists an equivalent LDIA with \( 2^{O(|\beta|+N)} \) states.

### 5.2 Probabilistic model checking

This section discusses how the above constructions can be used for probabilistic model checking. We will use a product between an MDP \( \mathcal{M} \) and a nondeterministic automaton \( A \) which was introduced in [28]. The construction is such that schedulers of the MDP are forced to resolve the nondeterminism of \( A \) by choosing the next state of the automaton. We assume that the automaton used to build the product has a single initial state, which holds for \( G_A^{GFM} \).

**Definition 5.5** Given an MDP \( \mathcal{M} = (S, s_0, Act, P, \Sigma, L) \) and TELA \( \mathcal{G} = (Q, \Sigma, \delta, (q_0), \alpha) \) we define the MDP

\[ \mathcal{M} \times \mathcal{G} = (S \times Q, (s_0, q_0), Act \times Q, P^\times, \Sigma, L^\times) \]

with \( L^\times((s, q)) = L(s) \)

\[ P^\times((s, q), (\alpha, p), (s', q')) = \]

\[ \begin{cases} P(s, \alpha, s') & \text{if } p = q' \text{ and } (q, L(s), q') \in \delta \\ 0 & \text{otherwise} \end{cases} \]

We define the accepting paths \( \Pi_{acc} \) of \( \mathcal{M} \times \mathcal{G} \) to be:

\[ \Pi_{acc} = \{(s_0, q_0)a_0(s_1, q_1)a_1 \ldots \in \text{Paths}(\mathcal{M} \times \mathcal{G}) | \]

\[ q_0, L(s_0), q_1, L(s_1), \ldots \models \alpha \} \]

As a first application, consider the qualitative model checking problem which asks to decide \( \text{Pr}_{\mathcal{M}}^{\max}(\mathcal{L}(A)) > 0 \), under the assumption that \( A \) is a limit-deterministic TELA. While NP-hardness follows from the fact that the problem is already hard for deterministic TELA [34, Thm. 5.13], we now show that it is also in NP. Furthermore, it is in P for automata with a fin-less acceptance condition. This was already known for LDIA [15], and our proof uses similar arguments.

**Proposition 5.3** Deciding \( \text{Pr}_{\mathcal{M}}^{\max}(\mathcal{L}(A)) > 0 \), given MDP \( \mathcal{M} \) and a limit-deterministic TELA \( A \), is NP-complete. If \( A \) has a fin-less acceptance condition, then the problem is in P.

**Proof** NP-hardness for general limit-deterministic TELA follows directly from NP-hardness for deterministic TELA (see [34, Thm. 5.13]). For the upper-bounds we use the fact discussed above that if \( A \) is limit-deterministic, then we can compute a partition \( Q_0, Q_N \) satisfying conditions 1-3 of Definition 5.1 in polynomial time.
In NP for limit-det. TELA. We now show that the problem of computing $P_{M}^\text{max}(L(A)) > 0$ is in NP if TELA $A = (Q, \Sigma, \delta, I, \alpha)$ is limit-deterministic. Let $Q_N, Q_D$ be a partition of $Q$ satisfying conditions 1-3 of Definition 5.1 and let $M \times A$ be the product-MDP defined in Definition 5.5 for a given MDP $M = (S, q_0, Act, P, \Sigma, L)$. Recall that an end-component of $M$ is a non-empty subset $S' \subseteq S$ together with a subset of the enabled actions $T(s) \subseteq Act(s)$ for each state $s \in S'$ such that the underlying graph is strongly connected and closed under probabilistic transitions [3, Definition 10.117]. An end-component $E$ of $M \times A$ naturally induces a set of transitions of $A$, which we denote by $\text{atrans}(E)$.

Claim. $P_{M}^\text{max}(L(A)) > 0$ holds iff there exists a reachable end-component $E$ of $M \times A$ such that $\text{atrans}(E) \models \alpha$.

Proof of claim. "\(\Rightarrow\)" Let $(s, q)$ be a state contained in such an end-component $E$. As $E$ is assumed to be reachable, there exists a finite path $\pi = s_0q_0s_1\ldots s_n\alpha_n$ through $M$ and a corresponding path $q_0 \xrightarrow{L(s_0, \ldots, s_n)} q$ through $A$. We start defining a scheduler $\mathcal{S}_1$ on $M$ which chooses action $\alpha_i$ for all prefixes of length $i$, if $0 \leq i \leq n$ and $\alpha_i$ is enabled. Clearly, $\mathcal{S}_1$ achieves a positive probability to realize the prefix $\pi$. As $E$ is an end-component, we can construct a scheduler $\mathcal{S}$ on $M \times A$ from state $(s, q)$ such that with probability one the set of transitions visited infinitely often is the set of all transitions of $E$ [3, Lemma 10.119]. Scheduler $\mathcal{S}$ induces a scheduler $\mathcal{S}_2$ on $M$ from $s$ which satisfies:

$$P_{M,\mathcal{S}}^\mathcal{S}_2(\{\pi \mid L(\pi) \text{ is accepted from } q \text{ in } A\}) = 1.$$

Combining schedulers $\mathcal{S}_1$ and $\mathcal{S}_2$ yields a scheduler witnessing $P_{M}^\text{max}(L(A)) > 0$.

"\(\Leftarrow\)" Every path $\pi$ from $s$ in $M$ induces a unique path $p_\pi(\pi)$ from $(s, q)$ in $M \times A$, if $q \in Q_D$ (assuming that $A$ is complete). We let $\text{Limit}(\pi)$ be the pair $(A, T)$ where $A$ is the set of states appearing infinitely often in path $\pi$ and $T : A \to 2^{\text{Act}}$ is the set of actions appearing infinitely often for each of the states in $A$. Given a state $q \in Q_D$ and an end-component $E$ of $M \times A$, we let

$$X_{q, E} = \{\pi \in \text{Paths}(M) \mid \text{there exist } \pi_1, \pi_2 \text{ s.t. } \pi = \pi_1\pi_2, I \xrightarrow{L(\pi_1)} A q \text{ and } \text{Limit}(p_\pi(\pi_2)) = E\},$$

where $I \xrightarrow{L(\pi_1)} A q$ means that a path in $A$ exists which starts in some state in $I$, ends in $q$ and is labeled by $L(\pi_1)$. Let $\mathcal{S}$ be a scheduler on $M$ satisfying $P_{M,\mathcal{S}}^\mathcal{S}(L(A)) > 0$. The $\mathcal{S}$-paths $\pi$ in $M$ satisfying both $L(\pi) \in L(A)$ and $\pi \notin \bigcup_{q \in Q_D} \{X_{q, E} \mid \text{atrans}(E) \models \alpha\}$ have probability zero. This is because a path $\pi$ satisfies the following property with probability one under $\mathcal{S}$: for all $i \geq 0$ and $q \in Q_D$: $\text{Limit}(p_\pi(\pi[i..]))$ forms an end-component in $M \times A$. It follows that there exists $q \in Q_D$ and an end-component $E$ of $M \times A$ satisfying $\text{atrans}(E) \models \alpha$ such that $P_{M,\mathcal{S}}(X_{q, E}) > 0$. But then $E$ must also be reachable in $M \times A$, which concludes the proof of the claim.

It is a direct consequence now that the problem is in NP, as we can guess the end-component $E$ and then check in polynomial time whether $\text{atrans}(E) \models \alpha$ holds.

In P for fin-less limit-det. TELA. If $\alpha$ is fin-less, then the existence of an end-component $E$ with $\text{atrans}(E) \models \alpha$ is equivalent to the existence of a maximal end-component satisfying the same property. This is because fin-less properties are preserved when adding additional transitions. As all maximal end-components can be enumerated in polynomial time (see Algorithm 47 in [3]), it follows by the above claim that $P_{M}^\text{max}(L(A)) > 0$ can be decided in polynomial time in this case.

\[\square\]

Our next aim is to show that the maximal probability in an MDP to satisfy a property specified as a TELA can be computed in single-exponential time. As a means to this end we will show that the automaton $G_{\text{GFM}}'$, as defined in Definition 5.4, is good-for-MDP (GFM) [24]. As before, we start with a TELA $A = (Q, \Sigma, \delta, I, \{\sqrt{1 - \sum_1^{n} \alpha_i}\})$ in DNF, with $\alpha_i = \text{Fin}(T_i^i) \land \bigcap_{1 \leq j \leq k} \text{Inf}(T_j^i)$ for $1 \leq i \leq m$.

The GFM-property is defined using the product construction introduced above (Definition 5.5). A Büchi automaton $G$ is called good-for-MDP (GFM) iff $P_{M}^\text{max}(L(G)) = P_{M, G_{\text{GFM}}'}^\text{max}(\Pi_{\text{acc}})$ holds for all MDP $M$ [24]. The inequality "$\geq\$" holds for all automata [28, Thm. 1], but the other direction requires, intuitively, that a scheduler on $M \times G$ is able to safely resolve the nondeterminism of the automaton based on the prefix of the run. This is trivially satisfied by deterministic automata, but good-for-games automata also have this property [28]. Limit-deterministic Büchi automata are not GFM in general, for example, $G_{\text{GFM}}'$ may not be (see Example 5.1).

The proof proceeds by fixing an arbitrary MDP $M$ and showing that $P_{M}^\text{max}(L(A)) \leq P_{M, G_{\text{GFM}}'}^\text{max}(\Pi_{\text{acc}})$. To this end it is enough to show that for any finite-memory scheduler $\mathcal{S}$ on $M$ one finds a scheduler $\mathcal{S}'$ on $M \times G_{\text{GFM}}'$ such that $P_{M, \mathcal{S}_{\text{GFM}}'}(L(A)) \leq P_{M, \mathcal{S}_{\text{GFM}}'}^\text{max}(\Pi_{\text{acc}})$. The restriction to finite-memory schedulers is allowed because the maximal probability to satisfy an $\omega$-regular property is always attained by such a scheduler [3, Sects. 10.6.3 and 10.6.4].

\\[\setcounter{section}{5}\\]
To define $\mathcal{G}'$ we analyze the behaviour of $\mathcal{G}$ in the product $\mathcal{M} \times \mathcal{D}$ of $\mathcal{M}$ with a specific deterministic automaton $\mathcal{D}$ for $\mathcal{L}(A)$, which we introduce later. In particular, we show that for each accepting bottom strongly connected component $B$ of the induced Markov chain of $\mathcal{M} \times \mathcal{D}$ under $\mathcal{G}$ there exists an accepting breakpoint component $\mathcal{B}_i$ such that, with probability one, words generated by $B$ are accepted by $\mathcal{B}_i$. To show this, we first prove a sufficient condition for a word to be accepted by $\mathcal{B}_i$.

**Lemma 5.4** Let $1 \leq i \leq m$, $w \in \Sigma^w$, $0 = j_0 < j_1 < j_2 < \ldots$ be an increasing sequence of natural numbers and $Q' \subseteq Q$ satisfying:

- for all $l \geq 0$: $\theta_i(Q', w[j_l..j_{l+1}-1]) = Q'$ and
- for all $l \geq 0$ and $q' \in Q'$ there exists $q \in Q'$ and a finite $\theta_i$-path from $q$ to $q'$ in $\mathcal{A}$ labeled by $w[j_l..j_{l+1}-1]$ which hits all transition-sets $T_1^i, \ldots, T_k^i$.

Then, $w$ is accepted from state $(Q', \emptyset, 0)$ in $\mathcal{B}_i$.

**Proof** As the first component of states in $\mathcal{B}_i$ just simulates the subset transition function $\theta_i$, the run from state $(Q', \emptyset, 0)$ of $\mathcal{B}_i$ for $w$ has the form:

$$(Q', \emptyset, 0) \xrightarrow{w[0..j_l-1]} (Q', B_1, h_1) \xrightarrow{w[j_l..j_{l-1}-1]} (Q', B_2, h_2) \xrightarrow{w[j_{l-1}..j_{l+1}-1]} \ldots$$

We show that each infix $(Q', B_l, h_l)$ $\xrightarrow{w[j_l..j_{l+1}-1]} (Q', B_{l+1}, h_{l+1})$ of the run above sees a transition in $q_{\text{break}}^i$. For contradiction, suppose that for some $l$ this is not the case. Then, the third component is not changed in any transition of the corresponding infix run. Let

$$(Q_0, A_0, h_1) \xrightarrow{w[j_l]} (Q_1, A_1, h_1) \xrightarrow{w[j_l+M-1]} \ldots \xrightarrow{w[j_l+M-1]} (Q_M, A_M, h_1)$$

be the sequence of the run between $(Q_0, A_0, h_1) = (Q', B_1, h_1)$ and $(Q_M, A_M, h_1) = (Q', B_{l+1}, h_1)$. Take an arbitrary element $q' \in Q' \setminus B_{l+1}$ and let $P$ be the source states of the incoming transitions of $q'$ labeled by $w[j_l+M-1]$ wrt. $\delta \setminus T'_0$. As a consequence we have that $P \cap A_{M-1}$ is empty and no $p \in Q_{M-1}$ satisfies $\rho$, $w[j_l+M-1], q' \in T'_h$. This is because otherwise we would either have $q' \in B_{l+1}$ or see a breakpoint transition. In particular, there is no path from any $p \in Q_{M-1}$ to $q'$ labeled by $w[j_l+M-1]$ that sees $T'_{h_l}$. We can continue this argument inductively until reaching $(Q_0, A_0, h_1)$ and conclude that there is no $q' \in Q'$ such that there exists a $\theta_i$-path from $q$ to $q'$ labeled by $w[j_l..j_{l+1}-1]$ which sees some transition in $T'_{h_l}$. This is in contradiction with the second property above.

Hence, the run sees a transition in $q_{\text{break}}^i$ infinitely often, which implies that it is accepting. $\square$

Recall that $\text{removeFin}(\text{split}(A)[i])$ is a GBA with two components: the initial component $Q_1$ is a copy of $A$ where no transition is accepting, and component $Q_2$ is a copy of $A$ without the transitions of $T'_j$. The acceptance condition is the GBA condition $\bigwedge_{1 \leq j \leq k_i} \text{Inf}(U_j^i)$ where $U_j^i$ is the set of transitions corresponding to $T_j^i \setminus T'_j$ in the component $Q_2$.

Now let $D_i = \text{det}(\text{removeFin}(\text{split}(A)[i]))$ be the Rabin automaton one gets by applying the construction of [42] to $\text{removeFin}(\text{split}(A)[i])$. Their soundness-proof [42, Thm. 1] can be modified slightly to prove the following statement, where $\gamma$ is the extended subset transition function of $\text{removeFin}(\text{split}(A)[i])$. It shows that an accepting lasso-path through $D_i$ induces a special path through the powerset automaton of $\text{removeFin}(\text{split}(A)[i])$. As a consequence, the corresponding word is accepted by $\text{removeFin}(\text{split}(A)[i])$. Furthermore, the word labeling the loop of the lasso satisfies a condition which matches the one of Lemma 5.4, and hence is accepted by $\mathcal{B}_i$.

**Lemma 5.5** Let $\rho = d_0d_1 \ldots$ be an accepting run of $D_i$ for $w$ satisfying Rabin pair $\text{Inf}(D_1) \wedge \text{Fin}(D_2)$ and $i_0$ be a position such that for all $l \geq i_0$ we have $(d_l, w[l], d_{l+1}) \notin D_2$ and for infinitely many $l \geq i_0$ we have $d_l = d_{i_0}$.

Then, a sequence $i_0 < i_1 < i_2 \ldots$ and a $Q' \subseteq Q_2$ exist such that

- $Q' \subseteq \gamma(I, w[0..i_0-1])$,
- for all $l \geq 0$: $Q' = \gamma(Q', w[i_l..i_{l+1}-1])$ and for all $q_2 \in Q'$ there exists $q_1 \in Q'$ and a $\gamma$-path from $q_1$ to $q_2$ labeled by $w[i_l..i_{l+1}-1]$ which hits all transition-sets $U_1^i, \ldots, U_k^i$.

**Proof** The following statement follows directly from the proof of [42, Thm. 1]: Let $j_0 < j_1 < \ldots$ be an infinite sequence of positions such that $(d_{j_l}, w[l], d_{j_{l+1}}) \notin D_2$ holds for $l \geq j_0$ and the sequence $d_{j_l}, w[j_l], d_{j_{l+1}} \ldots$ sees at least $k_i$ transitions in $D_i$ for all $l \geq 0$. Then, there exists a sequence $P_0, P_1, \ldots$, with all $P_l \subseteq Q_1 \cup Q_2$, such that we have $P_0 \subseteq \gamma(I, w[0..j_0-1])$, $P_{l+1} = \gamma(P_l, w[j_l..j_{l+1}-1])$ and for all $q_2 \in P_{l+1}$ there exists a $q_1 \in P_l$ and a $\gamma$-path from $q_1$ to $q_2$ labeled by $w[j_l..j_{l+1}-1]$ which hits all transition sets $U_1^i, \ldots, U_k^i$. Furthermore, the sequence is such that if $d_{j_l} = d_{j_{l+1}}$ then $P_{l+1} = P_l$.

Clearly a sequence $j_0 < j_1 < \ldots$ exists satisfying the above assumption (as $\rho$ is accepting) and which additionally has the property that $d_{j_l} = d_{j_{l+1}}$ for all $l \geq 0$ (by taking $j_0 = i_0$ and the fact that infinitely many positions see $d_{i_0}$). We may conclude that the sequence $P_0, P_1, \ldots$ exists as above and as $d_{j_l} = d_{j_{l+1}}$ for all $l \geq 0$, indeed $P_0 = P_l$ for all $l$. So we may take $Q' = P_0$, and the only thing left to show is that $Q' \subseteq
Lemma 5.5. Now, we show how these facts can be used to construct the scheduler $\mathcal{S}'$.

Recall that we are given a finite memory scheduler $\mathcal{S}$ of $\mathcal{M}$ and to prove the GFM-property we have to provide a scheduler $\mathcal{S}'$ such that $\Pr^{\mathcal{S}}(L(\mathcal{A})) \leq \Pr^{\mathcal{S}'}(\mathcal{M} \times G^{\text{GFM}}(\Pi_{\text{acc}}))$. Let $\mathcal{M} \times D$ be the product of the induced finite Markov chain $\mathcal{M}$ with $D = \bigotimes_{1 \leq i \leq m} D_i$. The scheduler $\mathcal{S}'$ is constructed as follows. It stays inside the initial component of $\mathcal{M} \times G^{\text{GFM}}$ and mimics the action chosen by $\mathcal{S}$ until the corresponding path in $\mathcal{M} \times D$ reaches an accepting strongly connected component (BSCC) $B$. This means that the transitions of $D$ induced by $B$ satisfy one of the Rabin pairs. The following lemma shows that in this case there exists a state in one of the accepting breakpoint components $\mathcal{B}_i$ to which $\mathcal{S}'$ can safely move.

Lemma 5.6 Let $s$ be a state in an accepting BSCC $B$ of $\mathcal{M} \times D$ and $\pi_1$ be a finite path that reaches $s$ from the initial state of $\mathcal{M} \times D$. Then, there exists $1 \leq i \leq m$ and $Q' \subseteq Q''$ such that

$$\Pr_s(Q \mid L(\pi) = 1) = 1.$$

Proof. There exists some Rabin pair $\text{Inf}(D'_1) \wedge \text{Fin}(D'_2)$ of $D$ such that no transition of $B$ is contained in $D'_2$, and some transition of $B$ is contained in $D'_1$. By construction of $D$, this pair directly corresponds to a Rabin pair $\text{Inf}(D'_1) \wedge \text{Fin}(D'_2)$ of one of the components $D_i$ (as the acceptance condition of $D$ is essentially the disjunction of the acceptance conditions of the $D_i$).

Hence for every path $\pi_2$ through $B$ starting in $s$ which sees all transitions in $B$ infinitely often we find an accepting run $d_0d_1 \ldots d_{j_0}$ of $D_i$ for $L(\pi(\pi_2))$ and a sequence $j_0 < j_1 < \ldots$ where $j_0 = |\pi_1|$ and such that:

- no transition of $d_{j_0}d_{j_0+1} \ldots$ is included in $D_2$,
- for all $l \geq 0$: $d_{j_l} = d_{j_0}$ and the run $d_0d_1 \ldots$ sees an accepting transition in between position $j_l$ and $j_{l+1}$ for all $l \geq 0$.

By Lemma 5.5 there exists a $Q' \subseteq \gamma(I, L(\pi_1)) \cap Q_2$ such that for all $l \geq 0$: $Q' = \gamma(Q', w[j_0 \ldots j_{l+1} - 1])$ and for all $q_1 \in Q'$ there exists a $q_2 \in Q'$ and a finite $\gamma$-path for $w[j_0 \ldots j_{l+1} - 1]$ hitting all transition-sets $U_{j_0}', \ldots, U_{j_{l+1}}'$. Let $Q''$ be the corresponding set of states of $A$, which implies $Q'' \subseteq \theta(I, L(\pi_1))$. As $Q' \subseteq Q_2$ we have: for all $q_1 \in Q''$ there exists a $q_2 \in Q'$ and a finite $\theta$-path for $w[j_0 \ldots j_{l+1} - 1]$ hitting all transition-sets $T_{j_0}', \ldots, T_{j_{l+1}}'$.

It follows that the conditions of Lemma 5.4 are satisfied and hence that $L(\pi_2)$ is accepted from $(Q'', \emptyset, 0)$ in $\mathcal{B}_i$. □

The lemma does not hold if we restrict ourselves to singleton $\{q\}$ (see Example 5.1). Hence, restricting $\delta_{\text{bridge}}$ to such transitions (as for $\delta_{\text{bridge}}$, see Definition 5.4) would not guarantee the GFM property.

Example 5.1 Consider the automaton $A$ with states $\{a_i, b_j \mid i, j \in [1, 2]\}$ such that $a_i b_j$ has transitions labeled by $a_i$ to $b_j a_1$ and $b_j a_2$. Transitions of states $b_j a_j$ are defined analogously, and all states in $\{a_i b_1 \mid i \in [1, 2]\}$ are initial (Fig. 5 shows the transitions of $a_1 b_1$). All transitions are accepting for a single Büchi condition, and hence $L(\mathcal{A}) = ((a_i b_1 \mid i \in [1, 2])^\omega)$.

Consider the Markov chain $M$ in Fig. 5 (transition probabilities are all 1/2 and omitted in the figure). Clearly, $\Pr_M(L(\mathcal{A})) = 1$. Figure 5 shows a part of the product of $\mathcal{M}$ with the breakpoint automaton $\mathcal{B}$ for $\mathcal{A}$ (Definition 5.3) starting from $(a_1, (\{a_1 b_1 \}, \emptyset, 0))$. As $b_1 a_1$ and $b_2 a_2$ have no $b_2$-transition, the state $(b_2, ((b_2 a_1, b_2 a_2), \emptyset, 0))$ is a trap state. Hence, the state $(a_1, (\{a_1 b_1 \}, \emptyset, 0))$ generates an accepting path with probability at most 1/2. This holds for all states $(s, (P', \emptyset, 0))$ of $\mathcal{M} \times \mathcal{B}$ where $P'$ is a singleton.

But using $\delta_{\text{bridge}}$ to connect initial and accepting components implies that any accepting path sees such a state. Hence, using $\delta_{\text{bridge}}$ to define $G^{\text{GF}}_{\mathcal{A}}$ would not guarantee the GFM property.

Using Lemma 5.6 we can define $\mathcal{S}'$ such that the probability accepting paths under $\mathcal{S}'$ in $\mathcal{M} \times G^{\text{GF}}_{\mathcal{A}}$ is at least as high as that of paths with label in $L(\mathcal{A})$ in $\mathcal{M}$. This is the non-trivial direction of the GFM property.

Lemma 5.7 For every finite-memory scheduler $\mathcal{S}$ on $\mathcal{M}$, there exists a scheduler $\mathcal{S}'$ on $\mathcal{M} \times G^{\text{GF}}_{\mathcal{A}}$ such that:

$$\Pr^{\mathcal{S}}_{\mathcal{M} \times G^{\text{GF}}_{\mathcal{A}}}(\Pi_{\text{acc}}) \geq \Pr^{\mathcal{S}'}_{\mathcal{M} \times G^{\text{GF}}_{\mathcal{A}}}(L(\mathcal{A}))$$

Proof. Let $\mathcal{S}$ be a finite-memory scheduler on $\mathcal{M}$ and $\mathcal{M} \times D$ be as above. As $\mathcal{S}$ uses only finite memory, the Markov chain $\mathcal{M} \times D$ has finitely many states. This, in turn, implies that a BSCC will be reached with probability one.

We construct the scheduler $\mathcal{S}'$ as follows. For every finite path $\pi_1$ of $\mathcal{M} \times G^{\text{GF}}_{\mathcal{A}}$, it needs to choose an action of $\mathcal{M}$ and whether to move from the initial component to one of the breakpoint components (this is the only nondeterministic choice in $G^{\text{GF}}_{\mathcal{A}}$). The action of $\mathcal{M}$ is always chosen in the same way as by $\mathcal{S}$ for the corresponding finite path of $\mathcal{M}$. If the path in $\mathcal{M} \times D$ corresponding to $\pi_1$ does not end in a BSCC, $\mathcal{S}'$ chooses to remain in the initial component. If it
Fig. 5 Restricting \( \delta_{\text{bridge}} \) to transitions with endpoints of the form \((s, ([q], \varnothing, 0)) \) (similar to \( \delta_{\text{LBRIDGE}} \)) would not guarantee the GFM property (see Example 5.1)

does reach a BSCC \( B \) we make a distinction on whether \( B \) is accepting or not. If \( B \) is not accepting, then \( \delta' \) can be defined arbitrarily for all prefixes that extend \( \pi_1 \). If \( B \) is accepting, then let us assume that the last state visited by \( \pi_1 \) does reach a BSCC

\[ \begin{align*}
(a) & \quad \begin{array}{c}
\text{b}_1 \text{a}_1 \\
\text{a}_1 \\
\text{b}_1 \text{a}_2
\end{array}
\end{align*} \]

(b) \quad \begin{align*}
\begin{array}{c}
\text{b}_2 \\
\text{a}_2 \\
\text{b}_2
\end{array}
\end{align*} \]

c) \quad \begin{align*}
\left( \begin{array}{c}
\text{a}_1, ([\text{a}_1, \text{b}_1], \varnothing, 0) \\
\text{a}_1 \\
\left( \begin{array}{c}
\text{b}_2, ([\text{b}_1, \text{a}_1, \text{b}_1, \text{a}_2], \varnothing, 0) \\
\text{b}_2 \end{array} \right) \right) \quad \frac{1}{2}
\end{align*}

\[ \begin{align*}
\left( \begin{array}{c}
\text{b}_1, ([\text{b}_1, \text{a}_1, \text{b}_1, \text{a}_2], \varnothing, 0) \\
\text{b}_1 \\
\end{array} \right) \quad \frac{1}{2}
\end{align*} \]

(398) T. John et al.

A \leftarrow \text{such that}

\[ \begin{align*}
\Pr_{, \mathcal{M}\mathcal{G}(\mathcal{L}(A))} & \leftarrow \Pr_{, \mathcal{M}GFM_{(acc)}} \\
\text{2 For every finite memory scheduler } \mathcal{S} \text{ of } \mathcal{M} & \leftarrow \text{there exists a scheduler } \mathcal{S}' \leftarrow \text{of } \mathcal{M} \leftarrow \text{such that}
\end{align*} \]

1. is proved by Lemma 5.7 and 2. is proved for arbitrary automaton in [28, Thm. 1].

To compute \( \Pr_{\mathcal{M}}^{\text{max}}(\mathcal{L}(B)) \) one can translate \( B \) into an equivalent TELA \( A \) in DNF, then construct \( G_{\mathcal{GFM}} \) and finally compute \( \Pr_{\mathcal{M} \times G_{\mathcal{GFM}}}(\Pi_{\text{acc}}) \). The automaton \( G_{\mathcal{GFM}} \) is single-exponential in the size of \( B \) by Theorem 5.1, and it is well known that \( \Pr_{\mathcal{M} \times G_{\mathcal{GFM}}}(\Pi_{\text{acc}}) \) can be computed in polynomial time in the size of \( \mathcal{M} \times G_{\mathcal{GFM}} \) [3, Thm. 10.127].

**Theorem 5.3** Given a TELA \( B \) (not necessarily in DNF) and an MDP \( \mathcal{M} \), the value \( \Pr_{\mathcal{M}}^{\text{max}}(\mathcal{L}(B)) \) can be computed in single-exponential time.

**6 Experimental evaluation**

The product approach combines a sequence of deterministic automata using the disjunctive product. We introduce the \textit{langcover heuristic}: the automata are “added” to the product one by one, but only if their language is not already subsumed by the automaton constructed so far. This leads to substantially smaller automata in many cases, but is only efficient if checking inclusion for the considered automata types is efficient. In our case this holds because we can reduce the needed checks to inclusion checks on Street automata, which are in P [2, Table 2]. Note, that this is not the case for arbitrary deterministic TELA, or nondeterministic automata.

\( \odot \) Springer
6.1 Implementation

We compare the following implementations of the constructions discussed above. Spot uses the TELA to GBA translator of Spot, simplifies (using Spot’s postprocessor with preference Small) and degeneralizes the result and [: : :] then determinizes using a version of Safra’s algorithm [18,38]. The removeFin function that is used is an optimized version of Definition 3.3. If the input is a weak, Rabin or Street automaton, Spot uses dedicated algorithms instead of removeFin to construct the GBA. In the constructions remFin → splitα, splitœ → remFin and remFin → rewriteα, the first step is replaced by the corresponding TELA to GBA construction (without the langcover heuristic). The intermediate GBA are also simplified. The construction G_A^LD is implemented in limit-det., using the Spot-library and parts of SEMINATOR [8]. We compare it to limit-det. via GBA, which concatenates the TELA to GBA construction of Spot with the limit-determinization of SEMINATOR. Similarly, good-for-MDP and good-for-MDP via GBA are the construction G_A^GFM applied to A directly, or to the GBA as constructed by Spot. Both constructions via GBA are in the worst case double-exponential. No post-processing, e.g. Spot’s postprocessor, is applied to any output automaton.

6.2 Experiments

Computations were performed on a computer with two Intel E5-2680 CPUs with 8 cores each at 2.70 GHz running Linux. Each individual experiment was limited to a single core, 15 GB of memory and 1200 seconds. We use versions 2.9.4 of SPOT (configured to allow 256 acceptance sets) and 2.0 of SEMINATOR.

Our first benchmark set (called random) consists of 1000 TELA with 4 to 50 states and 8 sets of transitions T_1,...,T_8 used to define the acceptance conditions. They are generated using Spot’s procedure random_graph() by specifying probabilities such that: a triple (q, a, q') ∈ Q × Σ × Q is included in the transition relation (3/|Q|) and such that a transition t is included in a set T_j (0.2). We use only transition systems that are nondeterministic. The acceptance condition is generated randomly using Spot’s procedure acc_code::random(). We transform the acceptance condition to DNF and keep those acceptance conditions whose lengths range between 2 and 21 and consist of at least two disjuncts. To quantify the amount of nondeterminism, we divide the number of pairs of transitions with common starting point and symbol (that is, of the form (q, a, q_1), (q, a, q_2), with q_1 ≠ q_2) of the automaton by its number of states.

Table 1 shows that the product produces smallest deterministic automata overall. Spot produces best results among the algorithms that go via a single GBA. One reason for this is that after GBA-simplifications of SPOT, the number of acceptance marks of the intermediate GBA are of about the same and Spot constructs GBA with fewer states. This difference in the number of states is expected, given the theoretical analysis in Sect. 3.2. The similarity of the numbers of acceptance marks is not supported by the theoretical analysis and can only be explained by SPOT’s advanced GBA-simplification algorithms that might suit Spot’s GBA better than the GBA constructed by the other approaches.

The limit-deterministic automata are in general much smaller than the deterministic ones, and limit-det via GBA, performs best in this category (see Fig. 6). However, the construction G_A^LD (limit-det.) resulted in fewer time- and memouts and, as shown in Fig. 6, it still produces for most cases outputs with fewer states than Spot.

For GFM automata we see that computing G_A^GFM directly, rather than first computing a GBA, yields much better results (good-for-MDP vs. good-for-MDP via GBA). However, the GFM automata suffer from significantly more time- and memouts than the other approaches. The automata sizes are comparable with Spot’s determinization (see Fig. 6). Their similarity to the pure limit-determinization constructions, and the fact that their acceptance condition is much simpler than for the deterministic automata, we believe that future work on optimizing this construction could make it a competitive alternative for probabilistic model checking using TELA.

We compare the two best performing determinization approaches, Spot and product, in more detail. Figure 6 shows that the results are widely spread, i.e. both approaches produce smaller results than the other one for a significant number of inputs. Figure 2 partitions the set of all input automata according to acceptance complexity (measured in the size of their DNF) and amount of nondeterminism. Each subset of input automata is of roughly the same size (159-180). The number of timeouts and memouts increases with the amount of nondeterminism and the size of the input acceptance condition and reaches up to 42%. The values “states”, “time” and “acceptance” are the median value of the ratio of these metrics. If only one of the approaches was able to construct a result within the given time and memory bounds we define the ratio as 0 (Spot did not construct a result) or infinity (product did not construct a result). If none of the approaches was able to construct a result, we do not consider this input automaton for the calculation. Values below 1 indicate that product produces smaller results and values above 1 indicate that Spot produces smaller results.

---

2 The source code and data of all experiments are available at [27].
The approach product produces for all subsets of input automata outputs with fewer states than Spot and the ratio of the computation time decreases the larger the amount of nondeterminism and the longer the acceptance condition is.

The second benchmark set (called DNF) consists of 500 TELA constructed randomly as above, apart from the acceptance conditions. They are in DNF with 2-3 disjuncts, with 2-3 Inf-atoms and 0-1 Fin-atoms each (all different). Such formulas tend to lead to larger CNF conditions, which benefits the new approaches. Table 1 shows that product produces way smaller results than Spot. Again, Table 2 partitions the set of input automata but we do not consider different lengths of acceptance conditions here because the subsets of input automata are already relatively small (140-193). Again, product performs better for automata with more nondeterminism.

The third benchmark set consists of the sequence of automata described in Fig. 4. Figure 7 shows the results for such input automata with 1-25 states. For these automata,
Table 2  Comparison of Spot and product, with input automata grouped by the size of the DNF of their acceptance condition and the amount of nondeterminism. “States”, “Time” and “Acceptance” refer to the median of the ratio product / Spot. The number of automata that is denoted in brackets is the number of input automata for which both approaches were able to construct a result within the given time and memory bounds.

| Benchmark | Amount of nondet. | Input acceptance | #Automata (no time-/memouts) | Timeouts | Memouts | Median of ratio Product / Spot |
|-----------|-------------------|------------------|-----------------------------|----------|---------|-------------------------------|
|           |                   |                  |                             | Spot     | product | Spot product | States | Time | Acceptance |
| random    | ≤ 0.66            | 2 ≤ |α| ≤ 11           | 175 (175)                   | 0.0%     | 0.0%    | 0.0%           | 0.0%   | 0.0% | 0.91 | 1.37 | 1.80 |
|          |                   | 12 ≤ |α| ≤ 21          | 180 (180)                   | 0.0%     | 0.0%    | 0.0%           | 0.0%   | 0.0% | 0.82 | 1.44 | 1.64 |
|          | > 0.66, ≤ 1.33    | 2 ≤ |α| ≤ 11           | 162 (159)                   | 0.0%     | 0.0%    | 0.0%           | 1.9%   | 0.0% | 0.75 | 1.19 | 1.80 |
|          |                   | 12 ≤ |α| ≤ 21          | 159 (144)                   | 0.0%     | 0.6%    | 0.0%           | 3.1%   | 7.5% | 0.75 | 1.26 | 1.82 |
|          | > 1.33            | 2 ≤ |α| ≤ 11           | 163 (128)                   | 1.8%     | 0.0%    | 0.0%           | 16.0%  | 11.7%| 0.75 | 0.84 | 1.72 |
|          |                   | 12 ≤ |α| ≤ 21          | 161 (82)                    | 1.2%     | 7.5%    | 0.0%           | 42.2%  | 28.0%| 0.43 | 0.80 | 1.67 |
| DNF      | ≤ 0.66            | 2 ≤ |α| ≤ 21           | 193 (193)                   | 0.0%     | 0.0%    | 0.0%           | 0.0%   | 0.0% | 0.81 | 1.17 | 1.14 |
|          | > 0.66, ≤ 1.33    | 2 ≤ |α| ≤ 21           | 167 (160)                   | 0.0%     | 0.0%    | 0.0%           | 3.0%   | 2.4% | 0.68 | 0.93 | 1.19 |
|          | > 1.33            | 2 ≤ |α| ≤ 21           | 140 (111)                   | 1.4%     | 0.0%    | 0.0%           | 18.6%  | 10.7%| 0.26 | 0.25 | 1.00 |
automata. The example also highlights the effect of the language heuristic: it benefits from the fact that the language of the input automaton under any individual disjunct of the DNF is the same. This is used to prune large parts of the state-space.

## 7 Conclusion

In this article, new approaches for determinization and limit-determinization of transition-based Emerson-Lei automata (TELA) have been suggested. One of these constructions produces limit-deterministic good-for-MDP automata, which can be used for quantitative probabilistic verification. The constructions have all been implemented and evaluated experimentally. The evaluation shows promising results in particular when applying the new product construction for determinization of TELA. Furthermore, we show that the complexity of limit-determinization of TELA is single-exponential rather than double-exponential (as compared to full determinization).

The presented work opens the door to numerous research questions and directions. At first, a general study on what properties can be naturally encoded directly into nondeterministic TELA would be of interest. This could be combined with a general evaluation of the developed approaches against real-world examples. It would also be interesting to see, whether and how much the new constructions could be further improved by relying on determinization procedures for GBA that use a general acceptance condition (rather than Rabin or parity) to reduce the number of states. Such an analysis could pave the way to develop a heuristic that selects for every GBA the determinization procedure that leads to the best overall result. Another important direction would be to consider translations from LTL to compact, nondeterministic TELA, which could then be used within (probabilistic) model-checking tools for LTL. A first step in this direction has already been made in [32]. Furthermore, the evaluation of good-for-MDP automata in the context of probabilistic model checking is another future research direction.

**Acknowledgements** We thank David Müller for suggesting to us the problem of determinizing Emerson-Lei automata and many discussions on the topic.

**Funding** Open Access funding enabled and organized by Projekt DEAL.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

## References

1. Babiak T, Blahoudek F, Duret-Lutz A, Klein J, Křetínský J, Müller D, Parker D, Strejček J (2015) The hanoi omega-automata format. In: Computer aided verification (CAV’15), LNCS. Springer
2. Baier C, Blahoudek F, Duret-Lutz A, Klein J, Müller D, Strejček J (2019) Generic emptiness check for fun and profit. In: Automated technology for verification and analysis (ATVA), LNCS. Springer
3. Baier C, Katoen JP (2008) Principles of model checking (representation and mind series). The MIT Press
4. Baier C, Kiefer S, Klein J, Klüppelholz S, Müller D, Worrell J (2016) Markov chains and unambiguous Büchi automata. In: Computer aided verification (CAV), LNCS. Springer
5. Ben-Ari M (2008) Principles of the spin model checker. Springer-Verlag, London
6. Blahoudek F (2018) Automata for formal methods: little steps towards perfection. Ph.D. thesis, Masaryk University, Faculty of Informatics
7. Blahoudek F, Duret-Lutz A, Klokocka M, Kretínsky M, Strejček J (2017) Seminar: a tool for Semi-Determinization of Omega-Automata. In: International conference on logic for programming, artificial intelligence and reasoning (LPAR), EPiC Series in Computing
8. Blahoudek F, Duret-Lutz A, Strejček J (2020) Seminar: a tool for Semi-Determinization of Omega-Automata. In: Lahiri SK, Wang C (eds) Computer aided verification. Springer International Publishing, Cham, pp 15–27
9. Blahoudek F, Major J, Strejček J (2019) LTL to smaller self-loop alternating automata and back. In: Theoretical aspects of computing (ICTAC), LNCS. Springer
10. Bloemen V, Duret-Lutz A, van de Pol J (2019) Model checking with generalized Rabin and Fin-less automata. Int J Soft Tools Technol Transf 21(3):307–324
11. Boker U (2018) Why these automata types? In: EPiC series in computing, 57, 143–163. EasyChair
12. Chatterjee K, Gaiser A, Křetínsky J (2013) Automata with generalized Rabin pairs for probabilistic model checking and LTL synthesis. In: Computer aided verification (CAV), LNCS. Springer
13. Colcombet T (2015) Unambiguity in automata theory. In: J. Shalit, A. Okhotin (eds.) Descriptive complexity of formal systems, lecture notes in computer science, pp. 3–18. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-319-19225-3_1
14. Courcoubetis C, Yannakakis M (1988) Verifying temporal properties of finite-state probabilistic programs. In: [Proceedings 1988] 29th annual symposium on foundations of computer science, pp. 338–345. 10.1109/SFCS.1988.21950
15. Courcoubetis C, Yannakakis M (1995) The complexity of probabilistic verification. J ACM 42(4):857–907
16. Couvreur JM (1999) On-the-fly verification of linear temporal logic. In: Formal methods (FM), LNCS. Springer
17. Duret-Lutz A (2017) Contributions to LTL and ω-automata for model checking. Habilitation thesis, Université Pierre et Marie Curie
18. Duret-Lutz A, Lewkowicz A, Fauchille A, Michaud T, Renault É, Xu L (2016) Spot 2.0–A framework for LTL and ω-automata manipulation. In: Automated technology for verification and analysis (ATVA), LNCS. Springer
19. Duret-Lutz A, Poitrenaud D, Couvreur JM (2009) On-the-fly emptiness check of transition-based Streett automata. In: Automated technology for verification and analysis (ATVA), LNCS. Springer

20. Emerson EA, Lei CL (1987) Modalities for model checking: branching time logic strikes back. Sci Comput Program 8(3):275–306

21. Esparza J, Křetínský J, Sickert S (2018) One theorem to rule them all: a unified translation of LTL into $\omega$-Automata. In: logic in computer science (LICS), ACM

22. Giannakopoulou D, Lerda F (2002) From states to transitions: improving translation of LTL formulae to Büchi automata. In: Formal techniques for networked and distributed systems (FORTE), LNCS. Springer

23. Hahn EM, Li G, Schewe S, Turrini A, Zhang L (2015) Lazy probabilistic model checking without determinisation. In: Concurrency theory (CONCUR)

24. Hahn EM, Perez M, Schewe S, Somenni F, Trivedi A, Wojtczak D (2020) Good-for-MDPs automata for probabilistic analysis and reinforcement learning. In: Tools and algorithms for the construction and analysis of systems (TACAS), LNCS. Springer

25. Jantsch S, Müller D, Baier C, Klein J (2019) From LTL to unambiguous Büchi automata via disambiguation of alternating automata. In: ter Beek MH, McIver A, Oliveira JN (eds) Formal methods–the next 30 years, LNCS. Springer

26. John T, Jantsch S, Baier C, Klein J (2019) From LTL to unambiguous Büchi automata via disambiguation of alternating automata. In: ter Beek MH, McIver A, Oliveira JN (eds) Formal methods–the next 30 years, LNCS. Springer

27. Jantsch S, Müller D, Baier C, Klein J (2019) From LTL to unambiguous Büchi automata via disambiguation of alternating automata. In: ter Beek MH, McIver A, Oliveira JN (eds) Formal methods–the next 30 years, LNCS. Springer

28. John T, Jantsch S, Baier C, Klein J (2019) From LTL to unambiguous Büchi automata via disambiguation of alternating automata. In: ter Beek MH, McIver A, Oliveira JN (eds) Formal methods–the next 30 years, LNCS. Springer

29. John T, Jantsch S, Baier C, Klein J (2019) From LTL to unambiguous Büchi automata via disambiguation of alternating automata. In: ter Beek MH, McIver A, Oliveira JN (eds) Formal methods–the next 30 years, LNCS. Springer

30. John T, Jantsch S, Baier C, Klein J (2019) From LTL to unambiguous Büchi automata via disambiguation of alternating automata. In: ter Beek MH, McIver A, Oliveira JN (eds) Formal methods–the next 30 years, LNCS. Springer

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.