Determinization and Limit-Determinization of Emerson-Lei Automata

Tobias John(✉), Simon Jantsch(✉), Christel Baier, and Sascha Klüppelholz

Technische Universität Dresden, Dresden, Germany
tobiasj@posteo.de,
{simon.jantsch,christel.baier,sascha.klueppelholz}@tu-dresden.de

Abstract. We study the problem of determinizing ω-automata whose acceptance condition is defined on the transitions using Boolean formulas, also known as *transition-based Emerson-Lei automata* (TELA). The standard approach to determinize TELA first constructs an equivalent *generalized Büchi automaton* (GBA), which is later determinized. We introduce three new ways of translating TELA to GBA. Furthermore, we give a new determinization construction which determinizes 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. We also study limit-determinization of TELA and 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.

1 Introduction

Automata on infinite words, also called ω-automata, play a fundamental role in the fields of verification and synthesis of reactive systems [11,32,35,39]. They can be used both to represent properties of systems and the systems themselves. For logical specification languages such as linear temporal logic (LTL), many verification systems, such as Spin [4] or Prism [25], use logic-to-automata translations internally to verify a given system against the specification.

A major research question in this area has been, and still is, the question of whether and how ω-automata can be determinized efficiently [26,31,33,36,37]. The first single-exponential and asymptotically optimal determinization for Büchi automata was presented in [36]. Deterministic automata are important

This work was funded by DFG grant 389792660 as part of TRR 248, the Cluster of Excellence EXC 2050/1 (CeTI, project ID 390696704, as part of Germany's Excellence Strategy), DFG-projects BA-1679/11-1 and BA-1679/12-1, and the Research Training Group QuantLA (GRK 1763).

from a practical point of view as classical automata-based solutions to reactive synthesis and probabilistic verification use deterministic automata [32,39].

The high complexity of determinization and most logic-to-automata translations have raised the question of more succinct representations of ω-automata. Using *generalized* acceptance conditions (e.g. generalized Büchi [12] or generalized Rabin [8,10]) and transition-based [18], 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 [16,35]. Together with a vast body of work on heuristics and dedicated procedures this standardized format has led to practically usable and mature tools and libraries such as Spot [14] and Owl [24] which support a wide range of operations on ω-automata. Special classes of nondeterministic automata with some of the desired properties of deterministic automata have also been studied. The classes of good-for-MDP [20] and good-for-games [23] automata can be used for quantitative probabilistic model checking of Markov decision processes [19,38], while limit-deterministic Büchi automata can be used for qualitative model-checking [11]. Dedicated translations from LTL directly to deterministic and limit-deterministic automata have been considered in [17].

This paper considers determinization and limit-determinization of TELA. In contrast to limit-determinization, the theoretical complexity of determinization is well understood (a tight, double-exponential, bound was given in [35,36]). However, it has not been studied yet from a practical point of view.

Contribution. We propose three new translations from TELA to GBA (Sect. 3) and give an example in which they perform exponentially better than state-of-the-art implementations. We introduce a new determinization procedure for TELA based on a product construction (Sect. 4). Our experiments show that it often outperforms the approaches based on determinizing a single GBA (Sect. 6). 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, Sect. 5.1). We show that deciding $\mathbf{Pr}^{\max}_{\mathcal{M}}(\mathcal{L}(\mathcal{A})) > 0$ is NP-complete for limit-deterministic TELA $\mathcal{A}$, and in P if the acceptance of $\mathcal{A}$ is fin-less (Proposition 5.9). Finally, we show how to limit-determinize TELA based on the breakpoint-construction. A version of this procedure yields *good-for-MDP* Büchi automata (Definition 5.6). Thereby $\mathbf{Pr}^{\max}_{\mathcal{M}}(\mathcal{L}(\mathcal{A}))$ is computable in single-exponential time for arbitrary MDP $\mathcal{M}$ and TELA $\mathcal{A}$ (Theorem 5.15).

Related Work. The upper-bound for TELA-determinization [35,36] relies on a translation to GBA which first transforms the acceptance formula into disjunctive normal form (DNF). We build on this idea. Another way of translating TELA to GBA was described in [13]. Translations from LTL to TELA have been proposed in [7,27,30], and all of them use product constructions to combine automata for subformulas. The emptiness-check for ω-automata under different types of acceptance conditions has been studied in [2,8,10,15], 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 [8,10] is equivalent to the special DNF that we use and a special case of the hyper-Rabin condition for which the emptiness problem is in P [9,16]. Probabilistic model checking for *deterministic* automata under this condition is considered in [10], while [8] is concerned with standard emptiness while allowing nondeterminism. A procedure to transform TELA into parity automata is presented in [34].[1]

2 Preliminaries

Automata. A *transition-based Emerson-Lei* automaton (TELA) $\mathcal{A}$ is a tuple $(Q, \Sigma, \delta, I, \alpha)$, where Q is a finite set of states, Σ is a finite alphabet, $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation, $I \subseteq Q$ is the set of initial states and α is a symbolic acceptance condition over δ, which is defined by:

$$\alpha ::= tt \mid ff \mid \mathrm{Inf}(T) \mid \mathrm{Fin}(T) \mid (\alpha \vee \alpha) \mid (\alpha \wedge \alpha), \text{ with } T \subseteq \delta$$

If α is tt, ff, $\mathrm{Inf}(T)$ or $\mathrm{Fin}(T)$, then it is called *atomic*. We denote by $|\alpha|$ 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 $T \subseteq \delta$. Their semantics is defined recursively as follows:

$$\begin{array}{lll} T \models tt & T \models \mathrm{Inf}(T') \text{ iff } \mathrm{T} \cap \mathrm{T}' \neq \emptyset & T \models \alpha_1 \vee \alpha_2 \text{ iff } \mathrm{T} \models \alpha_1 \text{ or } \mathrm{T} \models \alpha_2 \\ T \not\models ff & T \models \mathrm{Fin}(T') \text{ iff } \mathrm{T} \cap \mathrm{T}' = \emptyset & T \models \alpha_1 \wedge \alpha_2 \text{ iff } \mathrm{T} \models \alpha_1 \text{ and } \mathrm{T} \models \alpha_2 \end{array}$$

Two acceptance conditions α and β are *δ-equivalent* ($\alpha \equiv_\delta \beta$) if for all $T \subseteq \delta$ we have $T \models \alpha \iff T \models \beta$. A *run* of $\mathcal{A}$ for an infinite word $u = u_0 u_1 \ldots \in \Sigma^\omega$ is an infinite sequence of transitions $\rho = (q_0, u_0, q_1)(q_1, u_1, q_2) \ldots \in \delta^\omega$ that starts with an initial state $q_0 \in I$. The set of transitions that appear infinitely often in ρ are denoted by $\inf(\rho)$. A run ρ is *accepting* ($\rho \models \alpha$) iff $\inf(\rho) \models \alpha$. The language of $\mathcal{A}$, denoted by $\mathcal{L}(\mathcal{A})$, is the set of all words for which there exists an accepting run of $\mathcal{A}$. The sets of infinite words which are the language of some TELA are called *ω-regular*. A TELA $\mathcal{A}$ is *deterministic* if the set of initial states contains exactly one state and the transition relation is a function $\delta : Q \times \Sigma \to Q$. It is *complete*, if for all $(q, a) \in Q \times \Sigma$: $\delta \cap \{(q, a, q') \mid q' \in Q\} \neq \varnothing$. A *Büchi condition* is an acceptance condition of the form $\mathrm{Inf}(T)$ and a *generalized Büchi condition* is a condition of the form $\bigwedge_{1 \leq i \leq k} \mathrm{Inf}(T_i)$. We call the sets T_i appearing in a generalized Büchi condition its *acceptance sets*. Rabin (resp. Street) conditions are of the form $\bigvee_{1 \leq i \leq k}(\mathrm{Fin}(F_i) \wedge \mathrm{Inf}(T_i))$ (resp. $\bigwedge_{1 \leq i \leq k}(\mathrm{Fin}(F_i) \vee \mathrm{Inf}(T_i))$).

Probabilistic Systems. A labeled *Markov decision process* (MDP) $\mathcal{M}$ is a tuple $(S, s_0, \mathrm{Act}, P, \Sigma, L)$ where S is a finite set of states, $s_0 \in S$ is the initial state, Act is a finite set of actions, $P : S \times \mathrm{Act} \times S \to [0,1]$ defines the transition probabilities with $\sum_{s' \in S} P(s, \alpha, s') \in \{0, 1\}$ for all $(s, \alpha) \in S \times \mathrm{Act}$ and $L : S \to$

[1] All proofs are provided in the full version of the paper [21].

Σ is a labeling function of the states into a finite alphabet Σ. Action $\alpha \in \mathrm{Act}$ is *enabled* in s if $\sum_{s' \in S} P(s, \alpha, s') = 1$, and $\mathrm{Act}(s) = \{\alpha \mid \alpha \text{ is enabled in } s\}$. A *path* of $\mathcal{M}$ is an infinite sequence $s_0\alpha_0s_1\alpha_1 \ldots \in (S \times \mathrm{Act})^\omega$ such that $P(s_i, \alpha_i, s_{i+1}) > 0$ for all $i \geq 0$. The set of all paths of $\mathcal{M}$ is denoted by $\mathrm{Paths}(\mathcal{M})$ and $\mathrm{Paths}_{\mathrm{fin}}(\mathcal{M})$ denotes the *finite paths*. Given a path $\pi = s_0\alpha_0s_1\alpha_1 \ldots$, we let $L(\pi) = L(s_0)L(s_1) \ldots \in \Sigma^\omega$. A *Markov chain* is an MDP with $|\mathrm{Act}(s)| \leq 1$ for all states s. A scheduler of $\mathcal{M}$ is a function $\mathfrak{S} : (S \times \mathrm{Act})^* \times S \to \mathrm{Act}$ such that $\mathfrak{S}(s_0\alpha_0 \ldots s_n) \in \mathrm{Act}(s_n)$. It induces a Markov chain $\mathcal{M}_\mathfrak{S}$ and thereby a probability measure over $\mathrm{Paths}(\mathcal{M})$. The probability of a set of paths Π starting in s_0 under this measure is $\Pr^{\mathfrak{S}}_{\mathcal{M}}(\Pi)$. For an ω-regular property $\Phi \subseteq \Sigma^\omega$ we define $\mathbf{Pr}^{\max}_{\mathcal{M}}(\Phi) = \sup_{\mathfrak{S}} \Pr^{\mathfrak{S}}_{\mathcal{M}}(\{\pi \mid \pi \in \mathrm{Paths}(\mathcal{M}) \text{ and } L(\pi) \in \Phi\})$. See [3, Chapter 10] for more details.

3 From TELA to Generalized Büchi Automata

3.1 Operations on Emerson-Lei Automata

The first operator takes a TELA and splits it along the top-level disjuncts of the acceptance condition. Let $\mathcal{A} = (Q, \Sigma, \delta, I, \alpha)$ be a TELA where $\alpha = \bigvee_{1 \leq i \leq m} \alpha_i$ and the α_i are arbitrary acceptance conditions. We define $\mathrm{split}(\mathcal{A}) := (\mathcal{A}_1, \ldots, \mathcal{A}_m)$ with $\mathcal{A}_i = (Q, \Sigma, \delta, I, \alpha_i)$ for $1 \leq i \leq m$, and $\mathrm{split}(\mathcal{A})[i] := \mathcal{A}_i$.

Lemma 3.1. *It holds that* $\mathcal{L}(\mathcal{A}) = \bigcup_{1 \leq i \leq m} \mathcal{L}\big(\mathrm{split}(\mathcal{A})[i]\big)$.

The analogous statement does not hold for conjunction and intersection (cf [21, Fig. 5]). 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 [13] and similar constructions are used in [27,30]. While the sum operations yield smaller automata in general, only the product construction preserves determinism.

Definition 3.2. *Let* $\mathcal{A}_i = (Q_i, \Sigma, \delta_i, I_i, \alpha_i)$*, with* $i \in \{0, 1\}$*, be two complete TELA with disjoint state-spaces. The* sum *of* $\mathcal{A}_0$ *and* $\mathcal{A}_1$ *is defined as follows:*

$$\mathcal{A}_0 \oplus \mathcal{A}_1 = \big(Q_0 \cup Q_1, \Sigma, \delta_0 \cup \delta_1, I_0 \cup I_1, (\alpha_0 \wedge \mathrm{Inf}(\delta_0)) \vee (\alpha_1 \wedge \mathrm{Inf}(\delta_1)\big)$$

If $\alpha_i = \mathrm{Inf}(T^i_1) \wedge \ldots \wedge \mathrm{Inf}(T^i_k)$*, with* $i \in \{0, 1\}$*, (i.e. both automata are GBA), then we can use the* GBA-specific sum*:*

$$\mathcal{A}_0 \oplus_{GBA} \mathcal{A}_1 = \big(Q_0 \cup Q_1, \Sigma, \delta_0 \cup \delta_1, I_0 \cup I_1, (\mathrm{Inf}(T^0_1 \cup T^1_1) \wedge \ldots \wedge \mathrm{Inf}(T^0_k \cup T^1_k))\big)$$

The disjunctive product *is defined as follows:*

$$\mathcal{A}_0 \otimes \mathcal{A}_1 = \big(Q_0 \times Q_1, \Sigma, \delta_\otimes, I_0 \times I_1, (\uparrow(\alpha_0) \vee \uparrow(\alpha_1))\big)$$

with $\delta_\otimes = \big\{\big((q_0, q_1), a, (q'_0, q'_1)\big) \;\big|\; (q_0, a, q'_0) \in \delta_0 \text{ and } (q_1, a, q'_1) \in \delta_1\big\}$ *and* $\uparrow(\alpha_i)$ *is constructed by replacing every occurring set of transitions* T *in* α_i *by* $\big\{\big((q_0, q_1), u, (q'_0, q'_1)\big) \in \delta_\otimes \;\big|\; (q_i, u, q'_i) \in T\big\}$.

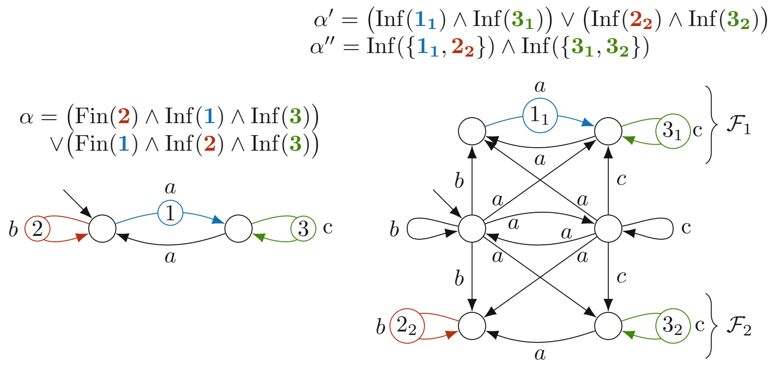
α′ = (Inf(1₁) ∧ Inf(3₁)) ∨ (Inf(2₂) ∧ Inf(3₂))
α″ = Inf({1₁, 2₂}) ∧ Inf({3₁, 3₂})
α = (Fin(2) ∧ Inf(1) ∧ Inf(3))
∨(Fin(1) ∧ Inf(2) ∧ Inf(3))
a
b
c
1
2
3
1₁
2₂
3₁
3₂
ℱ₁
ℱ₂

Fig. 1. Example of applying removeFin and $\text{removeFin}_{\text{GBA}}$ (Definition 3.4) to the automaton on the left. The result is the automaton on the right with acceptance α' (removeFin), respectively α'' ($\text{removeFin}_{\text{GBA}}$).

The additional $\text{Inf}(\delta_0)$ and $\text{Inf}(\delta_1)$ atoms in the acceptance condition of $\mathcal{A}_0 \oplus \mathcal{A}_1$ are essential (see [21, Fig. 6]). 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 [8,10]).

Definition 3.3 (DNF for TELA). *Let $\mathcal{A} = (Q, \Sigma, \delta, I, \alpha)$ be a TELA. We say that $\mathcal{A}$ is in* DNF *if α is of the form $\alpha = \bigvee_{1 \le i \le m} \alpha_i$, with $\alpha_i = \text{Fin}(T_0^i) \wedge \bigwedge_{1 \le j \le k_i} \text{Inf}(T_j^i)$ and such that all $k_i \ge 1$.*

The reason that a single Fin atom in each disjunct is enough is that $\text{Fin}(T_1) \wedge \text{Fin}(T_2) \equiv_\delta \text{Fin}(T_1 \cup T_2)$ for all T_1, T_2, δ. Taking $k_i \ge 1$ is also no restriction, as we can always add $\wedge \text{Inf}(\delta)$ to any disjunct. Using standard Boolean operations one can transform a TELA with acceptance β into DNF by just translating the acceptance formula into a formula α of the above form, with $|\alpha| \le 2^{|\beta|}$.

Fin-Less Acceptance. To transform a TELA in DNF (see Definition 3.3) into an equivalent one without Fin-atoms we use the idea of [8,13]: a main copy of $\mathcal{A}$ is connected to one additional copy for each disjunct α_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 1 shows an example.

Definition 3.4. *Let $\mathcal{F}_i = (Q_i, \Sigma, \delta_i, I_i, \phi_i)$, where $Q_i = \{q^{(i)} \mid q \in Q\}$, $\delta_i = \{(q^{(i)}, a, q'^{(i)}) \mid (q, a, q') \in \delta \setminus T_0^i\}$ and $\phi_i = \bigwedge_{1 \le j \le k_i} \text{Inf}(U_j^i)$, where $U_j^i = \{(q^{(i)}, a, q'^{(i)}) \mid (q, a, q') \in T_j^i \setminus T_0^i\}$. Let* $\text{removeFin}(\mathcal{A}) = (Q', \Sigma, \delta', I, \alpha')$ *and* $\text{removeFin}_{\text{GBA}}(\mathcal{A}) = (Q', \Sigma, \delta', I, \alpha'')$*, where $Q' = Q \cup \bigcup_{1 \le i \le m} Q_i$ and:*

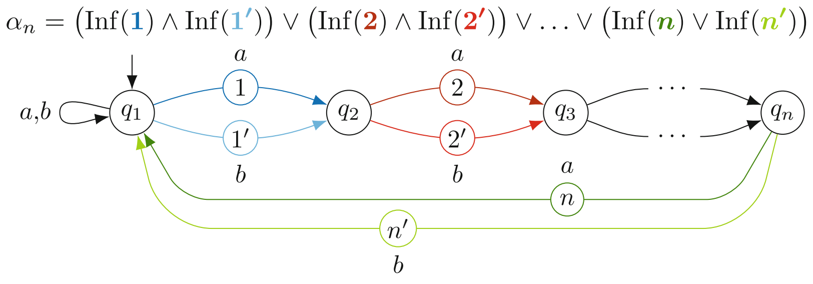
$\alpha_n = (\mathrm{Inf}(1) \wedge \mathrm{Inf}(1')) \vee (\mathrm{Inf}(2) \wedge \mathrm{Inf}(2')) \vee \ldots \vee (\mathrm{Inf}(n) \vee \mathrm{Inf}(n'))$
a,b
q_1
q_2
q_3
q_n
a
1
1′
b
a
2
2′
b
…
…
a
n
n′
b

Fig. 2. A class of TELA where generating the CNF leads to 2^n many conjuncts.

- $\delta' = \delta \cup \bigcup_{1 \le i \le m} \left(\delta_i \cup \{(q, a, q'^{(i)}) \mid (q, a, q') \in \delta\}\right)$
- $\alpha' = \bigvee_{1 \le i \le m} \phi_i$
- $\alpha'' = \bigwedge_{1 \le j \le k} \mathrm{Inf}(U_j^1 \cup \ldots \cup U_j^m)$, *with* $k = \max_i k_i$ *and* $U_j^i = \delta_i$ *if* $k_i < j \le k$.

Lemma 3.5. *It holds that* $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathrm{removeFin}(\mathcal{A})) = \mathcal{L}(\mathrm{removeFin}_{\mathrm{GBA}}(\mathcal{A}))$.

While removeFin($\mathcal{A}$) is from [8,13], removeFin$_{\mathrm{GBA}}$($\mathcal{A}$) is a variant that differs only in the acceptance and always produces GBA. Both consist of $m+1$ copies of $\mathcal{A}$ (with Fin-transitions removed).

3.2 Construction of Generalized Büchi Automata

Construction of Spot. The transformation from TELA to GBA from [13] is implemented in Spot [14]. It transforms the automaton into DNF and then applies (an optimized version of) removeFin. The resulting fin-less acceptance condition is translated into conjunctive normal form (CNF). As $\mathrm{Inf}(T_1) \vee \mathrm{Inf}(T_2) \equiv_\delta \mathrm{Inf}(T_1 \cup T_2)$ holds for all δ, one can rewrite any fin-less condition in CNF into a conjunction of Inf-atoms, which is a generalized Büchi condition. When starting with a TELA $\mathcal{B}$ with acceptance β and N states, one gets a GBA with $N\,2^{O(|\beta|)}$ 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. 2 shows such an example).

Copy-Based Approaches. We now describe three approaches (remFin→splitα, splitα→remFin and remFin→rewriteα), which construct GBA with at most $|\beta|$ acceptance sets. On the other hand, they generally produce automata with more states. They are based on [35] which first translates copies of $\mathcal{A}$ (corresponding to the disjuncts of the acceptance condition) to GBA, and then takes their sum. However, it is not specified in [35] how exactly Fin-atoms should be removed (they were concerned only with the theoretical complexity). We define:

$$\begin{aligned}
\text{remFin}{\rightarrow}\text{split}\alpha(\mathcal{A}) &:= \bigoplus_{1\leq i\leq m}{}_{\text{GBA}}\ \text{split}(\text{removeFin}(\mathcal{A}))[i] \\
\text{split}\alpha{\rightarrow}\text{remFin}(\mathcal{A}) &:= \bigoplus_{1\leq i\leq m}{}_{\text{GBA}}\ \text{removeFin}(\text{split}(\mathcal{A})[i]) \\
\text{remFin}{\rightarrow}\text{rewrite}\alpha(\mathcal{A}) &:= \text{removeFin}_{\text{GBA}}(\mathcal{A})
\end{aligned}$$

With removeFin as defined in Definition 3.4, the approaches remFin→splitα and splitα→remFin produce the same automata (after removing non-accepting SCC's in remFin→splitα), and all three approaches create $O(m)$ copies of $\mathcal{A}$. Our implementation uses an optimized variant of removeFin, as provided by Spot, which leads to different results for all three approaches.

4 Determinization

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 [37]. For a TELA $\mathcal{B}$ with n states and acceptance β, 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.

Determinization via a Product Construction. Another way to determinize a TELA $\mathcal{A}$ in DNF is to determinize the automata split$(\mathcal{A})[i]$ one by one and then combining them with the disjunctive product construction of Definition 3.2:

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

where "det" is any GBA-determinization procedure. Let $\mathcal{B}$ be a TELA with acceptance β and n states, and let α be an equivalent condition in DNF with m disjuncts. Assuming an optimal GBA-determinization procedure, the product combines m automata with $2^{O(n(\log n+\log|\beta|))}$ states and hence has $\left(2^{O(n\,(\log n+\log|\beta|))}\right)^m = 2^{O(2^{|\beta|}\cdot n(\log n+\log|\beta|))}$ states.

5 Limit-Deterministic TELA

Limit-determinism has been studied mainly in the context of Büchi automata [11, 38,39], 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. $\delta \cap (Q_D \times \Sigma \times Q_N) = \varnothing$,
2. *for all $(q, a) \in Q_D \times \Sigma$ there exists at most one q' such that $(q, a, q') \in \delta$,*
3. *every accepting run ρ of $\mathcal{A}$ satisfies $\inf(\rho) \cap (Q_N \times \Sigma \times Q_N) = \varnothing$.*

This is a semantic definition and as checking emptiness of deterministic TELA is already coNP-hard, checking whether a TELA is limit-deterministic is also.

Proposition 5.2. *Checking limit-determinism for TELA is coNP-complete.*

An alternative syntactic definition for TELA in DNF, which implies limit-determinism, is provided in Definition 5.3.

Definition 5.3. *A TELA $\mathcal{A} = (Q, \Sigma, \delta, \{q_0\}, \alpha)$ in DNF, with $\alpha = \bigvee_{1 \le i \le m} \alpha_i$, $\alpha_i = \mathrm{Fin}(T_0^i) \wedge \bigwedge_{1 \le j \le k_i} \mathrm{Inf}(T_j^i)$ and $k_i \ge 1$ 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 $i \le m$ and $1 \le j \le k_i$.*

5.1 Limit-Determinization

We first observe that replacing the product by a sum in the product-based determinization above yields limit-deterministic automata of single-exponential size (in contrast to the double-exponential lower-bound for determinization). Let $\mathcal{A}$ be a TELA in DNF with n states and acceptance $\alpha = \bigvee_{1 \le i \le m} \alpha_i$, where $\alpha_i = \mathrm{Fin}(T_0^i) \wedge \bigwedge_{1 \le j \le k_i} \mathrm{Inf}(T_j^i)$ (see Definition 3.3), and let $\mathcal{A}_i = \mathrm{split}(\mathcal{A})[i]$.

Proposition 5.4. *$\bigoplus_{1 \le i \le m} \det(\mathrm{removeFin}(\mathcal{A}_i))$ is limit-deterministic and of size $\sum_{1 \le i \le m} |\det(\mathcal{A}_i)| = m \cdot 2^{O(n\,(\log n + \log k))}$, where $k = \max\{k_i \mid 1 \le i \le m\}$.*

If "det" is instantiated by a GBA-determinization 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 [11] (for Büchi automata) and [5,6,19] (for GBA) to Emerson-Lei conditions in DNF. These constructions use an *initial* component and an *accepting breakpoint component* [28] for $\mathcal{A}$, 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 $\mathcal{A}$ without considering the Fin-transitions of that disjunct. To define the accepting components we use the subset transition function θ associated with δ: $\theta(P, a) = \bigcup_{q \in P} \{q' \mid (q, a, q') \in \delta\}$ for $(P, a) \in 2^Q \times \Sigma$, and additionally we define $\theta|_T(P, a) = \bigcup_{q \in P} \{q' \mid (q, a, q') \in \delta \cap T\}$. These functions are extended to finite words in the standard way.

Definition 5.5. *Let* $\theta_i = \theta|_{\delta\setminus T_0^i}$ *and define* $\mathcal{BP}_i = (Q_i, \Sigma, \delta_i, \{p_0\}, \mathrm{Inf}(\delta_i^{\mathrm{break}}))$ *with:* $Q_i = \{(R, B, l) \in 2^Q \times 2^Q \times \{0, \dots, k_i\} \mid B \subseteq R\}$, $p_0 = (I, \varnothing, 0)$ *and*

$$\delta_i^{\mathrm{main}} = \left\{ ((R_1, B_1, l), a, (R_2, B_2, l)) \mid \begin{array}{l} R_2 = \theta_i(R_1, a), \\ B_2 = \theta_i(B_1, a) \cup \theta_i|_{T_l^i}(R_1, a) \end{array} \right\}$$

$$\delta_i^{\mathrm{break}} = \left\{ ((R_1, B_1, l), a, (R_2, \varnothing, l') \mid \begin{array}{l} ((R_1, B_1, l), a, (R_2, B_2, l)) \in \delta_i^{\mathrm{main}}, \\ R_2 = B_2, \\ l' = (l+1) \bmod (k_i + 1) \end{array} \right\}$$

$$\delta_i = \left\{ ((R_1, B_1, l), a, (R_2, B_2, l)) \in \delta_i^{\mathrm{main}} \mid R_2 \neq B_2 \right\} \cup \delta_i^{\mathrm{break}}$$

In state (R, B, l), intuitively R is the set of states reachable for the prefix word in $\mathcal{A}$ without using transitions from T_0^i, while B are the states in R which have seen a transition in T_l^i since the last "breakpoint". The breakpoint-transitions are $\delta_i^{\mathrm{break}}$, which occur when all states in R have seen an accepting transition since the last breakpoint (namely if $R = B$). The breakpoint construction under-approximates the language of a given GBA, in general.

We define two limit-deterministic Büchi automata (LDBA) $\mathcal{G}_{\mathcal{A}}^{\mathrm{LD}}$ and $\mathcal{G}_{\mathcal{A}}^{\mathrm{GFM}}$ where $\mathcal{G}_{\mathcal{A}}^{\mathrm{GFM}}$ is additionally *good-for-MDP* (GFM) [20]. This means that $\mathcal{G}_{\mathcal{A}}^{\mathrm{GFM}}$ can be used to solve certain quantitative probabilistic model checking problems (see Sect. 5.2). Both use the above breakpoint automata as accepting components. While $\mathcal{G}_{\mathcal{A}}^{\mathrm{LD}}$ simply uses a copy of $\mathcal{A}$ as initial component, $\mathcal{G}_{\mathcal{A}}^{\mathrm{GFM}}$ uses the deterministic subset-automaton of $\mathcal{A}$ (it resembles the *cut-deterministic* automata of [5]). Furthermore, to ensure the GFM property, there are more transitions between initial and accepting copies in $\mathcal{G}_{\mathcal{A}}^{\mathrm{GFM}}$. The construction of $\mathcal{G}_{\mathcal{A}}^{\mathrm{GFM}}$ extends the approach for GBA in [19] (also used for probabilistic model checking) to TELA. We will distinguish elements from sets Q_i for different i from Definition 5.5 by using subscripts (e.g. $(R, P, l)_i$) and assume that these sets are pairwise disjoint.

Definition 5.6. ($\mathcal{G}_{\mathcal{A}}^{\mathrm{LD}}$ and $\mathcal{G}_{\mathcal{A}}^{\mathrm{GFM}}$). *Let* $Q_{\mathrm{acc}} = \bigcup_{1 \leq i \leq m} Q_i$, $\delta_{\mathrm{acc}} = \bigcup_{1 \leq i \leq m} \delta_i$ *and* $\alpha_{\mathrm{acc}} = \mathrm{Inf}(\bigcup_{1 \leq i \leq m} \delta_i^{\mathrm{break}})$. *Define*

$$\mathcal{G}_{\mathcal{A}}^{\mathrm{LD}} = (Q \cup Q_{\mathrm{acc}}, \Sigma, \delta^{\mathrm{LD}}, I, \alpha') \quad \textit{and} \quad \mathcal{G}_{\mathcal{A}}^{\mathrm{GFM}} = (2^Q \cup Q_{\mathrm{acc}}, \Sigma, \delta^{\mathrm{GFM}}, \{I\}, \alpha')$$

with

$$\delta^{\mathrm{LD}} = \delta \cup \delta_{\mathrm{bridge}}^{\mathrm{LD}} \cup \delta_{\mathrm{acc}} \qquad \textit{and} \qquad \delta^{\mathrm{GFM}} = \theta \cup \delta_{\mathrm{bridge}}^{\mathrm{GFM}} \cup \delta_{\mathrm{acc}}$$

$$\delta_{\mathrm{bridge}}^{\mathrm{LD}} = \left\{ \big(q, a, (\{q'\}, \varnothing, 0)_i\big) \mid (q, a, q') \in \delta \textit{ and } 1 \leq i \leq m \right\}$$

$$\delta_{\mathrm{bridge}}^{\mathrm{GFM}} = \left\{ \big(P, a, (P', \varnothing, 0)_i\big) \mid P' \subseteq \theta(P, a) \textit{ and } 1 \leq i \leq m \right\}$$

As $\delta_i^{\mathrm{break}} \subseteq \delta_{\mathrm{acc}}$ for all i, both $\mathcal{G}_{\mathcal{A}}^{\mathrm{LD}}$ and $\mathcal{G}_{\mathcal{A}}^{\mathrm{GFM}}$ are syntactically limit-deterministic. The proofs of correctness are similar to ones of the corresponding constructions for GBA [5, Thm. 7.6]. We show later in Proposition 5.14 that $\mathcal{G}_{\mathcal{A}}^{\mathrm{GFM}}$ is GFM.

Theorem 5.7. *$\mathcal{G}_{\mathcal{A}}^{\mathrm{LD}}$ and $\mathcal{G}_{\mathcal{A}}^{\mathrm{GFM}}$ are syntactically limit-deterministic and satisfy $\mathcal{L}(\mathcal{G}_{\mathcal{A}}^{\mathrm{LD}}) = \mathcal{L}(\mathcal{G}_{\mathcal{A}}^{\mathrm{GFM}}) = \mathcal{L}(\mathcal{A})$. Their number of states is in $O(n+3^n\, m\, k)$ for $\mathcal{G}_{\mathcal{A}}^{\mathrm{LD}}$ and $O(2^n + 3^n\, m\, k) = O(|\alpha|^2 \cdot 3^n)$ for $\mathcal{G}_{\mathcal{A}}^{\mathrm{GFM}}$, where $k = \max\{k_i \mid 1 \le i \le m\}$.*

Corollary 5.8. *Given TELA $\mathcal{B}$ (not necessarily in DNF) with acceptance condition β and N states, there exists an equivalent LDBA with $2^{O(|\beta|+N)}$ states.*

5.2 Probabilistic Model Checking

We now discuss how these constructions can be used for probabilistic model checking. First, consider the *qualitative* model checking problem to decide $\mathbf{Pr}_{\mathcal{M}}^{\max}(\mathcal{L}(\mathcal{A})) > 0$, under the assumption that $\mathcal{A}$ is a limit-deterministic TELA. While NP-hardness follows from the fact that the problem is already hard for deterministic TELA [29, 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 LDBA [11], and our proof uses similar arguments.

Proposition 5.9. *Deciding $\mathbf{Pr}_{\mathcal{M}}^{\max}(\mathcal{L}(\mathcal{A})) > 0$, given an MDP $\mathcal{M}$ and a limit-deterministic TELA $\mathcal{A}$, is NP-complete. If $\mathcal{A}$ has a fin-less acceptance condition, then the problem is in P.*

Now we show that $\mathcal{G}_{\mathcal{A}}^{\mathrm{GFM}}$ is good-for-MDP [20]. In order to define this property, we introduce the product of an MDP with a nondeterministic automaton in which, intuitively, the scheduler is forced to resolve the nondeterminism by choosing the next state of the automaton (see [20,23]). We assume that the automaton used to build the product has a single initial state, which holds for $\mathcal{G}_{\mathcal{A}}^{\mathrm{GFM}}$.

Definition 5.10. *Given an MDP $\mathcal{M} = (S, s_0, \mathrm{Act}, P, \Sigma, L)$ and TELA $\mathcal{G} = (Q, \Sigma, \delta, \{q_0\}, \alpha)$ we define the MDP $\mathcal{M} \times \mathcal{G} = (S \times Q, (s_0, q_0), \mathrm{Act} \times Q, P^\times, \Sigma, L^\times)$ with $L^\times((s,q)) = L(s)$ and*

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

We define the *accepting paths Π_{acc}* of $\mathcal{M} \times \mathcal{G}$ to be:

$$\Pi_{acc} = \{(s_0,q_0)\alpha_0(s_1,q_1)\alpha_1 \ldots \in \mathrm{Paths}(\mathcal{M}\times\mathcal{G}) \mid q_0, L(s_0), q_1, L(s_1)\ldots \models \alpha\}$$

A Büchi automaton $\mathcal{G}$ is good-for-MDP (GFM) if $\mathbf{Pr}_{\mathcal{M}}^{\max}(\mathcal{L}(\mathcal{G})) = \mathbf{Pr}_{\mathcal{M}\times\mathcal{G}}^{\max}(\Pi_{acc})$ holds for all MDP $\mathcal{M}$ [20]. The inequality "$\ge$" holds for all automata [23, Thm. 1], but the other direction requires, intuitively, that a scheduler on $\mathcal{M} \times \mathcal{G}$ 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 [23]. Limit-deterministic Büchi automata are not GFM in general, for example, $\mathcal{G}_{\mathcal{A}}^{\mathrm{LD}}$ may not be (see Example 5.12).

We fix an arbitrary MDP $\mathcal{M}$ and show that $\mathbf{Pr}_{\mathcal{M}}^{\max}(\mathcal{L}(\mathcal{A})) \le \mathbf{Pr}_{\mathcal{M}\times\mathcal{G}_{\mathcal{A}}^{\mathrm{GFM}}}^{\max}(\Pi_{acc})$. To this end we show that for any finite-memory scheduler

$\mathfrak{S}$ on $\mathcal{M}$ we find a scheduler $\mathfrak{S}'$ on $\mathcal{M} \times \mathcal{G}_{\mathcal{A}}^{\mathrm{GFM}}$ such that $\Pr_{\mathcal{M}}^{\mathfrak{S}}(\mathcal{L}(\mathcal{A})) \leq \Pr_{\mathcal{M}\times\mathcal{G}_{\mathcal{A}}^{\mathrm{GFM}}}^{\mathfrak{S}'}(\Pi_{acc})$. The restriction to finite-memory schedulers is allowed because the maximal probability to satisfy an ω-regular property is always attained by such a scheduler [3, Secs. 10.6.3 and 10.6.4]. Let $\mathcal{M}_{\mathfrak{S}} \times \mathcal{D}$ be the product of the induced finite Markov chain $M_{\mathfrak{S}}$ with $\mathcal{D} = \bigotimes_{1\leq i\leq m} \mathcal{D}_i$, where $\mathcal{D}_i = \det\big(\mathrm{removeFin}(\mathrm{split}(\mathcal{A})[i])\big)$ and "det" is the GBA-determinization procedure from [37], which makes $\mathcal{D}$ a deterministic Rabin automaton. The scheduler $\mathfrak{S}'$ is constructed as follows. It stays inside the initial component of $\mathcal{M} \times \mathcal{G}_{\mathcal{A}}^{\mathrm{GFM}}$ and mimics the action chosen by $\mathfrak{S}$ until the corresponding path in $\mathcal{M}_{\mathfrak{S}} \times \mathcal{D}$ reaches an accepting bottom strongly connected component (BSCC) B. This means that the transitions of $\mathcal{D}$ 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 breakpoint components to which $\mathfrak{S}'$ can safely move.

Lemma 5.11. *Let $\mathfrak{s}$ be a state in an accepting BSCC B of $\mathcal{M}_{\mathfrak{S}} \times \mathcal{D}$ and π_1 be a finite path that reaches $\mathfrak{s}$ from the initial state of $\mathcal{M}_{\mathfrak{S}} \times \mathcal{D}$. Then, there exists $1 \leq i \leq m$ and $Q' \subseteq \theta\big(I, L(\pi_1)\big)$ such that:*

$$\Pr_{\mathfrak{s}}(\{\pi \mid L(\pi) \textit{ is accepted from } (Q', \varnothing, 0) \textit{ in } \mathcal{BP}_i\}) = 1$$

The lemma does not hold if we restrict ourselves to singleton $\{q\} \subseteq \theta\big(I, L(\pi_1)\big)$ (see Example 5.12). Hence, restricting $\delta_{\mathrm{bridge}}^{\mathrm{GFM}}$ to such transitions (as for $\delta_{\mathrm{bridge}}^{\mathrm{LD}}$, see Definition 5.6) would not guarantee the GFM property.

Example 5.12. Consider the automaton $\mathcal{A}$ with states $\{\mathtt{a}_i\mathtt{b}_j \mid i,j \in \{1,2\}\} \cup \{\mathtt{b}_i\mathtt{a}_j \mid i,j \in \{1,2\}\}$, where $\mathtt{a}_i\mathtt{b}_j$ has transitions labeled by a_i to $\mathtt{b}_j\mathtt{a}_1$ and $\mathtt{b}_j\mathtt{a}_2$. Transitions of states $\mathtt{b}_i\mathtt{a}_j$ are defined analogously, and all states in $\{\mathtt{a}_i\mathtt{b}_j \mid i,j \in \{1,2\}\}$ are initial (Fig. 3a shows the transitions of $\mathtt{a}_1\mathtt{b}_1$). All transitions are accepting for a single Büchi condition, and hence $\mathcal{L}(\mathcal{A}) = (\{a_ib_j \mid i,j \in \{1,2\}\})^\omega$.

Consider the Markov chain $\mathcal{M}$ in Fig. 3b (transition probabilities are all 1/2 and omitted in the figure). Clearly, $\Pr_{\mathcal{M}}(\mathcal{L}(\mathcal{A})) = 1$. Figure 3c shows a part of the product of $\mathcal{M}$ with the breakpoint automaton $\mathcal{BP}$ for $\mathcal{A}$ (Definition 5.5) starting from $\big(a_1, (\{\mathtt{a}_1\mathtt{b}_1\}, \varnothing, 0)\big)$. The state $\big(b_2, (\{\mathtt{b}_1\mathtt{a}_1, \mathtt{b}_1\mathtt{a}_2\}, \varnothing, 0)\big)$ is a trap state as $\mathtt{b}_1\mathtt{a}_1$ and $\mathtt{b}_1\mathtt{a}_2$ have no b_2-transition. Hence, $\big(a_1, (\{\mathtt{a}_1\mathtt{b}_1\}, \varnothing, 0)\big)$ generates an accepting path with probability at most 1/2. This is true for all states $\big(s, (P', \varnothing, 0)\big)$ of $\mathcal{M}\times\mathcal{BP}$ where P' is a singleton. But using $\delta_{\mathrm{bridge}}^{\mathrm{LD}}$ to connect initial and accepting components implies that any accepting path sees such a state. Hence, using $\delta_{\mathrm{bridge}}^{\mathrm{LD}}$ to define $\mathcal{G}_{\mathcal{A}}^{\mathrm{GFM}}$ would not guarantee the GFM property.

Using Lemma 5.11 we can define $\mathfrak{S}'$ such that the probability accepting paths under $\mathfrak{S}'$ in $\mathcal{M} \times \mathcal{G}_{\mathcal{A}}^{\mathrm{GFM}}$ is at least as high as that of paths with label in $\mathcal{L}(\mathcal{A})$ in $\mathcal{M}_{\mathfrak{S}}$. This is the non-trivial direction of the GFM property.

Lemma 5.13. *For every finite-memory scheduler $\mathfrak{S}$ on $\mathcal{M}$, there exists a scheduler $\mathfrak{S}'$ on $\mathcal{M} \times \mathcal{G}_{\mathcal{A}}^{\mathrm{GFM}}$ such that:*

$$\Pr_{\mathcal{M}\times\mathcal{G}_{\mathcal{A}}^{\mathrm{GFM}}}^{\mathfrak{S}'}(\Pi_{acc}) \geq \Pr_{\mathcal{M}}^{\mathfrak{S}}(\mathcal{L}(\mathcal{A}))$$

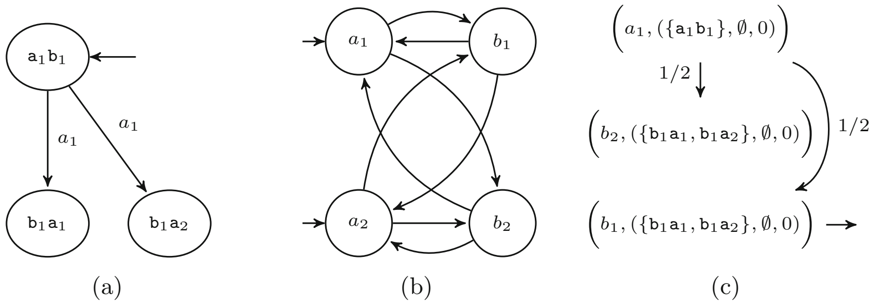
$\mathtt{a}_1\mathtt{b}_1$
a_1
a_1
$\mathtt{b}_1\mathtt{a}_1$
$\mathtt{b}_1\mathtt{a}_2$
a_1
b_1
a_2
b_2
$(a_1, (\{\mathtt{a}_1\mathtt{b}_1\}, \emptyset, 0))$
1/2
$(b_2, (\{\mathtt{b}_1\mathtt{a}_1, \mathtt{b}_1\mathtt{a}_2\}, \emptyset, 0))$
1/2
$(b_1, (\{\mathtt{b}_1\mathtt{a}_1, \mathtt{b}_1\mathtt{a}_2\}, \emptyset, 0))$
(a)
(b)
(c)

Fig. 3. Restricting $\delta^{\mathrm{GFM}}_{\mathrm{bridge}}$ to transitions with endpoints of the form $(s, (\{q\}, \varnothing, 0))$ (similar to $\delta^{\mathrm{LD}}_{\mathrm{bridge}}$) would not guarantee the GFM property (see Example 5.12).

Proposition 5.14. *The automaton $\mathcal{G}^{\mathrm{GFM}}_{\mathcal{A}}$ is* good-for-MDP.

To compute $\mathbf{Pr}^{\max}_{\mathcal{M}}(\mathcal{L}(\mathcal{B}))$ one can translate $\mathcal{B}$ into an equivalent TELA $\mathcal{A}$ in DNF, then construct $\mathcal{G}^{\mathrm{GFM}}_{\mathcal{A}}$ and finally compute $\mathbf{Pr}^{\max}_{\mathcal{M}\times\mathcal{G}^{\mathrm{GFM}}_{\mathcal{A}}}(\varPi_{\mathrm{acc}})$. The automaton $\mathcal{G}^{\mathrm{GFM}}_{\mathcal{A}}$ is single-exponential in the size of $\mathcal{B}$ by Theorem 5.7, and $\mathbf{Pr}^{\max}_{\mathcal{M}\times\mathcal{G}^{\mathrm{GFM}}_{\mathcal{A}}}(\varPi_{\mathrm{acc}})$ can be computed in polynomial time in the size of $\mathcal{M} \times \mathcal{G}^{\mathrm{GFM}}_{\mathcal{A}}$ [3, Thm. 10.127].

Theorem 5.15. *Given a TELA $\mathcal{B}$ (not necessarily in DNF) and an MDP $\mathcal{M}$, the value $\mathbf{Pr}^{\max}_{\mathcal{M}}(\mathcal{L}(\mathcal{B}))$ 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 *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 (the automata are deterministic with a disjunction of parity conditions as acceptance), but it is not the case for arbitrary deterministic TELA, or nondeterministic automata.

Implementation. We compare the following implementations of the constructions discussed above.[2] `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 [14,33]. The removeFin function that is used is an optimized version of Definition 3.4. In `remFin`→`split`α, `split`α→`remFin` and `remFin`→`rewrite`α, the first step is replaced by the corresponding TELA to GBA construction (using Spot's

[2] The source code and data of all experiments are available at [22].

Table 1. Evaluation of benchmarks *random* and *DNF*. Columns "States", "Time" and "Acceptance" refer to the respective median values, where mem-/timeouts are counted as larger than the rest. Values in brackets refer to the subset of input automata for which at least one determinization needed more than 0.5 s (447 (182) automata for benchmark *random* (*DNF*)).

	Algorithm	Timeouts	Memouts	States	Time	Acceptance	Intermediate GBA	
							States	Acceptance
random	Spot	0.5%	9.9%	3,414 (59,525)	<1 (1.5)	10 (17)	71	2
	`remFin`$\rightarrow$`split`α	0.5%	15.2%	8,639 (291,263)	<1 (9.7)	14 (24)	109	2
	`split`$\alpha\rightarrow$`remFin`	0.7%	17.8%	14,037 (522,758)	<1 (21.0)	14 (24)	119	2
	`remFin`$\rightarrow$`rewrite`α	1.6%	18.7%	15,859 (1,024,258)	<1 (40.2)	14 (26)	116	2
	`product`	1.3%	7.9%	3,069 (43,965)	<1 (1.2)	18 (29)		
	`product (no langcover)`	0.7%	9.0%	3,857 (109,908)	<1 (1.1)	24 (38)		
	`limit-det.`	0.0%	0.0%	778 (3,346)	<1 (<1)	1 (1)		
	`limit-det. via GBA`	1.6%	0.3%	463 (1,556)	<1 (1.6)	1 (1)		
	`good-for-MDP`	9.3%	13.4%	5,069 (192.558)	2.0 (139.6)	1 (1)		
	`good-for-MDP via GBA`	5.5%	44.0%	71,200 (–)	836.9 (–)	1 (–)		
DNF	Spot	0.4%	6.2%	5,980 (692,059)	<1 (18.3)	11 (25)	30	3
	`product`	0.0%	3.8%	2,596 (114,243)	<1 (4.6)	13 (24)		

removeFin). The product approach (also implemented using the Spot-library) is called `product` and `product (no langcover)` (without the langcover heuristic). The intermediate GBA are also simplified. The construction $\mathcal{G}_{\mathcal{A}}^{\mathrm{LD}}$ is implemented in `limit-det.`, using the Spot-library and parts of Seminator. 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 $\mathcal{G}_{\mathcal{A}}^{\mathrm{GFM}}$ applied to $\mathcal{A}$ directly, or to the GBA as constructed by Spot. Both constructions `via GBA` are in the worst case double-exponential. No post-processing is applied to any output automaton.

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 s. 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 $T_1, \ldots, T_8$ used to define the acceptance conditions. They are generated using Spot's procedure `random_graph()` by specifying probabilities such that: a triple $(q, a, q') \in Q \times \Sigma \times Q$ 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 of the form $(q, a, q_1), (q, a, q_2)$, with $q_1 \neq q_2$, of the automaton by its number of states.

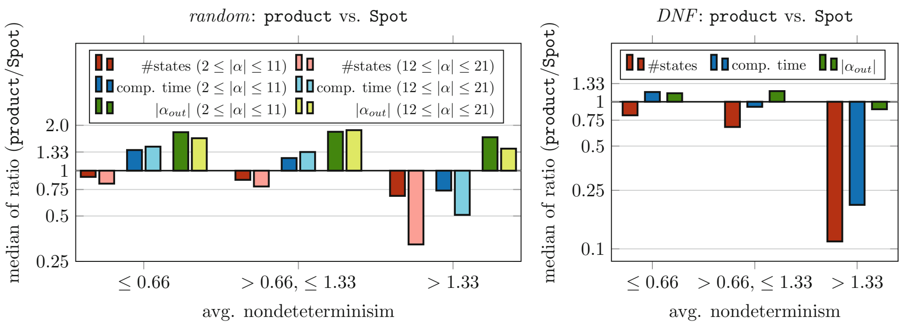

Fig. 4. Comparison of Spot and product, with input automata grouped by the size of the DNF of their acceptance condition and the amount of nondeterminism.

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 comparable. Figure 4 (left) compares Spot and product and partitions the 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) (see [21, Tab. 2]). The graph depicts the median of the ratio (product/Spot) for the measured values. For time- or memouts of Spot (product) we define the ratio as 0 (∞). If both failed, the input is discarded. The number of time- and memouts grows with the amount of nondeterminism and reaches up to 42%. The approach product performs better for automata with more nondeterminism and complex acceptance conditions as the results have fewer states and the computation times are smaller compared to Spot.

The limit-deterministic automata are generally much smaller than the deterministic ones, and `limit-det via GBA.` performs best in this category. However, the construction $\mathcal{G}_{\mathcal{A}}^{\mathrm{LD}}$ (`limit-det.`) resulted in fewer time- and memouts.

For GFM automata we see that computing $\mathcal{G}_{\mathcal{A}}^{\mathrm{GFM}}$ 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 on average with Spot's determinization (see [21, Fig. 7]). Given 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.

The second benchmark (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. Figure 4 (right) shows the median ratio of automata sizes, computation times and acceptance sizes, grouped by the amount of nondeterminism. We do not consider different lengths of acceptance conditions because the subsets of input automata are already relatively small (140–193). Again, `product` performs better for automata with more nondeterminism.

7 Conclusion

We have introduced several new approaches to determinize and limit-determinize automata under the Emerson-Lei acceptance condition. The experimental evaluation shows that in particular the product approach performs very well. Furthermore, we have shown that the complexity of limit-determinizing TELA is single-exponential (in contrast to the double-exponential blow-up for determinization). One of our constructions produces limit-deterministic good-for-MDP automata, which can be used for quantitative probabilistic verification.

This work leads to several interesting questions. The presented constructions would benefit from determinization procedures for GBA which trade a general acceptance condition (rather than Rabin or parity) for a more compact state-space of the output. Similarly, translations from LTL to compact, nondeterministic TELA would allow them to be embedded into (probabilistic) model-checking tools for LTL (a first step in this direction is made in [27]). It would be interesting to study, in general, what properties can be naturally encoded directly into nondeterministic TELA. Another open point is to evaluate the good-for-MDP automata in the context of probabilistic model checking in practice.

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

References

1. Babiak, T., et al.: The Hanoi omega-automata format. In: Kroening, D., Păsăreanu, C.S. (eds.) CAV 2015. LNCS, vol. 9206, pp. 479–486. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21690-4_31
2. Baier, C., Blahoudek, F., Duret-Lutz, A., Klein, J., Müller, D., Strejček, J.: Generic emptiness check for fun and profit. In: Chen, Y.-F., Cheng, C.-H., Esparza, J. (eds.) ATVA 2019. LNCS, vol. 11781, pp. 445–461. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-31784-3_26
3. Baier, C., Katoen, J.P.: Principles of Model Checking. Representation and Mind Series, The MIT Press, Cambridge (2008)
4. Ben-Ari, M.: Principles of the Spin Model Checker. Springer, London (2008). https://doi.org/10.1007/978-1-84628-770-1
5. Blahoudek, F.: Automata for formal methods: little steps towards perfection. Ph.D. thesis, Masaryk University, Faculty of Informatics (2018)
6. Blahoudek, F., Duret-Lutz, A., Klokocka, M., Kretínský, M., Strejcek, J.: Seminator: a tool for semi-determinization of omega-automata. In: International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR). EPiC Series in Computing (2017)

7. Blahoudek, F., Major, J., Strejček, J.: LTL to smaller self-loop alternating automata and back. In: Hierons, R.M., Mosbah, M. (eds.) ICTAC 2019. LNCS, vol. 11884, pp. 152–171. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-32505-3_10
8. Bloemen, V., Duret-Lutz, A., van de Pol, J.: Model checking with generalized Rabin and Fin-less automata. Int. J. Softw. Tools Technol. Transfer **21**(3), 307–324 (2019)
9. Boker, U.: Why these automata types? In: Logic for Programming, Artificial Intelligence and Reasoning (LPAR). EPiC Series in Computing (2018)
10. Chatterjee, K., Gaiser, A., Křetínský, J.: Automata with generalized Rabin pairs for probabilistic model checking and LTL synthesis. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 559–575. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_37
11. Courcoubetis, C., Yannakakis, M.: The complexity of probabilistic verification. J. ACM **42**(4), 857–907 (1995)
12. Couvreur, J.-M.: On-the-fly verification of linear temporal logic. In: Wing, J.M., Woodcock, J., Davies, J. (eds.) FM 1999. LNCS, vol. 1708, pp. 253–271. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48119-2_16
13. Duret-Lutz, A.: Contributions to LTL and ω-automata for model checking. Habilitation thesis, Université Pierre et Marie Curie (2017)
14. Duret-Lutz, A., Lewkowicz, A., Fauchille, A., Michaud, T., Renault, É., Xu, L.: Spot 2.0—a framework for LTL and ω-automata manipulation. In: Artho, C., Legay, A., Peled, D. (eds.) ATVA 2016. LNCS, vol. 9938, pp. 122–129. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46520-3_8
15. Duret-Lutz, A., Poitrenaud, D., Couvreur, J.-M.: On-the-fly emptiness check of transition-based Streett automata. In: Liu, Z., Ravn, A.P. (eds.) ATVA 2009. LNCS, vol. 5799, pp. 213–227. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04761-9_17
16. Emerson, E.A., Lei, C.L.: Modalities for model checking: branching time logic strikes back. Sci. Comput. Program. **8**(3), 275–306 (1987)
17. Esparza, J., Křetínský, J., Sickert, S.: One theorem to rule them all: a unified translation of LTL into ω-automata. In: Logic in Computer Science (LICS). ACM (2018)
18. Giannakopoulou, D., Lerda, F.: From states to transitions: improving translation of LTL formulae to Büchi automata. In: Peled, D.A., Vardi, M.Y. (eds.) FORTE 2002. LNCS, vol. 2529, pp. 308–326. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-36135-9_20
19. Hahn, E.M., Li, G., Schewe, S., Turrini, A., Zhang, L.: Lazy probabilistic model checking without determinisation. In: Concurrency Theory (CONCUR) (2015)
20. Hahn, E.M., Perez, M., Schewe, S., Somenzi, F., Trivedi, A., Wojtczak, D.: Good-for-MDPs automata for probabilistic analysis and reinforcement learning. In: Biere, A., Parker, D. (eds.) TACAS 2020. LNCS, vol. 12078, pp. 306–323. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45190-5_17
21. John, T., Jantsch, S., Baier, C., Klüppelholz, S.: Determinization and limit-determinization of Emerson-Lei automata. arXiv:2106.15892 [cs], June 2021
22. John, T., Jantsch, S., Baier, C., Klüppelholz, S.: Determinization and limit-determinization of Emerson-Lei automata. Supplementary material (ATVA 2021) (2021). https://doi.org/10.6084/m9.figshare.14838654.v2
23. Klein, J., Müller, D., Baier, C., Klüppelholz, S.: Are good-for-games automata good for probabilistic model checking? In: Dediu, A.-H., Martín-Vide, C., Sierra-Rodríguez, J.-L., Truthe, B. (eds.) LATA 2014. LNCS, vol. 8370, pp. 453–465. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-04921-2_37

24. Křetínský, J., Meggendorfer, T., Sickert, S.: Owl: a library for ω-words, automata, and LTL. In: Lahiri, S.K., Wang, C. (eds.) ATVA 2018. LNCS, vol. 11138, pp. 543–550. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01090-4_34
25. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_47
26. Löding, C., Pirogov, A.: Determinization of Büchi automata: unifying the approaches of Safra and Muller-Schupp. In: International Colloquium on Automata, Languages, and Programming (ICALP). LIPIcs (2019)
27. Major, J., Blahoudek, F., Strejček, J., Sasaráková, M., Zbončáková, T.: `ltl3tela`: LTL to small deterministic or nondeterministic Emerson-Lei automata. In: Chen, Y.-F., Cheng, C.-H., Esparza, J. (eds.) ATVA 2019. LNCS, vol. 11781, pp. 357–365. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-31784-3_21
28. Miyano, S., Hayashi, T.: Alternating finite automata on ω-words. Theoret. Comput. Sci. **32**(3), 321–330 (1984)
29. Müller, D.: Alternative automata-based approaches to probabilistic model checking. Ph.D. thesis, Technische Universität Dresden, November 2019
30. Müller, D., Sickert, S.: LTL to deterministic Emerson-Lei automata. In: Games, Automata, Logics and Formal Verification (GandALF). EPTCS (2017)
31. Muller, D.E., Schupp, P.E.: Simulating alternating tree automata by nondeterministic automata: new results and new proofs of the theorems of Rabin, McNaughton and Safra. Theoret. Comput. Sci. **141**(1), 69–107 (1995)
32. Pnueli, A., Rosner, R.: On the synthesis of a reactive module. In: Symposium on Principles of Programming Languages (POPL). Association for Computing Machinery (ACM), New York, NY, USA (1989)
33. Redziejowski, R.R.: An improved construction of deterministic omega-automaton using derivatives. Fund. Inform. **119**(3–4), 393–406 (2012)
34. Renkin, F., Duret-Lutz, A., Pommellet, A.: Practical "paritizing" of Emerson-Lei automata. In: Hung, D.V., Sokolsky, O. (eds.) ATVA 2020. LNCS, vol. 12302, pp. 127–143. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-59152-6_7
35. Safra, S., Vardi, M.Y.: On omega-automata and temporal logic. In: Symposium on Theory of Computing (STOC). Association for Computing Machinery (ACM), New York, NY, USA (1989)
36. Safra, S.: Complexity of automata on infinite objects. Ph.D. thesis, Weizmann Institute of Science, Rehovot, Israel (1989)
37. Schewe, S., Varghese, T.: Tight bounds for the determinisation and complementation of generalised Büchi automata. In: Chakraborty, S., Mukund, M. (eds.) ATVA 2012. LNCS, pp. 42–56. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33386-6_5
38. Sickert, S., Esparza, J., Jaax, S., Křetínský, J.: Limit-deterministic Büchi automata for linear temporal logic. In: Chaudhuri, S., Farzan, A. (eds.) CAV 2016. LNCS, vol. 9780, pp. 312–332. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-41540-6_17
39. Vardi, M.Y.: Automatic verification of probabilistic concurrent finite state programs. In: Symposium on Foundations of Computer Science (SFCS) (1985)