

# Deciding What is Good-for-MDPs

Sven Schewe  

University of Liverpool, UK

Qiyi Tang  

University of Liverpool, UK

Tansholpan Zhanabekova  

University of Liverpool, UK

---

## Abstract


Nondeterministic good-for-MDPs (GFM) automata are for MDP model checking and reinforcement learning what good-for-games automata are for reactive synthesis: a more compact alternative to deterministic automata that displays nondeterminism, but only so much that it can be resolved locally, such that a syntactic product can be analysed. GFM has recently been introduced as a property for reinforcement learning, where the simpler Büchi acceptance conditions it allows to use is key. However, while there are classic and novel techniques to obtain automata that are GFM, there has not been a decision procedure for checking whether or not an automaton is GFM. We show that GFM-ness is decidable and provide an EXPTIME decision procedure as well as a PSPACE-hardness proof.

**2012 ACM Subject Classification** Theory of computation → Automata over infinite objects; Mathematics of computing → Markov processes

**Keywords and phrases** Büchi automata, Markov Decision Processes, Omega-regular objectives, Reinforcement learning

**Digital Object Identifier** 10.4230/LIPIcs.CONCUR.2023.35

**Related Version** Full version hosted on arXiv [16].

**Funding**  This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement 956123 (FOCETA).

**Acknowledgements** We thank the anonymous reviewers of this paper for their constructive feedback.

## 1 Introduction

Omega-automata [20, 12] are formal acceptors of  $\omega$ -regular properties, which often result from translating a formula from temporal logics like LTL [14], as a specification for desired model properties in quantitative model checking and strategy synthesis [3], and reinforcement learning [19].

Especially for reinforcement learning, having a simple Büchi acceptance mechanism has proven to be a breakthrough [8], which led to the definition of the “good-for-MDPs” property in [9]. Just like for good-for-games automata in strategy synthesis for strategic games [10], there is a certain degree of nondeterminism allowed when using a nondeterministic automaton on the syntactic product with an MDP to learn how to control it, or to apply quantitative model checking. Moreover, the degree of freedom available to control MDPs is higher than the degree of freedom for controlling games. In particular, this always allows for using nondeterministic automata with a Büchi acceptance condition, both when using the classically used suitable limit deterministic automata [21, 6, 7, 17, 8] and for alternative GFM automata like the slim automata from [9].

This raises the question of whether or not an automaton is good-for-MDPs. While [9] has introduced the concept, there is not yet a decision procedure for checking the GFM-ness



© Sven Schewe, Qiyi Tang, Tansholpan Zhanabekova;  
licensed under Creative Commons License CC-BY 4.0

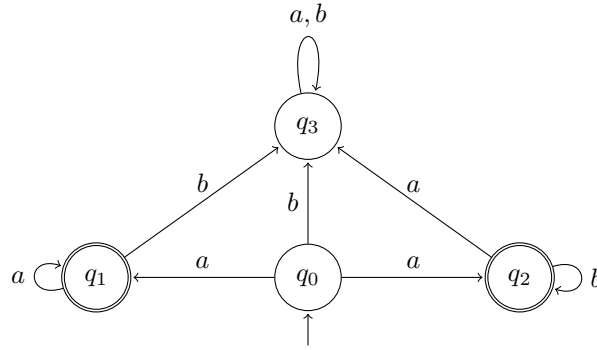
34th International Conference on Concurrency Theory (CONCUR 2023).

Editors: Guillermo A. Pérez and Jean-François Raskin; Article No. 35; pp. 35:1–35:17



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



■ **Figure 1** A nondeterministic Büchi word automaton over  $\{a, b\}$ . This NBW is complete and accepts the language  $\{a^\omega, ab^\omega\}$ .

of an automaton, let alone for the complexity of this test.

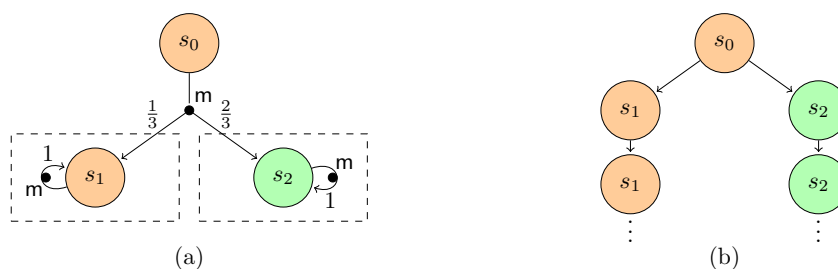
We will start by showing that the problem of deciding GFM-ness is PSPACE-hard by a reduction from the NFA universality problem [18]. We then define the auxiliary concept of *qualitative GFM*, QGFM, which relaxes the requirements for GFM to qualitative properties, and develop an automata based EXPTIME decision procedure for QGFM. This decision procedure is constructive in that it can provide a counter-example for QGFM-ness when such a counter-example exists. We then use it to provide a decision procedure for GFM-ness that uses QGFM queries for all states of the candidate automaton. Finally, we show that the resulting criterion for GFM-ness is also a necessary criterion for QGFM-ness, which leads to a collapse of the two concepts. This entails that the EXPTIME decision procedure we developed to test QGFM-ness can be used to decide GFM-ness, while our PSPACE-hardness proofs extend to QGFM-ness.

## 2 Preliminaries

We write  $\mathbb{N}$  for the set of nonnegative integers. Let  $S$  be a finite set. We denote by  $\text{Distr}(S)$  the set of probability distributions on  $S$ . For a distribution  $\mu \in \text{Distr}(S)$  we write  $\text{support}(\mu) = \{s \in S \mid \mu(s) > 0\}$  for its support. The cardinal of  $S$  is denoted  $|S|$ . We use  $\Sigma$  to denote a finite alphabet. We denote by  $\Sigma^*$  the set of finite words over  $\Sigma$  and  $\Sigma^\omega$  the set of  $\omega$ -words over  $\Sigma$ . We use the standard notions of prefix and suffix of a word. By  $w\alpha$  we denote the concatenation of a finite word  $w$  and an  $\omega$ -word  $\alpha$ . If  $L \subseteq \Sigma^\omega$  and  $w \in \Sigma^*$ , the residual language (left quotient of  $L$  by  $w$ ), denoted by  $w^{-1}L$  is defined as  $\{\alpha \in \Sigma^\omega \mid w\alpha \in L\}$ .

### 2.1 Automata

A nondeterministic Büchi **word automaton** (NBW) is a tuple  $\mathcal{A} = (\Sigma, Q, q_0, \delta, F)$ , where  $\Sigma$  is a finite alphabet,  $Q$  is a finite set of states,  $q_0 \in Q$  is the initial state,  $\delta : Q \times \Sigma \rightarrow 2^Q$  is the transition function, and  $F \subseteq Q$  is the set of final (or accepting) states. An NBW is *complete* if  $\delta(q, \sigma) \neq \emptyset$  for all  $q \in Q$  and  $\sigma \in \Sigma$ . Unless otherwise mentioned, we consider complete NBWs in this paper. A run  $r$  of  $\mathcal{A}$  on  $w \in \Sigma^\omega$  is an  $\omega$ -word  $q_0, w_0, q_1, w_1, \dots \in (Q \times \Sigma)^\omega$  such that  $q_i \in \delta(q_{i-1}, w_{i-1})$  for all  $i > 0$ . An NBW  $\mathcal{A}$  accepts exactly those runs, in which at least one of the infinitely often occurring states is in  $F$ . A word in  $\Sigma^\omega$  is accepted by the automaton if it has an accepting run, and the language of an automaton, denoted  $\mathcal{L}(\mathcal{A})$ , is the set of accepted words in  $\Sigma^\omega$ . An example of an NBW is given in Figure 1.



■ **Figure 2** (a) An MDP with initial state  $s_0$ . The set of labels is  $\{a, b\}$  and the labelling function for the MDP is as follows:  $\ell(s_0) = \ell(s_1) = a$ ,  $\ell(s_2) = b$ . The labels are indicated by different colours. Since each state has only one available action  $m$ , the MDP is actually an MC. There are two end-components in this MDP labelled with the two dashed boxes. (b) The tree that stems from unravelling of the MC with initial state  $s_0$  on the left, while disregarding probabilities.

Let  $C \subset \mathbb{N}$  be a finite set of colours. A nondeterministic parity **word automaton** (NPW) is a tuple  $P = (\Sigma, Q, q_0, \delta, \pi)$ , where  $\Sigma$ ,  $Q$ ,  $q_0$  and  $\delta$  have the same definitions as for NBW, and  $\pi : Q \rightarrow C$  is the priority (colouring) function that maps each state to a priority (colour). A run is accepting if and only if the highest priority (colour) occurring infinitely often in the infinite sequence is even. Similar to NBW, a word in  $\Sigma^\omega$  is accepted by an NPW if it has an accepting run, and the language of the NPW  $P$ , denoted  $\mathcal{L}(P)$ , is the set of accepted words in  $\Sigma^\omega$ . An NBW is a special case of an NPW where  $\pi(q) = 2$  for  $q \in F$  and  $\pi(q) = 1$  otherwise with  $C = \{1, 2\}$ .

A nondeterministic word automaton is *deterministic* if the transition function  $\delta$  maps each state and letter pair to a singleton set, a set consisting of a single state.

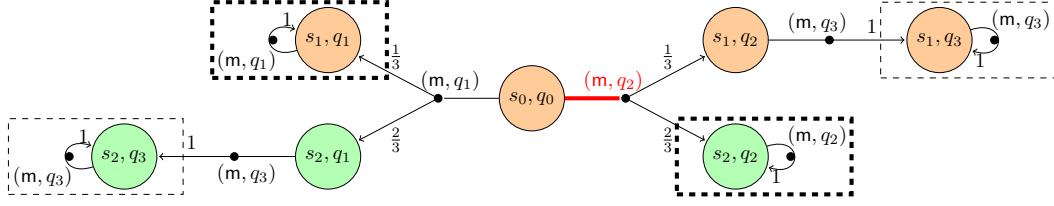
A nondeterministic automaton is called *good-for-games* (GFG) if it only relies on a limited form of nondeterminism: GFG automata can make their decision of how to resolve their nondeterministic choices on the history at any point of a run rather than using the knowledge of the complete word as a nondeterministic automaton normally would without changing their language. They can be characterised in many ways, including as automata that simulate deterministic automata. The NBW in Figure 1 is neither GFG nor good-for-MDPs (GFM) as shown later.

## 2.2 Markov Decision Processes (MDPs)

A (*finite, state-labelled*) *Markov decision process* (MDP) is a tuple  $\langle S, \text{Act}, P, \Sigma, \ell \rangle$  consisting of a finite set  $S$  of states, a finite set  $\text{Act}$  of actions, a partial function  $P : S \times \text{Act} \rightarrow \text{Distr}(S)$  denoting the probabilistic transition and a labelling function  $\ell : S \rightarrow \Sigma$ . The set of available actions in a state  $s$  is  $\text{Act}(s) = \{m \in \text{Act} \mid P(s, m) \text{ is defined}\}$ . An MDP is a (labelled) Markov chain (MC) if  $|\text{Act}(s)| = 1$  for all  $s \in S$ .

An infinite *run* (*path*) of an MDP  $\mathcal{M}$  is a sequence  $s_0 m_1 \dots \in (S \times \text{Act})^\omega$  such that  $P(s_i, m_{i+1})$  is defined and  $P(s_i, m_{i+1})(s_{i+1}) > 0$  for all  $i \geq 0$ . A finite run is a finite such sequence. Let  $\Omega(\mathcal{M})$  ( $\text{Paths}(\mathcal{M})$ ) denote the set of (finite) runs in  $\mathcal{M}$  and  $\Omega(\mathcal{M})_s$  ( $\text{Paths}(\mathcal{M})_s$ ) denote the set of (finite) runs in  $\mathcal{M}$  starting from  $s$ . Abusing the notation slightly, for an infinite run  $r = s_0 m_1 s_1 m_2 \dots$  we write  $\ell(r) = \ell(s_0)\ell(s_1)\dots \in \Sigma^\omega$ .

A *strategy* for an MDP is a function  $\mu : \text{Paths}(\mathcal{M}) \rightarrow \text{Distr}(\text{Act})$  that, given a finite run  $r$ , returns a probability distribution on all the available actions at the last state of  $r$ . A *memoryless* (positional) strategy for an MDP is a function  $\mu : S \rightarrow \text{Distr}(\text{Act})$  that, given a state  $s$ , returns a probability distribution on all the available actions at that state. The set



■ **Figure 3** An example of a product MDP  $\mathcal{M} \times \mathcal{N}$  with initial state  $(s_0, q_0)$  and  $F^\times = \{(s_1, q_1), (s_2, q_1), (s_1, q_2), (s_2, q_2)\}$  where  $\mathcal{M}$  is the MDP (MC) in Figure 2(a) and  $\mathcal{N}$  is the NBW in Figure 1. The states  $(s_0, q_0)$ ,  $(s_1, q_1)$ ,  $(s_1, q_2)$  and  $(s_1, q_3)$  are labelled with  $a$  while all the other states are labelled with  $b$ . Again, the four end-components of the MDP are labelled with dashed boxes; the upper left and lower right end-components are accepting (highlighted in thick dashed boxes).

of runs  $\Omega(\mathcal{M})_s^\mu$  is a subset of  $\Omega(\mathcal{M})_s$  that correspond to strategy  $\mu$  and initial state  $s$ . Given a memoryless/finite-memory strategy  $\mu$  for  $\mathcal{M}$ , an MC  $(\mathcal{M})_\mu$  is induced [3, Section 10.6].

A *sub-MDP* of  $\mathcal{M}$  is an MDP  $\mathcal{M}' = \langle S', \text{Act}', P', \Sigma, \ell' \rangle$ , where  $S' \subseteq S$ ,  $\text{Act}' \subseteq \text{Act}$  is such that  $\text{Act}'(s) \subseteq \text{Act}(s)$  for every  $s \in S'$ , and  $P'$  and  $\ell'$  are analogous to  $P$  and  $\ell$  when restricted to  $S'$  and  $\text{Act}'$ . In particular,  $\mathcal{M}'$  is closed under probabilistic transitions, i.e. for all  $s \in S'$  and  $m \in \text{Act}'$  we have that  $P'(s, m)(s') > 0$  implies that  $s' \in S'$ . An *end-component* [1, 3] of an MDP  $\mathcal{M}$  is a sub-MDP  $\mathcal{M}'$  of  $\mathcal{M}$  such that its underlying graph is strongly connected and it has no outgoing transitions. An example MDP is presented in Figure 2(a).

A strategy  $\mu$  and an initial state  $s \in S$  induce a standard probability measure on sets of infinite runs, see, e.g., [3]. Such measurable sets of infinite runs are called events or objectives. We write  $\text{Pr}_s^\mu$  for the probability of an event  $E \subseteq sS^\omega$  of runs starting from  $s$ .

► **Theorem 1.** (*End-Component Properties* [1, 3]). *Once an end-component  $E$  of an MDP is entered, there is a strategy that visits every state-action combination in  $E$  with probability 1 and stays in  $E$  forever. Moreover, for every strategy the union of the end-components is visited with probability 1.*

### 2.3 The Product of MDPs and Automata

Given an MDP  $\mathcal{M} = \langle S, \text{Act}, P, \Sigma, \ell \rangle$  with initial state  $s_0 \in S$  and an NBW  $\mathcal{N} = \langle \Sigma, Q, \delta, q_0, F \rangle$ , we want to compute an optimal strategy satisfying the objective that the run of  $\mathcal{M}$  is in the language of  $\mathcal{N}$ . We define the semantic satisfaction probability for  $\mathcal{N}$  and a strategy  $\mu$  from state  $s \in S$  as:  $\text{PSem}_{\mathcal{N}}^{\mathcal{M}}(s, \mu) = \text{Pr}_s^\mu\{r \in \Omega_s^\mu : \ell(r) \in \mathcal{L}(\mathcal{N})\}$  and  $\text{PSem}_{\mathcal{N}}^{\mathcal{M}}(s) = \sup_\mu (\text{PSem}_{\mathcal{N}}^{\mathcal{M}}(s, \mu))$ . In the case that  $\mathcal{M}$  is an MC, we simply have  $\text{PSem}_{\mathcal{N}}^{\mathcal{M}}(s) = \text{Pr}_s\{r \in \Omega_s : \ell(r) \in \mathcal{L}(\mathcal{N})\}$ .

The *product* of  $\mathcal{M}$  and  $\mathcal{N}$  is an MDP  $\mathcal{M} \times \mathcal{N} = \langle S \times Q, \text{Act} \times Q, P^\times, \Sigma, \ell^\times \rangle$  augmented with the initial state  $(s_0, q_0)$  and the Büchi acceptance condition  $F^\times = \{(s, q) \in S \times Q \mid q \in F\}$ . The labelling function  $\ell^\times$  maps each state  $(s, q) \in S \times Q$  to  $\ell(s)$ .

We define the partial function  $P^\times : (S \times Q) \times (\text{Act} \times Q) \rightarrow \text{Distr}(S \times Q)$  as follows: for all  $(s, m) \in \text{support}(P)$ ,  $s' \in S$  and  $q, q' \in Q$ , we have  $P^\times((s, q), (m, q'))((s', q')) = P(s, m)(s')$  for all  $q' \in \delta(q, \ell(s))$ <sup>1</sup>.

<sup>1</sup> When  $\mathcal{N}$  is complete, there always exists a state  $q'$  such that  $q' \in \delta(q, \ell(s))$ .

We define the syntactic satisfaction probability for the product MDP and a strategy  $\mu^\times$  from a state  $(s, q)$  as:  $\text{PSyn}_{\mathcal{N}}^{\mathcal{M}}((s, q), \mu^\times) = \Pr_{s, q}^{\mu^\times} \{r \in \Omega_{s, q}^{\mu^\times} : \ell^\times(r) \in \mathcal{L}(\mathcal{N})\}^2$  and  $\text{PSyn}_{\mathcal{N}}^{\mathcal{M}}(s) = \sup_{\mu^\times} (\text{PSyn}_{\mathcal{N}}^{\mathcal{M}}((s, q_0), \mu^\times))$ . The set of actions is  $\text{Act}$  in the MDP  $\mathcal{M}$  while it is  $\text{Act} \times Q$  in the product MDP. This makes  $\text{PSem}$  and  $\text{PSyn}$  potentially different. In general,  $\text{PSyn}_{\mathcal{N}}^{\mathcal{M}}(s) \leq \text{PSem}_{\mathcal{N}}^{\mathcal{M}}(s)$  for all  $s \in S$ , because accepting runs can only occur on accepted words. If  $\mathcal{N}$  is deterministic,  $\text{PSyn}_{\mathcal{N}}^{\mathcal{M}}(s) = \text{PSem}_{\mathcal{N}}^{\mathcal{M}}(s)$  holds for all  $s \in S$ .

End-components and runs are defined for products just like for MDPs. A run of  $\mathcal{M} \times \mathcal{N}$  is accepting if it satisfies the product's acceptance condition. An accepting end-component of  $\mathcal{M} \times \mathcal{N}$  is an end-component which contains some states in  $F^\times$ .

An example of a product MDP is presented in Figure 3. It is the product of the MDP in Figure 2(a) and the NBW in Figure 1. Since  $\ell(r)$  is in the language of the NBW for every run  $r$  of the MDP, we have  $\text{PSem}_{\mathcal{N}}^{\mathcal{M}}(s_0) = 1$ . However, the syntactic satisfaction probability  $\text{PSyn}_{\mathcal{N}}^{\mathcal{M}}(s_0) = \frac{2}{3}$  is witnessed by the memoryless strategy which chooses the action  $(m, q_2)$  at the initial state. We do not need to specify the strategy for the other states since there is only one available action for any remaining state. According to the following definition, the NBW in Figure 1 is not GFM as witnessed by the MDP in Figure 2(a).

► **Definition 2.** [9] *An NBW  $\mathcal{N}$  is good-for-MDPs (GFM) if, for all finite MDPs  $\mathcal{M}$  with initial state  $s_0$ ,  $\text{PSyn}_{\mathcal{N}}^{\mathcal{M}}(s_0) = \text{PSem}_{\mathcal{N}}^{\mathcal{M}}(s_0)$  holds.*

### 3 PSPACE-Hardness

We show that the problem of checking whether or not a given NBW is GFM is PSPACE-hard. The reduction is from the NFA universality problem, which is known to be PSPACE-complete [18]. Given an NFA  $\mathcal{A}$ , the NFA universality problem asks whether  $\mathcal{A}$  accepts every string, that is, whether  $\mathcal{L}(\mathcal{A}) = \Sigma^*$ .

We first give an overview of how this reduction works. Given a complete NFA  $\mathcal{A}$ , we first construct an NBW  $\mathcal{A}_f$  (Definition 4) which can be shown to be GFM (Lemma 6). Using this NBW  $\mathcal{A}_f$ , we then construct another NBW  $\text{fork}(\mathcal{A}_f)$  (Definition 7). We complete the argument by showing in Lemma 8 that the NBW  $\text{fork}(\mathcal{A}_f)$  is GFM if, and only if,  $\mathcal{A}$  accepts the universal language.

We start with the small observation that ‘for all finite MDPs’ in Definition 2 can be replaced by ‘for all finite MCs’.

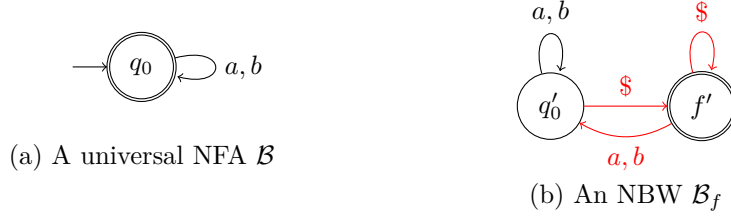
► **Theorem 3.** *An NBW  $\mathcal{N}$  is GFM iff, for all finite MCs  $\mathcal{M}$  with initial state  $s_0$ ,  $\text{PSyn}_{\mathcal{N}}^{\mathcal{M}}(s_0) = \text{PSem}_{\mathcal{N}}^{\mathcal{M}}(s_0)$  holds.*

**Proof. ‘if’:** This is the case because there is an optimal finite memory control for an MDP  $\mathcal{M}$ , e.g. by using a language equivalent DPW  $\mathcal{P}$  [13] and using its memory structure as finite memory. That is, we obtain an MC  $\mathcal{M}'$  by applying an optimal memoryless strategy for  $\mathcal{M} \times \mathcal{P}$  [4]. Naturally, if  $\mathcal{N}$  satisfies the condition for  $\mathcal{M}'$ , then it also satisfies it for  $\mathcal{M}$ .

‘only if’: MCs are just special cases of MDPs. ◀

Given a complete NFA  $\mathcal{A}$ , we construct an NBW  $\mathcal{A}_f$  by introducing a new letter  $\$$  and a new state. As an example, given an NFA (DFA)  $\mathcal{B}$  in Figure 4(a), we obtain an NBW  $\mathcal{B}_f$  in Figure 4(b). It is easy to see that  $\mathcal{L}(\mathcal{B}) = \Sigma^*$  where  $\Sigma = \{a, b\}$ .

<sup>2</sup> Let  $\text{inf}(r)$  be the set of states that appears infinite often in a run  $r$ . We also have  $\text{PSyn}_{\mathcal{N}}^{\mathcal{M}}((s, q), \mu^\times) = \Pr_{s, q}^{\mu^\times} \{r \in \Omega_{s, q}^{\mu^\times} : \text{inf}(r) \cap F^\times \neq \emptyset\}$ .



■ **Figure 4** (a)  $\mathcal{B}$  is a complete universal NFA. Let  $\Sigma = \{a, b\}$ . We have  $\mathcal{L}(\mathcal{A}) = \Sigma^*$ . (b) On the right is the corresponding complete NBW  $\mathcal{B}_f$ . The new final state  $f$  and the added transitions are highlighted in red. We have  $\mathcal{L}(\mathcal{B}_f) = \{w_1\$w_2\$w_3\$ \dots\}$  where  $w_i \in \Sigma^*$ .

► **Definition 4.** Given a complete NFA  $\mathcal{A} = (\Sigma, Q, q_0, \delta, F)$ , we define the NBW  $\mathcal{A}_f = (\Sigma_{\$}, Q_f, q_0, \delta_f, \{f\})$  with  $\Sigma_{\$} = \Sigma \cup \{\$\}$  and  $Q_f = Q \cup \{f\}$  for a fresh letter  $\$ \notin \Sigma$  and a fresh state  $f \notin Q$ , and with  $\delta_f(q, \sigma) = \delta(q, \sigma)$  for all  $q \in Q$  and  $\sigma \in \Sigma$ ,  $\delta_f(q, \$) = \{f\}$  for all  $q \in F$ ,  $\delta_f(q, \$) = \{q_0\}$  for all  $q \in Q \setminus F$ , and  $\delta_f(f, \sigma) = \delta_f(q_0, \sigma)$  for all  $\sigma \in \Sigma_{\$}$ .

The language of  $\mathcal{A}_f$  consists of all words of the form  $w_1\$w'_1\$w_2\$w'_2\$w_3\$w'_3\$ \dots$  such that, for all  $i \in \mathbb{N}$ ,  $w_i \in \Sigma_{\$}^*$  and  $w'_i \in \mathcal{L}(\mathcal{A})$ . This provides the following lemma.

► **Lemma 5.** Given two NFAs  $\mathcal{A}$  and  $\mathcal{B}$ ,  $\mathcal{L}(\mathcal{B}) \subseteq \mathcal{L}(\mathcal{A})$  if, and only if,  $\mathcal{L}(\mathcal{B}_f) \subseteq \mathcal{L}(\mathcal{A}_f)$ .

The following lemma simply states that the automaton  $\mathcal{A}_f$  from the above construction is GFM. This lemma is technical and is key to prove Lemma 8, the main lemma, of this section.

► **Lemma 6.** For every complete NFA  $\mathcal{A}$ ,  $\mathcal{A}_f$  is GFM.

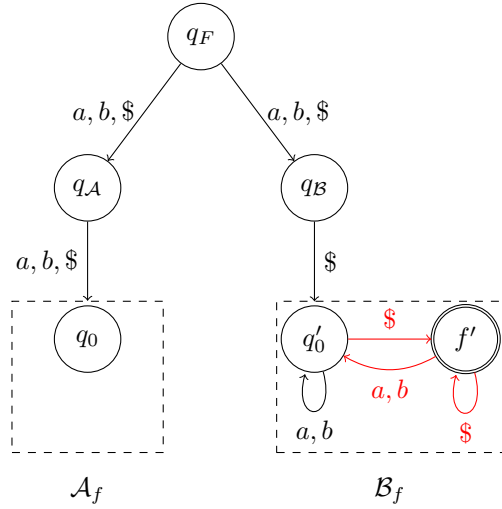
**Proof.** Consider an arbitrary MC  $\mathcal{M}$  with initial state  $s_0$ . We show that  $\mathcal{A}_f$  is good for  $\mathcal{M}$ , that is,  $\text{PSem}_{\mathcal{A}_f}^{\mathcal{M}}(s_0) = \text{PSyn}_{\mathcal{A}_f}^{\mathcal{M}}(s_0)$ . It suffices to show  $\text{PSyn}_{\mathcal{A}_f}^{\mathcal{M}}(s_0) \geq \text{PSem}_{\mathcal{A}_f}^{\mathcal{M}}(s_0)$  since by definition the converse  $\text{PSem}_{\mathcal{A}_f}^{\mathcal{M}}(s_0) \geq \text{PSyn}_{\mathcal{A}_f}^{\mathcal{M}}(s_0)$  always holds.

First, we construct a language equivalent deterministic Büchi automaton (DBW)  $\mathcal{D}_f$  by first determinising the NFA  $\mathcal{A}$  to a DFA  $\mathcal{D}$  by a standard subset construction and then obtain  $\mathcal{D}_f$  by Definition 4. Since  $\mathcal{L}(\mathcal{A}_f) = \mathcal{L}(\mathcal{D}_f)$ , we have that  $\text{PSem}_{\mathcal{A}_f}^{\mathcal{M}}(s_0) = \text{PSem}_{\mathcal{D}_f}^{\mathcal{M}}(s_0)$ . In addition, since  $\mathcal{D}_f$  is deterministic, we have  $\text{PSem}_{\mathcal{D}_f}^{\mathcal{M}}(s_0) = \text{PSyn}_{\mathcal{D}_f}^{\mathcal{M}}(s_0)$ .

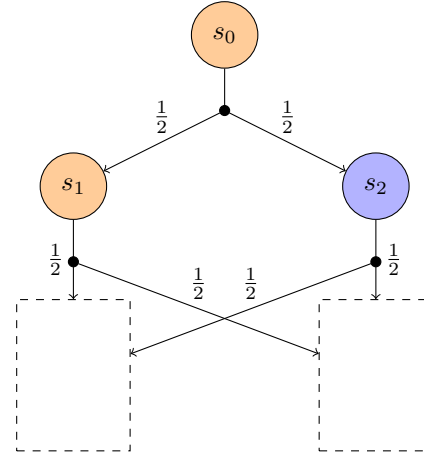
It remains to show  $\text{PSyn}_{\mathcal{A}_f}^{\mathcal{M}}(s_0) \geq \text{PSyn}_{\mathcal{D}_f}^{\mathcal{M}}(s_0)$ . For that, it suffices to show that for an arbitrary accepting run  $r$  of  $\mathcal{M} \times \mathcal{D}_f$ , there is a strategy for  $\mathcal{M} \times \mathcal{A}_f$  such that  $r'$  (the corresponding run in the product) is accepting in  $\mathcal{M} \times \mathcal{A}_f$  where the projections of  $r$  and  $r'$  on  $\mathcal{M}$  are the same.

Consider an accepting run of  $\mathcal{M} \times \mathcal{D}_f$ . Before entering an accepting end-component of  $\mathcal{M} \times \mathcal{D}_f$ , any strategy to resolve the nondeterminism in  $\mathcal{M} \times \mathcal{A}_f$  (thus  $\mathcal{A}_f$ ) can be used. This will not block  $\mathcal{A}_f$ , as it is a complete automaton, and  $\mathcal{A}_f$  is essentially re-set whenever it reads a  $\$$ . Once an accepting end-component of  $\mathcal{M} \times \mathcal{D}_f$  is entered, there must exist a word of the form  $\$w\$,$  where  $w \in \mathcal{L}(\mathcal{D})$  (and thus  $w \in \mathcal{L}(\mathcal{A})$ ), which is a possible initial sequence of letters produced from some state  $m$  of  $\mathcal{M} \times \mathcal{D}_f$  in this end-component. We fix such a word  $\$w\$$ ; such a state  $m$  of the end-component in  $\mathcal{M} \times \mathcal{D}_f$  from which this word  $\$w\$$  can be produced; and strategy of the NBW  $\mathcal{A}_f$  to follow the word  $w\$$  from  $q_0$  (and  $f$ ) to the accepting state  $f$ . (Note that the first  $\$$  always leads to  $q_0$  or  $f$ .)

In an accepting end-component we can be in two modes: *tracking*, or *not tracking*. If we are *not tracking* and reach  $m$ , we start to *track*  $\$w\$: we use  $\mathcal{A}_f$  to reach an accepting state after reading  $\$w\$$  (ignoring what happens in any other case) with the strategy we have fixed.$



■ **Figure 5** Given a complete NFA  $\mathcal{A}$ , an NBW  $\mathcal{A}_f$  and an NBW  $\text{fork}(\mathcal{A}_f)$  are constructed. In this example,  $\Sigma = \{a, b\}$  and  $\Sigma_{\$} = \{a, b, \$\}$ . From the state  $q_A$  (resp.  $q_B$ ), the NBW  $\text{fork}(\mathcal{A}_f)$  transitions to the initial state of  $\mathcal{A}_f$  (resp.  $\mathcal{B}_f$ ).



generate  $w_a \cdot \$$  repeatedly with probability one  
re-generate  $w_b \cdot \$$  repeatedly with probability one

■ **Figure 6** An example MC in the proof of Lemma 8. Assume  $\Sigma_{\$} = \{a, b, \$\}$ . The labelling of the MC is as follows:  $\ell(s_0) = \ell(s_1) = a$  and  $\ell(s_2) = \$$ .

Note that, after reading the first  $\$$ , we are in either  $q_0$  or  $f$ , such that, when starting from  $m$ , it is always possible, with a fixed probability  $p > 0$ , to read  $\$w\$$ , and thus to accept. If we read an unexpected letter (where the ‘expected’ letter is always the next letter from  $\$w\$$ ) or the end of the word  $\$w\$$  is reached, we move to *not tracking*.

The automata choices when *not tracking* can be resolved arbitrarily.

Once in an accepting end-component of  $\mathcal{M} \times \mathcal{D}_f$ , tracking is almost surely started infinitely often, and it is thus almost surely successful infinitely often. Thus, we have  $\text{PSyn}_{\mathcal{A}_f}^{\mathcal{M}}(s_0) \geq \text{PSyn}_{\mathcal{D}_f}^{\mathcal{M}}(s_0)$ . ◀

Let  $\mathcal{B}$  be an universal NFA in Figure 4(a) and  $\mathcal{B}_f = (\Sigma_{\$}, Q'_f, q'_0, \delta'_f, \{f'\})$  be the NBW in Figure 4(b). Assume without loss of generality that the state space of  $\mathcal{A}_f$ ,  $\mathcal{B}_f$ , and  $\{q_F, q_A, q_B\}$  are pairwise disjoint. We now define the fork operation. An example of how to construct an NBW  $\text{fork}(\mathcal{A}_f)$  is shown in Figure 5.

► **Definition 7.** Given an NBW  $\mathcal{A}_f = (\Sigma_{\$}, Q_f, q_0, \delta_f, \{f\})$ , we define the NBW  $\text{fork}(\mathcal{A}_f) = (\Sigma_{\$}, Q_F, q_F, \delta_F, \{f, f'\})$  with

- $Q_F = Q_f \cup Q'_{f'} \cup \{q_F, q_A, q_B\}$ ;
- $\delta_F(q, \sigma) = \delta_f(q, \sigma)$  for all  $q \in Q_f$  and  $\sigma \in \Sigma_{\$}$ ;
- $\delta_F(q, \sigma) = \delta_{f'}(q, \sigma)$  for all  $q \in Q'_{f'}$  and  $\sigma \in \Sigma_{\$}$ ;
- $\delta_F(q_F, \sigma) = \{q_A, q_B\}$  for all  $\sigma \in \Sigma_{\$}$ ;
- $\delta_F(q_A, \sigma) = \{q_0\}$  for all  $\sigma \in \Sigma_{\$}$ ;
- $\delta_F(q_B, \$) = \{q'_0\}$ , and  $\delta_F(q_B, \sigma) = \emptyset$  for all  $\sigma \in \Sigma$ .

Following from Lemma 5 and Lemma 6, we have:

► **Lemma 8.** The NBW  $\text{fork}(\mathcal{A}_f)$  is GFM if, and only if,  $\mathcal{L}(\mathcal{A}) = \Sigma^*$ .

**Proof.** We first observe that  $\mathcal{L}(\text{fork}(\mathcal{A}_f)) = \{\sigma\sigma'w \mid \sigma, \sigma' \in \Sigma_{\$}, w \in \mathcal{L}(\mathcal{A}_f)\} \cup \{\sigma\$w \mid \sigma \in \Sigma_{\$}, w \in \mathcal{L}(\mathcal{B}_f)\}$ .

**‘if’:** When  $\mathcal{L}(\mathcal{A}) = \Sigma^* = \mathcal{L}(\mathcal{B})$  holds, Lemma 5 provides  $\{\sigma\sigma'w \mid \sigma, \sigma' \in \Sigma_{\$}, w \in \mathcal{L}(\mathcal{A}_f)\} \supset \{\sigma\$w \mid \sigma \in \Sigma_{\$}, w \in \mathcal{L}(\mathcal{B}_f)\}$ , and therefore  $\mathcal{L}(\text{fork}(\mathcal{A}_f)) = \{\sigma\sigma'w \mid \sigma, \sigma' \in \Sigma_{\$}, w \in \mathcal{L}(\mathcal{A}_f)\}$ .

As  $\mathcal{A}_f$  is GFM by Lemma 6, this provides the GFM strategy “move first to  $q_{\mathcal{A}}$ , then to  $q_0$ , and henceforth follow the GFM strategy of  $\mathcal{A}_f$  for  $\text{fork}(\mathcal{A}_f)$ ”. Thus,  $\text{fork}(\mathcal{A}_f)$  is GFM in this case.

**‘only if’:** Assume  $\mathcal{L}(\mathcal{A}) \neq \Sigma^* = \mathcal{L}(\mathcal{B})$ , that is,  $\mathcal{L}(\mathcal{A}) \subset \mathcal{L}(\mathcal{B})$ . There must exist words  $w_a \in \mathcal{L}(\mathcal{A})$  and  $w_b \in \mathcal{L}(\mathcal{B}) \setminus \mathcal{L}(\mathcal{A})$ . We now construct an MC which witnesses that  $\text{fork}(\mathcal{A}_f)$  is not GFM.

The MC emits an  $a$  at the first step and then either an  $a$  or a  $\$$  with a chance of  $\frac{1}{2}$  at the second step. An example is provided in Figure 6.

After these two letters, it then moves to one of two cycles (independent of the first two chosen letters) with equal chance of  $\frac{1}{2}$ ; one of these cycles repeats a word  $w_a \cdot \$$  infinitely often, while the other repeats a word  $w_b \cdot \$$  infinitely often, where  $w_a \in \mathcal{L}(\mathcal{A})$ .

It is easy to see that the semantic chance of acceptance is  $\frac{3}{4}$  – failing if, and only if, the second letter is  $a$  and the word  $w_b \cdot \$$  is subsequently repeated infinitely often – whereas the syntactic chance of satisfaction is  $\frac{1}{2}$ : when the automaton first moves to  $q_{\mathcal{A}}$ , it accepts if, and only if, the word  $w_a \cdot \$$  is later repeated infinitely often, which happens with a chance of  $\frac{1}{2}$ ; when the automaton first moves to  $q_{\mathcal{B}}$ , it will reject when  $\$$  is not the second letter, which happens with a chance of  $\frac{1}{2}$ . ◀

It follows from Lemma 8 that the NFA universality problem is polynomial-time reducible to the problem of whether or not a given NBW is GFM.

► **Theorem 9.** *The problem of whether or not a given NBW is GFM is PSPACE-hard.*

Using the same construction of Definition 4, we can show that the problem of minimising a GFM NBW is PSPACE-hard. The reduction is from a problem which is similar to the NFA universality problem.

► **Theorem 10.** *Given a GFM NBW and a bound  $k$ , the problem whether there is a language equivalent GFM NBW with at most  $k$  states is PSPACE-hard. It is PSPACE-hard even for (fixed)  $k = 2$ .*

**Proof.** Using the construction of Definition 4, PSPACE-hardness follows from a reduction from the problem whether a nonempty complete NFA accepts all nonempty words. The latter problem is PSPACE-complete, following the PSPACE-completeness of the universality problem of (general) NFAs [18].

The reduction works because, for a nonempty complete NFA  $A$  the following hold:

(a) A GFM NBW equivalent to  $\mathcal{A}_f$  must have at least 2 states, one final and one nonfinal. This is because it needs a final state (as some word is accepted) as well as a nonfinal one (words that contain finitely many  $\$$ s are rejected).

(b) For a 2-state minimal GFM NBW equivalent to  $\mathcal{A}_f$ , there cannot be a word  $w \in \Sigma^+$  that goes from the final state back to it as this would produce an accepting run with finitely many  $\$$ s (as there is some accepting run). Therefore, when starting from the final state, any finite word can only go to the nonfinal state and stay there or block. But blocking is no option, as there is an accepted continuation to an infinite word. Thus, all letters in  $\Sigma$  lead from either state to the nonfinal state (only).



In order for a word starting with a letter in  $\Sigma$  to be accepted, there must therefore be a  $\$$  transition from the nonfinal to the final state.

These two points imply that for a nonempty complete NFA  $\mathcal{A}$  such that there is a 2-state GFM NBW equivalent to  $\mathcal{A}_f$  iff  $\mathcal{A}$  accepts all nonempty words.  $\blacktriangleleft$

#### 4 Decision Procedure for Qualitative GFM

In this section, we first define the notion of qualitative GFM (QGFM) and then provide an EXPTIME procedure that decides QGFM-ness.

The definition of QGFM is similar to GFM but we only need to consider MCs with which the semantic chance of success is one:

► **Definition 11.** *An NBW  $\mathcal{N}$  is qualitative good-for-MDPs (QGFM) if, for all finite MDPs  $\mathcal{M}$  with initial state  $s_0$  and  $\text{PSem}_{\mathcal{N}}^{\mathcal{M}}(s_0) = 1$ ,  $\text{PSyn}_{\mathcal{N}}^{\mathcal{M}}(s_0) = 1$  holds.*

Similar to Theorem 3, we can also replace ‘MDPs’ by ‘MCs’ in the definition of QGFM:

► **Theorem 12.** *An NBW  $\mathcal{N}$  is QGFM iff, for all finite MCs  $\mathcal{M}$  with initial state  $s_0$  and  $\text{PSem}_{\mathcal{N}}^{\mathcal{M}}(s_0) = 1$ ,  $\text{PSyn}_{\mathcal{N}}^{\mathcal{M}}(s_0) = 1$  holds.*

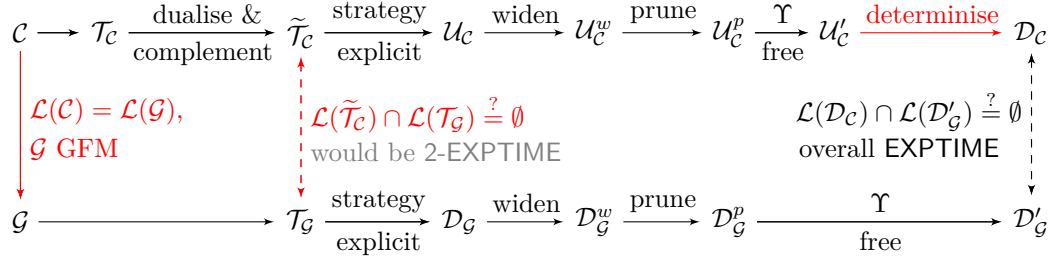
To decide QGFM-ness, we make use of the well known fact that qualitative acceptance, such as  $\text{PSem}_{\mathcal{N}}^{\mathcal{M}}(s_0) = 1$ , does not depend on the probabilities for an MC  $\mathcal{M}$ . This can, for example, be seen by considering the syntactic product  $\text{PSyn}_{\mathcal{D}}^{\mathcal{M}}(s_0) = 1$  with a deterministic parity automaton  $\mathcal{D}$  (for a deterministic automaton,  $\text{PSem}_{\mathcal{D}}^{\mathcal{M}}(s_0) = \text{PSyn}_{\mathcal{D}}^{\mathcal{M}}(s_0)$  trivially holds), where changing the probabilities does not change the end-components of the product MC  $\mathcal{M} \times \mathcal{D}$ , and the acceptance of these end-components is solely determined by the highest colour of the states (or transitions) occurring in it, and thus also independent of the probabilities: the probability is 1 if, and only if, an accepting end-component can be reached almost surely, which is also independent of the probabilities. As a result, we can search for the (regular) tree that stems from the unravelling of an MC, while disregarding probabilities. See Figure 2(b) for an example of such a tree.

This observation has been used in the synthesis of probabilistic systems before [15]. The set of directions (of a tree)  $\Upsilon$  could then, for example, be chosen to be the set of states of the unravelled finite MC; this would not normally be a full tree.

In the following, we show an exponential-time algorithm to decide whether a given NBW is QGFM or not. This procedure involves transformations of tree automata with different acceptance conditions. Because this is quite technical, we only provide an outline in the main paper. More notations (Section A.1) and details of the constructions (Section A.2) are provided in the full version [16].

For a given candidate NBW  $\mathcal{C}$ , we first construct a language equivalent NBW  $\mathcal{G}$  that we know to be GFM, such as the slim automaton from [9] or a suitable limit deterministic automaton [21, 6, 7, 17, 8]. For all known constructions,  $\mathcal{G}$  can be exponentially larger than  $\mathcal{C}$ . We use the slim automata from [9]; they have  $O(3^{|\mathcal{Q}|})$  states and transitions.

We then construct a number of tree automata as outlined in Figure 7. In a first construction, we discuss in the full version [16] how to build, for an NBW  $\mathcal{N}$ , a (symmetric) alternating Büchi tree automaton (ABS)  $\mathcal{T}_{\mathcal{N}}$  that accepts (the unravelling of) an MC (without probabilities, as discussed above)  $\mathcal{M}$  if, and only if, the syntactic product of  $\mathcal{N}$  and  $\mathcal{M}$  almost surely accepts. This construction is used twice: once to produce  $\mathcal{T}_{\mathcal{G}}$  for the GFM automaton  $\mathcal{G}$  we have constructed, and once to produce  $\mathcal{T}_{\mathcal{C}}$  for our candidate automaton  $\mathcal{C}$ . Since  $\mathcal{G}$  is QGFM,  $\mathcal{T}_{\mathcal{G}}$  accepts all the MCs  $\mathcal{M}$  that almost surely produce a run in  $\mathcal{L}(\mathcal{G})$  (which is the same as  $\mathcal{L}(\mathcal{C})$ ), that is,  $\text{PSem}_{\mathcal{G}}^{\mathcal{M}}(s_0) = \text{PSem}_{\mathcal{C}}^{\mathcal{M}}(s_0) = 1$ .



■ **Figure 7** Flowchart of the algorithms. The first algorithm is in 2-EXPTIME, which is to check the nonemptiness of the intersection of  $\tilde{\mathcal{T}}_{\mathcal{C}}$  and  $\mathcal{T}_{\mathcal{G}}$ . The second algorithm is in EXPTIME, which is to check the nonemptiness of the intersection of  $\mathcal{D}_{\mathcal{C}}$  and  $\mathcal{D}'_{\mathcal{G}}$ . The steps that have exponential blow-up are highlighted in red.

Therefore, to check whether or not our candidate NBW  $\mathcal{C}$  is QGFM, we can test language equivalence of  $\mathcal{T}_{\mathcal{C}}$  and  $\mathcal{T}_{\mathcal{G}}$ , e.g. by first complementing  $\mathcal{T}_{\mathcal{C}}$  to  $\tilde{\mathcal{T}}_{\mathcal{C}}$  and then checking whether or not  $\mathcal{L}(\tilde{\mathcal{T}}_{\mathcal{C}}) \cap \mathcal{L}(\mathcal{T}_{\mathcal{G}}) = \emptyset$  holds: the MCs in the intersection of the languages witness that  $\mathcal{C}$  is not QGFM. Thus,  $\mathcal{C}$  is QGFM if, and only if, these languages do not intersect, that is,  $\mathcal{L}(\tilde{\mathcal{T}}_{\mathcal{C}}) \cap \mathcal{L}(\mathcal{T}_{\mathcal{G}}) = \emptyset$ . This construction leads to a 2-EXPTIME procedure for deciding QGFM: we get the size of the larger automaton ( $\mathcal{G}$ ) and the complexity of the smaller automaton  $\mathcal{C}$ . The purpose of the following delicate construction is to contain the exponential cost to the syntactic material of the smaller automaton, while still obtaining the required level of entanglement between the structures and retaining the size advantage from the GFM property of  $\mathcal{G}$ .

Starting from  $\tilde{\mathcal{T}}_{\mathcal{C}}$ , we make a few transformations by mainly controlling the number of directions the alternating tree automaton needs to consider and the set of decisions player *accept*<sup>3</sup> has to make. This restricts the scope in such a way that the resulting intersection might shrink, but cannot become empty<sup>4</sup>.

We rein in the number of directions in two steps: in a first step, we *increase* the number of directions by widening the run tree with one more direction than the size of the state space of the candidate automaton  $\mathcal{C}$ . The larger amount of directions allows us to concurrently untangle the decisions of player *accept* within and between  $\tilde{\mathcal{T}}_{\mathcal{C}}$  and  $\mathcal{T}_{\mathcal{G}}$ , which intuitively creates one distinguished direction for each state  $q$  of  $\tilde{\mathcal{T}}_{\mathcal{C}}$  used by player *accept*, and one (different) distinguished direction for  $\mathcal{T}_{\mathcal{G}}$ . In a second step, we only keep these directions, resulting in an automaton with a *fixed* branching degree (just one bigger than the size of the state space of  $\mathcal{C}$ ), which is easy to analyse with standard techniques.

The standard techniques mean to first make the remaining choices of player *accept* in  $\tilde{\mathcal{T}}_{\mathcal{C}}$  explicit, which turns it into a universal co-Büchi automaton ( $\mathcal{U}_{\mathcal{C}}$ ). The automaton is then simplified to the universal co-Büchi automaton  $\mathcal{U}'_{\mathcal{C}}$  which can easily be determinised to a deterministic parity automaton  $\mathcal{D}_{\mathcal{C}}$ .

For  $\mathcal{T}_{\mathcal{G}}$ , a sequence of similar transformations are made; however, as we do not need to complement here, the automaton obtained from making the decisions explicit is already deterministic, which saves the exponential blow-up obtained in the determinisation of a universal automaton (determinising  $\mathcal{U}'_{\mathcal{C}}$  to  $\mathcal{D}_{\mathcal{C}}$ ).

<sup>3</sup> The acceptance of a tree by a tree automaton can be viewed as the outcome of a game played by player *accept* and player *reject*. We refer to the full version [16] for details.

<sup>4</sup> It can be empty to start with, of course, and will stay empty in that case. But if the intersection is not empty, then these operations will leave something in the language.

Therefore,  $\mathcal{D}_{\mathcal{C}}$  and  $\mathcal{D}'_{\mathcal{C}}$  can both be constructed from  $\mathcal{C}$  in time exponential in  $\mathcal{C}$ , and checking their intersection for emptiness can be done in time exponential in  $\mathcal{C}$ , too. With that, we obtain the membership in EXPTIME for QGFM-ness:

► **Theorem 13.** *The problem of whether or not a given NBW is QGFM is in EXPTIME.*

## 5 Membership in EXPTIME for GFM

In this section, we start out with showing a sufficient (Lemma 14) and necessary (Lemma 15) criterion for a candidate NBW to be GFM in Section 5.1.

We show in Section 5.2 that this criterion is also sufficient and necessary for QGFM-ness. This implies that GFM-ness and QGFM-ness collapse, so that the EXPTIME decision procedure from Section 4 can be used to decide GFM-ness, and the PSPACE hardness from Section 3 extends to QGFM.

### 5.1 Key Criterion for GFM-ness

In order to establish a necessary and sufficient criterion for GFM-ness, we construct two safety automata<sup>5</sup>  $\mathcal{S}$  and  $\mathcal{T}$ .

Given a candidate NBW  $\mathcal{C}$ , we define some notions for the states and transitions. We say a state  $q$  of  $\mathcal{C}$  is productive if  $\mathcal{L}(\mathcal{C}_q) \neq \emptyset$  where  $\mathcal{C}_q$  is the automaton obtained from  $\mathcal{C}$  by making  $q$  the initial state. A state  $q$  of the NBW  $\mathcal{C}$  is called QGFM if the automaton  $\mathcal{C}_q$  is QGFM. A transition  $(q, \sigma, r)$  is called residual if  $\mathcal{L}(\mathcal{C}_r) = \sigma^{-1}\mathcal{L}(\mathcal{C}_q)$  [11, 2]. In general,  $\mathcal{L}(\mathcal{C}_r) \subseteq \sigma^{-1}\mathcal{L}(\mathcal{C}_q)$  holds. See Figure 1 for an example of non-residual transitions. Selecting either of the two transitions from  $q_0$  will lose language: when selecting the transition to the left, the word  $a \cdot b^\omega$  is no longer accepted. Likewise, when selecting the transition to the right, the word  $a^\omega$  is no longer accepted. Thus, this automaton cannot make the decision to choose the left or the right transition, and neither  $(q_0, a, q_1)$  nor  $(q_0, a, q_2)$  is a residual transition.

Now we are ready to define  $\mathcal{S}$  and  $\mathcal{T}$ . In the NBW  $\mathcal{S}$ , we include the states from the candidate NBW  $\mathcal{C}$  that are productive and QGFM at the same time. We only include the residual transitions (in  $\mathcal{C}$ ) between those states. In the NBW  $\mathcal{T}$ , we include only the productive states of  $\mathcal{C}$  and the transitions between them. We then make both  $\mathcal{S}$  and  $\mathcal{T}$  safety automata by marking all states final. We first show that the criterion,  $\mathcal{L}(\mathcal{S}) = \mathcal{L}(\mathcal{T})$  and  $\mathcal{S}$  is GFG<sup>6</sup>, is sufficient for  $\mathcal{C}$  to be GFM. Similar to the proof of Lemma 6, to show the NBW  $\mathcal{C}$  is GFM, we show there exists a strategy for  $\mathcal{C}$  such that the syntactic and semantic chance of winning are the same for any MC.

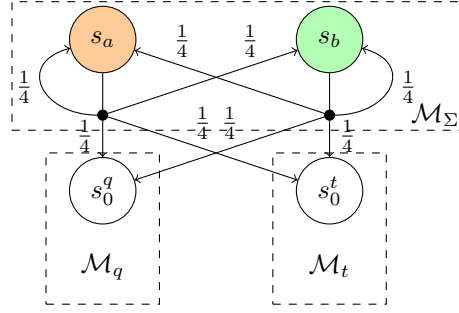
► **Lemma 14.** *If  $\mathcal{L}(\mathcal{S}) = \mathcal{L}(\mathcal{T})$  and  $\mathcal{S}$  is GFG, then the candidate NBW  $\mathcal{C}$  is GFM.*

**Proof.** As  $\mathcal{T}$  contains all states and transitions from  $\mathcal{S}$ ,  $\mathcal{L}(\mathcal{S}) \subseteq \mathcal{L}(\mathcal{T})$  always holds. We assume that  $\mathcal{L}(\mathcal{S}) \supseteq \mathcal{L}(\mathcal{T})$  holds and  $\mathcal{S}$  is GFG.

By Theorem 3, to show  $\mathcal{C}$  is GFM, it suffices to show that  $\mathcal{C}$  is good for an arbitrary MC  $\mathcal{M}$  with initial state  $s_0$ . We first determinise  $\mathcal{C}$  to a DPW  $\mathcal{D}$  [13]. Since  $\mathcal{D}$  is deterministic and  $\mathcal{L}(\mathcal{D}) = \mathcal{L}(\mathcal{C})$ , we have  $\text{PSyn}_{\mathcal{D}}^{\mathcal{M}}(s_0) = \text{PSem}_{\mathcal{D}}^{\mathcal{M}}(s_0) = \text{PSem}_{\mathcal{C}}^{\mathcal{M}}(s_0)$ . Since  $\text{PSem}_{\mathcal{C}}^{\mathcal{M}}(s_0) \geq$

<sup>5</sup> A safety automaton is one where all states are final. These automata can be viewed as NFAs where convenient.

<sup>6</sup> GFG as a general property is tricky, but  $\mathcal{S}$  is a safety automaton, and GFG safety automata are, for example, determinisable by pruning, and the property whether or not a safety automaton is GFG can be checked in polynomial time [5].



■ **Figure 8** An example MC in the proof of Lemma 15. In this example,  $\Sigma = \{a, b\}$ . Also, the state  $q$  of the candidate NBW  $\mathcal{C}$  is the only state that is not QGFM and the transition  $t$  of the NBW  $\mathcal{C}$  is the only non-residual transition. We have  $\ell(s_a) = a$  and  $\ell(s_b) = b$ . The states  $s_a$  and  $s_b$  and the transitions between them form  $\mathcal{M}_\Sigma$ .

$\text{PSyn}_{\mathcal{C}}^{\mathcal{M}}(s_0)$  always holds, we establish the equivalence of syntactic and semantic chance of winning for  $\mathcal{M} \times \mathcal{C}$  by proving  $\text{PSyn}_{\mathcal{C}}^{\mathcal{M}}(s_0) \geq \text{PSyn}_{\mathcal{D}}^{\mathcal{M}}(s_0) = \text{PSem}_{\mathcal{C}}^{\mathcal{M}}(s_0)$ .

For that, we show for an arbitrary accepting run  $r$  of  $\mathcal{M} \times \mathcal{D}$ , there is a strategy for  $\mathcal{M} \times \mathcal{C}$  such that  $r'$  (the corresponding run in the product) is accepting in  $\mathcal{M} \times \mathcal{C}$  where the projections of  $r$  and  $r'$  on  $\mathcal{M}$  are the same.

We define the strategy for  $\mathcal{M} \times \mathcal{C}$  depending on whether an accepting end-component of  $\mathcal{M} \times \mathcal{D}$  has been entered. Since  $r$  is accepting, it must enter an accepting end-component of  $\mathcal{M} \times \mathcal{D}$  eventually. Let the run  $r$  be  $(s_0, q_0^{\mathcal{D}})(s_1, q_1^{\mathcal{D}}) \cdots$  and it enters the accepting end-component on reaching the state  $(s_n, q_n^{\mathcal{D}})$ . Before  $r$  enters an accepting end-component of  $\mathcal{M} \times \mathcal{D}$ ,  $\mathcal{C}$  follows the GFG strategy for  $\mathcal{S}$  to stay within the states that are productive and QGFM. Upon reaching an accepting end-component of  $\mathcal{M} \times \mathcal{D}$ , the run  $r$  is in state  $(s_n, q_n^{\mathcal{D}})$ , assume the run for  $\mathcal{M} \times \mathcal{C}$  is in state  $(s_n, q_n^{\mathcal{C}})$  at this point. We then use the QGFM strategy of  $\mathcal{C}_{q_n^{\mathcal{C}}}$  from here since  $q_n^{\mathcal{C}}$  is QGFM.

We briefly explain why this strategy for  $\mathcal{M} \times \mathcal{C}$  would lead to  $\text{PSyn}_{\mathcal{C}}^{\mathcal{M}}(s_0) \geq \text{PSyn}_{\mathcal{D}}^{\mathcal{M}}(s_0)$ . Since  $(s_n, q_n^{\mathcal{D}})$  is in the accepting end-component of  $\mathcal{M} \times \mathcal{D}$ , we have  $\text{PSem}_{\mathcal{D}_{q_n^{\mathcal{D}}}}^{\mathcal{M}}(s_n) = \text{PSyn}_{\mathcal{D}_{q_n^{\mathcal{D}}}}^{\mathcal{M}}(s_n) = 1$  by Theorem 1. We show  $\text{PSem}_{\mathcal{C}_{q_n^{\mathcal{C}}}}^{\mathcal{M}}(s_n) = 1$  so that  $\text{PSyn}_{\mathcal{C}_{q_n^{\mathcal{C}}}}^{\mathcal{M}}(s_n) = \text{PSem}_{\mathcal{C}_{q_n^{\mathcal{C}}}}^{\mathcal{M}}(s_n) = 1$  as  $q_n^{\mathcal{C}}$  is QGFM. For that, it suffices to show  $\mathcal{L}(\mathcal{C}_{q_n^{\mathcal{C}}}) = \mathcal{L}(\mathcal{D}_{q_n^{\mathcal{D}}}) = w^{-1}\mathcal{L}(\mathcal{C})$  where  $w = \ell(s_0 s_1 \cdots s_{n-1})$ . This can be proved by induction in [16] over the length of the prefix of words from  $\mathcal{L}(\mathcal{S})$ . ◀

In order to show that this requirement is also necessary, we build an MC witnessing that  $\mathcal{C}$  is not GFM in case the criterion is not satisfied. An example MC is given in Figure 8. We produce the MC by parts. It has a central part denoted by  $\mathcal{M}_\Sigma$ . The state space of  $\mathcal{M}_\Sigma$  is  $S_\Sigma$  where  $S_\Sigma = \{s_\sigma \mid \sigma \in \Sigma\}$ . Each state  $s_\sigma$  is labelled with  $\sigma$  and there is a transition between every state pair.

For every state  $q$  that is not QGFM, we construct an MC  $\mathcal{M}_q = \langle S_q, P_q, \Sigma, \ell_q \rangle$  from Section 4 witnessing that  $\mathcal{C}_q$  is not QGFM, that is, from a designated initial state  $s_0^q$ ,  $\text{PSyn}_{\mathcal{C}_q}^{\mathcal{M}_q}(s_0^q) \neq \text{PSem}_{\mathcal{C}_q}^{\mathcal{M}_q}(s_0^q) = 1$ , and for every non-residual transition  $t = (q, \sigma, r)$  that is not in  $\mathcal{S}$  due to  $\mathcal{L}(\mathcal{C}_r) \neq \sigma^{-1}\mathcal{L}(\mathcal{C}_q)$ , we construct an MC  $\mathcal{M}_t = \langle S_t, P_t, \Sigma, \ell_t \rangle$  such that, from an initial state  $s_0^t$ , there is only one ultimately periodic word  $w_t$  produced, such that  $w_t \in \sigma^{-1}\mathcal{L}(\mathcal{C}_q) \setminus \mathcal{L}(\mathcal{C}_r)$ .

Finally, we produce an MC  $\mathcal{M}$ , whose states are the disjoint union of the MCs  $\mathcal{M}_\Sigma$ ,  $\mathcal{M}_q$  and  $\mathcal{M}_t$  from above. The labelling and transitions within the MCs  $\mathcal{M}_q$  and  $\mathcal{M}_t$  are preserved

while, from the states in  $S_\Sigma$ ,  $\mathcal{M}$  also transitions to all initial states of the individual  $\mathcal{M}_q$  and  $\mathcal{M}_t$  from above. It remains to specify the probabilities for the transitions from  $S_\Sigma$ : any state in  $S_\Sigma$  transitions to its successors uniformly at random.

► **Lemma 15.** *If  $\mathcal{L}(\mathcal{S}) \neq \mathcal{L}(\mathcal{T})$  or  $\mathcal{S}$  is not GFG, the candidate NBW  $\mathcal{C}$  is not GFM.*

**Proof.** Assume  $\mathcal{L}(\mathcal{S}) \neq \mathcal{L}(\mathcal{T})$ . There must exist a word  $w = \sigma_0, \sigma_1, \dots \in \mathcal{L}(\mathcal{T}) \setminus \mathcal{L}(\mathcal{S})$ .

Let us use  $\mathcal{M}$  with initial state  $s_{\sigma_0}$  as the MC which witnesses that  $\mathcal{C}$  is not GFM. We first build the product MDP  $\mathcal{M} \times \mathcal{C}$ . There is a non-zero chance that, no matter how the choices of  $\mathcal{C}$  (thus, the product MDP  $\mathcal{M} \times \mathcal{C}$ ) are resolved, a state sequence  $(s_{\sigma_0}, q_0), (s_{\sigma_1}, q_1), \dots, (s_{\sigma_k}, q_k)$  with  $k \geq 0$  is seen, and  $\mathcal{C}$  selects a successor  $q$  such that  $(q_k, \sigma_k, q)$  is not a transition in  $\mathcal{S}$ .

For the case that this is because  $\mathcal{C}_q$  is not QGFM, we observe that there is a non-zero chance that the product MDP moves to  $(s'_0, q)$ , such that  $\text{PSyn}_{\mathcal{C}}^{\mathcal{M}}(s_{\sigma_0}) < \text{PSem}_{\mathcal{C}}^{\mathcal{M}}(s_{\sigma_0})$  follows.

For the other case that this is because the transition  $t = (q_k, \sigma_k, q)$  is non-residual, that is,  $\mathcal{L}(\mathcal{C}_q) \neq \sigma_k^{-1} \mathcal{L}(\mathcal{C}_{q_k})$ , we observe that there is a non-zero chance that the product MDP moves to  $(s'_0, q)$ , such that  $\text{PSyn}_{\mathcal{C}}^{\mathcal{M}}(s_{\sigma_0}) < \text{PSem}_{\mathcal{C}}^{\mathcal{M}}(s_{\sigma_0})$  follows.

For the case that  $\mathcal{S}$  is not GFG, no matter how the nondeterminism of  $\mathcal{C}$  is resolved, there must be a shortest word  $w = \sigma_0, \dots, \sigma_k$  ( $k \geq 0$ ) such that  $w$  is a prefix of a word in  $\mathcal{L}(\mathcal{S})$ , but the selected control leaves  $\mathcal{S}$ . For this word, we can argue in the same way as above. ◀

Lemma 14 and Lemma 15 suggest that GFM-ness of a NBW can be decided in EXPTIME by checking whether the criterion holds or not. However, as shown in the next section that QGFM = GFM, we can apply the EXPTIME procedure from Section 5 to check QGFM-ness, and thus, GFM-ness.

## 5.2 QGFM = GFM

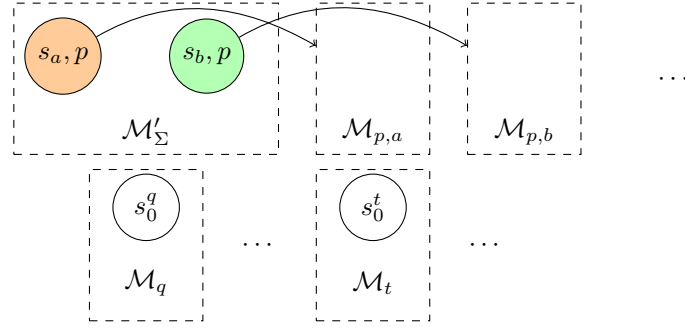
To show that QGFM = GFM, we show that the same criterion from the previous section is also sufficient and necessary for QGFM. By definition, if an NBW is GFM then it is QGFM. Thus, the sufficiency of the criterion follows from Lemma 14. We are left to show the necessity of the criterion. To do that, we build an MC  $\mathcal{M}'$  witnessing that  $\mathcal{C}$  is not QGFM in case the criterion of Lemma 15 is not satisfied. We sketch in Figure 9 the construction of  $\mathcal{M}'$ .

The principle difference between the MC  $\mathcal{M}'$  constructed in this section and  $\mathcal{M}$  from the previous section is that the new MC  $\mathcal{M}'$  needs to satisfy that  $\text{PSem}_{\mathcal{C}}^{\mathcal{M}'}(s_0) = 1$  ( $s_0$  is the initial state of  $\mathcal{M}'$ ), while still forcing the candidate NBW  $\mathcal{C}$  to make decisions that lose probability of success, leading to  $\text{PSyn}_{\mathcal{C}}^{\mathcal{M}'}(s_0) < 1$ . This makes the construction of  $\mathcal{M}'$  more complex, but establishes that qualitative and full GFM are equivalent properties.

The MC will also be constructed by parts and it has a central part. It will also have the MCs  $\mathcal{M}_q$  for each non QGFM state  $q$  and  $\mathcal{M}_t$  for each non-residual transition  $t$  from the previous section. We now describe the three potential problems of  $\mathcal{M}$  of the previous section and the possible remedies.

The first potential problem is in the central part as it might contain prefixes that cannot be extended to words in  $\mathcal{L}(\mathcal{C})$ . Such prefixes should be excluded. This can be addressed by building a cross product with a deterministic safety automaton that recognises all the prefixes of  $\mathcal{L}(\mathcal{C})$  (the safety hull of the language of  $\mathcal{C}$ ) and removing the states that have no outgoing transitions in the product.

The second problem is caused by the transitions to all MCs  $\mathcal{M}_q$  and  $\mathcal{M}_t$  from every state in the central part. This can, however, be avoided by including another safety automaton in



■ **Figure 9** An illustration of the MC in the proof of Lemma 16. In this example, we have  $\Sigma = \{a, b\}$ . The new central part  $\mathcal{M}'_\Sigma$  is obtained by removing the states that have no outgoing transitions in the cross product of  $\mathcal{M}_\Sigma$  and  $\mathcal{D}'$ . For the states  $(s_a, p)$  and  $(s_b, p)$  of  $\mathcal{M}'_\Sigma$ , we have  $\ell((s_a, p)) = a$  and  $\ell((s_b, p)) = b$ . For each state  $(s_\sigma, p)$  of the central part, we construct an MC  $\mathcal{M}_{p,\sigma}$  using  $\mathcal{D}'$ . There is a transition from each state  $(s_\sigma, p)$  of  $\mathcal{M}'_\Sigma$  to the initial state of  $\mathcal{M}_{p,\sigma}$ . The MCs  $\mathcal{M}_q$  and  $\mathcal{M}_t$  are as before. Whether there is a transition from a state from  $\mathcal{M}'_\Sigma$  to the MCs  $\mathcal{M}_q$  and  $\mathcal{M}_t$  is determined by the overestimation provided by  $\mathcal{D}'$ .

the product that tracks, which of these transitions *could* be used by our candidate NBW  $\mathcal{C}$  at the moment, and only using those transitions. Note that this retains all transitions used in the proof to establish a difference between  $\text{PSem}_{\mathcal{C}}^{\mathcal{M}'}(s_0)$  and  $\text{PSyn}_{\mathcal{C}}^{\mathcal{M}'}(s_0)$  while guaranteeing that the semantic probability of winning after progressing to  $\mathcal{M}_q$  or  $\mathcal{M}_t$  is 1.

Removing the transitions to some of the  $\mathcal{M}_q$  and  $\mathcal{M}_t$  can potentially create a third problem, namely that no transitions to  $\mathcal{M}_q$  and  $\mathcal{M}_t$  are left so that there is no way leaving the central part of the MC. To address this problem, we can create a new MC for each state in the central part so that a word starting from the initial state of the MC can always be extended to an accepting word in  $\mathcal{L}(\mathcal{C})$  by transitioning to this new MC. How such MCs can be constructed is detailed later.

Zooming in on the construction, the MC  $\mathcal{M}$  should satisfy that all the finite runs that start from an initial state  $s_0$ , before transitioning to  $\mathcal{M}_q$  or  $\mathcal{M}_t$ , can be extended to a word in the language of  $\mathcal{C}$  are retained. The language of all such initial sequences is a safety language, and it is easy to construct an automaton that (1) recognises this safety language and (2) retains the knowledge of how to complete each word in the language of  $\mathcal{C}$ . To create this automaton, we first determinise  $\mathcal{C}$  to a deterministic parity word automaton  $\mathcal{D}$  [13]. We then remove all non-productive states from  $\mathcal{D}$  and mark all states final, yielding the safety automaton<sup>7</sup>  $\mathcal{D}'$ .

The two automata  $\mathcal{D}'$  and  $\mathcal{D}$  can be used to address all the problems we have identified. To address the first problem, we build a cross product MC of  $\mathcal{M}_\Sigma$  (the central part of  $\mathcal{M}$  in last section) and  $\mathcal{D}'$ . We then remove all the states in the product MC without any outgoing transitions and make the resulting MC the new central part denoted by  $\mathcal{M}'_\Sigma$ . Every state in  $\mathcal{M}'_\Sigma$  is of the form  $(s_\sigma, p)$  where  $s_\sigma \in S_\Sigma$  is from  $\mathcal{M}_\Sigma$  and  $p \in \mathcal{D}'$ .

The states of the deterministic automaton  $\mathcal{D}$  (and thus those of  $\mathcal{D}'$ ) also provide information about the possible states of  $\mathcal{C}$  that could be after the prefix we have seen so far. To address the second problem, we use this information to overestimate whether  $\mathcal{C}$  could be in

<sup>7</sup> From every state  $p$  in  $\mathcal{D}'$ , we can construct an extension to an accepted word by picking an accepted lasso path through  $\mathcal{D}$  that starts from  $p$ . Note that  $\mathcal{D}'$  is not complete, but every state has some successor.

some state  $q$ , or use a transition  $t$ , which in turn is good enough for deciding whether or not to transition to the initial states of  $\mathcal{M}_q$  resp.  $\mathcal{M}_t$  from every state of the new central part.

To address the third problem, we build, for every state  $p$  of  $\mathcal{D}'$  and every letter  $\sigma \in \Sigma$  such that  $\sigma$  can be extended to an accepted word from state  $p$ , an MC  $\mathcal{M}_{p,\sigma}$  that produces a single  $\omega$ -regular word (sometimes referred to as lasso word)  $w_{p,\sigma}$  with probability 1. The word  $\sigma \cdot w_{p,\sigma}$  will be accepted from state  $p$  (or: by  $\mathcal{D}_p$ ). From every state  $(s_\sigma, p)$  of the central part, there is a transition to the initial state of  $\mathcal{M}_{p,\sigma}$ .

The final MC transitions uniformly at random, from a state  $(s_\sigma, p)$  in  $\mathcal{M}'_\Sigma$ , to one of its successor states, which comprise the initial state of  $\mathcal{M}_{p,\sigma}$  and the initial states of some of the individual MCs  $\mathcal{M}_q$  and  $\mathcal{M}_t$ .

► **Lemma 16.** *If  $\mathcal{L}(S) \neq \mathcal{L}(T)$  or  $S$  is not GFG, the candidate NBW  $\mathcal{C}$  is not QGFM.*

**Proof.** The proof of the difference in the probability of winning in case  $\mathcal{L}(S) \neq \mathcal{L}(T)$  or in case  $S$  is not GFG are the same as in Lemma 15.

We additionally have to show that  $\text{PSem}_{\mathcal{C}}^{\mathcal{M}'}(s_0) = 1$ . But this is easily provided by the construction: when we move on to some  $\mathcal{M}_{p,\sigma}$ ,  $\mathcal{M}_q$ , or  $\mathcal{M}_t$ , we have sure, almost sure, and sure satisfaction, respectively, of the property by construction, while staying for ever in the central part of the new MC happens with probability 0. ◀

By definition, if a candidate NBW  $\mathcal{C}$  is GFM, it is QGFM. Together with Lemma 14 and Lemma 16, we have that  $\mathcal{L}(S) = \mathcal{L}(T)$  and  $S$  is GFG iff the candidate NBW  $\mathcal{C}$  is QGFM. Thus, we have

► **Theorem 17.** *The candidate NBW  $\mathcal{C}$  is GFM if, and only if,  $\mathcal{C}$  is QGFM.*

By Theorem 13 and Theorem 17, we have:

► **Corollary 18.** *The problem of whether or not a given NBW is GFM is in EXPTIME.*

Likewise, by Theorem 9, Theorem 10, and Theorem 17, we have:

► **Corollary 19.** *The problem of whether or not a given NBW is QGFM is PSPACE-hard. Given a QGFM NBW and a bound  $k$ , the problem whether there is a language equivalent QGFM NBW with at most  $k$  states is PSPACE-hard. It is PSPACE-hard even for (fixed)  $k = 2$ .*

## 6 Discussion

We have started out with introducing the prima facie simpler auxiliary concept of *qualitative* GFM-ness.

We have then established that deciding GFM-ness is PSPACE-hard by a reduction from the NFA universality problem and developed an algorithm for checking *qualitative* GFM-ness in EXPTIME.

We then closed with first characterising GFM-ness with a heavy use of QGFM-ness tests, only to find that this characterisation also proves to be a necessary requirement for QGFM-ness, which led to a collapse of the qualitative and full GFM-ness. The hardness results for GFM-ness therefore carry over to QGFM-ness, while the decision procedure for QGFM-ness proves to be a decision procedure for GFM-ness by itself.

**Acknowledgement.** We thank an anonymous reviewer for raising the excellent question (to the original version) of whether or not QGFM and GFM are different. They proved not to be. Without their clever question, we would not have considered this question, and thus not strengthened this paper accordingly.

## References

- 1 Luca Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, Stanford, CA, USA, 1998.
- 2 Marc Bagnol and Denis Kuperberg. Büchi good-for-games automata are efficiently recognizable. In *FSTTCS*, 2018.
- 3 Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.
- 4 Andrea Bianco and Luca de Alfaro. Model checking of probabalistic and nondeterministic systems. In P. S. Thiagarajan, editor, *Foundations of Software Technology and Theoretical Computer Science, 15th Conference, Bangalore, India, December 18-20, 1995, Proceedings*, volume 1026 of *Lecture Notes in Computer Science*, pages 499–513. Springer, 1995.
- 5 Thomas Colcombet. Forms of determinism for automata (invited talk). In Christoph Dürr and Thomas Wilke, editors, *29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012, February 29th - March 3rd, 2012, Paris, France*, volume 14 of *LIPICs*, pages 1–23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2012.
- 6 Costas Courcoubetis and Mihalis Yannakakis. The complexity of probabilistic verification. *J. ACM*, 42(4):857–907, July 1995.
- 7 Ernst Moritz Hahn, Guangyuan Li, Sven Schewe, Andrea Turrini, and Lijun Zhang. Lazy probabilistic model checking without determinisation. In *Concurrency Theory*, pages 354–367, 2015.
- 8 Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. Omega-regular objectives in model-free reinforcement learning. In Tomáš Vojnar and Lijun Zhang, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 395–412, Cham, 2019. Springer International Publishing.
- 9 Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. Good-for-MDPs automata for probabilistic analysis and reinforcement learning. In Armin Biere and David Parker, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 306–323, Cham, 2020. Springer International Publishing.
- 10 Thomas A. Henzinger and Nir Piterman. Solving games without determinization. In *Computer Science Logic*, pages 394–409, September 2006. LNCS 4207.
- 11 Denis Kuperberg and Michał Skrzypczak. On determinisation of good-for-games automata. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming*, pages 299–310, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.
- 12 Dominique Perrin and Jean-Éric Pin. *Infinite Words: Automata, Semigroups, Logic and Games*. Elsevier, 2004.
- 13 Nir Piterman. From Nondeterministic Büchi and Streett Automata to Deterministic Parity Automata. *Log. Methods Comput. Sci.*, 3(3), 2007.
- 14 Amir Pnueli. The temporal logic of programs. In *IEEE Symposium on Foundations of Computer Science*, pages 46–57, 1977.
- 15 Sven Schewe. Synthesis for probabilistic environments. In *4th International Symposium on Automated Technology for Verification and Analysis (ATVA 2006)*, pages 245–259. Springer Verlag, 2006.
- 16 Sven Schewe, Qiyi Tang, and Tansholpan Zhanabekova. Deciding what is good-for-mdps, 2023. [arXiv:2202.07629](https://arxiv.org/abs/2202.07629).
- 17 Salomon Sickert, Javier Esparza, Stefan Jaax, and Jan Křetínský. Limit-deterministic Büchi automata for linear temporal logic. In *Computer Aided Verification*, pages 312–332, 2016. LNCS 9780.
- 18 Larry J. Stockmeyer and Albert R. Meyer. Word problems requiring exponential time: Preliminary report. In Alfred V. Aho, Allan Borodin, Robert L. Constable, Robert W. Floyd, Michael A. Harrison, Richard M. Karp, and H. Raymond Strong, editors, *Proceedings of the 5th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1973, Austin, Texas, USA*, pages 1–9. ACM, 1973.



- 19 Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, second edition, 2018.
- 20 Wolfgang Thomas. *Handbook of Theoretical Computer Science*, chapter Automata on Infinite Objects, pages 133–191. The MIT Press/Elsevier, 1990.
- 21 Moshe Y. Vardi. Automatic verification of probabilistic concurrent finite state programs. In *Foundations of Computer Science*, pages 327–338, 1985.