Verifying ω-Regular Properties of Markov Chains

Doron Bustan[1], Sasha Rubin[2], and Moshe Y. Vardi[1]

[1] Rice University***
[2] The University of Auckland

Abstract. In this work we focus on model checking of probabilistic models. Probabilistic models are widely used to describe randomized protocols. A Markov chain induces a probability measure on sets of computations. The notion of correctness now becomes probabilistic. We solve here the general problem of linear-time probabilistic model checking with respect to ω-regular specifications. As specification formalism, we use alternating Büchi infinite-word automata, which have emerged recently as a generic specification formalism for developing model checking algorithms. Thus, the problem we solve is: given a Markov chain $\mathcal{M}$ and automaton $\mathcal{A}$, check whether the probability induced by $\mathcal{M}$ of $L(\mathcal{A})$ is one (or compute the probability precisely). We show that these problem can be solved within the same complexity bounds as model checking of Markov chains with respect to LTL formulas. Thus, the additional expressive power comes at no penalty.

1 Introduction

In model checking, we model a system as a transition system $\mathcal{M}$ and a specification as a temporal formula ψ. Then, using formal methods, we check whether $\mathcal{M}$ satisfies ψ [7]. One of the most significant developments in this area is the discovery of algorithmic methods for verifying temporal logic properties of *finite-state* systems [25, 19, 6, 35]. This derives its significance both from the fact that many synchronization and communication protocols can be modelled as finite-state programs, as well as from the great ease of use of fully algorithmic methods. Looking at model-checking algorithms more closely, we can classify these algorithms according to two criteria. The first criterion is the type of model that we use – nondeterministic or probabilistic. The second criterion is the specification language.

For nondeterministic models and linear temporal logic (LTL), a close and fruitful connection with the theory of automata over infinite words has been developed [34–36]. The basic idea is to associate with each LTL formula a nondeterministic Büchi automaton over infinite words (NBW) that accepts exactly all the computations that satisfy the formula. This enables the reduction of various decision problems, such as satisfiability and model checking, to known automata-theoretic problems, yielding clean and asymptotically optimal algorithms. Furthermore, these reductions are very helpful for implementing temporal-logic based verification algorithms, cf. [14]. This connection to

*** Supported in part by NSF grants CCR-9988322, CCR-0124077, CCR-0311326, IIS-9908435, IIS-9978135, and EIA-0086264, by BSF grant 9800096, and by a grant from the Intel Corporation.

automata theory can also be extended to languages beyond LTL, such as ETL [36] and μTL [32].

In this paper we focus on model checking of probabilistic models. Probabilistic models are widely used to describe randomized protocols, which are often used in distributed protocols [5], communication protocols [8], robotics [29], and more. We use Markov chains as our probabilistic model, cf. [31]. A Markov chain induces a probability measure on sets of computations. The notion of correctness now becomes probabilistic: we say here that a program is correct if the probability that a computation satisfies the specification is one (we also discuss a quantitative notion of correctness, where we compute the probability that a computation satisfies the specification). Early approaches for probabilistic model checking of LTL formulas [20, 31] required determinization of NBW, which involves an additional exponential blow-up over the construction of automata from formulas [26], requiring exponential space complexity, unlike the polynomial space complexity of standard model-checking algorithms for LTL, cf. [30].

The exponential gap for probabilistic model checking was bridged in [9, 10], who provided a polynomial-space algorithm, matching the lower bound of [28]. The algorithm in [9, 10] is specialized to LTL (and ETL) specifications, and proceeds by induction on the structure of the formula. An automata-theoretic account of this algorithm was given in [11]. It is shown there that LTL formulas can be translated to a special type of NBW, which they call *separated automata*. (An NBW is separated if every two states that are located in the same strongly connected component have disjoint languages). As with the standard translation of LTL formulas to NBW, the translation to separated automata is exponential. It is then shown in [11] how to model check Markov chains, in nondeterministic logarithmic space, with respect to separated NBW as complemented specification. This yields a polynomial space upper bound for probabilistic model checking of LTL formulas.

The automata-theoretic framework in [11] is very specifically tailored to LTL. As mentioned earlier contrast, the automata-theoretic framework for model checking nondeterministic models is quite general and can also handle more expressive specification languages such as ETL and μTL. This is not a mere theoretical issue. There has been a major recent emphasis on the development of industrial specification languages. These efforts resulted in a several languages [18, 23, 4, 2], culminating in an industrial standard, PSL 1.01 (`www.accellera.com`). Most of the new languages have the full power of NBW, i.e., they can express all ω-regular languages. Thus, they are strictly more expressive than LTL [37], and, thus, not covered by the framework of [9–11].

In this paper we solve the general problem of probabilistic model checking with respect to ω-regular specifications. As specification formalism, we use alternating Büchi infinite-word automata (ABW); see discussion below. Thus, the problem we solve is: Given a Markov chain $\mathcal{M}$ and an ABW $\mathcal{A}$, check whether the probability induced by $\mathcal{M}$ of $L(\mathcal{A})$ is one (i.e., whether $P_{\mathcal{M}}(L(\mathcal{A})) = 1$). (A more refined problem is to calculate the probability precisely, see Appendix D.)

The motivation for using ABWs as a specification formalism is derived from recent developments in the area of linear specification languages. First, ABWs have been used as an intermediate formalism between LTL formulas and nondeterministic Büchi word automata (NBW). As shown in [12, 13], one can exploit the linear translation from LTL

formulas to ABWs ([33]) for an early minimization, before the exponential translation to NBW. Second, not only can logics such as ETL and μTL be easily translated to ABW, but also most of the new industrial languages can be translated to ABW. Furthermore, for some of them efficient such translations are known (cf. [1]). Thus, ABW can serve as a generic specification formalism for developing model checking algorithms. Note that applying the techniques of [9, 10] to ABW specifications requires converting them first to NBW at an exponential cost, which we succeed here in avoiding.

We present here an algorithm for model checking of Markov chains, using ABWs as specifications. The space complexity of the algorithm is polylogarithmic in $\mathcal{M}$ and polynomial in $\mathcal{A}$. The linear translation of LTL to ABW implies that this complexity matches the lower bound for this problem.

As in [10, 11], our algorithm uses the subset construction to capture the language of every subset of states of $\mathcal{A}$ (an infinite word w is in the language of a set Q of states if $w \in L(s)$ for every state $s \in Q$ and $w \notin L(s)$ for every $s \notin Q$). While for LTL, a straightforward subset construction suffices, this is not the case for ABW. A key technical innovation of this paper is our use of *two* nondeterministic structures that correspond to the alternating automaton $\mathcal{A}$ to capture the language of every set of automaton states. The first nondeterministic structure is an NBW $\mathcal{A}_f$ called the *full automaton*, and the second a "slim" version of the full automaton without accepting conditions, which we call the *local transition system* T_A. Every state q of $\mathcal{A}_f$ and T_A corresponds to a set Q of states in $\mathcal{A}$. While is possible, however, that a several states of $\mathcal{A}_f$ correspond to the same set of states of $\mathcal{A}$, every state of T_A corresponds to a unique set of states of $\mathcal{A}$. The model-checking algorithm make use of the products G and G_f of the Markov chain M with T_A and A_f, respectively.

2 Preliminaries

2.1 Automata

Definition 1. *A* nondeterministic Büchi word automaton *(NBW) is* $\mathcal{A} = \langle \Sigma, S, S_0, \delta, F \rangle$, *where* Σ *is a finite alphabet,* S *is a finite set of states,* $\delta : S \times \Sigma \to 2^S$ *is a transition function,* $S_0 \subseteq S$ *is a set of initial states, and* $F \subseteq S$ *is a set of accepting states.*

Let $w = w_0, w_1, \ldots$ be an infinite word over Σ. For $i \in \mathbb{N}$, let $w^i = w_i, w_{i+1}, \ldots$ denote the suffix of w from its i'th letter. A sequence $\rho = s_0, s_1, \ldots$ in S^ω is a *run* of $\mathcal{A}$ over an infinite word $w \in \Sigma^\omega$, if $s_0 \in S_0$ and for every $i > 0$, we have $s_{i+1} \in \delta(s_i, w_i)$. We use $inf(\rho)$ to denote the set of states that appear infinitely often in ρ. A run ρ of $\mathcal{A}$ is *accepting* if $inf(\rho) \cap F \neq \emptyset$. An NBW $\mathcal{A}$ accepts a word w if $\mathcal{A}$ has an accepting run over w. We use $L(\mathcal{A})$ to denote the set of words that are accepted by $\mathcal{A}$. For $s \in S$, we denote by $\mathcal{A}^{(s)}$ the automaton $\mathcal{A}$ with a single initial state s. We write $L(s)$ (the language of s) for $L(\mathcal{A}^{(s)})$ when $\mathcal{A}$ is clear from the context.

Before we define an alternating Büchi word automaton, we need the following definition. For a given set X, let $\mathcal{B}^+(X)$ be the set of positive Boolean formulas over X (i.e., Boolean formulas built from elements in X using $\wedge$ and $\vee$), where we also allow the formulas **true** and **false**. Let $Y \subseteq X$. We say that Y *satisfies* a formula $\theta \in \mathcal{B}^+(X)$ if the truth assignment that assigns *true* to the members of Y and assigns *false* to the members of $X \setminus Y$ satisfies θ.

Example 1. The sets $\{s_1, s_3\}$ and $\{s_1, s_4\}$ both satisfy the formula $(s_1 \vee s_2) \wedge (s_3 \vee s_4)$, while the set $\{s_1, s_2\}$ does not satisfy this formula.□

Definition 2. *A tree is a set $X \subseteq \mathbb{N}^*$, such that for $x \in \mathbb{N}^*$ and $n \in \mathbb{N}$, if $xn \in X$ then $x \in X$.*

We call the node ϵ the root of the tree. A node x is the *parent* of another node t and t is a *child* of x if t is of the form xn. The *level* of a node x, denoted $|x|$, is its distance from the root ε; in particular, $|\varepsilon| = 0$. A *branch* $\beta = x_0, x_1, \ldots$ of a tree is a maximal sequence of nodes such that x_0 is the root ε and x_i is the parent of x_{i+1} for all $0 < i < |\beta|$. Note that β can be finite or infinite. A *Σ-labelled tree*, for a finite alphabet Σ, is a pair $(\tau, \mathcal{T})$, where τ is a tree and $\mathcal{T}$ is a mapping from the nodes of τ to Σ that assigns to every node of τ a label in Σ. We often consider $\mathcal{T}$ as the labelled tree. A branch $\beta = x_0, x_1, \ldots$ of τ defines a word $\mathcal{T}(\beta) = \mathcal{T}(x_0), \mathcal{T}(x_1), \ldots$ consisting of the sequence of labels along the branch.

Definition 3. *An* alternating Büchi word automaton *(ABW) is $\mathcal{A} = \langle \Sigma, S, s_0, \delta, F \rangle$, where Σ, S, and F are as in NBW, $s_0 \in S$ is a single initial state, and $\delta : S \times \Sigma \to \mathcal{B}^+(S)$ is a transition function.*

Because of the universal choice in alternating transitions, a run of an alternating automaton is a tree rather than a sequence. A run of $\mathcal{A}$ on an infinite word $w = w_0, w_1, \ldots$ is a (possibly infinite) S-labelled tree τ such that $\mathcal{T}(\varepsilon) = s_0$ and the following holds:

> if $|x| = i$, $\mathcal{T}(x) = s$, and $\delta(s, w_i) = \theta$, then x has k children $x \cdot 1, \ldots, x \cdot k$, for some $k \leq |S|$, and $\{\mathcal{T}(x \cdot 1), \ldots, \mathcal{T}(x \cdot k)\}$ satisfies θ.

The run τ is *accepting* if every infinite branch in $\mathcal{T}$ includes infinitely many labels in F. Note that the run can also have finite branches; if $|x| = i$, $\tau(x) = s$, and $\delta(s, w_i) = \mathbf{true}$, then x need not have children.

We define the projected graph of ABW $\mathcal{A} = \langle \Sigma, S, \delta, s_0, F \rangle$ as a graph $\langle S, E \rangle$ where (s_1, s_2) is an edge in E if s_2 appears in $\delta(s_1, \sigma)$ for some $\sigma \in \Sigma$.

Definition 4. *An* alternating weak word automaton *(*AWW*) is an ABW $\mathcal{A} = \langle \Sigma, S, , s_0\delta, F \rangle$ such that for every strongly connected component C of the projected graph of $\mathcal{A}$, either $C \subseteq F$ or $C \cap F = \emptyset$.*

Lemma 1. [17] *Let $\mathcal{A}$ be an ABW. Then there exists an AWW $\mathcal{A}_w$ such that $L(\mathcal{A}) = L(\mathcal{A}_w)$ and the size of $\mathcal{A}_w$ is quadratic in the size of $\mathcal{A}$. Furthermore, $\mathcal{A}_w$ can be constructed in time quadratic in the size of $\mathcal{A}$.*

Given two AWW $\mathcal{A}_1$ and $\mathcal{A}_2$, we can construct AWW for $\Sigma^\omega \setminus L(\mathcal{A}_1)$, $L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$, and $L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$, which are linear in their size, relative to $\mathcal{A}_1$ and $\mathcal{A}_2$ [24].

Next, we present a translation of ABWs into NBWs. Both the translation and its correctness are derived directly from [22]. We say that a tuple $(Q_1, Q_2) \subseteq 2^{S \times S}$ is consistent with respect to $\mathcal{A}$ if $Q_2 \subseteq Q_1 \setminus F$.

Definition 5. *Given an ABW $\mathcal{A} = \langle \Sigma, S, s_0, \delta, F \rangle$, we define $N(\mathcal{A})$ as the NBW $\mathcal{A}_n = \langle \Sigma, S_n, (\{s_0\}, \{s_0\} \setminus F), \delta_n, F_n \rangle$, such that:*

- *S_n is the set of consistent tuples with respect to $\mathcal{A}$.*
- *(Q'_1, Q'_2) is in $\delta_n((Q_1, Q_2), \sigma)$ iff $Q'_1 \models \bigwedge_{s \in Q_1} \delta(s, \sigma)$ and either:*
  - *$Q_2 = \emptyset$ and $Q'_2 = Q'_1 \setminus F$, or*
  - *$Q_2 \neq \emptyset$ and there exists a set $Y_2 \subseteq Q'_1$ such that $Y_2 \models \bigwedge_{s \in Q_2} \delta(s, \sigma)$, and $Q'_2 = Y_2 \setminus F$.*
- *$F_n = \{(Q_1, Q_2) \in S_n | Q_2 = \emptyset\}$.*

Theorem 1. [22] *Let Q_1 be a subset of S, and let Q_2 be a subset of $Q_1 \setminus F$. Then $\cap_{s \in Q_1} L(\mathcal{A}^{(s)}) = L(\mathcal{A}_n^{(Q_1, Q_2)})$.*

2.2 Markov chains

We model probabilistic systems by finite *Markov chains*. The basic intuition is that transitions between states are governed by some probability distribution.

Definition 6. *A Markov chain is a tuple $\mathcal{M} = \langle X, P_T, P_I \rangle$ such that X is a set of states, $P_T : (X \times X) \to [0, 1]$ is a transition probability distribution that assigns to every transition (x_1, x_2) its probability. P_T satisfies that for every $x_1 \in X$ we have $\sum_{x_2 \in X} P_T(x_1, x_2) = 1$. $P_I : X \to [0, 1]$ is an initial probability distribution that satisfies $\sum_{x \in X} P_I(x) = 1$.*

We denote by $\mathcal{M}^{(x)}$ the Markov chain $\mathcal{M}$ with P_I that maps x to 1. Sometimes we consider a Markov chain as a graph $\langle X, E \rangle$ where $(x_1, x_2) \in E$ iff $P_T(x_1, x_2) > 0$. For an alphabet $\Sigma = 2^{AP}$, let $V : X \to \Sigma$ be a labelling function, then each path ρ in X^* or in X^ω in $\mathcal{M}$ is mapped by V to a word w in Σ^* or in Σ^ω respectively. For simplicity we assume that $\Sigma = X$ and that $V(x) = x$ for every $x \in X$, this simplification does not change the complexity of verifying the Markov chain [10]. Note, that every infinite path of $\mathcal{M}$ is a word in X^ω but the converse does not necessarily hold.

As in the standard theory of Markov processes (see [15, 16]), we define a probability space called the *sequence space* $\Psi_M = (\Omega, \Delta, P_M)$, where $\Omega = \Sigma^\omega$ is the set of all infinite sequences of states, Δ is the *Borel field* generated by the *basic cylindric sets*

$$\Delta(w_0, w_1, \ldots, w_n) = \{w \in \Omega | \text{ such that } w \text{ has prefix } w_0, w_1, \ldots, w_n\},$$

and P_M is a probability distribution defined by

$$P_M(\Delta(w_0, w_1, \ldots, w_n)) =$$

$$P_I(w_0) \cdot P_T(w_0, w_1) \cdot P_T(w_1, w_2) \cdot \ldots \cdot P_T(w_{n-1}, w_n).$$

(Δ is the smallest set that contains the basic cylindric sets, and is closed under negation, and countable union.) Consider a language $L \subseteq \Sigma^\omega$, viewed as a specification, that is measurable w.r.t. the sequence space Ψ_M. We say that $\mathcal{M}$ *almost surely satisfies* L if $P_M(L) = 1$, that is, if "almost" all computations of $\mathcal{M}$ are in L. In this paper we are concerned with the language of an ABW $\mathcal{A}$, thus we want to determine whether $P_M(L(\mathcal{A})) = 1$. It is shown in [31] that ω-regular languages are measurable.

The following property of Markov chains is called *ergodicity* and is proved in [16]. Let $\mathcal{M}$ be a Markov chain, then a path of $\mathcal{M}$, with probability one, enters a bottom

strongly connected component (BSCC) K of $\mathcal{M}$, and contains every finite path in K infinitely often. In other words, let L_e be the set of infinite words of $\mathcal{M}$ such that every word w in L_e has a suffix that is contained in a BSCC K of $\mathcal{M}$, and contains every finite path in K infinitely often. Then, $P_M(L_e) = 1$.

3 The Full Automaton and The Local Transition System

In this section we capture the behavior of an AWW $\mathcal{A}$ using two nondeterminstic systems. First we define the full automaton, which captures the languages of subsets of the states of $\mathcal{A}$. Then we define the local transition system, which captures the local relations between subsets of states of $\mathcal{A}$.

3.1 The Full Automaton

Given a AWW $\mathcal{A} = \langle \Sigma, S, \delta, F \rangle$ (we ignore its initial state), we define its dual AWW $\hat{\mathcal{A}} = \langle \Sigma, S, \hat{\delta}, \hat{F} \rangle$, where the Boolean formula $\hat{\delta}(s, \sigma)$ is obtained from $\delta(s, \sigma)$ by replacing every **true** with **false** and vice versa, and every $\vee$ with $\wedge$ and vice versa, in addition we define $\hat{F} = S \setminus F$. It is easy to see that $\hat{\mathcal{A}}$ is an AWW.

Lemma 2. [24] *Let $\mathcal{A}$ be an AWW and $\hat{\mathcal{A}}$ be its dual AWW. For every state s we have that $L(\mathcal{A}^{(s)}) = \Sigma^\omega \setminus L(\hat{\mathcal{A}}^{(s)})$.*

Given an AWW $\mathcal{A}$ and its dual AWW $\hat{\mathcal{A}}$ we define the state space of the full automaton as a subset of $2^S \times 2^S \times 2^S \times 2^S$. We start with the following definition.

Definition 7. *A tuple (Q_1, Q_2, Q_3, Q_4) is* consistent *if $Q_2 = S \setminus Q_1$, $Q_3 \subseteq Q_1 \setminus F$, and $Q_4 \subseteq Q_2 \setminus \hat{F}$.*

Definition 8. *Given an AWW $\mathcal{A} = \langle \Sigma, S, \delta, F \rangle$ we define its* full automaton *as the NBW $\mathcal{A}_f = \langle \Sigma, S_f, \delta_f, F_f \rangle$ where*

- *S_f is the set of consistent tuples over $2^S \times 2^S \times 2^S \times 2^S$.*
- *A state (Q_1', Q_2', Q_3', Q_4') is in $\delta_f((Q_1, Q_2, Q_3, Q_4), \sigma)$ if $Q_1' \models \wedge_{s \in Q_1} \delta(s, \sigma)$, $Q_2' \models \wedge_{s \in Q_2} \hat{\delta}(s, \sigma)$, and either:*
  1. *$Q_3 = Q_4 = \emptyset$, $Q_3' = Q_1' \setminus F$, and $Q_4' = Q_2' \setminus \hat{F}$*
  2. *$Q_3 \neq \emptyset$ or $Q_4 \neq \emptyset$, there exists $Y_3 \subseteq Q_1'$ such that $Y_3 \models \wedge_{s \in Q_3} \delta(s, \sigma)$ and $Q_3' = Y_3 \setminus F$, and there exists $Y_4 \subseteq Q_2'$ such that $Y_4 \models \wedge_{s \in Q_4} \hat{\delta}(s, \sigma)$ and $Q_4' = Y_4 \setminus \hat{F}$*
- *$F_f = \{(Q_1, Q_2, Q_3, Q_4) \in S_f | Q_3 = Q_4 = \emptyset)\}$*

The full automaton can be considered as an intersection of the two NBWs $N(\mathcal{A})$ and $N(\hat{\mathcal{A}})$. The definition of the full automaton restricts the states of the intersection to consistent tuples.

Theorem 2. *Let $\mathcal{A}$ be an AWW and let $\mathcal{A}_f$ be its full automaton, let $Q \subseteq S$ be a set of states, then for every state (Q_1, Q_2, Q_3, Q_4) such that $Q_1 = Q$ we have that*

$$\bigcap_{s \in Q} L(\mathcal{A}^{(s)}) \bigcap_{s \notin Q} \overline{L(\mathcal{A}^{(s)})} = L(\mathcal{A}_f^{(Q_1, Q_2, Q_3, Q_4)})$$

Theorem 2 is proved in Appendix A.1. We now present more properties of the full automaton. We use these properties later.

Definition 9. *Let $\mathcal{A}$ be an ABW and let w be an infinite word. We define the* type *of w w.r.t. $\mathcal{A}$ as the set $type_{\mathcal{A}}(w) = \{s | \mathcal{A}^{(s)}$ accepts $w\}$.*

The following lemma is a direct consequence of Theorem 2.

Lemma 3. *Let $\mathcal{A}_f$ be full automaton, and let (Q_1, Q_2, Q_3, Q_4) and (Q'_1, Q'_2, Q'_3, Q'_4) be states of $\mathcal{A}_f$. Then,*

- *If $Q_1 = Q'_1$ then $L(\mathcal{A}_f^{(Q_1,Q_2,Q_3,Q_4)}) = L(\mathcal{A}_f^{(Q'_1,Q'_2,Q'_3,Q'_4)})$.*
- *If $Q_1 \neq Q'_1$ then $L(\mathcal{A}_f^{(Q_1,Q_2,Q_3,Q_4)}) \cap L(\mathcal{A}_f^{(Q'_1,Q'_2,Q'_3,Q'_4)}) = \emptyset$.*

Lemma 3 and Theorem 2 imply that the first element Q_1 of the states of $\mathcal{A}_f$ characterizes a distinct language.

Definition 10. *The language of Q_1 is defined as $L(Q_1) = L(\mathcal{A}_f^{(Q_1, S \setminus Q_1, \emptyset, \emptyset)})$.*

3.2 The local transition system

As observed above, it is sufficient to look at Q_1 in order to determine the language of a state of $\mathcal{A}_f$. Recall that $L(Q_1) = L(\mathcal{A}^{(Q_1, S\setminus Q_1, \emptyset, \emptyset)})$. We observe that if there exists a transition $((Q_1, Q_2, Q_3, Q_4), \sigma, (Q'_1, Q'_2, Q'_3, Q'_4))$ in δ_f, then for every state of the form (Q_1, Q_2, Y_3, Y_4) there exists a state (Q'_1, Q'_2, Y'_3, Y'_4) such that the transition $((Q_1, Q_2, Y_3, Y_4), \sigma, (Q'_1, Q'_2, Y'_3, Y'_4))$ is in δ_f.

These observations imply that there are some local relationships between the languages of the states of $\mathcal{A}_f$. Indeed, if a word w is in $L((Q'_1, Q'_2, Q'_3, Q'_4))$ then for the word $\sigma \cdot w$ that is in $L((Q_1, Q_2, Q_3, Q_4))$, there exists a state of the form (Q'_1, Q'_2, Y'_3, Y'_4) that is in $\delta_f((Q_1, Q_2, Q_3, Q_4), \sigma)$. Thus, we can say that there exists a transition on σ from $L(Q_1)$ to $L(Q'_1)$. The local transition system captures these relationships.

Definition 11. *Given an AWW $\mathcal{A} = \langle \Sigma, S, \delta, F \rangle$ we define its* local transition system *as $T_{\mathcal{A}} = \langle \Sigma, S_T, \delta_T \rangle$ where*

- *S_T is the set of subsets of S and δ_T is a function from S_T to 2^{S_T}.*
- *A state Q' is in $\delta_T(Q, \sigma)$ if $Q' \models \wedge_{s \in Q} \delta(s, \sigma)$ and $(S \setminus Q') \models \wedge_{s \notin Q} \hat{\delta}(s, \sigma)$.*

Example 2. We now present an example of a full automaton and a local transition system. The example is presented at Figure 1. For simplicity we use a deterministic automaton $\mathcal{A}$. The figure shows $\mathcal{A}$'s dual automaton $\hat{\mathcal{A}}$, the full automaton $\mathcal{A}_f$, and the local transition system T_A. Note that $\hat{\mathcal{A}}$ has $\hat{F} = S$, thus for every state (Q_1, Q_2, Q_3, Q_4) of $\mathcal{A}_f$, we have $Q_4 = \emptyset$. For this reason, and since Q_2 is always equal to $S \setminus Q_1$, we only write the sets Q_1 and Q_3 inside the states. □

The definitions of the full automaton and the local transition system implies the following lemma:

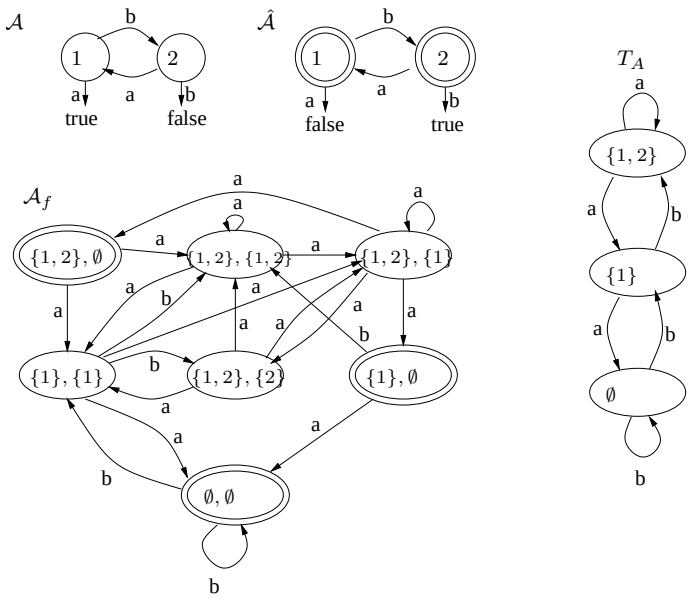

Fig. 1. An example of a full automaton $\mathcal{A}_f$ and a local transition system T_A.

Lemma 4. *Let $\mathcal{A}$ be an AWW and let $\mathcal{A}_f$ and $T_{\mathcal{A}}$ be its full automaton and local transition system respectively. Let (Q_1, Q_2, Q_3, Q_4) be a state of $\mathcal{A}_f$, and let σ be a letter in Σ. Then, for every state Q_1' we have that Q_1' is in $\delta_T(Q_1, \sigma)$ iff there exists a state of the form (Q_1', Q_2', Q_3', Q_4') in $\delta_f((Q_1, Q_2, Q_3, Q_4), \sigma)$.*

The proof of Lemma 4 is straightforward from the definitions of $\mathcal{A}_f$ and T_A. In particular for every state (Q_1, Q_2, Q_3, Q_4) and infinite word w we have that $\mathcal{A}^{(Q_1,Q_2,Q_3,Q_4)}$ has a run on w iff $T_A^{(Q_1)}$ has a run on w. However, we do not define accepting conditions for the local transition system. Thus, it is possible that $T_A^{(Q_1)}$ has a run on w, but $\mathcal{A}_f^{(Q_1,Q_2,Q_3,Q_4)}$ does not have an accepting run on w.

Lemma 5. *Let T_A be a local transition system, and let Q, Q' and Q'' be states of $T_{\mathcal{A}}$. Let σ be a letter in Σ. If $Q'' \in \delta_T(Q', \sigma)$ and $Q'' \in \delta_T(Q, \sigma)$, then $Q = Q'$.*

Lemma 5 is proved in Appendix A.2. When a transition system satisfies the property shown in Lemma 5, we say that the transition system is *reverse deterministic*.

4 Verifying Markov Chains

In this section we construct a product $G_{\mathcal{M},\mathcal{A}}$ of the Markov chain $\mathcal{M}$ and the local transition system T_A. We show that the problem of checking whether $P_M(L(\mathcal{A})) = 1$, can be reduced to checking for a state (x, Q) of G whether the probability of $L(Q) \cap x \cdot \Sigma^\omega$ is positive. Then, we show how to use the full automaton to solve this problem.

Definition 12. *Let $\mathcal{A}$ be an AWW, $\mathcal{T}_A$ be $\mathcal{A}$'s local transition system, and $\mathcal{M}$ be a Markov chain. We define the graph $G_{\mathcal{M},\mathcal{A}}$ as having vertex set (x, Q) such that x is a state of $\mathcal{M}$ and Q is a state of T_A. An edge $(x, Q) \to (x', Q')$ is included in $G_{\mathcal{M},\mathcal{A}}$ if $\mathcal{M}$ has a transition $x \to x'$ and (Q, x, Q') is a transition in T_A.*

When $\mathcal{A}$ and $\mathcal{M}$ are clear from the context, we write G instead of $G_{\mathcal{M},\mathcal{A}}$. Lemma 5 implies that for every three states (x, Q), (x', Q'), and (x'', Q''), if there is a transition from (x, Q) to (x'', Q'') and there is a transition from (x', Q') to (x'', Q''), then $x \neq x'$. We say that G is *reverse deterministic*.

Example 3. We present in Figure 2 two Markov chains $\mathcal{M}_1$ and $\mathcal{M}_2$. We assume that the initial probability for each state in the Markov chains is $\frac{1}{2}$. The figure also presents the products G_1 and G_2 of $\mathcal{M}_1$ and $\mathcal{M}_2$ respectively, with the local transition system T_A from Example 2. □

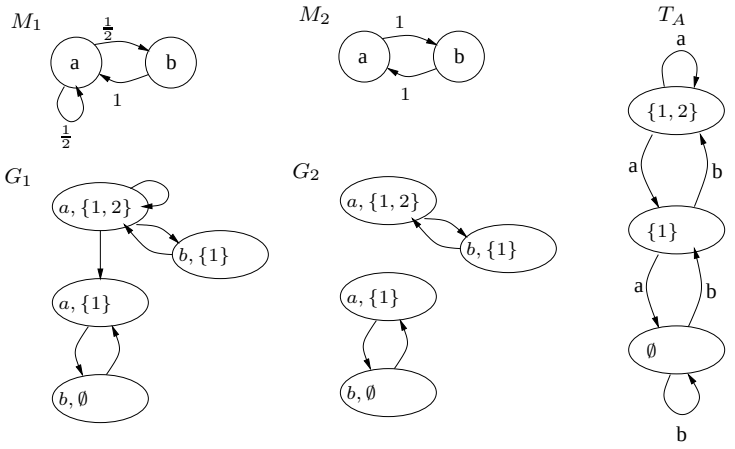

Fig. 2. Two Markov chains $\mathcal{M}_1$ and $\mathcal{M}_2$, and the graphs they impose G_1 and G_2.

Every infinite path in G projected on the first component gives a path in $\mathcal{M}$. Conversely, every path of $\mathcal{M}$ is the projection of at least one path in G. In fact, let $w = x_0 \cdot x_1 \cdot \ldots$ be a path of $\mathcal{M}$. For each j, let Q_j be the type of the suffix $x_j \cdot x_{j+1} x_{j+2} \cdots$. Then for each j there is a transition (Q_j, x_j, Q_{j+1}) in δ_T and thus an edge $(x_j, Q_j) \to (x_{j+1}, Q_{j+1})$ in G. We call this path the *augmented path* corresponding to w.

Definition 13. *For a state (x, Q) of G we denote by $P(x, Q)$ the probability that a path that starts in state x has type Q, namely $P(x, Q) = P_M(\mathcal{M}^{(x)}$ has type $Q)$. We call* (x, Q) probable *if $P(x, Q) > 0$.*

The importance of the probable states is demonstrated in the following two lemmas, which are proved in Appendix B.

Lemma 6. $P_M(L(Q)) = \sum_{x \in X} P_I(x) \cdot P(x, Q)$.

Let x be a state of a Markov chain with $P_I(x) > 0$. Then for every state (x, Q) we have that if (x, Q) is probable, then $P_M(L(Q)) > 0$. Thus we conclude Lemma 7.

Lemma 7. *Let s be a state of an AWW $\mathcal{A}$ and let $\mathcal{M}$ be a Markov chain. Then, $P_M(L(s)) < 1$ iff there exists a state x of $\mathcal{M}$ and a set $Q \subseteq S$ such that $P_I(x) > 0$, $s \notin Q$ and the state (x, Q) is probable.*

Thus, in order to determine whether $P_M(L(s_0)) = 1$, it is enough to determine the probable states of G. In the rest of this section we show how to identify the probable states. We define H as the restriction of G to the probable states.

Example 4. Look again at Figure 2. It is easy to see that given that a path of $\mathcal{M}_1$ starts at state a, the path is of the form $a((ba)+a)^\omega$ with probability 1. $a((ba)+a)^\omega$ is contained in $L(\{1,2\}) \cup \{(ab)^\omega\}$, since $P_{\mathcal{M}_1}((ab)^\omega) = 0$, we have $P(a, \{1,2\}) = 1$. Similarly, a path of $\mathcal{M}_1$ that starts at b is with probability one in $L(\{1\})$, thus, $P(b, \{1\}) = 1$. This implies that in G_1, H is the subgraph induced by the states $(a, \{1,2\})$ and $(b, \{1\})$. On the other hand, looking at $\mathcal{M}_2$, we see that a path that starts at a is of the form $(ab)^\omega$ and a path that starts at b is of the form $(ba)^\omega$. Thus, in G_2, H is the subgraph induced by the states $(a, \{1\})$ and $(b, \emptyset)$. □

We start with the following observation. Partition the language $L(Q)$ according to the first letter of a word and the type of the suffix that starts from the second letter. Then, for every state (x, Q) of G we have $P(x, Q) = \sum_{(x,Q)\to(x',Q')} P_T(x, x') \cdot P(x', Q')$. Note that if $(x, Q) \to (x', Q')$ is an edge of G, then $P_T(x, x') > 0$ and thus $P(x, Q) \geq P_T(x, x') \cdot P(x', Q')$. Hence, if (x', Q') is probable then all its ancestors are probable. This implies that it is sufficient to identify the BSCCs of H and then to construct the set of probable states using backward reachability.

Let C be an SCC of G. If it contains some probable state (x, Q), then since all the states in C are ancestors of (x, Q), all states in C are probable. That is, either C is an SCC of H or $C \cap H = \emptyset$. Recall that every path in C projects to a path in $\mathcal{M}$. So the set of first components of all members of C are in the same SCC, say K, of $\mathcal{M}$, which is the SCC of $\mathcal{M}$ containing x. We say that C *corresponds* to K. Note that distinct SCC's of G may correspond to the same K.

Theorem 3 characterizes the SCCs of G which are the BSCCs of H. Before we present the theorem we need the following notation. For a tuple $E = \langle E_1, E_2, \ldots E_n \rangle$, we define $\pi_i(E) = E_i$ to be the i'th element in E. This notation is extended naturally to sequences of tuples.

Definition 14. *A finite path ρ_G in G is* fulfilling *if there exists a path ρ_f in $\mathcal{A}_f$ such that $\pi_2(\rho_G) = \pi_1(\rho_f)$, the first state of ρ_f is of the form $(Q_1, Q_2, (Q_1 \setminus F), (Q_2 \setminus \hat{F}))$, and the last state of ρ_f is of the form $(Q_1', Q_2', \emptyset, \emptyset)$.*

Theorem 3. *Let C be an SCC of G. Then C is a BSCC of H iff it satisfies the following conditions:*

1. *C corresponds to a BSCC K of $\mathcal{M}$.*
2. *Every finite path of K is a projection of a path in C.*
3. *C contains a fulfilling path.*

The proof of Theorem 3 is highly nontrivial and is presented at Appendix B.1.

5 Algorithms

In Figure 3 we present the algorithm that determines for an AWW $\mathcal{A}$ and a Markov chain $\mathcal{M}$ whether $P_M(L(\mathcal{A})) = 1$. An extension for exact probability is presented in Appendix D. Theorem 3 implies that the algorithm mark the BSCCs of H. Thus, B is the set of probable states. Lemma 7 implies that the algorithm returns **true** iff $P_M(L(\mathcal{A})) = 1$. Finding SCCs in G that correspond to BSCCs in $\mathcal{M}$, and doing

```
Inputs: Markov chain M = ⟨X, P_I, P_T⟩, AWW A = ⟨Σ, S, s_0, δ, F⟩.
Construct the full automaton A_f of A.
Construct the local transition system T_A and the graph G.
Mark all SCCs C of G that satisfy:
   1. C corresponds to a BSCC K of M.
   2. Every finite path of K is a projection of a path in C.
   3. C contains a fulfilling path.
Construct the set B of all states of G from which the marked
SCCs are reachable.
return true iff for every state (x, Q) ∈ B, if P_I(x) > 0, then s_0 ∈ Q.
```

Fig. 3. The model-checking algorithm

backward reachability can be done in time linear in the size of G. The most complex part of the algorithm is to identify, SCCs C of G that satisfy:

1. C corresponds to a BSCC K of $\mathcal{M}$.
2. Every finite path of K is a projection of a path in C.
3. C contains a fulfilling path.

The following lemma is proved in [10]. The only property of G that they use is that G is reverse deterministic.

Lemma 8. *Let C be an SCC of G that corresponds to an SCC K of $\mathcal{M}$. Then the following are equivalent:*

1. *Every finite path in K is a projection of a path in C.*
2. *No other SCC of G corresponding to K is an ancestor of C.*

Lemma 8 implies that the second task is equivalent to checking whether there is no ancestor SCC of C that corresponds to K. This check can be easily done while scanning the SCCs of G.

Example 5. In G_1 at Figure 2 there are two SCC's that correspond to the single BSCC of $\mathcal{M}_1$. The SCC of $(a, \{1, 2\})$ and $(b, \{1\})$ does not have ancestors and contains the fulfilling path $(a, \{1, 2\}), (a, \{1, 2\}), (a, \{1, 2\})$ that corresponds to the path $(\{1, 2\}, \{1, 2\}), (\{1, 2\}, \{1\}), (\{1, 2\}, \emptyset)$ in $\mathcal{A}_f$, thus, this SCC is the BSCC of H. In G_2 there are two SCCs that correspond to the single BSCC of $\mathcal{M}_2$ and neither of them have an ancestor. However, only the SCC of $(a, \{1\})$ and $(b, \emptyset)$ has a fulfilling path, thus it is the BSCC of H.□

We now explain how to check whether an SCC C of G contains a fulfilling path. We construct the product $G_f = \mathcal{M} \times \mathcal{A}_f$, similarly to the construction of G.

Definition 15. *Let $\mathcal{A}$ be an AWW, $\mathcal{A}_f$ be $\mathcal{A}$'s full automaton, and $\mathcal{M}$ be a Markov chain. We define the* full graph G_f *as having vertex set $(x, (Q_1, Q_2, Q_3, Q_4))$ such that x is a state of $\mathcal{M}$ and (Q_1, Q_2, Q_3, Q_4) is a state of $\mathcal{A}_f$. An edge $(x, (Q_1, Q_2, Q_3, Q_4)) \to (x', (Q'_1, Q'_2, Q'_3, Q'_4))$ is included in G_f if $\mathcal{M}$ has a transition $x \to x'$ and (Q'_1, Q'_2, Q'_3, Q'_4) is in $\delta_f((Q_1, Q_2, Q_3, Q_4), x)$.*

Lemma 9. *An SCC C of G contains a fulfilling path iff there exists a path $(x_0, (Q^0_1, Q^0_2, Q^0_3, Q^0_4)), (x_1, (Q^1_1, Q^1_2, Q^1_3, Q^1_4)), \ldots, (x_n, (Q^n_1, Q^n_2, \emptyset, \emptyset))$ in G_f such that the path $(x_0, Q^0_1), (x_1, Q^1_1), \ldots (x_n, Q^n_1)$ is contained in C, $Q^0_3 = Q^0_1 \setminus F$, and $Q^0_4 = Q^0_2 \setminus \hat{F}$.*

Lemma 9 is proved in Appendix C.

Complexity Finding SCCs in G that correspond to BSCCs in $\mathcal{M}$, and doing backward reachability can be done in linear time and polylogarithmic space in $|G|$. Constructing BSCCs of $\mathcal{M}$ can be done in time linear in $|\mathcal{M}|$, identifying SCCs of G that correspond to these BSCC can be done in time linear in $|G|$. Marking SCCs that do not have ancestors that correspond to the same BSCC in $\mathcal{M}$ can also be done in time linear in $|G|$. Checking that an SCC of G contains a fulfilling path can be done in time linear in $|G_f|$, simply by scanning G_f and G in parallel, thus, the algorithm can be implemented in time linear in $|\mathcal{M} \times \mathcal{A}_f|$. Since the size of $\mathcal{A}_f$ is $2^{O(|\mathcal{A}|)}$, we have that the time complexity of the algorithm is $|\mathcal{M}| \cdot 2^{O(|\mathcal{A}|)}$.

As for space complexity we show that algorithm works in space polynomial in $|\mathcal{A}|$ and polylogarithmic in $|\mathcal{M}|$. We rewrite the conditions of Theorem 3, Lemma 7, and Lemma 8 as follows: $P_M(L(\mathcal{A})) < 1$ iff there exists a probable state (x_0, Q_0) such that $s_0 \notin Q$ and $P_I(x_0) > 0$. This is true iff (x_0, Q_0) reaches a state (x, Q) that is in a BSCC of H, $s_0 \notin Q_0$, and $P_I(x_0) > 0$. That is

1. (x, Q) is reachable from a state (x_0, Q_0) such that $P_I(x_0) > 0$ and $s_0 \notin Q_0$.
2. x is in a BSCC of $\mathcal{M}$ (Theorem 3, (1)). This condition is equivalent to the following: for every state x' of $\mathcal{M}$ we have that if there exists a path in $\mathcal{M}$ from x to x' then there exists a path from x' to x.
3. No other SCC of G that corresponds to the SCC of x in $\mathcal{M}$ is the ancestor of the SCC of (x, Q) (Lemma 8). This condition is equivalent to the following: for every state (x', Q'), if there exists a path from (x', Q') to (x, Q), then either there exists a path from (x, Q) to (x', Q'), or there is no path from x to x'.
4. The SCC of (x, Q) contains a fulfilling path (Theorem 3, (3)). By Lemma 9 this condition is equivalent to the following: there exists a path in G_f from a state $(x', (Q'_1, Q'_2, Q'_1 \setminus F, Q'_2 \setminus \hat{F}))$ to a state $(x'', (Q''_1, Q''_2, \emptyset, \emptyset))$ such that the projection of the path on G is contained in the SCC of (x, Q). This condition is equivalent to: there is a path from a state $(x', (Q'_1, Q'_2, Q'_1 \setminus F, Q'_2 \setminus \hat{F}))$ to a state $(x'', (Q''_1, Q''_2, \emptyset, \emptyset))$ in G_f and there are paths from (x, Q) to (x'', Q''_1), from (x'', Q''_1) to (x', Q'), and from $((x', Q')$ to (x, Q) in G.

In [27] it is shown that checking whether there is a path from one state to another in a graph with n states requires $log^2(n)$ space. This implies that the conditions above can be checked in space $O(log^2(|G_f|)) = log^2(|\mathcal{M}| \cdot 2^{O(|\mathcal{A}|)}) = O(log^2(|\mathcal{M}|) + log(|\mathcal{M}|) \cdot |\mathcal{A}| + |\mathcal{A}|^2) = O(log^2(|\mathcal{M}|) + |\mathcal{A}|^2)$.

6 Concluding remarks

We presented here an optimal solution to the general problem of linear-time probabilistic model checking with respect to ω-regular specifications, expressed by alternating automata. Beyond the interest in the problem itself, our solution is interesting from a theoretical perspective, since the concept of full automaton may have other applications. More work is needed in reducing our result to practice. One direction is to extend the *ProbaTaf* system, which currently handles LTL specifications of Markov chain [11], to alternating automata specifications. Another, is to combine the symbolic approach to alternating automata [21] with the symbolic approach to probabilistic model checking [3].

References

1. R. Armoni, D. Bustan, O. Kupferman, and M. Y. Vardi. Resets vs. aborts in linear temporal logic. In *Int'l Conf. on Tools and Algorithms for Construction and Analysis of Systems*, pages 65 – 80, 2003.
2. R. Armoni, L. Fix, R. Gerth, B. Ginsburg, T. Kanza, A. Landver, S. Mador-Haim, A. Tiemeyer, E. Singerman, M.Y. Vardi, and Y. Zbar. The ForSpec temporal language: A new temporal property-specification language. In *Proc. 8th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'02)*, LNCS 2280, pages 296–311, 2002.
3. C. Baier, E.M. Clarke, V. Hartonas-Garmhausen, M.Z. Kwiatkowska, and M. Ryan. Symbolic model checking for probabilistic processes. In *Automata, Languages and Programming, 24th Inte.l Colloq.*, LNCS 1256, pages 430–440, 1997.
4. I. Beer, S. Ben-David, C. Eisner, D. Fisman, A. Gringauze, and Y. Rodeh. The temporal logic sugar. In *Proc. Conf. on Computer-Aided Verification*, LNCS 2102, pages 363–367, 2001.
5. P. Berman and J.A Garay. Randomized distributed agreement revisited. In *Proceedings of the 23rd Inte'l Symp. on Fault-Tolerant Computing (FTCS '93)*, pages 412–421, 1993.
6. E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. on Programming Languages and Systems*, 8(2):244–263, 1986.
7. E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.
8. R. Cole, B.M. Maggs, F. M. auf der Heide, A. Richa M. Mitzenmacher, K. Schroder, R. Sitaraman, and B. Vocking. Randomized protocols for low-congestion circuit routing in multistage interconnection networks. In *30th ACM Symp. on Theo. of Comp. (STOC)*, pages 378–388, 1998.
9. C. Courcoubetis and M. Yannakakis. Markov decision processes and regular events. In *Proc. 17th Int'l Coll. on Automata Languages and Programming*, volume 443, pages 336–349. LNCS, 1990.

10. C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, 1995.
11. J. M. Couvreur, N. Saheb, and G. Sutre. An optimal automata approach to LTL model checking of probabilistic systems. In *Proc. 10th Int. Conf. Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'2003), Almaty, Kazakhstan, Sep. 2003*, volume 2850 of *Lecture Notes in Artificial Intelligence*. Springer, 2003.
12. C. Fritz and T. Wilke. State space reductions for alternating Büchi automata: Quotienting by simulation equivalences. In *FST TCS 2002: Foundations of Software Technology and Theoretical Computer Science: 22nd Conf.*, volume 2556 of *LNCS*, pages 157–168, 2002.
13. P. Gastin and D. Oddoux. Fast LTL to Büchi automata translation. In *Computer Aided Verification, Proc. 13th Inte'l Conf.*, volume 2102 of *LNCS*, pages 53–65, 2001.
14. G.J. Holzmann. The model checker SPIN. *IEEE Trans. on Software Engineering*, 23(5):279–295, May 1997. Special issue on Formal Methods in Software Practice.
15. J.G. Kemeny and J.L. Snell. *Finite Markov Chains*. Van Nostrad, Princeton, 1960.
16. J.G. Kemeny, J.L. Snell, and A.W. Knapp. *Denumerable Markov Chains*. Springer-Verlag, 1976.
17. O. Kupferman and M.Y. Vardi. Weak alternating automata are not that weak. In *Proc. 5th Israeli Symp. on Theory of Computing and Systems*, pages 147–158. IEEE Computer Society Press, 1997.
18. R.P. Kurshan. *FormalCheck User's Manual*. Cadence Design, Inc., 1998.
19. O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *Proc. 12th ACM Symp. on Principles of Programming Languages*, pages 97–107, 1985.
20. O. Lichtenstein, A. Pnueli, and L. Zuck. The glory of the past. In *Logics of Programs*, volume 193 of *Lecture Notes in Computer Science*, pages 196–218, Brooklyn, June 1985. Springer-Verlag.
21. S. Merz. Weak alternating automata in Isabelle/HOL. In J. Harrison and M. Aagaard, editors, *Theorem Proving in Higher Order Logics: 13th International Conference*, volume 1869 of *Lecture Notes in Computer Science*, pages 423–440. Springer-Verlag, 2000.
22. S. Miyano and T. Hayashi. Alternating finite automata on ω-words. *Theoretical Computer Science*, 32:321–330, 1984.
23. M.J. Morley. Semantics of temporal $\boldsymbol{e}$. In T. F. Melham and F.G. Moller, editors, Banff'99 *Higher Order Workshop (Formal Methods in Computation)*. University of Glasgow, Department of Computing Science Technic al Report, 1999.
24. D.E. Muller, A. Saoudi, and P.E. Schupp. Alternating automata, the weak monadic theory of the tree and its complexity. In *Proc. 13th Intel Colloq. on Automata, Languages and Programming*, volume 226 of *LNCS*, 1986.
25. J.P. Queille and J. Sifakis. Specification and verification of concurrent systems in Cesar. In *Proc. 5th Inte'l Symp. on Programming*, volume 137 of *LNCS*, pages 337–351, 1981.
26. S. Safra. On the complexity of ω-automata. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 319–327, White Plains, October 1988.
27. W.J. Savitch. Relationship between nondeterministic and deterministic tape complexities. *Journal on Computer and System Sciences*, 4:177–192, 1970.
28. A.P. Sistla and E.M. Clarke. The complexity of propositional linear temporal logic. *J. ACM*, 32:733–749, 1985.
29. S. Thrun. Probabilistic algorithms in robotics. *AI Magazine*, 21(4):93–109, 2000.
30. M. Y. Vardi. Probabilistic linear-time model checking: An overview of the automata-theoretic approach. In *Formal Methods for Real-Time and Probabilistic Systems: 5th Inte'l AMAST Workshop*, volume 1601 of *LNCS*, pages 265–276, 1999.

31. M.Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proc. 26th IEEE Symp. on Foundations of Computer Science*, pages 327–338, Portland, October 1985.
32. M.Y. Vardi. A temporal fixpoint calculus. In *Proc. 15th ACM Symp. on Principles of Programming Languages*, pages 250–259, San Diego, January 1988.
33. M.Y. Vardi. Nontraditional applications of automata theory. In *Proc. Inte'l Symp. on Theoretical Aspects of Computer Software*, volume 789, pages 575–597. LNCS, 1994.
34. M.Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Logics for Concurrency: Structure versus Automata*, volume 1043 of *LNCS*, pages 238–266, 1996.
35. M.Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proc. 1st Symp. on Logic in Computer Science*, pages 332–344, Cambridge, June 1986.
36. M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, November 1994.
37. P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1–2):72–99, 1983.

A Proofs for Section 3

A.1 Proof of Theorem 2

Lemma 10. *Let $\mathcal{A}$ be an AWW, let s be a state in S, and let σ be a letter in Σ. Then, for every $Q \subseteq S$, we have that $Q \models \delta(s,\sigma)$ iff $S \setminus Q \not\models \hat{\delta}(s,\sigma)$.*

Lemma 10 can be proved by induction on the structure of the formula $\delta(s,\sigma)$.

Lemma 11. *Let $\mathcal{A}$ be an AWW, let w be an infinite word, and let Q be a subset of S s.t w is in $\bigcap_{s\in Q} L(\mathcal{A}^{(s)}) \bigcap_{s\notin Q} \overline{L(\mathcal{A}^{(s)})}$. Then, $Q = type_{\mathcal{A}}(w)$.*

Proof. Since $w \in \bigcap_{s\in Q} L(\mathcal{A}^{(s)})$, we have $Q \subseteq type_{\mathcal{A}}(w)$. Since $w \in \bigcap_{s\notin Q} \overline{L(\mathcal{A}^{(s)})}$, we have $type_{\mathcal{A}}(w) \subseteq Q$, thus, $Q = type_{\mathcal{A}}(w)$. □

Lemma 12. *Let $\mathcal{A}$ be an AWW, let $\hat{\mathcal{A}}$ be its dual, and let w be an infinite word. Then $S \setminus type_{\mathcal{A}}(w) = type_{\hat{\mathcal{A}}}(w)$.*

Proof. Let s be a state, then $s \in S \setminus type_{\mathcal{A}}(w)$ iff $w \notin L(\mathcal{A}^{(s)})$. By Lemma 2, $w \notin L(\mathcal{A}^{(s)})$ iff $w \in L(\hat{\mathcal{A}}^{(s)})$ iff $s \in type_{\hat{\mathcal{A}}}(w)$. □

Lemma 13. *Let $\mathcal{A}$ be an AWW and let w be an infinite word. Then for every i and every $s \in type_{\mathcal{A}}(w^i)$, we have $type_{\mathcal{A}}(w^{i+1}) \models \delta(s, w_i)$*

Proof. Let s be a state in $type_{\mathcal{A}}(w^i)$, then $\mathcal{A}^{(s)}$ accepts w^i. Let Q' be the set of successors of s in an accepting run tree of $\mathcal{A}^{(s)}$ on w^i. Then, for every $t \in Q'$ we have that $\mathcal{A}^{(t)}$ accepts w^{i+1}, thus $Q' \subseteq type_{\mathcal{A}}(w^{i+1})$. Since, $Q' \models \delta(s, w_i)$, we have $type_{\mathcal{A}}(w^{i+1}) \models \delta(s, w_i)$. □

Definition 16. *Let $\mathcal{A}$ be an AWW and $\mathcal{A}_f$ be the full automaton of $\mathcal{A}$. Let w be an infinite word. We define $Max_{\mathcal{A}}(w)$ to be the state $(type_{\mathcal{A}}(w), type_{\hat{\mathcal{A}}}(w), type_{\mathcal{A}}(w) \setminus F, type_{\hat{\mathcal{A}}}(w) \setminus \hat{F})$ of $\mathcal{A}_f$.*

Lemma 12 implies that $Max_A(w)$ is a consistent tuple.

Lemma 14. *Let w be an infinite word and let $\mathcal{A}$ be an AWW. Then there exists a run of $\mathcal{A}_f$ on a prefix of w that starts at $Max_{\mathcal{A}}(w)$ and that reaches a state in F_f.*

Proof. Let $(Q_1^0, Q_2^0, Q_3^0, Q_4^0) = Max_{\mathcal{A}}(w)$. We define two sets of trees. Φ_3 is a set of $|Q_1^0|$ many run trees s.t. for every $s \in Q_1^0$ there exists a unique tree in Φ_3 that is an accepting run tree of $\mathcal{A}^{(s)}$ on w. Similarly, we define Φ_4 as a set of $|Q_2^0|$ many run trees s.t. for every $t \notin Q_1^0$ there exists a unique tree in Φ_4 that is an accepting run tree of $\hat{\mathcal{A}}^{(t)}$ on w. We define $\Phi = \Phi_3 \cup \Phi_4$. Note that every path of a tree in Φ_3 contains infinitely many accepting states in F, and every path of a tree in Φ_4 contains infinitely many accepting states in $\hat{F}$. Furthermore, the branching degree of every node in these tree is bounded by $|S|$. By Königs lemma, there exists an index i s.t. for every tree in Φ and every path ξ in the tree there exists $k \leq i$ s.t. ξ_k is an accepting state.

We construct a run of $\mathcal{A}_f^{(Q_1^0, Q_2^0, Q_3^0, Q_4^0)}$ on w inductively, until it reaches a state in F_f. The run satisfies the following properties:

1. For every $k \leq i$ we have that if a state s is in Q_3^k, then there exists a path ξ in a tree in Φ_3 s.t. $\xi_0, \xi_1, \ldots, \xi_k$ does not contain states in F and $\xi_k = s$.
2. For every $k \leq i$ we have that if a state t is in Q_4^k, then there exists a path ξ in a tree in Φ_4 s.t. $\xi_0, \xi_1, \ldots, \xi_k$ does not contain states in $\hat{F}$ and $\xi_k = t$.

The first state of the run is $(Q_1^0, Q_2^0, Q_3^0, Q_4^0)$. It easy to see that the properties above hold for $k = 0$. Suppose that for some $k < i$, we have a prefix of a run of $\mathcal{A}_f$ that satisfies the properties above. We define $Q_1^{k+1} = type_{\mathcal{A}}(w^{k+1})$, $Q_2^{k+1} = S \setminus Q_1^{k+1}$. As for Q_3^{k+1} and Q_4^{k+1}, the induction hypothesis implies that for every state s in Q_3^k there exists a path ξ^s in a tree in Φ_3 s.t. $\xi_0^s, \xi_1^s, \ldots, \xi_k^s$ does not contain states in F and $\xi_k^s = s$. Let Y_3^s be the set of successors of ξ_k^s in the tree. We define $Y_3 = \cup_{s \in Q_3^k} Y_3^s$, and $Q_3^{k+1} = Y_3 \setminus F$. Since for every s in Q_3^k we have that the successors of ξ_k^s in the run tree accepts w^{k+1}, we have that $Y_3 \subseteq type_{\mathcal{A}}(w^{k+1})$, thus, $Y_3 \subseteq Q_1^{k+1}$. In a similar way we define $Y_4 \subseteq Q_2^{k+1}$ using trees of Φ_4, and define $Q_4^{k+1} = Y_4 \setminus \hat{F}$. Since $Y_3 \subseteq Q_1^{k+1}$ and $Y_4 \subseteq Q_2^{k+1}$, the state $(Q_1^{k+1}, Q_2^{k+1}, Q_3^{k+1}, Q_4^{k+1})$ is consistent.

Obviously, the properties above holds for $k+1$. We prove that $(Q_1^{k+1}, Q_2^{k+1}, Q_3^{k+1}, Q_4^{k+1})$ is in $\delta_f((Q_1^k, Q_2^k, Q_3^k, Q_4^k), w_k)$. Lemma 13 implies that for every state $s \in Q_1^k$ we have $Q_1^{k+1} \models \delta(s, w_k)$. Lemma 12 implies that $Q_2^{k+1} = type_{\hat{\mathcal{A}}}(w^{k+1})$, Lemma 13 implies that for every $t \in Q_2^k$, we have $Q_2^{k+1} \models \hat{\delta}(t, w_k)$. The definition of Y_3 implies that for every state s in Q_3^k we have $Y_3 \models \delta(s, w_k)$. Similarly, for every $t \in Q_4^k$ we have $Y_4 \models \hat{\delta}(s, w_k)$. This completes the proof that $(Q_1^{k+1}, Q_2^{k+1}, Q_3^{k+1}, Q_4^{k+1})$ is in $\delta_f((Q_1^k, Q_2^k, Q_3^k, Q_4^k), w_k)$.

Since every path in a tree in Φ contains an accepting state before the i'th state, the properties above imply that the run reaches an accepting state before the i'th state. □

Lemma 15. *Let $\mathcal{A}$ be a AWW and let $\mathcal{A}_f$ be its full automaton, let $Q \subseteq S$ be a set of states, then for every state (Q_1, Q_2, Q_3, Q_4) s.t. $Q_1 = Q$ we have that*

$$\bigcap_{s \in Q} L(\mathcal{A}^{(s)}) \bigcap_{s \notin Q} \overline{L(\mathcal{A}^{(s)})} \subseteq L(\mathcal{A}_f^{(Q_1, Q_2, Q_3, Q_4)})$$

Proof. We prove that every w in $\bigcap_{s \in Q} L(\mathcal{A}^{(s)}) \bigcap_{s \notin Q} \overline{L(\mathcal{A}^{(s)})}$ is also in $L(\mathcal{A}_f^{(Q_1, Q_2, Q_3, Q_4)})$. We construct the run of $\mathcal{A}_f$ inductively in finite sequences, s.t. for every sequence, the last state $(Q_1^i, Q_2^i, Q_3^i, Q_4^i)$ in the sequence is $Max_{\mathcal{A}}(w^i)$. Moreover, every sequence except for the first, contains an accepting state. Since, the run contains infinitely many sequences, the run is accepting.

- Base: Lemma 11 implies that $Q_1 = type_{\mathcal{A}}(w)$. Lemma 19 implies that there exists a transition from (Q_1, Q_2, Q_3, Q_4) to $Max_{\mathcal{A}}(w^1)$ on w_0. This is the first sequence.
- Induction step: Suppose that the i'th sequence ends at state $(Q_1^k, Q_2^k, Q_3^k, Q_4^k)$ s.t. $(Q_1^k, Q_2^k, Q_3^k, Q_4^k) = Max_{\mathcal{A}}(w^k)$. Then, Lemma 14 implies that there exists a run of $\mathcal{A}_f$ that starts from $(Q_1^k, Q_2^k, Q_3^k, Q_4^k)$ and reaches an accepting state $(Q_1^j, Q_2^j, \emptyset, \emptyset)$ for some $j > k$. The definition of $\mathcal{A}_f$ implies that there exists a transition from $(Q_1^j, Q_2^j, \emptyset, \emptyset)$ to $Max_{\mathcal{A}}(w^{j+1})$ on w_j. We take $(Q_1^{k+1}, Q_2^{k+1}, Q_3^{k+1}, Q_4^{k+1}), \ldots (Q_1^j, Q_2^j, \emptyset, \emptyset), Max_{\mathcal{A}}(w^{j+1})$ to be the $i+1$'th sequence. □

Lemma 16. *Let $\mathcal{A}$ be a AWW and let $\mathcal{A}_f$ be its full automaton, let $Q \subseteq S$ be a set of states, then for every state (Q_1, Q_2, Q_3, Q_4) s.t. $Q_1 = Q$ we have that $\bigcap_{s\in Q} L(\mathcal{A}^{(s)}) \bigcap_{s\notin Q} \overline{L(\mathcal{A}^{(s)})} \supseteq L(\mathcal{A}_f^{(Q_1,Q_2,Q_3,Q_4)})$*

Proof. We prove that every w in $L(\mathcal{A}_f^{(Q_1,Q_2,Q_3,Q_4)})$ is also in $\bigcap_{s\in Q} L(\mathcal{A}^{(s)}) \bigcap_{s\notin Q} \overline{L(\mathcal{A}^{(s)})}$. Let $\rho_f = (Q_1^0, Q_2^0, Q_3^0, Q_4^0), (Q_1^1, Q_2^1, Q_3^1, Q_4^1), \ldots$ be an accepting run of $\mathcal{A}_f^{(Q_1,Q_2,Q_3,Q_4)}$ on w and let s be a state in Q_1^0. We need to show that w is in $\mathcal{A}^{(s)}$. Let $\mathcal{A}_n = N(\mathcal{A})$ as defined in Definition 5. We prove that $\mathcal{A}_n^{(Q_1,Q_3)}$ has an accepting run on w, by Theorem 1 this implies that w is in $L(\mathcal{A}^{(s)})$. Similarly, we can prove that w is in $L(\hat{A}^{(t)})$ for every state t in Q_2^0.

Let $i_0 = -1$ and let $i_1, i_2, \ldots$ be an infinite sequence of indices s.t. for every $j \geq 1$, the i_j'th state of ρ is in F_f. We construct a run $\rho_n = (Y_1^0, Y_2^0), (Y_1^1, Y_2^1), \ldots$ s.t. For every j and $i_j < k \leq i_{j+1}$ we define $Y_1^k = Q_1^k$, and

$$Y_2^k = \begin{cases} Y_1^k \setminus F & \text{if } Y_2^l = \emptyset \text{ for some } i_j < l < k \\ Q_3^k & \text{otherwise} \end{cases}$$

We prove that ρ_n is an accepting run of $\mathcal{A}_n$ on w. We prove by induction on j that for every $j \geq 0$ we have that

1. $Y_1^{i_j+1} = Q_1^{i_j+1}$ and $Y_2^{i_j+1} = Q_3^{i_j+1}$.
2. For every $i_j < k \leq i_{j+1}$, there exists a transition from (Y_1^k, Y_2^k) to (Y_1^{k+1}, Y_2^{k+1}) on w_k.
3. There exists $i_j < k \leq i_{j+1}$ s.t. $Y_2^k = \emptyset$

Since for every $j \geq 0$ there exists $k > i_j$ s.t. $Y_2^k = \emptyset$, the run is accepting.

We start with two observations. The definitions of $\mathcal{A}_f$ and $\mathcal{A}_n$ imply the following observation.

Observation **17** *For a state (Q_1, Q_2, Q_3, Q_4) of $\mathcal{A}_f$ s.t. $Q_3 \neq \emptyset$, a transition from (Q_1, Q_2, Q_3, Q_4) to (Q_1', Q_2', Q_3', Q_4') on σ in $\mathcal{A}_f$ implies a transition from (Q_1, Q_3) to (Q_1', Q_3') in $\mathcal{A}_n$.*

Recall that if there exists a transition in $\mathcal{A}_f$ from (Q_1, Q_2, Q_3, Q_4) to (Q_1', Q_2', Q_3', Q_4') on σ, then there exists a transition in $\mathcal{A}_f$ from (Q_1, Q_2, Q_3, Q_4) to $(Q_1', Q_2', Q_1' \setminus F, Q_2' \setminus \hat{F})$ on σ. This implies that:

Observation **18** *If there exists a transition in $\mathcal{A}_f$ from (Q_1, Q_2, Q_3, Q_4) to (Q_1', Q_2', Q_3', Q_4') on σ, then there exist a transition in $\mathcal{A}_n$ from (Q_1, Q_3) to $(Q_1', Q_1' \setminus F)$ on σ.*

Next, we prove the induction claim.

– Base case $j = 0$: The definition of ρ_n implies property 1. Let l be the smallest index s.t. $i_j < l \leq i_{j+1}$ and $Q_3^l = \emptyset$. Then Observation 17 implies that for every $i_j < k < l$ property 2 holds. By the definition of $\mathcal{A}_n$, property 3 holds in index $k = l$. Observation 18 implies that property 2 holds for $l \leq k \leq i_{j+1}$.

– Induction step: Assume that the three properties hold for j, we prove that they hold for $j+1$. Since $(Q_1^{i_j}, Q_2^{i_j}, Q_3^{i_j}, Q_4^{i_j})$ is in F_f, we have $Q_3^{i_j+1} = Q_1^{i_j+1} \setminus F$. The induction hypothesis implies that for some $i_{j-1} < k \leq i_j$, we have $Y_2^k = \emptyset$ thus $Y_2^{i_j+1} = Y_1^{i_j+1} \setminus F$, thus property 1 holds. Properties 2 and 3 can be proved as in the base case. □

Lemma 16 completes the proof of Theorem 2.

A.2 Proofs for the rest of Section 3

Lemma 19. *Let $\mathcal{A}$ be an AWW, let $\mathcal{A}_f$ be its full automaton, and let w be an infinite word. Then, there exist a transition in $\mathcal{A}_f$ from every state of the form $(type_{\mathcal{A}}(w), S \setminus type_{\mathcal{A}}(w), Q_3, Q_4)$ to $Max_{\mathcal{A}}(w^1)$ on w_0.*

Proof. By Lemma 12, we have $S \setminus type_{\mathcal{A}}(w) = type_{\hat{\mathcal{A}}}(w)$. Lemma 13 implies that for every state s in $type_{\mathcal{A}}(w)$, we have $type_{\mathcal{A}}(w^1) \models \delta(s, w_0)$, and that for every state t in $type_{\hat{\mathcal{A}}}(w)$, we have $type_{\hat{\mathcal{A}}}(w^1) \models \hat{\delta}(t, w_0)$. If $Q_3 = Q_4 = \emptyset$, then the definition of δ_f implies the lemma directly. Otherwise, since $(type_{\mathcal{A}}(w), S \setminus type_{\mathcal{A}}(w), Q_3, Q_4)$ is consistent, we have that $Q_3 \subseteq type_{\mathcal{A}}(w) \setminus F$, and $Q_4 \subseteq type_{\hat{\mathcal{A}}}(w) \setminus \hat{F}$. We take $Y_3 = type_{\mathcal{A}}(w^1)$ and $Y_4 = type_{\hat{\mathcal{A}}}(w^1)$. These sets satisfy the conditions of δ_f. □

Theorem 2 implies the following lemma.

Lemma 20. *For an AWW $\mathcal{A}$ and set of state Q, we have $L(Q) = \{w | type_{\mathcal{A}}(w) = Q\}$.*

Lemma 5 Let $T_{\mathcal{A}}$ be a local transition system, and let Q, Q' and Q'' be states of $T_{\mathcal{A}}$. Let σ be a letter in Σ. If $Q'' \in \delta_T(Q', \sigma)$ and $Q'' \in \delta_T(Q, \sigma)$, then $Q = Q'$.

Proof. Assume to the contrary that the lemma does not hold, meaning, $Q'' \in \delta_T(Q', \sigma)$ and $Q'' \in \delta_T(Q, \sigma)$, but $Q \neq Q'$. Then, w.l.o.g. there exists a state s in Q that is not in Q'. The definition of $T_{\mathcal{A}}$ implies that $Q'' \models \delta(s, \sigma)$. Lemma 10 implies that $S \setminus Q'' \not\models \hat{\delta}(s, \sigma)$. Thus, $Q'' \notin \delta_T(Q', \sigma)$, contradiction. □

B Proofs for Section 4

Lemma 6 $P_M(L(Q)) = \sum_{x \in X} P_I(x) \cdot P(x, Q)$.

Proof. We partition the language $L(Q)$ according to the first letter of the word. Then, we have that $P_M(L(Q) \cap \{w | w \text{ starts with } x\}) = P_I(x) \cdot P(x, Q)$. Thus, $P_M(L(Q)) = \sum_{x \in X} P_I(x) \cdot P(x, Q)$.□

Let x be a state of a Markov chain with $P_I(x) > 0$. Then for every state (x, Q) we have that if (x, Q) is probable, then $P_M(L(Q)) > 0$. Thus we conclude the following lemma.

Lemma 7 Let s be a state of an AWW $\mathcal{A}$ and let $\mathcal{M}$ be a Markov chain. Then, $P_M(L(s)) < 1$ iff there exists a state x of $\mathcal{M}$ and a set $Q \subseteq S$ s.t. $P_I(x) > 0$, $s \notin Q$ and the state (x, Q) is probable.

Proof. Let w be a word in $\overline{L(s)}$, then $s \notin type_A(w)$. By Lemma 20, $w \in L(Q)$ for some Q s.t. $s \notin Q$. This implies that $\overline{L(s)} \subseteq \cup_{s \notin Q} L(Q)$. By Theorem 2, for every Q s.t. $s \notin Q$ we have $L(Q) \subseteq \overline{L(s)}$, thus, $\overline{L(s)} = \cup_{s \notin Q} L(Q)$. Then, $P_M(L(s)) < 1$ iff $P_M(\overline{L(s)}) > 0$ iff there exists a set Q s.t. $s \notin Q$ and $P_M(L(Q)) > 0$ iff there exists Q s.t. $s \notin Q$, a state x such that $P_I(x) > 0$, and $P(x, Q) > 0$. □

B.1 Proof of Theorem 3

Theorem 3 Let C be an SCC of G. Then C is a BSCC of H iff it satisfies the following conditions:

1. C corresponds to a BSCC K of $\mathcal{M}$.
2. Every finite path of K is a projection of a path in C.
3. C contains a fulfilling path.

In the rest of the section we prove theorem 3. We start by showing that every $BSCC$ of H satisfies the conditions of the theorem. Lemma 21 makes a connection between the paths of $\mathcal{M}$ and the pathes of H.

Lemma 21. *Let C be a BSCC of H. Let (x, Q) be a state in C, and let w be a path of $\mathcal{M}$ that starts at x and has type Q. Then, with probability one, the augmented path of w is contained in C.*

Proof. Let $\Theta = \{w | w$ starts at x, has type Q, and its augmented path is not contained in $C\}$. We prove that $P_{\mathcal{M}^{(x)}}(\Theta) = 0$. First, note that all the augmented pathes of the paths in Θ start at (x, Q). Since non-probable states do not have probable successors, for every word w in Θ there exists a unique i such that the i'th state in the augmented path of w is in H and the $i+1$'th state of the augmented path of w is not. We define

$$\Theta_i = \{w | w \text{ starts at } x, \text{ has type } Q, \text{ and its augmented path exits } C \text{ in its } i\text{'th state}\},$$

then we have $\Theta = \cup_{i=1}^{\infty} \Theta_i$. Thus, it is sufficient to prove that for every i we have $P_{\mathcal{M}^{(x)}}(\Theta_i) = 0$. We further partition Θ_i according to the type Q' of the suffix of w that starts at w_{i+1} and the prefix $\rho = w_0, w_1, \ldots, w_{i+1}$. Note that (w_{i+1}, Q') is not probable. For a set Q' of states and a finite path ρ of length $i+1$, we define

$$\Theta_i^{Q',\rho} = \{w | Q' \text{ is the type of } w^{i+1}, \rho = w_0, w_1, \ldots, w_{i+1}, \text{ and } P(w_{i+1}, Q') = 0\}$$

Then $\Theta_i \subseteq \cup_{\rho \in X^{i+1}, Q' \subseteq S} \Theta_i^{Q',\rho}$. For the set $\Theta_i^{Q',\rho}$ we have $P_{\mathcal{M}^{(x)}}(\Theta_i^{Q',\rho}) \leq P_{\mathcal{M}^{(x)}}(\rho) \cdot P(w_{i+1}, Q')$. Since $P(w_{i+1}, Q') = 0$, we have that $P_{\mathcal{M}^{(x)}}(\Theta_i^{Q',\rho}) = 0$. This implies that $P_{\mathcal{M}^{(x)}}(\Theta_i) = 0$, so $P_{\mathcal{M}^{(x)}}(\Theta) = 0$. □

Lemma 22. *Let C be a BSCC of H that corresponds to an SCC K of $\mathcal{M}$. Then K is a BSCC of $\mathcal{M}$.*

Proof. Let (x, Q) be a state in a BSCC C of H. Suppose K is not a BSSC of $\mathcal{M}$. Then a path ρ starting from $x \in K$ almost surely leaves K, meaning it has a suffix with no letters in K. Since (x, Q) is in H, with positive probability ρ has type Q. This implies that with positive probability the augmented path of ρ starts at (x, Q) and its projection has a suffix with no letters in K. This implies with positive probability the augmented path of ρ starts at (x, Q) and exits C, meaning exits H. This contradicts Lemma 21. □

Lemma 23. *Let C be a BSCC of H that corresponds to a BSCC K of $\mathcal{M}$. Then every finite path in K is the projection of some path in C.*

Proof. Suppose that α is a finite path in K that is not the projection of a path in C. Let x be a state in K, and let (x, Q) be a state in C. Since (x, Q) is in C, with positive probability a path w of $\mathcal{M}$ that starts at x has type Q. By ergodicity, a path that starts at x almost surely eventually takes α. Thus, with positive probability, a path w that starts at x has type Q and contains α. Let ρ be the augmented path of a path w that starts at x and has type Q. By the definition of augmented path, ρ starts at (x, Q), and since the projection of ρ contains α, ρ exits C. This implies that with positive probability the augmented path of a path of $\mathcal{M}$ that starts at x exits C, which contradicts Lemma 21. □

We now prove that every BSCC of H contains a fulfilling path. We start by showing that every augmented path in G contains a fulfilling path.

Lemma 24. *Let w be a path in $\mathcal{M}$. Then the augmented path ρ_G of w contains a fulfilling path.*

Proof. Let Q_1 be the type of w. By Lemma 20, there exists an accepting run ρ_f of $\mathcal{A}^{(Q_1,Q_2,\emptyset,\emptyset)}$ on w. For every i, since ρ_f^i is a run of $\mathcal{A}^{(Q_1^i,Q_2^i,Q_3^i,Q_4^i)}$ on w^i, the first element Q_1^i of the ith state of ρ_f is $type_{\mathcal{A}}(w^i)$. Together with the definition of augmented path this implies that $\pi_1(\rho_f) = \pi_2(\rho_G)$. Since ρ_f is an accepting run, it contains infinitely many accepting states of the form $(Y_1, Y_2, \emptyset, \emptyset)$. Recall that from a state of the form $(Y_1, Y_2, \emptyset, \emptyset)$ all the outgoing transitions in $\mathcal{A}_f$ enter states of the form $(Y_1', Y_2', (Y_1' \setminus F), (Y_2' \setminus \hat{F}))$. This implies that ρ_G contains a fulfilling path. □

Lemma 25. *Let C be a BSCC of H, then C contains a fulfilling path.*

Proof. Let (x, Q) be a state in C. Since C is in H, (x, Q) is probable. Thus, with positive probability, a path that starts with x is of type Q. Lemma 21 implies that with probability one an augmented path of a path that starts in x and has a type Q is contained in C. Thus, with positive probability, the augmented path of a path that starts at x is contained in C. Since there is at least one outgoing path out of x, this implies that there exists a path that starts at x such that its augmented path is contained in C. Together with Lemma 24, this implies that C contains a fulfilling path. □

This completes the proof that of the first direction of Theorem 3. In the rest of this section we prove the other direction of Theorem 3. Let C be an SCC of G, we prove that if C satisfies the conditions of the theorem, then C is a BSCC of H. Let C be an SCC of G that corresponds to a BSCC K of $\mathcal{M}$ such that every finite path in K is a projection of a path in C. The next four lemmas prove that if C contains a fulfilling path, then C is a BSCC of H.

Lemma 26. *Let C be an SCC of G that corresponds to a BSCC K of $\mathcal{M}$ such that every finite path of K is a projection of a path in C. Let ρ_C be an infinite path in C and let ρ_K be the projection of ρ_C. If ρ_K contains every finite path in K infinitely many times, then ρ_C contains every finite path in C infinitely many times.*

Proof. Every suffix of ρ_K contains every finite path in K infinitely many times. Assume that ρ_C contains some finite path α of C only finitely many times, then there exists a suffix ρ'_C of ρ_C such that its projection ρ'_K on $\mathcal{M}$ contains every finite path in K infinitely often and ρ'_c does not contain α at all. Thus, it is enough to prove that ρ_C contains every finite path of C. Assume now to the contrary that there is a finite path α in C that is not contained in ρ_C. For every finite path ρ in C we define $\Xi(\rho)$ to be the set of finite paths ρ' in C that have the same projection on K as ρ and do not contain α. Let $\xi(\rho)$ be the set of states in C that are the last states of the paths in $\Xi(\rho)$.

We define a sequence $\alpha_0, \alpha_1, \ldots$ of paths in C such that for every i such that $|\xi(\alpha_i))| \geq 1$, we have that $|\xi(\alpha_i)| > |\xi(\alpha_{i+1})|$, and α_i contains α. Since $\xi(\alpha_0)$ is a finite set, there exists n such that $|\xi(\alpha_n)| = 0$. Let β_i be the projection of α_i, for $i \geq 0$. Then all paths in C with projection β_n contain α. Since ρ_K contains β_n, we have that ρ_C contains α, contradiction.

We define $\alpha_0 = \alpha$. Let α_i be the i'th path in the sequence. Let (x, Q) be a state in $\xi(\alpha_i)$, and let ρ be the path in $\Xi(\alpha_i)$ (so it has projection β_i) that ends in (x, Q). We define α_{i+1} to be $\alpha\gamma\rho$ where γ is some path that connect α to ρ. Since C is an SCC, there exists such a path. Note that α_{i+1} has a suffix with projection β_i thus $\xi(\alpha_{i+1}) \subseteq \xi(\alpha_i)$, furthermore, the path with projection β_{i+1} that ends in (x, Q) contains α, thus $(x, Q) \notin \xi(\alpha_{i+1})$. □

Lemma 19, Lemma 4, and the definition of G imply the following lemma.

Lemma 27. *Let $\rho_G = (x^0, Q^0), (x^1, Q^1), \ldots$ be a finite or infinite path in G, then:*

1. *Let ρ_f be a run of $\mathcal{A}_f$ such that $\pi_1(\rho_f) = \pi_2(\rho_g)$. Then ρ_f is a run of $\mathcal{A}_f$ on $\pi_1(\rho_G)$.*
2. *The sequence $(Q^0, S \setminus Q^0, Q^0 \setminus F, (S \setminus Q^0) \setminus \hat{F}), (Q^1, S \setminus Q^1, Q^1 \setminus F, (S \setminus Q^1) \setminus \hat{F}), (Q^2, S \setminus Q^2, Q^2 \setminus F, (S \setminus Q^2) \setminus \hat{F}), \ldots$ is a run of $\mathcal{A}^f$ on $x^0, x^1, \ldots$.*

Lemma 28. *Let α be a fulfilling path in G. Let ρ_G be a path in G that starts at a state (x, Q_1) and contains α infinitely often. Then $\mathcal{A}_f^{(Q_1, Q_2, (Q_1 \setminus F), (Q_2 \setminus \hat{F}))}$ accepts the projection $\pi_1(\rho_G)$ of ρ_G on $\mathcal{M}$.*

Proof. We show an accepting run of $\mathcal{A}_f^{(Q_1, Q_2, (Q_1 \setminus F), (Q_2 \setminus \hat{F}))}$ on $\pi_1(\rho_G)$. Let $i_1 = 0$ and $i_2, i_3, \ldots$ be an infinite sequence of indices such that for every $j \geq 1$, we have that the subpath $\rho_G^{i_{2j}}, \rho_G^{i_{2j}+1}, \ldots, \rho_G^{i_{2j+1}}$ of ρ_G is α. Let α_f be the path of $\mathcal{A}_f$ that corresponds to α as defined in Definition 14.

We define a run of $\mathcal{A}_f$ as follows, for every $j \geq 1$, the part of the run that starts at i_{2j} and ends at i_{2j+1} is α_f. For every $j \geq 0$ and $i_{2j+1} < k < i_{2j+2}$, the k'th state of the run is $Q_1^k = \rho_G^k$, $Q_2^k = S \setminus Q_1^k$, $Q_3^k = Q_1^k \setminus F$, and $Q_4^k = Q_2^k \setminus \hat{F}$.

Lemma 27 implies that the path that we describe is a run of $\mathcal{A}_f^{(Q_1, Q_2, (Q_1 \setminus F), (Q_2 \setminus \hat{F}))}$ on the projection of ρ_G. Since ρ_G contains infinitely many fulfilling paths, the run contains infinitely many accepting states. □

Lemma 29. *Let C be an SCC of G that corresponds to a BSCC K of $\mathcal{M}$ and satisfies the conditions below.*

- *Every finite path of K is a projection of a path in C.*
- *C contains a fulfilling path.*

Then, C is a BSCC of H.

Proof. First we pove that C is contained in H. Since for every SCC C of G either $C \subseteq H$ or $C \cap H = \emptyset$, it is enough to prove that C contains at least one probable state. We prove that with probability one a path ρ_K that starts from a state x in K is a projection of a path ρ_C that starts at a state (x, Q_1) in C, such that $\mathcal{A}_f^{(Q_1, S\setminus Q_1, (Q_1\setminus F), (S\setminus Q_1\setminus \hat{F}))}$ accepts ρ_K, that is, $type_{\mathcal{A}}(\rho_K) = Q_1$. This implies that with probability one, a path that starts from a state x in K has type Q_1 such that (x, Q_1) is in C. This implies that at least one state in C is probable.

Let ρ_K be a path of K that starts at x. Let ρ_C be a path in C with projection ρ_K on K. Then, by ergodicity (see Section 2.2) ρ_K contains every finite path in K infinitely often. Lemma 26 implies that ρ_C contains every finite path in C infinitely often, and in particular it contains the fulfilling path infinitely often. Lemma 28 implies that $\mathcal{A}_f^{(Q_1, Q_2, (Q_1\setminus F), (Q_2\setminus \hat{F}))}$ accepts ρ_K.

Finally, we prove that C is a BSCC of H. Since C corresponds to a BSCC K of $\mathcal{M}$, every successor SCC C' of C in G corresponds to K too. Lemma 8 implies that there exists a finite path in K that is not the projection of a path in C'. By Lemma 23 C' is not a BSCC of H. Since C is a SCC of H and no successor SCC of C is a BSCC of H, we conclude that C is a BSCC of H □

C Proofs for Section 5

Lemma 30. *Let ρ be a path in G_f, then the projection of the states of ρ on G is a path in G.*

Proof. Let $((x, (Q_1, Q_2, Q_3, Q_4)), (x', (Q_1', Q_2', Q_3', Q_4')))$ be a transition in G_f, we prove that $((x, Q_1), (x', Q_1'))$ is a transition of G. The definition of G_f implies that $P_T(x, x') > 0$ and that $(Q_1', Q_2', Q_3', Q_4') \in \delta_f((Q_1, Q_2, Q_3, Q_4), x)$. Lemma 4 implies that (Q_1, x, Q_1') is a transition in T_A. By the definition of G, we have that $((x, Q_1), (x', Q_1'))$ is a transition of G. □

Lemma 9 An SCC C of G contains a fulfilling path iff there exists a path $(x_0, (Q_1^0, Q_2^0, Q_3^0, Q_4^0)), (x_1, (Q_1^1, Q_2^1, Q_3^1, Q_4^1)), \ldots, (x_n, (Q_1^n, Q_2^n, \emptyset, \emptyset))$ in G_f such that the path $(x_0, Q_1^0), (x_1, Q_1^1), \ldots (x_n, Q_1^n)$ is contained in C, $Q_3^0 = Q_1^0 \setminus F$, and $Q_4^0 = Q_2^0 \setminus \hat{F}$.
Lemma 9 is followed directly from Lemma 30.

Example 6. We present in Figure 4 the product G_{2f} of $\mathcal{A}_f$ and $\mathcal{M}_2$ which are presented in Figures 1 and 2. It easy to see that the only SCC in G_2 that contains a fulfilling path is the SCC of $(a, \{1\})$ and $(b, \emptyset)$ that contains the path $(a, \{1\}), (b, \emptyset)$ that correspond to the path $(a, \{1\}, \{1\}), (b, \emptyset, \emptyset)$ of G_{2f}. □

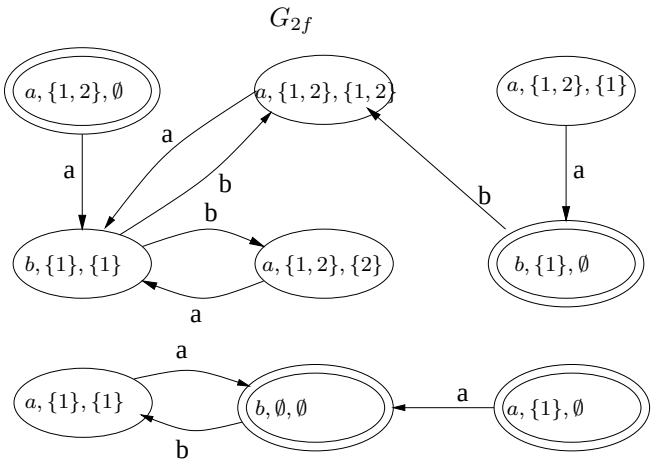

Fig. 4. The graph G_{2f}.

D Exact Probability of $L(\mathcal{A})$

In order to compute the exact probability $P_M(L(\mathcal{A}))$, we prove that the graph H can be defined as a Markov chain that refines $\mathcal{M}$. First we define a Markov chain over H. Define the initial probability $P_I^H(x,Q)$ as $P_I(x) \cdot P(x,Q)$. Define the probability of a transition $(x,Q) \to (x',Q')$ to be

$$P_T^H((x,Q),(x',Q')) = \frac{P_T(x,x')P(x',Q')}{P(x,Q)}.$$

Lemma 31. *H is a Markov Chain.*

Proof. The sum of the initial probabilities in H is equal to

$$\sum_{(x,Q)\in G} P_I(x)P(x,Q) = \sum_x P_I(x) \sum_Q P(x,Q)$$

Since every path that starts at x has a unique type, we have $\Sigma_Q P(x,Q) = 1$. Thus,

$$\sum_x P_I(x) \sum_Q P(x,Q) = \sum_x P_I(x) = 1.$$

Next, we show that for every state in H, the probabilities of its outgoing transitions are sum to one. Consider $(x,Q) \in H$. Then, the sum of the probabilities of the outgoing transitions from (x,Q) is:

$$\sum_{(x,Q)\to(x',Q')} \frac{P_T(x,x')P(x',Q')}{P(x,Q)} = \frac{1}{P(x,Q)} \cdot \sum_{x\to x'} P_T(x,x') \cdot \sum_{Q \xrightarrow{x} Q'} P(x',Q')$$

Thus, it is enough to prove that

$$\sum_{x \to x'} P_T(x, x') \cdot \sum_{Q \xrightarrow{x} Q'} P(x', Q') = P(x, Q).$$

We can partition the probability that a path that starts at x is accepted by Q according to the second letter of the path. For every x' the probability that a path starts at x its second letter is x' and it is accepted by Q is $P_T(x, x') \cdot \sum_{Q \xrightarrow{x'} Q'} P(x', Q')$.□

Next, we prove that H refines $\mathcal{M}$. We start by proving that for every regular language L we have $P_M(L) = P_H(L)$. We define a labelling function V for H such that $V(x, Q) = x$ thus both Markov chains are defined over the same alphabet.

Lemma 32. *For every ω-regular language L we have $P_M(L) = P_H(L)$.*

Proof. It is sufficient to show that $\mathcal{M}$ and H agree on the basic cylindrical sets. That is, for every finite word $w = x_x, x_1, \ldots, x_k$, we have $P_M(\Delta(w)) = P_H(\Delta(w))$. Recall that T_A is reverse deterministic. So for a fixed $w = x_0, \cdots, x_k$ and Q_k, there exists a unique path $\rho = (x_0, Q_0), \cdots, (x_k, Q_k)$ such that $V(\rho) = w$ and ρ ends in (x_k, Q_k). We define $\rho^{w,Q}$ to be the unique path in H such that $V(\rho^{w,Q}) = w$ and $\rho_k^{w,Q} = Q$. So,

$$\begin{aligned}
& P_H(\Delta(w)) \\
&= \sum_{V(\rho)=w} P_H(\rho), \text{ we partition according to the } k\text{'th state of } \rho \\
&= \sum_{Q} P_H(\rho^{w,Q})\}) \\
&= \sum_{Q} [P_I^H(\rho_0^{w,Q}) \prod_{0 \le i < k} P_T^H(\rho_i^{w,Q}, \rho_{i+1}^{w,Q})] \\
&= \sum_{Q} [P_I(x_0) P(\rho_0^{w,Q}) \prod_{0 \le i < q} P_T(x_i, x_{i+1}) \frac{P(\rho_{i+1}^{w,Q})}{P(\rho_i^{w,Q})}], \text{ definition of } P_T^H \\
&= \sum_{Q} [P_I(x_0) [\prod_{0 \le i < k} P_T(x_i, x_{i+1})]\, P(\rho_k^{w,Q})] \\
&= \sum_{Q} [P_M(w) \cdot P(\rho_k^{w,Q})] \text{ since } \rho_k^{w,Q} = (x_k, Q), \\
&= P_M(w). \square
\end{aligned}$$

In order to complete the proof that H refines $\mathcal{M}$ we need to prove that $P_M(L(Q_1, Q_2, Q_3, Q_4)) = \sum_{x \in X} P_I^H(x, Q_1)$. This is proved in Lemma 33.

Lemma 33. $P_M(L((Q_1, Q_2, Q_3, Q_4))) = \sum_{x \in X} P_I^H(x, Q_1)$.

Proof. Lemma 32 implies that $P_M(\{w | w_0 = x \text{ and } w \text{ has type } Q_1\}) = P_H(\{w | w_0 = x \text{ and } w \text{ has type } Q_1\})$. We need to prove that $P_H(\{w | w_0 = x \text{ and } w \text{ has type } Q_1\}) = P_I^H(x, Q_1)$. It is enough to prove that a projection of a path that starts in (x, Q_1) is with

probability one of type Q_1 meaning is accepted with probability one by $\mathcal{A}_f^{(Q_1,Q_2,(Q_1\setminus F),(Q_2\setminus \hat{F}))}$. Let ρ be a path that starts at (x,Q_1). By argodicity, ρ reaches a BSCC of H and then pass through every finite path of the BSCC infinitely often. In particulary, ρ pass infinitely often through a fulfilling path. Lemma 28 implies that $\mathcal{A}_f^{(Q_1,Q_2,(Q_1\setminus F),(Q_2\setminus \hat{F}))}$ accepts the projection of ρ on $\mathcal{M}$.□

Given a state s of $\mathcal{A}$, we have $L(s) = \cup_{s\in Q} L(Q)$, which implies that $P_M(L(s)) = \sum_{s\in Q}\sum_{x\in X} P_I^H(x,Q)$. Thus, it left to calculate the exact probabilities of H. We need to compute for every state (x,Q) of H, the probability $P(x,Q)$. In [10] it is shown that these probabilities form a unique solution to the following linear system:

1. for every state (x,Q) of H we have $P(x,Q) = \sum_{((x,Q),(x',Q'))\in G} P_T(x,x')P(x',Q')$.
2. For every state x of $\mathcal{M}$ we have $\sum_{Q\in 2^S} P(x,Q) = 1$.