Improved Algorithms for the Automata-Based Approach to Model-Checking

Laurent Doyen[1] and Jean-François Raskin[2]

[1] I&C, Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland
[2] CS, Université Libre de Bruxelles (ULB), Belgium

Abstract. We propose and evaluate new algorithms to support the automata-based approach to model-checking: algorithms to solve the universality and language inclusion problems for nondeterministic Büchi automata. To obtain those new algorithms, we establish the existence of pre-orders that can be exploited to efficiently evaluate fixed points on the automata defined during the complementation step (that we keep implicit in our approach). We evaluate the performance of our new algorithm to check for universality of Büchi automata experimentally using the random automaton model recently proposed by Tabakov and Vardi. We show that on the difficult instances of this probabilistic model, our algorithm outperforms the standard ones by several orders of magnitude. This work is an extension to the infinite words case of new algorithms for the finite words case that we and co-authors have presented in a recent paper [DDHR06].

1 Introduction

In the automata-based approach to model-checking [VW86,VW94], programs and properties are modeled by finite automata. Let A be a finite automaton that models a program and let B be a finite automaton that models a specification that the program should satisfy: all the traces of the program (executions) should be traces of the specification, that is $\mathcal{L}(A) \subseteq \mathcal{L}(B)$. To solve the inclusion problem, the classical automata-theoretic solution consists in complementing the language of the automaton B and then to check that $\mathcal{L}(A) \cap \mathcal{L}^c(B)$ is empty (the later intersection being computed as a product).

In the finite case, the program and the specification are finite automata over finite words (NFA) and the construction for the complementation is conceptually simple: it is achieved by a classical subset construction. In the case of infinite words, the program and (or at least) the specification are nondeterministic Büchi automata (NBW). The NBW are also complementable; this was first proved by Büchi in the late sixties [BL69]. However, the result is much harder to obtain than in the case of NFA. The orginal construction of Büchi has a $O(2^{2^n})$ worst case complexity (where n is the size of the automaton to complement) which is not optimal. In the late eighties Safra in [Saf88], and later Kupferman and Vardi in [KV97], have given optimal complementation procedures that have $O(2^{n \log n})$ complexity (see [Mic88] for the lower bound). While for finite words, the classical algorithm has been implemented and shown practically usable, for infinite words, the theoretically optimal solution is difficult to implement and very poor results are known about their practical behavior. The actual attemps to implement them have shown very limited in the size of the specifications that can be handled:

automata with more than around ten states are intractable [Tab06,GKSV03]. Such sizes are clearly not sufficient in pratcice. As a consequence, tools like SPIN [RH04] that implement the automata-theoretic approach to model-checking ask either that the complement of the specification is explicitly given or they limit the specification to properties that are expressible in LTL.

In this paper, we propose a new approach to check $\mathcal{L}(A) \subseteq \mathcal{L}(B)$ that can handle much larger Büchi automata. In a recent paper, we have shown that the classical subset construction can be avoided and kept implicit for checking language inclusion and language universality for NFA and their alternating extensions [DDHR06]. Here, we adapt and extend that technique to the more intricate automata on infinite words.

To present the intuition behind our new techniques, let us consider a simpler setting of the problem. Assume that we are given a NBW B and we want to check if $\Sigma^\omega \subseteq \mathcal{L}(B)$, that is to check if $\mathcal{L}(B)$ is universal. First, remember that $\mathcal{L}(B)$ is universal when $\mathcal{L}^c(B)$ is empty. The classical algorithm first complements B and then checks for emptiness. The language of a NBW is nonempty if there exists an infinite run of the automaton that visits accepting locations infinitely often. The existence of such a run can be established in polynomial time by computing the following fixed point $\mathcal{F} \equiv \nu y \cdot \mu x \cdot (\mathsf{Pre}(x) \cup (\mathsf{Pre}(y) \cap \alpha))$ where Pre is the predecessor operator of the automaton (given a set L of locations it returns the set of locations that can reach L in one step) and α is the set of accepting locations of the automaton. The automaton is non-empty if and only if its initial location is a member of the fixed point $\mathcal{F}$. This well-known algorithm is quadratic in the size of the automaton. Unfortunately, the automaton that accepts the language $\mathcal{L}^c(B)$ is usally huge and the evaluation of the fixed point is unfeasable for all but the smallest specifications B. To overcome this difficulty, we make the following observation: if $\preceq$ is a *simulation* pre-order on the locations of B^c ($\ell_1 \preceq \ell_2$ means ℓ_1 can simulate ℓ_2) which is compatible with the accepting condition (if $\ell_1 \preceq \ell_2$ and $\ell_2 \in \alpha$ then $\ell_1 \in \alpha$), then the sets that are computed during the evaluation of $\mathcal{F}$ are all $\preceq$*-closed* (if an element ℓ is in the set then all $\ell' \preceq \ell$ are also in the set). Then $\preceq$-closed sets can be represented by their $\preceq$-maximal elements and if operations on such sets can be computed directly on their representation, we have the ingredients to evaluate the fixed point in a more efficient way. For an automaton $\mathcal{B}$ over finite words, set inclusion would be a typical example of a simulation relation for $\mathcal{B}^c$ [DDHR06].

We show that the classical constructions for Büchi automata that are used in the automata-theoretic approach to model-checking are all equipped with a simulation pre-order that exists by construction and does not need to be computed. On that basis we propose new algorithms to check universality of NBW, language inclusion for NBW, and emptiness of alternating Büchi automata (ABW).

We evaluate an implementation of our new algorithm for the universality problem of NBW and on a randomized model recently proposed by Tabakov and Vardi. We show that the performance of the new algorithm on this randomized model outperforms by several order of magnitude the existing implementations of the Kupferman-Vardi algorithm [Tab06,GKSV03]. When the classical solution is limited to automata of size 8 for some parameter values of the randomized model, we are able to handle automata with more than one hundred locations for the same parameter values. We have identified the hardest instances of the randomized model for our algorithms and show that we can still handle problems with several dozens of locations for those instances.

Structure of the paper In Section 2, we recall the Vardi-Kupferman and Miyano-Hayashi constructions that are used for complementation of NBW. In Section 3, we recall the notion of simulation pre-order for a Büchi automaton and prove that the fixed point needed to establish emptiness of nondeterministic Büchi automata handles only closed sets for such pre-orders. We use this observation in Section 4 to define a new algorithm to decide emptiness of ABW. In Section 5, we adapt the technique for the universality problem of NBW. In Section 6, we report on the performances of the new algorithm for universality. In Section 7, we extend those ideas to obtain a new algorithm for language inclusion of NBW. Section 8 concludes the paper.

2 Büchi Automata and Classical Algorithms

An *alternating Büchi automaton* (ABW) is a tuple $\mathcal{A} = \langle \mathsf{Loc}, \iota, \Sigma, \delta, \alpha \rangle$ where:

- Loc is a finite set of states (or locations). The *size* of $\mathcal{A}$ is $|\mathcal{A}| = |\mathsf{Loc}|$;
- $\iota \in \mathsf{Loc}$ is the *initial* state;
- Σ is a finite *alphabet*;
- $\delta : \mathsf{Loc} \times \Sigma \to \mathcal{B}^+(\mathsf{Loc})$ is the *transition function* where $\mathcal{B}^+(\mathsf{Loc})$ is the set of positive boolean formulas over Loc, *i.e.* formulas built from elements in $\mathsf{Loc} \cup \{\mathsf{true}, \mathsf{false}\}$ using the boolean connectives $\wedge$ and $\vee$;
- $\alpha \subseteq \mathsf{Loc}$ is the acceptance condition.

We say that a set $X \subseteq \mathsf{Loc}$ *satisfies* a formula $\varphi \in \mathcal{B}^+(\mathsf{Loc})$ (noted $X \models \varphi$) iff the truth assignment that assigns true to the members of X and assigns false to the members of $\mathsf{Loc} \backslash X$ satisfies φ.

A *run* of $\mathcal{A}$ on an infinite word $w = \sigma_0 \cdot \sigma_1 \dots$ is a DAG $T_w = \langle V, v_\iota, \to \rangle$ where:

- $V = \mathsf{Loc} \times \mathbb{N}$ is the set of nodes. A node (ℓ, i) represents the state ℓ after the first i letters of the word w have been read by $\mathcal{A}$. Nodes of the form (ℓ, i) with $\ell \in \alpha$ are called *α-nodes*;
- $v_\iota = (\iota, 0)$ is the root of the DAG;
- and $\to \subseteq V \times V$ is such that (i) if $(\ell, i) \to (\ell', i')$ then $i' = i + 1$ and (ii) for every $(\ell, i) \in V$, the set $\{\ell' \mid (\ell, i) \to (\ell', i + 1)\}$ satisfies the formula $\delta(\ell, \sigma_i)$.
  We say that $(\ell', i + 1)$ is a *successor* of (ℓ, i) if $(\ell, i) \to (\ell', i + 1)$, and we say that (ℓ', i') is *reachable* from (ℓ, i) if $(\ell, i) \to^* (\ell', i')$.

A run $T_w = \langle V, v_\iota, \to \rangle$ of $\mathcal{A}$ on an infinite word w is *accepting* iff all its infinite paths π rooted at v_ι (thus $\pi \in \mathsf{Loc}^\omega$) visit α-nodes infinitely often. An infinite word $w \in \Sigma^\omega$ is *accepted* by $\mathcal{A}$ iff there exists an accepting run on it. We denote by $\mathcal{L}(\mathcal{A})$ the set of infinite words accepted by $\mathcal{A}$, and by $\mathcal{L}^c(\mathcal{A})$ the set of infinite words that are not accepted by $\mathcal{A}$.

A *nondeterministic Büchi automaton* (NBW) is an ABW whose transition function is restricted to disjunctions over Loc. Runs of NBW reduce to (linear) traces. The transition function of NBW is often seen as a function $[Q \times \Sigma \to 2^Q]$ and we write $\delta(\ell, \sigma) = \{\ell_1, \dots, \ell_n\}$ instead of $\delta(\ell, \sigma) = \ell_1 \vee \ell_2 \vee \dots \vee \ell_n$. We note by $\mathsf{Pre}^{\mathcal{A}}_\sigma(L)$ the set of predecessors by σ of the set L: $\mathsf{Pre}^{\mathcal{A}}_\sigma(L) = \{\ell \in \mathsf{Loc} \mid \exists \ell' \in L : \ell' \in \delta(\ell, \sigma)\}$. Let $\mathsf{Pre}^{\mathcal{A}}(L) = \{\ell \in \mathsf{Loc} \mid \exists \sigma \in \Sigma : \ell \in \mathsf{Pre}^{\mathcal{A}}_\sigma(L)\}$.

Problems The *emptiness problem* for NBW is to decide, given an NBW $\mathcal{A}$, whether $\mathcal{L}(\mathcal{A}) = \emptyset$. This problem is solvable in polynomial time. The symbolic approach through fixed point computation is quadratic in the size of $\mathcal{A}$.

The *universality problem* for NBW is to decide, given an NBW $\mathcal{A}$ over the alphabet Σ whether $\mathcal{L}(\mathcal{A}) = \Sigma^\omega$ where Σ^ω is the set of all infinite words on Σ. This problem is PSPACE-complete [SVW87]. The classical algorithm to decide universality is to first complement the NBW and then to check emptiness of the complement. The difficult step is the complementation as it may cause an exponential blow-up in the size of the automaton. There exists two types of construction, one is based on a determinization of the automaton [Saf88] and the other uses ABW as an intermediate step [KV97]. We review the second construction below.

The *language inclusion problem* for NBW is to decide, given two NBW $\mathcal{A}$ and $\mathcal{B}$, whether $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$. This problem is central in model-checking and it is PSPACE-complete. The classical solution consists in checking the emptiness of $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}^c(\mathcal{B})$, which again requires the expensive complementation of $\mathcal{B}$.

The *emptiness problem* for ABW is to decide, given an ABW $\mathcal{A}$, whether $\mathcal{L}(\mathcal{A}) = \emptyset$. This problem is also PSPACE-complete and it can be solved using a translation from ABW to NBW that preserves the language of the automaton [MH84]. Again, this construction involves an exponential blow-up that makes straight implementations feasible only for automata limited to around ten states. However, the emptiness problem for ABW is very important in practice for LTL model-checking as there exist efficient polynomial translations from LTL formulas to ABW [GO01]. The classical construction is presented below.

Kupferman-Vardi construction Complementation of ABW is straightforward by dualizing the transition function (by swapping $\wedge$ and $\vee$, and swapping true and false in each formulas) and interpreting the accepting condition α as a co-Büchi condition, *i.e.* a run T_w is accepted if all its infinite paths have a suffix that contains no α-nodes.

The result is an alternating co-Büchi automaton (ACW). The accepting runs of ACW have a layered structure that has been studied in [KV97], where the notion of *ranks* is defined. The rank is a positive number associated to each node of a run T_w of an ACW on a word w. Let $G_0 = T_w$. Nodes of rank 0 are those nodes from which only finitely many nodes are reachable in G_0. Let G_1 be the run T_w from which all nodes of rank 0 have been removed. Then, nodes of rank 1 are those nodes of G_1 from which no α-node is reachable in G_1. For $i \geq 1$, let G_i be the the run T_w from which all nodes of rank $0, \dots, i-1$ have been removed. Then, nodes of rank $2i$ are those nodes of G_{2i} from which only finitely many nodes are reachable in G_{2i}, and nodes of rank $2i+1$ are those nodes of G_{2i+1} from which no α-node is reachable in G_{2i+1}. Intuitively, the rank of a node (ℓ, i) hints how difficult it is to prove that all the paths of T_w that start in (ℓ, i) visit α-nodes only finitely many times. It can be shown that every node has a rank between 0 and $2(|\mathsf{Loc}| - |\alpha|)$, and all α-nodes have an even rank [GKSV03].

The layered structure of the runs of ACW induces a construction to complement ABW [KV97]. We present this construction directly for NBW. Given a NBW $\mathcal{A} = \langle \mathsf{Loc}, \iota, \Sigma, \delta, \alpha \rangle$ and an even number $k \in \mathbb{N}$, let $\mathsf{KV}(\mathcal{A}, k) = \langle \mathsf{Loc}', \iota', \Sigma, \delta', \alpha' \rangle$ be an ABW such that:

- $\mathsf{Loc}' = \mathsf{Loc} \times [k]$ where $[k] = \{0, 1, \ldots, k\}$. Intuitively, the automaton $\mathsf{KV}(\mathcal{A}, k)$ is in state (ℓ, n) after the first i letters of the input word w have been read if it guesses that the rank of the node (ℓ, i) in a run of $\mathcal{A}$ on w is at most n;
- $\iota' = (\iota, k)$;
- $\delta'((\ell, i), \sigma) = \mathsf{false}$ if $\ell \in \alpha$ and i is odd, and otherwise $\delta'((\ell, i), \sigma) = \bigvee_{i' \leq i}(\ell_1, i') \wedge \bigvee_{i' \leq i}(\ell_2, i') \wedge \cdots \wedge \bigvee_{i' \leq i}(\ell_n, i')$ if $\delta(\ell, \sigma) = \ell_1 \vee \ell_2 \vee \cdots \vee \ell_n$; For example, if $\delta(\ell, \sigma) = \ell_1 \vee \ell_2$ then $\delta'((\ell, 2), \sigma) = ((\ell_1, 2) \vee (\ell_1, 1) \vee (\ell_1, 0)) \wedge ((\ell_2, 2) \vee (\ell_2, 1) \vee (\ell_2, 0))$.
- $\alpha' = \mathsf{Loc} \times [k]^{odd}$ where $[k]^{odd}$ is the set of odd numbers in $[k]$.

The ABW that the Kupferman-Vardi construction specifies accepts the complement language and its size is quadratic in the size of the original automaton.

Theorem 1 ([KV97]) *For all NBW $\mathcal{A} = \langle \mathsf{Loc}, \iota, \Sigma, \delta, \alpha \rangle$, for all $0 \leq k' \leq k$, we have $\mathcal{L}(\mathsf{KV}(\mathcal{A}, k')) \subseteq \mathcal{L}(\mathsf{KV}(\mathcal{A}, k))$ and for $k = 2(|\mathsf{Loc}| - |\alpha|)$, we have $\mathcal{L}(\mathsf{KV}(\mathcal{A}, k)) = \mathcal{L}^c(\mathcal{A})$.*

Miyano-Hayashi construction Classically, to check emptiness of ABW, a variant of the subset construction is applied that transforms the ABW into a NBW that accepts the same language [MH84]. Intuitively, the NBW maintains a set s of states of the ABW that corresponds to a whole level of a guessed run DAG of the ABW. In addition, the NBW maintains a set o of states that "owe" a visit to an accepting state. Whenever the set o gets empty, meaning that every path of the guessed run has visited at least one accepting state, the set o is initiated with the current level of the guessed run. It is asked that o gets empty infinitely often in order to ensure that every path of the run DAG visits accepting states infinitely often. The construction is as follows.

Given an ABW $\mathcal{A} = \langle \mathsf{Loc}, \iota, \Sigma, \delta, \alpha \rangle$, let $\mathsf{MH}(\mathcal{A}) = \langle 2^{\mathsf{Loc}} \times 2^{\mathsf{Loc}}, (\{\iota\}, \emptyset), \Sigma, \delta', \alpha' \rangle$ be a NBW where $\alpha' = 2^{\mathsf{Loc}} \times \{\emptyset\}$ and δ' is defined, for all $\langle s, o \rangle \in 2^{\mathsf{Loc}} \times 2^{\mathsf{Loc}}$ and $\sigma \in \Sigma$, as follows:

- If $o \neq \emptyset$, then $\delta'(\langle s, o \rangle, \sigma) = \{\langle s', o' \setminus \alpha \rangle \mid o' \subseteq s', s' \models \bigwedge_{\ell \in s} \delta(\ell, \sigma)$ and $o' \models \bigwedge_{\ell \in o} \delta(\ell, \sigma)\}$;
- If $o = \emptyset$, then $\delta'(\langle s, o \rangle, \sigma) = \{\langle s', s' \setminus \alpha \rangle \mid s' \models \bigwedge_{\ell \in s} \delta(\ell, \sigma)\}$.

The size of the Miyano-Hayashi construction is exponential in the size of the original automaton.

Theorem 2 ([MH84]) *For all ABW $\mathcal{A}$, we have $\mathcal{L}(\mathsf{MH}(\mathcal{A})) = \mathcal{L}(\mathcal{A})$.*

The size of the automaton obtained after the Kupferman-Vardi and the Miyano-Hayashi construction is an obstacle to the straight implementation of the method. In Section 3, we propose a new approach that circumvents this problem.

Direct complementation In our solution, we implicitly use the two constructions to complement Büchi automata but, as we will see, we do not construct the automata. For the sake of clarity, we give below the specification of the automaton that would result from the composition of the two constructions. In the definition of the state space, we omit the states (ℓ, i) for $\ell \in \alpha$ and i odd, as those states have no successor in the Kupferman-Vardi construction.

Definition 3 Given a NBW $\mathcal{A} = \langle \mathsf{Loc}, \iota, \Sigma, \delta, \alpha \rangle$ and an even number $k \in \mathbb{N}$, let $\mathsf{KVMH}(\mathcal{A}, k) = \langle Q_k \times Q_k, q_\iota, \Sigma, \delta', \alpha' \rangle$ be a NBW such that:

- $Q_k = 2^{(\mathsf{Loc} \times [k]) \setminus (\alpha \times \mathbb{N}^{odd})}$ where $\mathbb{N}^{odd}$ is the set of odd natural numbers;
- $q_\iota = (\{(\iota, k)\}, \emptyset)$;
- Let $\mathsf{odd} = \mathsf{Loc} \times [k]^{odd}$; δ' is defined for all $s, o \in Q_k$ and $\sigma \in \Sigma$, as follows:
  - If $o \neq \emptyset$, then $\delta'(\langle s, o \rangle, \sigma)$ is the set of pairs $\langle s', o' \setminus \mathsf{odd} \rangle$ such that:
    (i) $o' \subseteq s'$;
    (ii) $\forall (\ell, n) \in s \cdot \forall \ell' \in \delta(\ell, \sigma) \cdot \exists (\ell', n') \in s' : n' \leq n$;
    (iii) $\forall (\ell, n) \in o \cdot \forall \ell' \in \delta(\ell, \sigma) \cdot \exists (\ell', n') \in o' : n' \leq n$.
  - If $o = \emptyset$, then $\delta'(\langle s, o \rangle, \sigma)$ is the set of pairs $\langle s', s' \setminus \mathsf{odd} \rangle$ such that: $\forall (\ell, n) \in s \cdot \forall \ell' \in \delta(\ell, \sigma) \cdot \exists (\ell', n') \in s' : n' \leq n$.
- $\alpha' = 2^{\mathsf{Loc} \times [k]} \times \{\emptyset\}$;

We write $\langle s, o \rangle \xrightarrow{\sigma}_{\delta'} \langle s', o' \rangle$ to denote $\langle s', o' \rangle \in \delta'(\langle s, o \rangle, \sigma)$.

Theorem 4 ([KV97,MH84]) *For all NBW $\mathcal{A} = \langle \mathsf{Loc}, \iota, \Sigma, \delta, \alpha \rangle$, for all $0 \leq k' \leq k$, we have $\mathcal{L}(\mathsf{KVMH}(\mathcal{A}, k')) \subseteq \mathcal{L}(\mathsf{KVMH}(\mathcal{A}, k))$ and for $k = 2(|\mathsf{Loc}| - |\alpha|)$, we have $\mathcal{L}(\mathsf{KVMH}(\mathcal{A}, k)) = \mathcal{L}^c(\mathcal{A})$.*

3 Simulation Pre-Orders and Fixed Points

Let $\mathcal{A} = \langle \mathsf{Loc}, \iota, \Sigma, \delta, \alpha \rangle$ be a NBW. Let $\langle 2^{\mathsf{Loc}}, \subseteq, \cup, \cap, \emptyset, \mathsf{Loc} \rangle$ be the powerset lattice of locations. The fixed point $\mathcal{F}_\mathcal{A} \equiv \nu y \cdot \mu x \cdot (\mathsf{Pre}^\mathcal{A}(x) \cup (\mathsf{Pre}^\mathcal{A}(y) \cap \alpha))$ can be used to check emptiness of $\mathcal{A}$ as we have $\mathcal{L}(\mathcal{A}) \neq \emptyset$ iff $\iota \in \mathcal{F}_\mathcal{A}$.
Let $\preceq \subseteq \mathsf{Loc} \times \mathsf{Loc}$ be a pre-order and let $\ell_1 \prec \ell_2$ iff $\ell_1 \preceq \ell_2$ and $\ell_2 \not\preceq \ell_1$.

Definition 5 A pre-order $\preceq$ is a *simulation*[3] for $\mathcal{A}$ iff the following properties hold:

- for all $\ell_1, \ell_2, \ell_3 \in \mathsf{Loc}$, for all $\sigma \in \Sigma$, if $\ell_3 \preceq \ell_1$ and $\ell_2 \in \delta(\ell_1, \sigma)$ then there exists $\ell_4 \in \mathsf{Loc}$ such that $\ell_4 \preceq \ell_2$ and $\ell_4 \in \delta(\ell_3, \sigma)$;
- for all $\ell \in \alpha$, for all $\ell' \in \mathsf{Loc}$, if $\ell' \preceq \ell$ then $\ell' \in \alpha$.

A set $L \subseteq \mathsf{Loc}$ is $\preceq$*-closed* iff for all $\ell_1, \ell_2 \in \mathsf{Loc}$, if $\ell_1 \preceq \ell_2$ and $\ell_2 \in L$ then $\ell_1 \in L$. The $\preceq$*-closure* of L, is the set $\downarrow L = \{\ell \in \mathsf{Loc} \mid \exists \ell' \in L : \ell \preceq \ell'\}$. We denote by $\mathsf{Max}(L)$ the set of $\preceq$-maximal elements of L: $\mathsf{Max}(L) = \{\ell \in L \mid \nexists \ell' \in L : \ell \prec \ell'\}$. When the context is ambiguous, we sometimes write $\downarrow_\preceq$ and $\mathsf{Max}_\preceq$ with the intended pre-order in subscript. For any $\preceq$-closed set $L \subseteq \mathsf{Loc}$, we have $L = \downarrow \mathsf{Max}(L)$. Furthermore, if $\preceq$ is a partial order, then $\mathsf{Max}(L)$ is an antichain of elements and it is a canonical representation of L. The following lemma states interesting properties of $\preceq$-closed sets of locations.

Lemma 6 *For all NBW $\mathcal{A} = \langle \mathsf{Loc}, \iota, \Sigma, \delta, \alpha \rangle$, for all simulations $\preceq$ for $\mathcal{A}$, the following properties hold:*

1. for all $\preceq$-closed set $L \subseteq \mathsf{Loc}$, for all $\sigma \in \Sigma$, $\mathsf{Pre}^\mathcal{A}_\sigma(L)$ is $\preceq$-closed;

[3] Several notions of simulation pre-orders have been defined for Büchi automata, see [EWS05] for a survey.

2. *for all $\preceq$-closed sets $L_1, L_2 \subseteq \mathsf{Loc}$, $L_1 \cup L_2$ and $L_1 \cap L_2$ are $\preceq$-closed;*
3. *the set α is $\preceq$-closed.*

We can take advantage of Lemma 6 to compute the fixed point $\mathcal{F}_{\mathcal{A}}$ more efficiently in terms of space consumption and execution time. First, we represent $\preceq$-closed sets by their maximal elements. This way, the size of the sets is usually drastically reduced. As we will see later, this can potentially save an exponential factor. Second, the union of $\preceq$-closed sets can be computed efficiently using this representation as we have $\mathsf{Max}(L_1 \cup L_2) = \mathsf{Max}(\mathsf{Max}(L_1) \cup \mathsf{Max}(L_2))$. Third, we will see that the NBW that we have to analyze in the automata-based approach to model-checking are all equipped with a simulation pre-order that can be exploited to compute efficiently the intersection and the predecessors of $\preceq$-closed sets of locations.

Intuitively, when computing the sequence of approximations for $\mathcal{F}_{\mathcal{A}}$, we can concentrate on maximal elements for a simulation pre-order as those locations are such that if they have an accepting run in $\mathcal{A}$, then all the locations that are smaller for the pre-order also have an accepting run in $\mathcal{A}$.

4 Emptiness of ABW

We now show how to apply Lemma 6 to check more efficiently the emptiness of ABW. Let $\mathcal{A}_1 = \langle \mathsf{Loc}_1, \iota_1, \Sigma, \delta_1, \alpha_1 \rangle$ be an ABW for which we want to decide whether $\mathcal{L}(\mathcal{A}_1) = \emptyset$. We know that the (exponential) Miyano-Hayashi construction gives a NBW $\mathcal{A}_2 = \mathsf{MH}(\mathcal{A}_1)$ such that $\mathcal{L}(\mathcal{A}_2) = \mathcal{L}(\mathcal{A}_1)$. We show that the emptiness of $\mathcal{A}_1$ (or equivalently of $\mathcal{A}_2$) can be decided more efficiently by computing the fixed point $\mathcal{F}_{\mathcal{A}_2}$ and without constructing explicitly $\mathcal{A}_2$. To do so, we show that there exists a simulation for $\mathcal{A}_2$ for which we can compute $\cup$, $\cap$ and Pre by manipulating only maximal elements of closed sets of locations.

Let $\mathsf{MH}(\mathcal{A}_1) = \langle \mathsf{Loc}_2, \iota_2, \Sigma, \delta_2, \alpha_2 \rangle$. Remember that $\mathsf{Loc}_2 = 2^{\mathsf{Loc}_1} \times 2^{\mathsf{Loc}_1}$. Define the pre-order $\preceq_{\mathsf{alt}} \subseteq \mathsf{Loc}_2 \times \mathsf{Loc}_2$ such that for all $\langle s, o \rangle, \langle s', o' \rangle \in \mathsf{Loc}_2$, we have $\langle s, o \rangle \preceq_{\mathsf{alt}} \langle s', o' \rangle$ iff (i) $s \subseteq s'$, (ii) $o \subseteq o'$, and (iii) $o = \emptyset$ iff $o' = \emptyset$. Note that this pre-order is a partial order. As a consequence, given a set of pairs $L = \{\langle s_1, o_1 \rangle, \langle s_2, o_2 \rangle, \ldots, \langle s_n, o_n \rangle\}$, the set $\mathsf{Max}(L)$ is an antichain and identifies L.

Lemma 7 *For all ABW $\mathcal{A}_1$, the partial order $\preceq_{\mathsf{alt}}$ is a simulation for* $\mathsf{MH}(\mathcal{A}_1)$.

Proof. Let $\mathcal{A}_1 = \langle \mathsf{Loc}_1, \iota_1, \Sigma, \delta_1, \alpha_1 \rangle$ and $\mathsf{MH}(\mathcal{A}_1) = \langle \mathsf{Loc}_2, \iota_2, \Sigma, \delta_2, \alpha_2 \rangle$. First, let $\sigma \in \Sigma$ and $\langle s_1, o_1 \rangle$, $\langle s_2, o_2 \rangle$, $\langle s_3, o_3 \rangle \in \mathsf{Loc}_2$ be such that $\langle s_3, o_3 \rangle \preceq_{\mathsf{alt}} \langle s_1, o_1 \rangle$ and $\langle s_1, o_1 \rangle \xrightarrow{\sigma}_{\delta_2} \langle s_2, o_2 \rangle$. We show that there exists $\langle s_4, o_4 \rangle \in \mathsf{Loc}_2$ such that $\langle s_3, o_3 \rangle \xrightarrow{\sigma}_{\delta_2} \langle s_4, o_4 \rangle$ and $\langle s_4, o_4 \rangle \preceq_{\mathsf{alt}} \langle s_3, o_3 \rangle$. First, let us consider the case where $o_1 = \emptyset$. In this case, we have $o_3 = \emptyset$ by definition of $\preceq_{\mathsf{alt}}$ and $\delta_2(\langle s_1, o_1 \rangle, \sigma) = \{\langle s', s' \setminus \alpha_1 \rangle \mid s' \models \bigwedge_{l \in s_1} \delta_1(l, \sigma)\}$, this set being contained in $\delta_2(\langle s_3, o_3 \rangle, \sigma) = \{\langle s', s' \setminus \alpha_1 \rangle \mid s' \models \bigwedge_{l \in s_3} \delta_1(l, \sigma)\}$ as s_3 puts less constraints than s_1 since $s_3 \subseteq s_1$. A similar reasoning holds if $o_1 \neq \emptyset$. Second, let $\langle s_1, o_1 \rangle \in \alpha_2$ and let $\langle s_2, o_2 \rangle \preceq_{\mathsf{alt}} \langle s_1, o_1 \rangle$. By definition of α_2, we know that $o_2 = \emptyset$, and by definition of $\preceq_{\mathsf{alt}}$ we have $o_2 = \emptyset$ and so $\langle s_2, o_2 \rangle \in \alpha_2$. ■

Algorithm 1: Algorithm for $\mathsf{Pre}_{\mathsf{alt}}(\cdot)$.

Data : An ABW $\mathcal{A}_1 = \langle \mathsf{Loc}_1, \iota_1, \Sigma, \delta_1, \alpha_1 \rangle$, $\sigma \in \Sigma$ and $\langle s', o' \rangle \in 2^{\mathsf{Loc}_1} \times 2^{\mathsf{Loc}_1}$ such that $o' \subseteq s'$.

Result : The $\preceq_{\mathsf{alt}}$-antichain $\mathsf{Pre}^{\mathsf{alt}}_{\sigma}(\langle s', o' \rangle)$.

begin

1 $L_{\mathsf{Pre}} \leftarrow \emptyset$;

2 $o \leftarrow \{\ell \in \mathsf{Loc}_1 \mid o' \cup (s' \cap \alpha_1) \models \delta_1(\ell, \sigma)\}$;

3 **if** $o' \not\subseteq \alpha_1 \vee o' = \emptyset$ **then**

4 $\quad L_{\mathsf{Pre}} \leftarrow \{\langle o, \emptyset \rangle\}$;

5 **if** $o \neq \emptyset$ **then**

6 $\quad s \leftarrow \{\ell \in \mathsf{Loc}_1 \mid s' \models \delta_1(\ell, \sigma)\}$;

7 $\quad L_{\mathsf{Pre}} \leftarrow L_{\mathsf{Pre}} \cup \{\langle s, o \rangle\}$;

8 **return** L_{Pre};

end

So, we know according to Lemma 6 that all the sets that we compute to evaluate $\mathcal{F}_{\mathcal{A}_2}$ are $\preceq_{\mathsf{alt}}$-closed. So, we explain now how to compute intersections and pre-operations by manipulating maximal elements only. Given $\langle s_1, o_1 \rangle, \langle s_2, o_2 \rangle$, we can compute $\langle s, o \rangle$ such that $\downarrow \langle s, o \rangle = \downarrow \langle s_1, o_1 \rangle \cap \downarrow \langle s_2, o_2 \rangle$ as follows. If $o_1 \cap o_2 \neq \emptyset$ then $\langle s, o \rangle = \langle s_1 \cap s_2, o_1 \cap o_2 \rangle$, and if $o_1 = o_2 = \emptyset$ then $\langle s, o \rangle = \langle s_1 \cap s_2, \emptyset \rangle$; otherwise the intersection is empty. Algorithm 1 computes the predecessors of a $\preceq_{\mathsf{alt}}$-closed set by just manipulating its maximal elements. It runs in time $O(|\mathsf{Loc}_1| \cdot \|\delta_1\|)$ where $\|\delta_1\|$ is the size of the transition relation, defined as the maximal number of boolean connectives in a formula $\delta_1(\ell, \sigma)$.

Theorem 8 *Given an ABW $\mathcal{A}_1 = \langle \mathsf{Loc}_1, \iota_1, \Sigma, \delta_1, \alpha_1 \rangle$, $\sigma \in \Sigma$ and $\langle s', o' \rangle \in 2^{\mathsf{Loc}_1} \times 2^{\mathsf{Loc}_1}$ such that $o' \subseteq s'$, the set $L_{\mathsf{Pre}} = \mathsf{Pre}^{\mathsf{alt}}_{\sigma}(\langle s, o \rangle)$ computed by Algorithm 1 is an $\preceq_{\mathsf{alt}}$-antichain such that $\downarrow L_{\mathsf{Pre}} = \mathsf{Pre}^{\mathcal{A}_2}_{\sigma}(\downarrow \{\langle s', o' \rangle\})$ where $\mathcal{A}_2 = \mathsf{MH}(\mathcal{A}_1)$.*

Proof. Let $\mathcal{A}_2 = \mathsf{MH}(\mathcal{A}_1) = \langle \mathsf{Loc}_2, \iota_2, \Sigma, \delta_2, \alpha_2 \rangle$. We show that (1) $L_{\mathsf{Pre}} \subseteq \mathsf{Pre}^{\mathcal{A}_2}_{\sigma}(\downarrow \{\langle s', o' \rangle\})$ and (2) for all $\langle s_1, o_1 \rangle \in \mathsf{Pre}^{\mathcal{A}_2}_{\sigma}(\downarrow \{\langle s', o' \rangle\})$, there exists $\langle s, o \rangle \in L_{\mathsf{Pre}}$ such that $\langle s_1, o_1 \rangle \preceq_{\mathsf{alt}} \langle s, o \rangle$. This entails that $\downarrow L_{\mathsf{Pre}} = \mathsf{Pre}^{\mathcal{A}_2}_{\sigma}(\downarrow \{\langle s', o' \rangle\})$.

To prove (1), we first show that $\langle s, o \rangle \xrightarrow{\sigma}_{\delta_2} \langle s', o' \rangle$ where $\langle s, o \rangle$ is added to L_{Pre} at line 7 of Algorithm 1. By the test of line 5, we have $o \neq \emptyset$. According to the definition of $\mathsf{MH}(\cdot)$ (see Section 2), we have to check that there exists a set $o'' \subseteq s'$ such that $o' = o'' \setminus \alpha_1$ (we take $o'' = o' \cup (s' \cap \alpha_1)$), and the following conditions hold:

(i) $s' \models \bigwedge_{\ell \in s} \delta_1(\ell, \sigma)$ since we have $s' \models \delta_1(\ell, \sigma)$ for all $\ell \in s$ by line 6 of Alg. 1.

(ii) $o'' \models \bigwedge_{\ell \in o} \delta_1(\ell, \sigma)$ since we have $o'' \models \delta_1(\ell, \sigma)$ for all $\ell \in o$ by line 2 of Alg. 1.

Second, we show that $\langle o, \emptyset \rangle \xrightarrow{\sigma}_{\delta_2} \langle s'', o'' \rangle$ for some $\langle s'', o'' \rangle \preceq_{\mathsf{alt}} \langle s', o' \rangle$ where $\langle o, \emptyset \rangle$ is added to L_{Pre} at line 4 of Algorithm 1.

We take $s'' = o' \cup (s' \cap \alpha_1)$ and $o'' = s'' \setminus \alpha_1$. Since $o' \subseteq s'$, we have (a) $s'' \subseteq s'$, and we have (b) $o'' = o' \setminus \alpha_1 \subseteq o'$. Let us establish that (c) $o' = \emptyset$ iff $o'' = \emptyset$. If $o' = \emptyset$ then $o'' = \emptyset$ since $o'' \subseteq o'$. If $o' \neq \emptyset$ then by the test of line 3, we have

$o' \not\subseteq \alpha_1$ and thus $o'' \neq \emptyset$. Hence we have $\langle s'', o'' \rangle \preceq_{\mathsf{alt}} \langle s', o' \rangle$, and by line 2 of the algorithm, we have $s'' \models \delta_1(\ell, \sigma)$ for all $\ell \in o$, and thus $s'' \models \bigwedge_{\ell \in o} \delta_1(\ell, \sigma)$. Therefore $\langle o, \emptyset \rangle \xrightarrow{\sigma}_{\delta_2} \langle s'', o'' \rangle$.

To prove (2), assume that there exist $\langle s_1, o_1 \rangle$ and $\langle s_1', o_1' \rangle$ such that $\langle s_1, o_1 \rangle \xrightarrow{\sigma}_{\delta_2} \langle s_1', o_1' \rangle$ and $\langle s_1', o_1' \rangle \preceq_{\mathsf{alt}} \langle s', o' \rangle$. We have to show that there exists $\langle s, o \rangle \in L_{\mathsf{Pre}}$ such that $\langle s_1, o_1 \rangle \preceq_{\mathsf{alt}} \langle s, o \rangle$.

First, assume that $o_1 \neq \emptyset$. Since $\langle s_1, o_1 \rangle \xrightarrow{\sigma}_{\delta_2} \langle s_1', o_1' \rangle$, we have:

(i) for all $\ell \in s_1$, $s_1' \models \delta_1(\ell, \sigma)$ and since $s_1' \subseteq s'$ also $s' \models \delta_1(\ell, \sigma)$. Let s be the set defined at line 6 of Algorithm 1. For all $\ell \in \mathsf{Loc}$, if $s' \models \delta_1(\ell, \sigma)$ then $\ell \in s$. Hence, $s_1 \subseteq s$.

(ii) for all $\ell \in o_1$, $o_1'' \models \delta_1(\ell, \sigma)$ for some $o_1'' \subseteq s_1'$ such that $o_1' = o_1'' \setminus \alpha_1$. Hence necessarily $o_1'' \subseteq o_1' \cup (s_1' \cap \alpha_1) \subseteq o' \cup (s' \cap \alpha_1)$ and thus for all $\ell \in o_1$, $o' \cup (s' \cap \alpha_1) \models \delta_1(\ell, \sigma)$. Let o be the set defined at line 2 of Algorithm 1. For all $\ell \in \mathsf{Loc}$, if $o' \cup (s' \cap \alpha_1) \models \delta_1(\ell, \sigma)$ then $\ell \in o$. Hence, $o_1 \subseteq o$ and $o \neq \emptyset$.

Hemce, $\langle s, o \rangle$ which is added to L_{Pre} by Alg. 1 at line 7 satisfies $\langle s_1, o_1 \rangle \preceq_{\mathsf{alt}} \langle s, o \rangle$.

Second, assume that $o_1 = \emptyset$. Since $\langle s_1, o_1 \rangle \xrightarrow{\sigma}_{\delta'} \langle s_1', o_1' \rangle$ and $o_1 = \emptyset$, we know that for all $\ell \in s_1$, $s_1' \models \delta_1(\ell, \sigma)$ and $o_1' = s_1' \setminus \alpha_1$. Let $s'' = o' \cup (s' \cap \alpha_1)$ so we have (a) $s_1' \cap \alpha_1 \subseteq s' \cap \alpha_1 \subseteq s''$ and (b) $s_1' \setminus \alpha_1 = o_1' \subseteq o' \subseteq s''$. Hence, $s_1' \subseteq s''$ and thus for all $\ell \in s_1$, $s'' \models \delta_1(\ell, \sigma)$. Let o be the set defined at line 2 of Algorithm 1. For all $\ell \in \mathsf{Loc}$, if $s'' \models \delta_1(\ell, \sigma)$ then $\ell \in o$. Hence, $s_1 \subseteq o$ and $\langle s_1, \emptyset \rangle \preceq_{\mathsf{alt}} \langle o, \emptyset \rangle$ where $\langle o, \emptyset \rangle$ is added to L_{Pre} by Algorithm 1 at line 4. Notice that the test at line 3 is satisfied because $o_1' = s_1' \setminus \alpha_1$ implies that $o_1' \not\subseteq \alpha_1 \vee o_1' = \emptyset$ and since $\langle s_1', o_1' \rangle \preceq_{\mathsf{alt}} \langle s', o' \rangle$, we have $o' \not\subseteq \alpha_1 \vee o' = \emptyset$. ∎

5 Universality of NBW

Given the NBW $\mathcal{A} = \langle \mathsf{Loc}, \iota, \Sigma, \delta, \alpha \rangle$, we define the pre-order $\preceq_{\mathsf{univ}} \subseteq (2^{\mathsf{Loc} \times \mathbb{N}} \times 2^{\mathsf{Loc} \times \mathbb{N}}) \times (2^{\mathsf{Loc} \times \mathbb{N}} \times 2^{\mathsf{Loc} \times \mathbb{N}})$ as follows: for $s, s', o, o' \subseteq \mathsf{Loc} \times \mathbb{N}$, let $\langle s, o \rangle \preceq_{\mathsf{univ}} \langle s', o' \rangle$ iff the following conditions hold:

- for all $(\ell, n) \in s$, there exists $(\ell, n') \in s'$ such that $n' \leq n$;
- for all $(\ell, n) \in o$, there exists $(\ell, n') \in o'$ such that $n' \leq n$;
- $o = \emptyset$ iff $o' = \emptyset$.

This relation formalizes the intuition that it is easier to accept a word in $\mathsf{KVMH}(\mathcal{A}, k)$ from a given location with a high rank than with a low rank. This is because the rank is always decreasing along every path of the runs of $\mathsf{KV}(\mathcal{A}, k)$, and so a rank is always simulated by a greater rank. Hence essentially the minimal rank of s and o is relevant to define the pre-order $\preceq_{\mathsf{univ}}$. The third condition requires that accepting states are simulated by accepting states.

The relation $\preceq_{\mathsf{univ}}$ is a simulation for the NBW $\mathsf{KVMH}(\mathcal{A}, k)$ (with state space $Q_k \times Q_k$) defined in Section 2.

Lemma 9 *For all NBW $\mathcal{A}$, for all even numbers $k \in \mathbb{N}$, the restriction of $\preceq_{\mathsf{univ}}$ to $(Q_k \times Q_k) \times (Q_k \times Q_k)$ is a simulation for the NBW $\mathsf{KVMH}(\mathcal{A}, k)$ of Definition 3.*

The proof of Lemma 9 is given in the appendix.

According to Lemma 6, all the intermediate sets that are computed by the fixed point $\mathcal{F}_{\mathcal{A}^c}$ to check emptiness of $\mathcal{A}^c = \mathsf{KVMH}(\mathcal{A}, k)$ for $k = 2(|\mathsf{Loc}| - |\alpha|)$ (and thus universality of $\mathcal{A}$) are $\preceq_{\mathsf{univ}}$-closed.

Before computing $\cup$, $\cap$ and Pre for $\preceq_{\mathsf{univ}}$-closed sets, we make the following useful observation. Given a set $s \in Q_k$, define its *characteristic function* $f_s : \mathsf{Loc} \to \mathbb{N} \cup \{\infty\}$ such that $f_s(\ell) = \inf\{n \mid (\ell, n) \in s\}$ with the usual convention that $\inf \emptyset = \infty$.

Lemma 10 *For all sets $s, s', o, o' \in Q_k$, if $f_s = f_{s'}$ and $f_o = f_{o'}$, then $\langle s, o\rangle \preceq_{\mathsf{univ}} \langle s', o'\rangle$ and $\langle s', o'\rangle \preceq_{\mathsf{univ}} \langle s, o\rangle$.*

Let f, g, f', g' be characteristic functions. We write $f \leq f'$ iff for all $\ell \in \mathsf{Loc}$, $f(\ell) \leq f'(\ell)$ and we write $\langle f, g\rangle \leq \langle f', g'\rangle$ iff $f \leq f'$ and $g \leq g'$. Let $\max(f, f')$ be the function f'' such that $f''(\ell) = \max\{f(\ell), f'(\ell)\}$ for all $\ell \in \mathsf{Loc}$. We write $f_\emptyset$ for the function such that $f_\emptyset(\ell) = \infty$ for all $\ell \in \mathsf{Loc}$. Given an even number $k \in \mathbb{N}$, define the set $[\![f]\!]_k = \{s \in Q_k \mid f_s = f\}$ and the set $[\![\langle f, g\rangle]\!]_k = \{\langle s, o\rangle \mid s \in [\![f]\!]_k \wedge o \in [\![g]\!]_k \wedge o \subseteq s\}$. Observe that $f \leq f'$ iff $[\![f']\!]_k \subseteq [\![f]\!]_k$. We extend the operator $[\![\cdot]\!]_k$ to sets of pairs of characteristic functions as expected. According to Lemma 10, the set $[\![\langle f, g\rangle]\!]_k$ is an equivalence class for the equivalence relation induced by $\preceq_{\mathsf{univ}}$, and a $\preceq_{\mathsf{univ}}$-closed set (as well as its $\preceq_{\mathsf{univ}}$-maximal elements) is a union of equivalence classes, so it can be equivalently seen as a union of pairs of characteristic functions.

Now, we show how to compute efficiently $\cup$, $\cap$ and Pre for $\preceq_{\mathsf{univ}}$-closed sets that are represented by characteristic functions. Let L_1, L_2 be two sets of pairs of characteristic functions, let $L_\cup$ be the set of $\leq$-minimal elements of $L_1 \cup L_2$, and let $L_\cap = \{\langle \max(f_s, f_{s'}), \max(f_o, f_{o'})\rangle \mid \langle f_s, f_o\rangle \in L_1 \wedge \langle f_{s'}, f_{o'}\rangle \in L_2 \wedge \max(f_o, f_{o'}) \neq f_\emptyset\} \cup \{\langle \max(f_s, f_{s'}), f_\emptyset\rangle \mid \langle f_s, f_\emptyset\rangle \in L_1 \wedge \langle f_{s'}, f_\emptyset\rangle \in L_2\}$. Then, we have $[\![L_\cup]\!]_k = \mathsf{Max}(\downarrow[\![L_1]\!]_k \cup \downarrow[\![L_2]\!]_k)$ and $[\![L_\cap]\!]_k = \mathsf{Max}(\downarrow[\![L_1]\!]_k \cap \downarrow[\![L_2]\!]_k)$.

To compute $\mathsf{Pre}_\sigma(\cdot)$ of a single pair of characteristic functions, we propose Algorithm 2 whose correctness is established by Theorem 11. Computing the predecessors of a set of characteristic functions is then straightforward using the algorithm for union of sets of pairs of characteristic functions since

$$\mathsf{Pre}^{\mathsf{KVMH}(\mathcal{A},k)}(L) = \bigcup_{\sigma \in \Sigma} \bigcup_{\ell \in L} \mathsf{Pre}_\sigma^{\mathsf{KVMH}(\mathcal{A},k)}(\ell)$$

.

Theorem 11 *Given a NBW $\mathcal{A} = \langle \mathsf{Loc}, \iota, \Sigma, \delta, \alpha\rangle$, $\sigma \in \Sigma$, an even number k, and a pair of characteristic functions $\langle f_{s'}, f_{o'}\rangle$ such that $f_{s'} \leq f_{o'}$, the set $L_{\mathsf{Pre}} = \mathsf{Pre}_\sigma^{\mathsf{univ}}(\langle f_{s'}, f_{o'}\rangle)$ computed by Algorithm 2 is such that $\downarrow[\![L_{\mathsf{Pre}}]\!]_k = \mathsf{Pre}_\sigma^{\mathsf{KVMH}(\mathcal{A},k)}(\downarrow[\![\langle f_{s'}, f_{o'}\rangle]\!]_k)$ and $\forall \langle f_s, f_o\rangle \in L_{\mathsf{Pre}} : f_s \leq f_o$.*

The proof of Theorem 11 is given in the appendix. In Algorithm 2, we represent ∞ by any number strictly greater than k, and we adapt the definition of $\leq$ as follows: $f \leq f'$ iff for all $\ell \in \mathsf{Loc}$, either $f(\ell) \leq f'(\ell)$ or $f'(\ell) > k$. In the algorithm, we use the notations $\lceil n\rceil^{\mathsf{odd}}$ for the least odd number n' such that $n' \geq n$, and $\lceil n\rceil^{\mathsf{even}}$ for the least even number n' such that $n' \geq n$.

Algorithm 2: Algorithm for $\mathsf{Pre}^{\mathsf{univ}}_{\sigma}(\cdot)$.

Data : A NBW $\mathcal{A} = \langle \mathsf{Loc}, \iota, \Sigma, \delta, \alpha \rangle$, $\sigma \in \Sigma$, an even number k and a pair $\langle f_{s'}, f_{o'} \rangle$ of characteristic functions.

Result : The set $\mathsf{Pre}^{\mathsf{univ}}_{\sigma}(\langle f_{s'}, f_{o'} \rangle)$.

begin

1   **foreach** $\ell \in \mathsf{Loc}$ **do**

2      $f_o(\ell) \leftarrow 0$;

3   **foreach** $\ell' \in \delta(\ell, \sigma)$ **do**

4   **if** $\ell' \in \alpha$ **then** $f_o(\ell) \leftarrow \max\{f_o(\ell), f_{o'}(\ell')\}$;

5   **else** $f_o(\ell) \leftarrow \max\{f_o(\ell), \min\{f_{o'}(\ell'), \lceil f_{s'}(\ell') \rceil^{\mathsf{odd}}\}\}$;

6   **if** $\ell \in \alpha$ **then** $f_o(\ell) \leftarrow \lceil f_o(\ell) \rceil^{\mathsf{even}}$;

7    $L_{\mathsf{Pre}} \leftarrow \{\langle f_o, f_{\emptyset} \rangle\}$;

8   **if** $\exists \ell : f_o(\ell) \leq k$ *(i.e. $o \neq \emptyset$)* **then**

9   **foreach** $\ell \in \mathsf{Loc}$ **do**

10        $f_s(\ell) \leftarrow \max\{f_{s'}(\ell') \mid \ell' \in \delta(\ell, \sigma)\}$;

11   **if** $\ell \in \alpha$ **then** $f_s(\ell) \leftarrow \lceil f_s(\ell) \rceil^{\mathsf{even}}$;

12      $L_{\mathsf{Pre}} \leftarrow L_{\mathsf{Pre}} \cup \{\langle f_s, f_o \rangle\}$;

13   **return** L_{Pre};

end

The structure of Algorithm 2 is similar to Algorithm 1, but the computations are expressed in terms of characteristic functions, thus in terms of ranks. For example, lines 4-5 compute the equivalent of line 2 in Algorithm 1, where α_1 corresponds here to the set of odd-ranked locations, and thus contains no α-nodes. Details are given in the proof of Theorem 11. Algorithm 2 runs in time $O(|\mathsf{Loc}|^2)$, which is no more computationally expensive than the classical Pre. However, there is often an exponential factor between the number of elements in the argument of Pre in the two approaches. For example, the set $\alpha' = 2^{\mathsf{Loc} \times [k]} \times \{\emptyset\}$ with an exponential number of elements is represented by the unique pair $\langle f_s, f_{\emptyset} \rangle$ where $f_s(\ell) = 0$ for all $\ell \in \mathsf{Loc}$, which makes the new approach much more efficient in practice.

6 Implementation and Practical Evaluation

The randomized model To evaluate our new algorithm for universality of NBW and compare with the existing implementations of the Kupferman-Vardi and Miyano-Hayashi constructions, we use a random model to generate NBW. This model was first proposed by Tabakov and Vardi to compare the efficiency of some algorithms for automata in the context of finite words automata [TV05] and more recently in the context of infinite words automata [Tab06]. In the model, the input alphabet is fixed to $\Sigma = \{0, 1\}$, and for each letter $\sigma \in \Sigma$, a number k_σ of different state pairs $(\ell, \ell') \in \mathsf{Loc} \times \mathsf{Loc}$ are chosen uniformly at random before the corresponding transitions (ℓ, σ, ℓ') are added to the automaton. The ratio $r_\sigma = \frac{k_\sigma}{|\mathsf{Loc}|}$ is called the *transition density* for σ. This ratio represents the average outdegree of each state for σ. In all

Table 1. Automata size for which the median execution time for checking universality is less than 20 seconds. The symbol $\propto$ means *more than 1500*.

f \ r	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0
0.1	$\propto$	$\propto$	$\propto$	550	200	120	60	40	30	40	50	50	70	90	100
0.3	$\propto$	$\propto$	$\propto$	500	200	100	40	30	40	70	100	120	160	180	200
0.5	$\propto$	$\propto$	$\propto$	500	200	120	60	60	90	120	120	120	140	260	500
0.7	$\propto$	$\propto$	$\propto$	500	200	120	70	80	100	200	440	1000	$\propto$	$\propto$	$\propto$
0.9	$\propto$	$\propto$	$\propto$	500	180	100	80	200	600	$\propto$	$\propto$	$\propto$	$\propto$	$\propto$	$\propto$

experiments, we choose $r_0 = r_1$, and denote the transition density by r. The model contains a second parameter: the *density f of accepting states*. There is only one initial state, and the number m of accepting states is linear in the total number of states, as determined by $f = \frac{m}{|\mathsf{Loc}|}$. The accepting states themselves are chosen uniformly at random. Observe that since the transition relation is not always total, automata with $f = 1$ are not necessarily universal.

Tabakov and Vardi have studied the space of parameter values for this model and argue that "interesting" automata are generated by the model as the two parameters r and f vary. They also study the density of universal automata in [Tab06].

Performance comparison We have implemented our algorithm to check the universality of randomly generated NBW. The code is written in `C` with an explicit representation for characteristic functions, as arrays of integers. All the experiments are conducted on a biprocessor Linux station (two 3.06Ghz Intel Xeons with 4GB of RAM).

Fig. 1 shows as a function of r (transition density) and f (density of accepting states) the median execution times for testing universality of 100 random automata with $|\mathsf{Loc}| = 30$. It shows that the universality test was the most difficult for $r = 1.8$ and $f = 0.1$ with a median time of 11 seconds. The times for $r \leq 1$ and $r \geq 2.8$ are not plotted because they were always less than 250ms. A similar shape and maximal median time is reported by Tabakov for automata of size 6, that is for automata that are five times smaller [Tab06]. Another previous work reports prohibitive execution times for complementing NBW of size 6, showing that explicitly constructing the complement is not a reasonable approach [GKSV03].

To evaluate the scalability of our algorithm, we have ran the following experiment. For a set of parameter values, we have evaluated the maximal size of automata (measured in term of number of locations) for which our algorithm could analyze 50 over 100 instances in less than 20 seconds. We have tried automata sizes from 10 to 1500, with a fine granularity for small sizes (from 10 to 100 with an increment of 10, from 100 to 200 with an increment of 20, and from 200 to 500 with an increment of 30) and a rougher granularity for large sizes (from 500 to 1000 with an increment of 50, and from 1000 to 1500 with an increment of 100).

The results are shown in Fig. 2, and the corresponding values are given in Table 1. The vertical scale is logarithmic. For example, for $r = 2$ and $f = 0.5$, our algorithm was able to handle at least 50 automata of size 120 in less than 20 seconds and was not

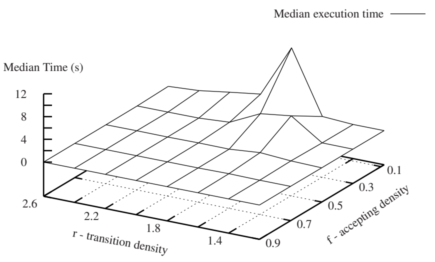

Fig. 1. Median time to check universality of 100 automata of size 30 for each sample point.

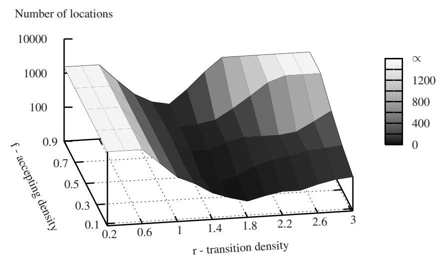

Fig. 2. Automata size for which the median execution time to check universality is less than 20 seconds (log scale).

able to do so for automata of size 140. In comparison, Tabakov and Vardi have studied the behavior of Kupferman-Vardi and Miyano-Hayashi constructions for different implementation schemes. We compare with the performances of their symbolic approach which is the most efficient. For the same parameter values ($r = 2$ and $f = 0.5$), they report that their implementation can handle NBW with at most 8 states in less than 20 seconds [Tab06].

In Fig. 3, we show the median execution time to check universality for relatively difficult instances ($r = 2$ and f vary from 0.3 to 0.7). The vertical scale is logarithmic, so the behavior is roughly exponential in the size of the automata. Similar analyzes are reported in [Tab06] but for sizes below 10.

Finally, we give in Fig. 4 the distribution of execution times for 100 automata of size 50 with $r = 2.2$ and $f = 0.5$, so that roughly half of the instances are universal. Each point represents one automaton, and one point lies outside the figure with an execution time of 675s for a non universal automaton. The existence of very few instances that are very hard was often encountered in the experiments, and this is why we use the median for the execution times. If we except this hard instance, Fig. 4 shows that universal automata (average time 350ms) are slightly easier to analyze than non-universal automata (average time 490ms). This probably comes from the fact that we stop the computation of the (greatest) fixed point whenever the initial state is no more $\preceq_{\mathsf{univ}}$-less than the successive approximations. Indeed, in such case, since the approximations are $\preceq_{\mathsf{univ}}$-decreasing, we know that the initial state would also not lie in the fixed point. Of course, this optimization applies only for non-universal automata.

7 Language Inclusion for Büchi automata

Let $\mathcal{A}_1 = \langle \mathsf{Loc}_1, \iota_1, \Sigma, \delta_1, \alpha_1 \rangle$ and $\mathcal{A}_2$ be two NBW defined on the same alphabet Σ for which we want to check language inclusion: $\mathcal{L}(\mathcal{A}_1) \subseteq^? \mathcal{L}(\mathcal{A}_2)$. To solve this problem, we check emptiness of $\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}^c(\mathcal{A}_2)$. As we have seen, we can use the Kupferman-Vardi and Miyano-Hayashi construction to specify a NBW $\mathcal{A}_2^c = \langle \mathsf{Loc}_2, \iota_2, \Sigma, \delta_2, \alpha_2 \rangle$ that accepts the complement of the language of $\mathcal{A}_2$.

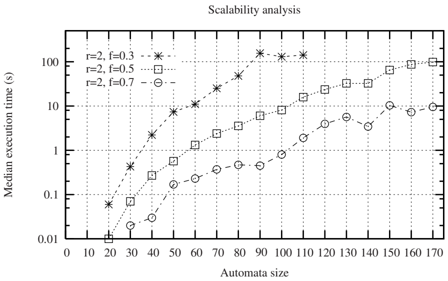

Fig. 3. Median time to check universality (of 100 automata for each sample point).

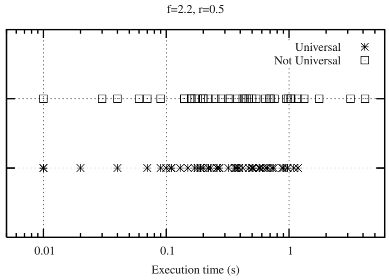

Fig. 4. Execution time to check universality of 100 automata, 57 of which were universal.

Using the classical product construction, let $\mathcal{B}$ be a finite automaton with set of locations $\mathsf{Loc}_{\mathcal{B}} = \mathsf{Loc}_1 \times \mathsf{Loc}_2$, initial state $\iota_{\mathcal{B}} = (\iota_1, \iota_2)$, and tranition function $\delta_{\mathcal{B}}$ such that $\delta_{\mathcal{B}}((\ell_1, \ell_2), \sigma) = \delta_1(\ell_1, \sigma) \times \delta_2(\ell_2, \sigma)$. We equip $\mathcal{B}$ with the generalized Büchi condition $\{\beta_1, \beta_2\} = \{\alpha_1 \times \mathsf{Loc}_2, \mathsf{Loc}_1 \times \alpha_2\}$, thus asking for a run of $\mathcal{B}$ to be accepting that it visits β_1 and β_2 infinitely often. It is routine to show that we have $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2^c)$. The following fixed point

$$\mathcal{F}'_{\mathcal{B}} \equiv \nu y \cdot \Big(\mu x_1 \cdot \big[\mathsf{Pre}^{\mathcal{B}}(x_1) \cup (\mathsf{Pre}^{\mathcal{B}}(y) \cap \beta_1)\big] \cap \mu x_2 \cdot \big[\mathsf{Pre}^{\mathcal{B}}(x_2) \cup (\mathsf{Pre}^{\mathcal{B}}(y) \cap \beta_2)\big]\Big)$$

can be used to check emptiness of $\mathcal{B}$ as we have $\mathcal{L}(\mathcal{B}) \neq \emptyset$ iff $\iota_B \in \mathcal{F}'_{\mathcal{B}}$. We now define the pre-order $\preceq_{\mathsf{inc}}$ over the locations of $\mathcal{B}$: for all $(\ell_1, \ell_2), (\ell'_1, \ell'_2) \in \mathsf{Loc}_{\mathcal{B}}$, let $(\ell_1, \ell_2) \preceq_{\mathsf{inc}} (\ell'_1, \ell'_2)$ iff $\ell_1 = \ell'_1$ and $\ell_2 \preceq_{\mathsf{univ}} \ell'_2$.

Lemma 12 *The relation $\preceq_{\mathsf{inc}}$ is a simulation for $\mathcal{B}$.*

As a consequence of the last lemma, we know that all the sets that we have to manipulate to solve the language inclusion problem using the fixed point $\mathcal{F}'_{\mathcal{B}}$ are $\preceq_{\mathsf{inc}}$-closed. The operators $\cup$, $\cap$ and Pre can be thus computed efficiently, using the same algorithms and data structures as for universality. In particular, let $\mathsf{Pre}^{\mathsf{inc}}_{\sigma}(\ell'_1, \ell'_2) = \mathsf{Pre}^{\mathcal{A}_1}_{\sigma}(\ell'_1) \times \mathsf{Pre}^{\mathsf{univ}}_{\sigma}(\ell'_2)$ where $\mathsf{Pre}^{\mathsf{univ}}_{\sigma}$ is computed by Algorithm 2 (with input $\mathcal{A}_2$). It is easy to show as a corollary of Theorem 11 that $\downarrow\mathsf{Pre}^{\mathsf{inc}}_{\sigma}(\ell'_1, \ell'_2) = \mathsf{Pre}^{\mathcal{B}}_{\sigma}(\downarrow\{(\ell'_1, \ell'_2)\})$.

8 Conclusion

We have shown that the expensive complementation constructions for nondeterministic Büchi automata can be avoided for solving classical problems like universality and language inclusion. Our approach is based on fixed points computation and the existence of simulation relations for the (exponential) constructions used in complementation of Büchi automata. Those simulations are used to dramatically reduce the amount of computations needed to decide classical problems. Their definition relies on the structure of the original automaton and do not require explicit complementation.

The resulting algorithms evaluate a fixed point formula and avoid redundant computations by maintaining sets of maximal elements according to the simulation relation. In practice, the computation of the predecessor operator, which is the key of the approach, is efficient because it is done on antichain of elements only. Eventhough the classical approaches (as well as ours) have the same worst case complexity, our prototype implementation outperforms those approaches where complementation is explicit. The huge gap of performances holds over the entire parameter space of the randomized model proposed by Tabakov and Vardi.

Applications of this paper go beyond universality and language inclusion for NBW, as we have shown that the methodology applies to alternating Büchi automata for which efficient translations from LTL formula are known [GO01]. The hope rises then that significant improvements can be brought to the model-checking problem of LTL.

References

[BL69] J. Richard Büchi and Lawrence H. Landweber. Definability in the monadic second-order theory of successor. *J. Symb. Log.*, 34(2):166–170, 1969.

[DDHR06] M. De Wulf, L. Doyen, T. A. Henzinger, and J.-F. Raskin. Antichains: A new algorithm for checking universality of finite automata. In *Proceedings of CAV 2006*, *LNCS* 4144, pp. 17–30. Springer.

[EWS05] K. Etessami, T. Wilke, and R. A. Schuller. Fair simulation relations, parity games, and state space reduction for bu"chi automata. *SIAM J. Comput.*, 34(5):1159–1175, 2005.

[GKSV03] S. Gurumurthy, O. Kupferman, F. Somenzi, and M. Y. Vardi. On complementing nondeterministic büchi automata. In *Proc. of CHARME 2003*, *LNCS* 2860, pp. 96–110. Springer.

[GO01] P. Gastin and D. Oddoux. Fast LTL to Büchi automata translation. In *Proc. of CAV 2001*, *LNCS* 2102, pp. 53–65. Springer.

[KV97] O. Kupferman and M. Y. Vardi. Weak alternating automata are not that weak. In *Proceedings of ISTCS'97*, pp. 147–158. IEEE Computer Society Press.

[MH84] Satoru Miyano and Takeshi Hayashi. Alternating finite automata on omega-words. In *CAAP*, pages 195–210, 1984.

[Mic88] Max Michel. Complementation is more difficult with automata on infinite words. CNET, Paris, 1988.

[RH04] Theo C. Ruys and Gerard J. Holzmann. Advanced spin tutorial. In *SPIN*, *LNCS* 2989, pp. 304–305. Springer, 2004.

[Saf88] Shmuel Safra. On the complexity of ω-automata. In *Proc. of FOCS: Foundations of Computer Science*, pages 319–327. IEEE, 1988.

[SVW87] A. P. Sistla, M. Y. Vardi and P. Wolper. The Complementation Problem for Büchi Automata with Applications to Temporal Logic. *Th. Comp. Sci.*, 49:217–237, 1987.

[Tab06] D. Tabakov. Experimental evaluation of explicit and symbolic approaches to complementation of non-deterministic buechi automata. *Talk at "Games and Verification" workshop, Newton Institute for Math. Sciences*. July 2006.

[TV05] D. Tabakov and M. Y. Vardi. Experimental evaluation of classical automata constructions. In *LPAR* 2005, *LNCS* 3835, pp. 396–411. Springer.

[VW86] M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification (prelim. report). In *LICS* 1986, pp. 332–344. IEEE.

[VW94] Moshe Y. Vardi and Pierre Wolper. Reasoning about infinite computations. *Inf. Comput.*, 115(1):1–37, 1994.

9 Appendix

Proof of Lemma 9. Let $\mathcal{A} = \langle \mathsf{Loc}, \iota, \Sigma, \delta, \alpha \rangle$ and $\mathsf{KVMH}(\mathcal{A}, k) = \langle Q_k \times Q_k, q_\iota, \Sigma, \delta', \alpha' \rangle$. First, we show that for all $\langle s_1, o_1 \rangle, \langle s_2, o_2 \rangle, \langle s_3, o_3 \rangle \in Q_k \times Q_k$, for all $\sigma \in \Sigma$, if $\langle s_1, o_1 \rangle \xrightarrow{\sigma}_{\delta'} \langle s_2, o_2 \rangle$ and $\langle s_3, o_3 \rangle \preceq \langle s_1, o_1 \rangle$ then $\langle s_3, o_3 \rangle \xrightarrow{\sigma}_{\delta'} \langle s_2, o_2 \rangle$. Notice that we have trivially $\langle s_2, o_2 \rangle \preceq_{\mathsf{univ}} \langle s_2, o_2 \rangle$. We give the proof for $o_1 \neq \emptyset$. The case $o_1 = \emptyset$ is proven similarly. According to Definition 3, since $\langle s_1, o_1 \rangle \xrightarrow{\sigma}_{\delta'} \langle s_2, o_2 \rangle$ we have (i) $\forall (\ell, n_1) \in s_1 \cdot \forall \ell' \in \delta(\ell, \sigma) \cdot \exists (\ell', n_2) \in s_2 : n_2 \leq n_1$ and (ii) $\forall (\ell, n_1) \in o_1 \cdot \forall \ell' \in \delta(\ell, \sigma) \cdot \exists (\ell', n_2) \in o_2 : n_2 \leq n_1$. Since $\langle s_3, o_3 \rangle \preceq \langle s_1, o_1 \rangle$, we have $o_3 \neq \emptyset$ and (i') $\forall (\ell, n_3) \in s_3 \cdot \exists (\ell, n_1) \in s_1 : n_1 \leq n_3$ and (ii') $\forall (\ell, n_3) \in o_3 \cdot \exists (\ell, n_1) \in o_1 : n_1 \leq n_3$.

Combining (i) and (i') yields $\forall (\ell, n_3) \in s_3 \cdot \forall \ell' \in \delta(\ell, \sigma) \cdot \exists (\ell', n_2) \in s_2 : n_2 \leq n_1 \leq n_3$, and combining (ii) and (ii') yields $\forall (\ell, n_3) \in o_3 \cdot \forall \ell' \in \delta(\ell, \sigma) \cdot \exists (\ell', n_2) \in o_2 : n_2 \leq n_1 \leq n_3$. Since $o_3 \neq \emptyset$, this implies that $\langle s_3, o_3 \rangle \xrightarrow{\sigma}_{\delta'} \langle s_2, o_2 \rangle$.

Second, for all $\langle s, o \rangle \in \alpha'$ we have $o = \emptyset$, and thus for all $\langle s', o' \rangle \subseteq \mathsf{Loc} \times [k]$, if $\langle s', o' \rangle \preceq \langle s, o \rangle$ then $o' = \emptyset$ so that $\langle s', o' \rangle \in \alpha'$.

Hence $\preceq_{\mathsf{univ}}$ is a simulation for $\mathsf{KVMH}(\mathcal{A}, k)$. ■

Proof of Theorem 11. Let $\mathcal{A}^c = \mathsf{KVMH}(\mathcal{A}, k) = \langle Q_k \times Q_k, q_\iota, \Sigma, \delta', \alpha' \rangle$.

We show that (1) $[\![L_{\mathsf{Pre}}]\!]_k \subseteq \mathsf{Pre}_\sigma^{\mathcal{A}^c}(\downarrow [\![\langle f_{s'}, f_{o'} \rangle]\!]_k)$ and (2) for all $\langle s_1, o_1 \rangle \in \mathsf{Pre}_\sigma^{\mathcal{A}^c}(\downarrow [\![\langle f_{s'}, f_{o'} \rangle]\!]_k)$, there exists $\langle s, o \rangle \in [\![L_{\mathsf{Pre}}]\!]_k$ such that $\langle s_1, o_1 \rangle \preceq_{\mathsf{univ}} \langle s, o \rangle$. This entails that $\downarrow [\![L_{\mathsf{Pre}}]\!]_k = \mathsf{Pre}_\sigma^{\mathcal{A}^c}(\downarrow [\![\langle f_{s'}, f_{o'} \rangle]\!]_k)$.

We make the following preliminary remarks: if $\langle s, o \rangle \xrightarrow{\sigma}_{\delta'} \langle s', o' \rangle$, then for all $\langle s_1, o_1 \rangle \in [\![\langle f_s, f_o \rangle]\!]_k$, there exists $\langle s_1', o_1' \rangle \in [\![\langle f_{s'}, f_{o'} \rangle]\!]_k$ such that $\langle s_1, o_1 \rangle \xrightarrow{\sigma}_{\delta'} \langle s_1', o_1' \rangle$, which we write $\langle f_s, f_o \rangle \xrightarrow{\sigma}_{\delta'} \langle f_{s'}, f_{o'} \rangle$. Similarly, if $\langle s, o \rangle \preceq_{\mathsf{univ}} \langle s', o' \rangle$, then for all $\langle s_1', o_1' \rangle \in [\![\langle f_{s'}, f_{o'} \rangle]\!]_k$ we have $\langle s, o \rangle \preceq_{\mathsf{univ}} \langle s_1', o_1' \rangle$, which we write $\langle s, o \rangle \preceq_{\mathsf{univ}} \langle f_{s'}, f_{o'} \rangle$.

To prove (1), we first show that $\langle f_s, f_o \rangle \xrightarrow{\sigma}_{\delta'} \langle f_{s'}, f_{o'} \rangle$ for every pair $\langle f_s, f_o \rangle$ added to L_{Pre} at line 12. Moreover, we show that $f_s \leq f_o$. By the test of line 8, we have $[\![f_o]\!]_k \neq \{\emptyset\}$. Let $\langle s', o' \rangle \in [\![\langle f_{s'}, f_{o'} \rangle]\!]_k$ ($\langle s', o' \rangle$ exists because $f_{s'} \leq f_{o'}$). According to the definition of $\mathsf{KVMH}(\mathcal{A}, k)$ (see Section 2), we have to check that there exists a set $o'' \subseteq s'$ such that $o' = o'' \setminus \mathsf{odd}$ (we take $o'' = o' \cup (s' \cap \mathsf{odd})$), and the following conditions hold for all $\langle s, o \rangle \in [\![\langle f_s, f_o \rangle]\!]_k$:

(i) $\forall (\ell, n) \in s \cdot \forall \ell' \in \delta(\ell, \sigma) \cdot \exists n' \leq n : (\ell', n') \in s'$.
Observe that for all $\ell \in \mathsf{Loc}$, for all $\ell' \in \delta(\ell, \sigma)$, we have $f_{s'}(\ell') \leq f_s(\ell)$ (lines 10,11 of Algorithm 2). Since $f_s(\ell) \leq n$, we take $n' = f_{s'}(\ell')$ and we have $n' \leq f_s(\ell) \leq n$ with $(\ell', n') \in s'$.

(ii) $\forall (\ell, n) \in o \cdot \forall \ell' \in \delta(\ell, \sigma) \cdot \exists n' \leq n : (\ell', n') \in o''$.
Since $o'' = o' \cup (s' \cap \mathsf{odd})$, we have $f_{o''}(\ell') = f_o(\ell')$ for $\ell' \in \alpha$ and $f_{o''}(\ell') = \min\{f_{o'}(\ell'), \lceil f_{s'}(\ell') \rceil^{\mathsf{odd}}\}$ for $\ell' \notin \alpha$. Now, for all $\ell \in \mathsf{Loc}$, for all $\ell' \in \delta(\ell, \sigma)$, we have either $\ell' \in \alpha$ and then $f_o(\ell) \geq n'$ for $n' = f_{o'}(\ell')$, or $\ell' \notin \alpha$ and then $f_o(\ell) \geq n'$ for $n' = \min\{f_{o'}(\ell'), \lceil f_{s'}(\ell') \rceil^{\mathsf{odd}}\}$ (lines 4-6 of Algorithm 2). In both cases, for $(\ell, n) \in o$ we have $f_{o''}(\ell') \leq n' \leq f_o(\ell) \leq n$ and $(\ell', n') \in o''$.

Moreover, we prove that:

(iii) $f_s \leq f_o$.
Since $f_{s'} \leq f_{o'}$, we have for all $\ell \in \mathsf{Loc}$ either $f_{o'}(\ell) > k$ or $f_{o'}(\ell) \geq f_{s'}(\ell)$. By lines 4-6 of Algorithm 2, we have for all $\ell \in \mathsf{Loc}$, for all $\ell' \in \delta(\ell, \sigma)$ either $f_o(\ell) \geq f_{o'}(\ell')$ or $f_o(\ell) \geq \lceil f_{s'}(\ell') \rceil^{\mathsf{odd}}$, and thus either $f_o(\ell) > k$ or $f_o(\ell) \geq f_{s'}(\ell')$. Hence, we have for all $\ell \in \mathsf{Loc}$ either $f_o(\ell) > k$ or $f_o(\ell) \geq \max\{f_{s'}(\ell') \mid \ell' \in \delta(\ell, \sigma)\}$. Therefore, by lines 10-11 of Algorithm 2, if $\ell \notin \alpha$, then $f_o(\ell) > k$ or $f_o(\ell) \geq f_s(\ell)$, and if $\ell \in \alpha$, then $f_o(\ell)$ is even (line 6) and thus either $f_o(\ell) > k$ or $f_o(\ell) \geq \lceil \max\{f_{s'}(\ell') \mid \ell' \in \delta(\ell, \sigma)\} \rceil^{\mathsf{even}} = f_s(\ell)$. In all cases, $f_s \leq f_o$.

(iv) $\forall \ell \in \alpha : f_s(\ell)$ and $f_o(\ell)$ are even.
This is enforced by line 11 and line 6 of the algorithm.

Second, we show that for every pair $\langle f_o, \emptyset \rangle$ added to L_{Pre} at line 7, we have $\langle f_o, \emptyset \rangle \xrightarrow{\sigma}_{\delta'} \langle s'', o'' \rangle$ for some $\langle s'', o'' \rangle \preceq_{\mathsf{univ}} \langle s', o' \rangle$ with $\langle s', o' \rangle \in [\![\langle f_{s'}, f_{o'} \rangle]\!]_k$ ($\langle s', o' \rangle$ exists because $f_{s'} \leq f_{o'}$). We take $s'' = o' \cup (s' \cap \mathsf{odd})$ and $o'' = s'' \setminus \mathsf{odd}$. Since $o' \subseteq s'$, we have (a) $s'' \subseteq s'$, and we have (b) $o'' = o' \setminus \mathsf{odd} \subseteq o'$. Moreover, if $o' \neq \emptyset$, then $(\ell, k) \in o'$ for some $\ell \in \mathsf{Loc}$, and since k is even, $(\ell, k) \in o''$ and thus $o'' \neq \emptyset$. Since $o'' \subseteq o'$, we have (c) $o' \neq \emptyset$ iff $o'' \neq \emptyset$. Hence $\langle s'', o'' \rangle \preceq_{\mathsf{univ}} \langle s', o' \rangle$. The fact that $\langle f_o, \emptyset \rangle \xrightarrow{\sigma}_{\delta'} \langle s'', o'' \rangle$ is proven similarly to (ii).

To prove (2), assume that there exist $\langle s_1, o_1 \rangle$ and $\langle s'_1, o'_1 \rangle$ such that $\langle s_1, o_1 \rangle \xrightarrow{\sigma}_{\delta'} \langle s'_1, o'_1 \rangle$ and $\langle s'_1, o'_1 \rangle \in \downarrow [\![\langle f_{s'}, f_{o'} \rangle]\!]_k$. We have to show that there exists $\langle f_s, f_o \rangle \in L_{\mathsf{Pre}}$ such that $\langle s_1, o_1 \rangle \preceq_{\mathsf{univ}} \langle f_s, f_o \rangle$.

First, assume that $o_1 \neq \emptyset$. Notice that since $\langle s'_1, o'_1 \rangle \preceq_{\mathsf{univ}} \langle f_{s'}, f_{o'} \rangle$, we have for all $\ell \in \mathsf{Loc}$ either $f_{s'_1}(\ell) > k$ or $f_{s'_1}(\ell) \geq f_{s'}(\ell)$, and either $f_{o'_1}(\ell) > k$ or $f_{o'_1}(\ell) \geq f_{o'}(\ell)$. Since $\langle s_1, o_1 \rangle \xrightarrow{\sigma}_{\delta'} \langle s'_1, o'_1 \rangle$, we have:

(i) for all $(\ell, n_1) \in s_1$, for all $\ell' \in \delta(\ell, \sigma)$, $n_1 \geq f_{s_1}(\ell) \geq f_{s'_1}(\ell')$ and thus either $n_1 \geq k$ or $n_1 \geq f_{s'}(\ell')$. Hence, for all $\ell \in \mathsf{Loc}$ either $f_{s_1}(\ell) > k$ or $f_{s_1}(\ell) \geq \max\{f_{s'}(\ell') \mid \ell' \in \delta(\ell, \sigma)\} = f_s(\ell)$ (where f_s is computed by line 10 of Algorithm 2). Thus, $f_s \leq f_{s_1}$.

(ii) for all $(\ell, n_2) \in o_1$, for all $\ell' \in \delta(\ell, \sigma)$, $n_2 \geq\geq f_{o_1}(\ell) \geq f_{o''_1}(\ell')$ for some set o''_1 such that $o''_1 \subseteq s'_1$ and $o''_1 \setminus \mathsf{odd} = o'_1$. Hence necessarily $o''_1 \subseteq o'_1 \cup (s'_1 \cap \mathsf{odd}) \subseteq o' \cup (s' \cap \mathsf{odd})$ and thus $f_{o''_1} \geq f_{o' \cup (s' \cap \mathsf{odd})}$. Hence, for all $\ell \in \mathsf{Loc}$ either $f_{o_1}(\ell) > k$ or $f_{o_1}(\ell) \geq f_o(\ell)$ (where f_o is computed at lines 1-6 of Algorithm 2). Thus, $f_o \leq f_{o_1}$.

Hence, $\langle f_s, f_o \rangle$ added to L_{Pre} by Algorithm 2 at line 12 satisfies $\langle s_1, o_1 \rangle \preceq_{\mathsf{univ}} \langle f_s, f_o \rangle$.

Second, assume that $o_1 = \emptyset$. Let $s'' = o' \cup (s' \cap \mathsf{odd})$. Since $\langle s_1, o_1 \rangle \xrightarrow{\sigma}_{\delta'} \langle s'_1, o'_1 \rangle$ and $o_1 = \emptyset$, we have $o'_1 = s'_1 \setminus \mathsf{odd}$. Since $\langle s'_1, o'_1 \rangle \preceq_{\mathsf{univ}} \langle f_{s'}, f_{o'} \rangle$, we have (a) $s'_1 \subseteq s'$ and thus $s'_1 \cap \mathsf{odd} \subseteq s' \cap \mathsf{odd} \subseteq s''$ and (b) $o'_1 = s'_1 \setminus \mathsf{odd} \subseteq o' \subseteq s''$. Hence $s'_1 \subseteq s''$ which is equivalent to say that for all $\ell \in \mathsf{Loc}$ either $f_{s'_1}(\ell) > k$ or $f_{s'_1}(\ell) \geq f_{s''}(\ell)$. Now, by the fact that $\langle s_1, o_1 \rangle \xrightarrow{\sigma}_{\delta'} \langle s'_1, o'_1 \rangle$, we know that for all $(\ell, n_1) \in s_1$, for all $\ell' \in \delta(\ell, \sigma)$, $n_1 \geq f_{s'_1}(\ell')$ and thus either $n_1 \geq k$ or $n_1 \geq f_{s''}(\ell')$. Notice that $f_o(\ell) = \max\{f_{s''}(\ell') \mid \ell' \in \delta(\ell, \sigma)\}$ where f_o is computed at lines 1-6 of Algorithm 2. Thus, either $n_1 \geq k$ or $n_1 \geq f_o(\ell)$ for all $\ell \in \mathsf{Loc}$ and therefore $f_o \leq f_{s_1}$ so that $\langle s_1, \emptyset \rangle \preceq_{\mathsf{univ}} \langle f_o, \emptyset \rangle$ where $\langle f_o, \emptyset \rangle$ is added to L_{Pre} by Algorithm 2 at line 12. ∎