Revealing vs. Concealing: More Simulation Games for Büchi Inclusion

Milka Hutagalung, Martin Lange, and Etienne Lozes

School of Electr. Eng. and Computer Science, University of Kassel, Germany*

Abstract. We address the problem of deciding language inclusion between two non-deterministic Büchi automata. It is known to be PSPACE-complete and finding techniques that are efficient in practice is still a challenging problem. We introduce two new sequences of simulation relations, called multi-letter simulations, in which Verifier has to reproduce Refuter's moves taking advantage of a forecast. We compare these with the multi-pebble games introduced by Etessami. We show that multi-letter simulations, despite being more restrictive than multi-pebble ones, have a greater potential for an incremental inclusion test, for their size grows generally slower. We evaluate this idea experimentally and show that incremental inclusion testing may outperform the most advanced Ramsey-based algorithms by two orders of magnitude.

1 Introduction

Nondeterministic Büchi automata (NBA) are an important formalism and defacto standard for the specification and verification of reactive systems. In the automata-theoretic approach to formal verification [18], problems about programs get translated into decision problems on automata, for instance NBA. Satisfiability of a specification corresponds to language non-emptiness which is NLOGSPACE-complete for NBA. Hence, it boils down to some reachability questions which can be solved efficiently.

The step from wanted to unwanted (or vice-versa) program behaviour corresponds to the complementation problem for NBA. It is known that this is inherently exponential. This explains why other decision problems on NBA are harder than emptiness, for example inclusion—corresponding to the question of whether one specification supersedes another—is PSPACE-complete. Note that $L(\mathcal{A}) \subseteq L(\mathcal{B})$ iff $L(\mathcal{A}) \cap \overline{L(\mathcal{B})} = \emptyset$. It is possible to construct an NBA recognising $L(\mathcal{A}) \cap \overline{L(\mathcal{B})}$ which is in general exponentially larger than $\mathcal{B}$ and which can then be tested for emptiness. This results in an NPSPACE procedure, and Savitch's Theorem [15] brings it down to PSPACE.

A large amount of work in the area of automata theory for verification is devoted to avoiding explicit complementation because none of the existing complementation procedures [14, 17, 12] is widely accepted to be good enough for

* The European Research Council has provided financial support under the European Community's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement no 259267.

practical purposes. Regarding the inclusion problem, there is for instance the so-called Ramsey-based approach [7, 1, 2] which essentially uses the computational content of Büchi's original proof of complementability of NBA. It clearly does not avoid PSPACE-hardness but it can sometimes outperform algorithms based on explicit complementation.

Since the early works of Dill et al [4], there has also been an approach based on simulation relations. The notion of simulation that reflects the Büchi acceptance condition, called *fair simulation* [10], can be described in terms of a game between two players, Refuter and Verifier. Consider two NBA $\mathcal{A}$ and $\mathcal{B}$ over some alphabet Σ. The game proceeds for possibly infinitely many rounds, starting with two pebbles being placed on the initial states of $\mathcal{A}$ and $\mathcal{B}$. In each round, Refuter chooses some $a \in \Sigma$ and moves the $\mathcal{A}$-pebble along some a-transition. Verifier responds by moving the $\mathcal{B}$-pebble along some a-transition, too. Refuter wins if the infinite sequence of states pebbled in $\mathcal{A}$ forms an accepting run, and the one in $\mathcal{B}$ does not. A player loses as soon as he/she cannot move anymore.

There is a relatively simple encoding of this game as a parity game with 3 priorities. Thus, the winner in this game can be decided even in polynomial time. This game entails language inclusion, in the sense that language inclusion holds whenever Verifier has a winning strategy, but without surprise, the converse does not hold in general (otherwise we would have PSPACE=PTIME!). It is therefore reasonable to ask whether the fair simulation games can be refined in order to obtain something that is "closer" to language inclusion. The answer is yes, and it is helpful to consult some simple game-theoretic concepts for its explanation.

Fair simulation games are games of *perfect information*: at each time both players have full knowledge about the state of the game. Language inclusion, on the other hand, can be seen as a game of *imperfect information*, where refuter cannot observe Verifier's pebble, and Verifier can always revise his choice provided it remains consistent with all previous rounds. The anti-chain approach to language inclusion [5] can be seen as solving such a game, and, like the Ramsey-based approach, it is in general exponential but can outperform methods based on explicit complementation.

This reading of the problem suggests that an approximation of language inclusion based on fair simulation can be obtained by introducing some form of "opacity" in Verifier's moves. Since an approximation is either complete or it is not, it is useful to stratify the approximation based on some parameter $k \in \mathbb{N}$, such that smaller values of k, raising simpler games, can be tested first, yielding an incremental algorithm for deciding language inclusion. Such a parameterized opacity is obtained for the so-called k-pebble game [6]. This essentially gives Verifier a limited amount of resources in order to conceal information from Refuter. He now has up to k pebbles that he moves along a-transitions in $\mathcal{B}$ in order to respond to Refuter's choice a. The fair simulation game becomes the same as the 1-pebble fair simulation game, and with growing k, the k-pebble game gets "closer" to language inclusion.

In this paper, we introduce another family of approximations for language inclusion. That problem can be characterised by a simple one-round simulation game: first Refuter chooses an infinite word $w \in L(\mathcal{A})$, then Verifier produces an accepting run for it in $\mathcal{B}$. This is not a game of imperfect information anymore but it can be seen as a game with *infinite forecast.* A natural finite approximation of this is to let Verifier have a finite forecast on Refuter's moves. We call the result the *k-letter game.* It works similar to the ordinary fair simulation game but requires Refuter to always choose the next k letters of the infinite word to be constructed. This makes Refuter *reveal* more information about his subsequent choices to Verifier, whereas pebble games permit Verifier to *conceal* information about his past choices.

The k-letter games provide a new approach to the NBA inclusion problem. They can be used in order to devise an incremental language inclusion test. It successively solves games for increasing parameters k. Each iteration – which fixes k – can be done in polynomial time. The complexity grows exponentially with k but the base of this exponential is only the fixed (and often small) size of the underlying alphabet whereas, in the case of the k-pebble games, the base is the variable and usually not so small size of one of the input automata.

The rest of the paper is organised as follows. Sect. 2 recalls the necessary background about NBA and simulation games. Sect. 3 defines and examines so-called *static multi-letters simulations.* Sect. 4 refines them to *dynamic multi-letters simulations* which are theoretically better suited for approximating language inclusion. Sect. 5 reports on experiments that compare incremental inclusion tests based on multi-letter and multi-pebble games with one of the most advanced version of the Ramsey-based inclusion test.

2 Background

Words and Büchi automata. As usual, we use Σ to denote a finite *alphabet* with $|\Sigma| \geq 2$, ε for the empty word, $\Sigma^*/\Sigma^+/\Sigma^\omega$ for the sets of all finite / non-empty finite / infinite words over Σ. For some $k \in \mathbb{N}$, Σ^k (resp $\Sigma^{\leq k}$) denotes the set of all words of length exactly (resp. at most) k. In the following, all language-theoretic concept are to be understood with respect to some fixed alphabet.

A non-deterministic Büchi automaton (NBA) is a tuple $\mathcal{A} = (Q, q_0, \Delta, F)$ where Q is a finite set of states with q_0 being a designated starting state, $\Delta \subseteq Q \times \Sigma \times Q$ is a transition relation, and $F \subseteq Q$ is the set of accepting states. The *size* of $\mathcal{A}$ is measured in terms of its state space: $|\mathcal{A}| := |Q|$.

A *run* of $\mathcal{A}$ on a word $w = a_0a_1 \cdots \in \Sigma^\omega$ is an infinite sequence $\rho = q_0, q_1, \ldots$ such that $(q_i, a_i, q_{i+1}) \in \Delta$ for all $i \geq 0$. Let $\mathit{inf}(q_0, q_1, \ldots) = \{q \mid \text{there are infinitely many } i \text{ with } q = q_i\}$. When using this notation we implicitly assume that $q_0, q_1, \ldots$ is an infinite sequence. The run is *accepting* if $\mathit{inf}(q_0, q_1, \ldots) \cap F \neq \emptyset$. The language of $\mathcal{A}$ is the set $L(\mathcal{A})$ of infinite words for which there exists an accepting run.

We write $q \xrightarrow{a} q'$ when $(q, a, q') \in \Delta$ assuming that the underlying NBA can be inferred from the context.

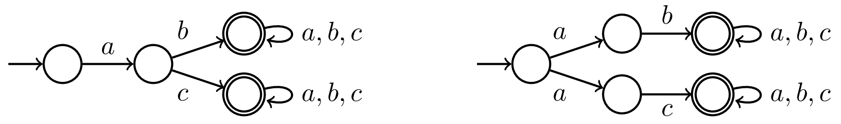
The *language inclusion* problem for NBA is the following. Given two NBA $\mathcal{A}$ and $\mathcal{B}$, decide whether or not $L(\mathcal{A}) \subseteq L(\mathcal{B})$ holds. We simply write $\mathcal{A} \sqsubseteq_{\mathsf{incl}} \mathcal{B}$ in that case.

Fair Simulation. The *fair simulation game* $\mathcal{G}_{\mathsf{fair}}$ [10] is played between Refuter and Verifier on two NBA $\mathcal{A} = (Q_{\mathcal{A}}, q^{\mathcal{A}}_{\mathsf{init}}, \Delta_{\mathcal{A}}, F_{\mathcal{A}})$ and $\mathcal{B} = (Q_{\mathcal{B}}, q^{\mathcal{B}}_{\mathsf{init}}, \Delta_{\mathcal{B}}, F_{\mathcal{B}})$, both controlling a pebble each, according to the following description.

Fair simulation game $\mathcal{G}_{\mathsf{fair}}(\mathcal{A},\mathcal{B})$	
start:	$q_0 := q^{\mathcal{A}}_{\mathsf{init}}$, $q'_0 := q^{\mathcal{B}}_{\mathsf{init}}$
round i: $(i = 0, 1, \ldots)$	(1) Refuter chooses some $a \in \Sigma$ (2) Refuter chooses q_{i+1} such that $q_i \xrightarrow{a} q_{i+1}$ (3) Verifier chooses q'_{i+1} such that $q'_i \xrightarrow{a} q'_{i+1}$
winner:	Refuter wins if ... • Verifier cannot move anymore • $\mathit{inf}(q_0, q_1, \ldots) \cap F_{\mathcal{A}} \neq \emptyset$ and $\mathit{inf}(q'_0, q'_1, \ldots) \cap F_{\mathcal{B}} = \emptyset$ Verifier wins if ... • Refuter cannot move anymore • $\mathit{inf}(q_0, q_1, \ldots) \cap F_{\mathcal{A}} = \emptyset$ or $\mathit{inf}(q'_0, q'_1, \ldots) \cap F_{\mathcal{B}} \neq \emptyset$

We write $\mathcal{A} \sqsubseteq_{\mathsf{fair}} \mathcal{B}$ if Verifier has a winning strategy for $\mathcal{G}_{\mathsf{fair}}(\mathcal{A},\mathcal{B})$. It is well-known that fair simulation approximates language inclusion in the sense that $\mathcal{A} \sqsubseteq_{\mathsf{fair}} \mathcal{B}$ implies $\mathcal{A} \sqsubseteq_{\mathsf{incl}} \mathcal{B}$ but not vice-versa. A counterexample for the converse implication is the following.

Example 1. Consider the following two NBA $\mathcal{A}$ (left) and $\mathcal{B}$ (right) over the alphabet $\Sigma = \{a, b, c\}$.



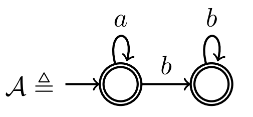
Clearly, we have $\mathcal{A} \sqsubseteq_{\mathsf{incl}} \mathcal{B}$. On the other hand, Refuter can win $\mathcal{G}_{\mathsf{fair}}(\mathcal{A},\mathcal{B})$ because Verifier has to commit in the first round to some a-successor, and then Refuter can choose a b- or a c-successor whereas Verifier only has one of these choices.

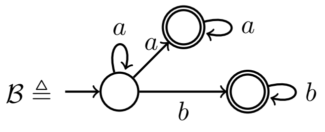
The game $\mathcal{G}_{\mathsf{fair}}(\mathcal{A},\mathcal{B})$ can be reduced to a parity game [9] on a graph of size $\mathcal{O}(|\mathcal{A}| \cdot |\mathcal{B}| \cdot |\Sigma|)$. Thus, the winner of a fair simulation game is decidable. In the vertices, we keep track of the current state of $\mathcal{A}$, $\mathcal{B}$, and the letter a chosen by Refuter. The vertex reached by a move $q'_j \xrightarrow{a} q'_{j+1}$ of Verifier has a priority 2 if q'_{j+1} is a final state, and the vertex reached by a move $q_j \xrightarrow{a} q_{j+1}$ of Refuter has a priority 1 if q_{j+1} is a final state. Other vertices have priority 0. It is not too hard to check that player 0 wins the parity game iff Verifier has a winning strategy for $\mathcal{G}_{\mathsf{fair}}(\mathcal{A},\mathcal{B})$. In fact, winning strategies in these two games can easily be derived from one another.

Multi-Pebble Games. The *k-pebble game* for some $k \geq 1$ is a refinement of the fair simulation game [6]. Here we only give a brief sketch of its definition and state some important properties. A detailed definition is not necessary in order to follow the rest of this paper. The interested reader is referred to the literature for that purpose [6, 3].

In the k-pebble game $\mathcal{G}^k_{\mathsf{peb}}(\mathcal{A},\mathcal{B})$ on two NBA $\mathcal{A}$ and $\mathcal{B}$, Refuter controls and moves a pebble on $\mathcal{A}$ as is done in the fair simulation game. Verifier now controls k pebbles on $\mathcal{B}$. In response to a letter a chosen by Refuter he chooses k (not necessarily different) a-successors of the currently k pebbled states in $\mathcal{B}$ and moves the pebbles there. The winning conditions for finite plays are the same as above. An infinite play is won by Verifier if the constructed infinite run in $\mathcal{A}$ is not accepting or if it can be guaranteed in some way that the pebbling in $\mathcal{B}$ "contained an accepting run". As an example, $\mathcal{A} \sqsubseteq^2_{\mathsf{peb}} \mathcal{B}$ holds for the two NBA of Ex. 1: in the first round, Verifier pushes a pebble in each of the two branches, and in the second round, he drops the irrelevant one, and continues like in the standard fair simulation game. It is hard to determine whether a sequence of pebblings contains an accepting run, roughly for the same reasons that make the power set construction unsuitable for determinising Büchi automata. However, since multi-pebble games only approximate language inclusion, it is possible to relax the winning condition even further and check whether the sequence of pebblings contains a run that is accepting w.r.t. a co-Büchi condition or, equivalently, whether all runs contained in it are accepting w.r.t. the Büchi condition. The determinisation problem for co-Büchi automata is conceptually simpler [13], and using this together with the possibility to drop pebbles yields a reasonable approximation and a direct reduction to finite parity games.

Example 2. For every $k \geq 1$, the k-pebble game for the two automata





is won by Refuter. He wins by always playing a: Verifier must keep a pebble on the first state of $\mathcal{B}$ to be ready for a possible b, i.e. he must not drop it at any time. Since this pebble does not visit an accepting state infinitely often, Refuter wins the game. Note that $L(\mathcal{A}) = L(\mathcal{B})$ nonetheless.

We write $\mathcal{A} \sqsubseteq^k_{\mathsf{peb}} \mathcal{B}$ if Verifier has a winning strategy for $\mathcal{G}^k_{\mathsf{peb}}(\mathcal{A},\mathcal{B})$. Obviously, we have $\sqsubseteq^1_{\mathsf{peb}} = \sqsubseteq_{\mathsf{fair}}$.

Proposition 3 ([6]). *For every $k \geq 1$ and NBA $\mathcal{A},\mathcal{B}$ over Σ there is a parity game of size at most $|\mathcal{A}| \cdot (2 \cdot |\mathcal{B}| + 1)^k \cdot (|\Sigma| + 1)$ and index 3 that is won by player 0 iff $\mathcal{A} \sqsubseteq^k_{\mathsf{peb}} \mathcal{B}$.*

Furthermore, these games give rise to the following hierarchy

$$\sqsubseteq^1_{\mathsf{peb}} \subsetneq \sqsubseteq^2_{\mathsf{peb}} \subsetneq \sqsubseteq^3_{\mathsf{peb}} \subsetneq \ \cdots \ \subsetneq \bigcup_{k\geq 1} \sqsubseteq^k_{\mathsf{peb}} \subsetneq \sqsubseteq_{\mathsf{incl}} \ . \tag{1}$$

3 The Static Multi-Letter Games

We consider a new parametrised simulation game. In this game, Verifier moves only one pebble, but benefits from Refuter being forced to reveal more information to him/her in a single round.

Definition. Let $k \geq 1$. The *static k-letter game* is played between Refuter and Verifier on two NBA $\mathcal{A} = (Q_\mathcal{A}, q^\mathcal{A}_{\mathsf{init}}, \Delta_\mathcal{A}, F_\mathcal{A})$ and $\mathcal{B} = (Q_\mathcal{B}, q^\mathcal{B}_{\mathsf{init}}, \Delta_\mathcal{B}, F_\mathcal{B})$, similar to the fair simulation game. However, in each round both players advance by k moves through the NBA instead of just 1 as it is done in the fair simulation game.

Static k-letters game $\mathcal{G}^k_{\mathsf{stat}}(\mathcal{A},\mathcal{B})$	
start:	$q_0 := q^\mathcal{A}_{\mathsf{init}}$, $q'_0 := q^\mathcal{B}_{\mathsf{init}}$, $j := 0$
round i:	(1) Refuter chooses some $w = a_1 \dots a_k \in \Sigma^k$ (2) Refuter chooses $q_{j+1}, \dots, q_{j+k}$ such that $q_j \xrightarrow{a_1} q_{j+1} \xrightarrow{a_2} q_{j+2} \xrightarrow{a_3} \dots \xrightarrow{a_k} q_{j+k}$ (3) Verifier chooses $q'_{j+1}, \dots, q'_{j+k}$ such that $q'_j \xrightarrow{a_1} q'_{j+1} \xrightarrow{a_2} q'_{j+2} \xrightarrow{a_3} \dots \xrightarrow{a_k} q'_{j+k}$ (4) $j := j + k$
winner:	same as in fair simulation game

We write $\mathcal{A} \sqsubseteq^k_{\mathsf{stat}} \mathcal{B}$ if Verifier has a winning strategy for $\mathcal{G}^k_{\mathsf{stat}}(\mathcal{A},\mathcal{B})$.

The game $\mathcal{G}^k_{\mathsf{stat}}(\mathcal{A},\mathcal{B})$ can be modeled by a parity game on a graph of size $\mathcal{O}(|\mathcal{A}| \cdot |\mathcal{B}| \cdot (|\Sigma|^k + 1))$. In the vertices, we keep track the current state of $\mathcal{A}$, $\mathcal{B}$, and the word $w = a_1 \dots a_k$ chosen by Refuter. The vertex reached by a move $q'_j \xrightarrow{w} q'_{j+k}$ of Verifier has a priority 2 if a final state is seen along the move. The vertex reached by a move $q_j \xrightarrow{w} q_{j+k}$ of Refuter has priority 1 if a final state is seen along the move. Other vertices have priority 0. We can use a suitable result on parity games of index 3 [11], and obtain the following theorem.

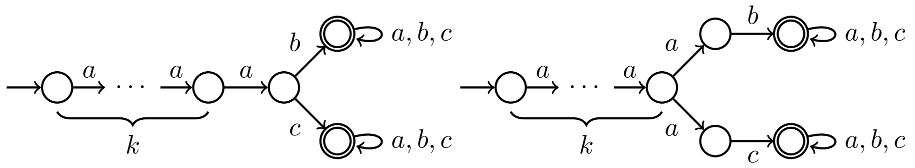
Theorem 4. *For every $k \geq 1$ and NBA $\mathcal{A}$ and $\mathcal{B}$, $\mathcal{G}^k_{\mathsf{stat}}(\mathcal{A},\mathcal{B})$ is decidable in time $\mathcal{O}((|\mathcal{A}| \cdot |\mathcal{B}| \cdot (|\Sigma|^k + 1))^3)$.*

Properties. It is quite obvious to see that the 1-letter static game is the same as the fair simulation game. Thus we have $\sqsubseteq^1_{\mathsf{stat}} = \sqsubseteq_{\mathsf{fair}}$. Furthermore, the static games approximate language inclusion in the following sense.

Theorem 5. *For every $k \geq 1$ and all NBA $\mathcal{A}$, $\mathcal{B}$ we have: $\mathcal{A} \sqsubseteq^k_{\mathsf{stat}} \mathcal{B}$ implies $\mathcal{A} \sqsubseteq_{\mathsf{incl}} \mathcal{B}$.*

Thus, static multi-letter games look like a good alternative to the multi-pebble games for incrementally searching for a proof of language inclusion. Indeed, the size of these games, while still growing exponentially, grows "only" as $\mathcal{O}(|\Sigma|^k)$, hence much slower than the multi-pebble games for typical inputs in which $\Sigma \ll |\mathcal{B}|$ (see Prop. 3).

However, the static multi-letter games do not form a hierarchy, at least in the same sense as the k-pebble games do: for instance, for every $k \geq 2$, there are NBA $\mathcal{A}$, $\mathcal{B}$ such that $\mathcal{A} \sqsubseteq^k_{\mathsf{stat}} \mathcal{B}$ but $\mathcal{A} \not\sqsubseteq^{k+1}_{\mathsf{stat}} \mathcal{B}$. Indeed, the following two NBA $\mathcal{A}_k$ (left) and $\mathcal{B}_k$ (right) are such a counter-example.



Instead, the static multi-letter games form a lattice which is isomorphic to the one of naturals numbers under the division ordering.

Theorem 6. *For all $k, k' > 0$:*

$$\sqsubseteq^{\mathsf{gcd}(k,k')}_{\mathsf{stat}} \quad\subseteq\quad \sqsubseteq^k_{\mathsf{stat}} \cap \sqsubseteq^{k'}_{\mathsf{stat}} \quad\subseteq\quad \sqsubseteq^k_{\mathsf{stat}} \cup \sqsubseteq^{k'}_{\mathsf{stat}} \quad\subseteq\quad \sqsubseteq^{\mathsf{lcm}(k,k')}_{\mathsf{stat}} \quad .$$

For the purpose of an algorithm that solves static multi-letter games incrementally in k, this result tells that some sequences of values for k could introduce more incompleteness, like e.g. iterating over the powers of 2. Of course, increasing k by 1 in each step would alleviate this problem.

The lower complexity of the static multi-letter games with respect to the multi-pebble games comes at the price of being more incomplete.

Theorem 7. *The following holds.*

1. *For all $k \geq 1$, there are NBA $\mathcal{A}$, $\mathcal{B}$ such that $\mathcal{A} \sqsubseteq^2_{\mathsf{peb}} \mathcal{B}$ (and thus $\mathcal{A} \sqsubseteq_{\mathsf{incl}} \mathcal{B}$), but $\mathcal{A} \not\sqsubseteq^k_{\mathsf{stat}} \mathcal{B}$.*
2. *$\bigcup_{k\geq 1} \sqsubseteq^k_{\mathsf{stat}} \subsetneq \bigcup_{k \geq 1} \sqsubseteq^k_{\mathsf{peb}}$.*

Proof. (1) Consider NBA $\mathcal{A}'$, $\mathcal{B}'$ obtained by slightly modifying NBA $\mathcal{A}$, $\mathcal{B}$ from Ex. 1, by adding an a-loop in the initial state of $\mathcal{A}$ and $\mathcal{B}$. Whatever k is, Refuter can take the a-loop $k-1$ times and then the outgoing a-transition in $\mathcal{A}'$'s initial state. Verifier has to respond with a move that makes him/her commit to the following b or c. In the next round, Refuter can choose ca^{k-1} or ba^{k-1} and Verifier will be stuck. On the other hand, we have $\mathcal{A}' \sqsubseteq^2_{\mathsf{peb}} \mathcal{B}'$.

(2) Assume $\mathcal{A} \sqsubseteq^k_{\mathsf{stat}} \mathcal{B}$. Then Verifier has a winning strategy with $|\mathcal{B}|$ pebbles, since he can mimic a play of the static k-letters game by just pushing all pebbles

following all available choices, except every k rounds in which he only keeps live a single pebble on the state defined by his winning strategy for the k-letters game. $\square$

Another drawback of the static multi-letter games in the perspective of an incremental algorithm is that determining whether the search is hopeless is harder than for the multi-pebble games: the latter are ensured to become stationary, whereas for all $k \geq 1$, it holds that $|\mathcal{G}^k_{\mathsf{stat}}(\mathcal{A},\mathcal{B})| < |\mathcal{G}^{k+1}_{\mathsf{stat}}(\mathcal{A},\mathcal{B})|$.

4 The Dynamic Multi-Letter Games

In this section we consider a further refinement of the fair simulation game in which, again, Refuter is forced to reveal more information to Verifier. Moreover, Verifier is given additional power in this game which remedies the lack of a linear hierarchy for static games.

Definition. Let $k \geq 1$. The *dynamic k-letter game* is played between Refuter and Verifier on two NBA $\mathcal{A} = (Q_\mathcal{A}, q^\mathcal{A}_{\mathsf{init}}, \Delta_\mathcal{A}, F_\mathcal{A})$ and $\mathcal{B} = (Q_\mathcal{B}, q^\mathcal{B}_{\mathsf{init}}, \Delta_\mathcal{B}, F_\mathcal{B})$, similar to the static k-letter game. However, Verifier now can choose how far both players need to advance in each round.

Dynamic k-letter game $\mathcal{G}^k_{\mathsf{dyn}}(\mathcal{A},\mathcal{B})$	
start:	$q_0 := q^\mathcal{A}_{\mathsf{init}}$, $q'_0 := q^\mathcal{B}_{\mathsf{init}}$, $j := 0$
round i:	(1) Verifier chooses some h with $1 \leq h \leq k$ (2) Refuter chooses some $w = a_1 \dots a_h \in \Sigma^h$ (3) Refuter chooses $q_{j+1}, \dots, q_{j+h}$ such that $q_j \xrightarrow{a_1} q_{j+1} \xrightarrow{a_2} q_{j+2} \xrightarrow{a_3} \dots \xrightarrow{a_h} q_{j+h}$ (4) Verifier chooses $q'_{j+1}, \dots, q'_{j+h}$ such that $q'_j \xrightarrow{a_1} q'_{j+1} \xrightarrow{a_1} q'_{j+2} \xrightarrow{a_2} \dots \xrightarrow{a_h} q'_{j+h}$ (5) $j := j + h$
winner:	same as in fair simulation game

We write $\mathcal{A} \sqsubseteq^k_{\mathsf{dyn}} \mathcal{B}$ if Verifier has a winning strategy for $\mathcal{G}^k_{\mathsf{dyn}}(\mathcal{A},\mathcal{B})$.

The game $\mathcal{G}^k_{\mathsf{dyn}}(\mathcal{A},\mathcal{B})$ can be reduced—similar to the k-letter static game—to a parity game of size $\mathcal{O}(|\mathcal{A}| \cdot |\mathcal{B}| \cdot (|\Sigma|^{k+1} + k))$. In the vertices we keep track of the current state of $\mathcal{A}$, $\mathcal{B}$, the h chosen by Verifier, and the word $w = a_1 \dots a_h$ chosen by Refuter. We use the same priority assignment as in the parity game for the static k-letter game.

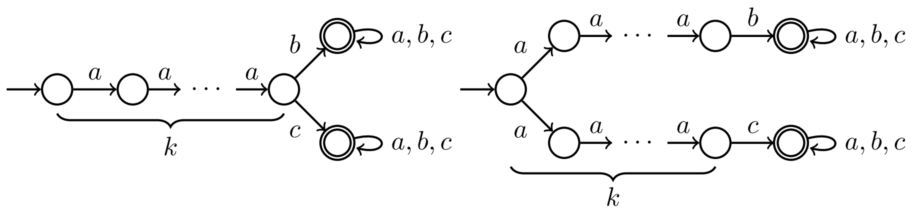
Theorem 8. *For every $k \geq 1$ and NBA $\mathcal{A}$ and $\mathcal{B}$, $\mathcal{A} \sqsubseteq^k_{\mathsf{dyn}} \mathcal{B}$ is decidable in time $\mathcal{O}((|\mathcal{A}| \cdot |\mathcal{B}| \cdot (|\Sigma|^{k+1} + k))^3)$.*

Properties. It is quite obvious to see that the 1-letter dynamic game is the same as the 1-letter static game, hence the same as the fair simulation game. Therefore we also have $\sqsubseteq^1_{\mathsf{dyn}} = \sqsubseteq_{\mathsf{fair}}$. As with the static games, the dynamic games approximate language inclusion. However, the dynamic games do form a hierarchy similar to the k-pebble games.

Theorem 9. *The following holds.*

$$\sqsubseteq^1_{\mathsf{dyn}} \subsetneq \sqsubseteq^2_{\mathsf{dyn}} \subsetneq \sqsubseteq^3_{\mathsf{dyn}} \subsetneq \cdots \subsetneq \bigcup_{k\geq 1} \sqsubseteq^k_{\mathsf{dyn}} \subsetneq \sqsubseteq_{\mathsf{incl}}$$

Proof. By definition it is clear that $\sqsubseteq^k_{\mathsf{dyn}} \subseteq \sqsubseteq^{k+1}_{\mathsf{dyn}}$ for every $k \geq 1$. However for every k, there are NBA $\mathcal{A}$, $\mathcal{B}$ with $\mathcal{A} \sqsubseteq^{k+1}_{\mathsf{dyn}} \mathcal{B}$ but $\mathcal{A} \not\sqsubseteq^k_{\mathsf{dyn}} \mathcal{B}$ as we consider the following two NBA $\mathcal{A}$ (left) and $\mathcal{B}$ (right).



□

As mentioned above, for $k = 1$, the static and dynamic games coincide. For larger parameters, the dynamic games are nonetheless more powerful than the static ones.

Theorem 10. *For $k \geq 2$, we have $\sqsubseteq^k_{\mathsf{stat}} \subsetneq \sqsubseteq^k_{\mathsf{dyn}}$.*

Proof. By definition, $\sqsubseteq^k_{\mathsf{stat}} \subseteq \sqsubseteq^k_{\mathsf{dyn}}$. To show $\sqsubseteq^k_{\mathsf{dyn}} \not\subseteq \sqsubseteq^k_{\mathsf{stat}}$ for $k \geq 3$, consider any NBA $\mathcal{A}_k$ and $\mathcal{B}_k$ for which $\mathcal{A}_{k-1} \not\sqsubseteq^k_{\mathsf{stat}} \mathcal{B}_{k-1}$ but $\mathcal{A}_{k-1} \sqsubseteq^{k-1}_{\mathsf{stat}} \mathcal{B}_{k-1}$ holds. By the inclusions $\sqsubseteq^{k-1}_{\mathsf{stat}} \subseteq \sqsubseteq^{k-1}_{\mathsf{dyn}} \subseteq \sqsubseteq^k_{\mathsf{dyn}}$, it holds that $\mathcal{A}_{k-1} \sqsubseteq^k_{\mathsf{dyn}} \mathcal{B}_{k-1}$, which ends the proof. The case $k = 2$ is similar. □

Despite being better approximations of language inclusion than static games, dynamic games suffer from the same drawbacks compared to multi-pebble games.

Theorem 11. *The following holds.*

1. *For all $k \geq 1$, there are NBA $\mathcal{A}$, $\mathcal{B}$ such that $\mathcal{A} \sqsubseteq^2_{\mathsf{peb}} \mathcal{B}$ (and thus $\mathcal{A} \sqsubseteq_{\mathsf{incl}} \mathcal{B}$), but $\mathcal{A} \not\sqsubseteq^k_{\mathsf{dyn}} \mathcal{B}$.*
2. *$\bigcup_{k\geq 1} \sqsubseteq^k_{\mathsf{dyn}} \subsetneq \bigcup_{k\geq 1} \sqsubseteq^k_{\mathsf{peb}}$.*

Although the sequence of games is not stationary, there is an upper bound on the indices k that are worth being tried while looking for a proof of language inclusion.

Theorem 12. *For all NBA $\mathcal{A},\mathcal{B}$, if there is $k \geq 0$ such that $\mathcal{A} \sqsubseteq^{k}_{\mathsf{dyn}} \mathcal{B}$, then $\mathcal{A} \sqsubseteq^{k_0}_{\mathsf{dyn}} \mathcal{B}$ for $k_0 := 2^{(|\mathcal{A}|+|\mathcal{B}|)^3}$.*

This result could be seen as the indication of a faster convergence of the multi-pebble games towards a fixpoint (since the corresponding upper bound for pebble games is "only" $|B|$). However, these upper bounds are often not tractable in practice, and should be considered with some care.

To conclude, it may be noticed that typical examples are such that $|\Sigma| \ll |\mathcal{B}|$. In this context, the size of dynamic games, like the ones for static games, is expected to grow much slower than those of multi-pebble games. Therefore, the next section compares these games from an experimental point of view.

5 Experiments

We implemented the three incremental inclusion tests using OCaml and the PGSolver library [8] with the recursive algorithm by Zielonka [19]. We evaluated our implementation by comparing it to Rabit [2], a recent Java implementation of optimized Ramsey-based methods (we used the options recommended by the authors, namely `-q -b -rd -fplus -SFS -qr -c -l`). We ran experiments on a machine with 16 Intel Xeon cores at 1.87GH, including experiments with Rabit (note however that Java code naturally tends to run slower than OCaml code). Due to the possible divergence of our algorithms, we fixed a timeout of 1 hour for every incremental inclusion test and every pair of automata. The results as well as the OCaml code are available online.[1]

For our experiments, we used the same benchmarks over which Rabit was tested[2], restricting to pairs of automata $\mathcal{A},\mathcal{B}$ for which language inclusion can be expected (namely Rabit does not report that $\mathcal{A} \not\sqsubseteq_{\mathsf{incl}} \mathcal{B}$). Rabit benchmarks were obtained from (1) several mutual exclusion protocols with possible errors injected into the code, and (2) the Tabakov-Vardi random model [16], parametrized by the number of states of the automata, the transition density (d_{tr}), and the acceptance density (d_{acc}).

The results for the benchmarks on protocol verification are summarised in Figure 1. The performances of all algorithms are unsurprisingly very similar for the cases where fair simulation holds (k=1). For the cases where fair simulation does not hold, Rabit tends to be the better tool. An exception is BakeryV2, for which Rabit times out, whereas multi-letter simulations perform quite well.

The Tabakov-Vardi benchmarks consist of (1) 4000 pairs of automata with 15 states, d_{tr} ranging from 1.5 to 3, and d_{acc} from 0.1 to 1.0, (2) 100 pairs of automata with 30 states each, $d_{\mathsf{tr}} = 2$, $d_{\mathsf{acc}} = 0.1$, and (3) 100 pairs of automata with 50 states each, $d_{\mathsf{tr}} = 3$, $d_{\mathsf{acc}} = 0.6$. For these parameters one can expect a substantial amount of positive instances for the language inclusion problem, so-called "hard" inclusion in case (2) and easy but non-trivial inclusion in case (3) [2]. We call a method successful on a pair of automata if it can show language

[1] see http://carrick.fmv.informatik.uni-kassel.de/~milka/iit

[2] see http://languageinclusion.org/CONCUR2011

mutual exclusion protocols							
	multi-letter				multi-pebble		Rabit
	k	dynamic	k	static	k	pebble	
Mcs	1	22.94s	1	23.48s	1	25.41s	39.00s
FischerV2	1	0.07s	1	0.07s	1	0.07s	0.09s
Peterson	1	0.01s	1	0.01s	1	0.01s	0.03s
Bakery	1	7.22s	1	7.16s	1	7.23s	4.43s
Phils	1	0.02s	1	0.02s	1	0.02s	0.11s
Fischer	1	40.80s	1	47.30s	1	47.37s	3.41s
FischerV3	-	>1h	-	>1h	-	>1h	7.63s
FischerV4	-	>1h	-	>1h	-	>1h	2136.70s
BakeryV2	2	36.01s	2	7.06s	-	>1h	>1h

random NBA								
	size = 30, $d_{\mathsf{acc}} = 0.1$, $d_{\mathsf{tr}} = 2$				size = 50, $d_{\mathsf{acc}} = 0.6$, $d_{\mathsf{tr}} = 3$			
	dynamic	static	pebble	Rabit	dynamic	static	pebble	Rabit
time	59.12s	52.78s	18.22s	1655.84s	0.55s	0.52s	2.09s	127.53s
success %	80%	82%	86%	58%	100%	100%	100%	100%
average k	3.66	4.33	1.93	-	1.01	1.01	1.01	-

Fig. 1. Experimental comparisons between multi-letter simulations and other methods.

inclusion before the timeout, and for fixed size, d_{acc}, and d_{tr}, its percentage of success is the rate of successful pairs of automata over all pairs for which one may expect language inclusion. The results for sizes 30 and 50 are given in Figure 1, too. Any incremental inclusion test is almost always two orders of magnitudes faster than Rabit. Incremental inclusion tests are also more successful in general, although the percentage of success of multi-letter games may fall to $\sim 50\%$ for automata of size 15 with timeout 10 minutes (see detailed benchmarks). The pebble game typically explores smaller values of k only. It heavily depends on the size of the automata, in contrast to the multi-letter game.

The experiments demonstrate that incremental inclusion testing is a very reasonable heuristic for proving language inclusion. This heuristic, when it succeeds, is still comparable to the recently optimised Ramsey-based approach, and may perform well when simulation does not hold ($k > 1$).

In order to tackle the lack of completenes, it is tempting to try to combine incremental inclusion and the Ramsey-based approach. It remains to be seen whether or not the results of the multi-letter simulation tests can be used for quotienting. We aim to develop a combination of the two approaches in the future.

References

1. Abdulla, P.A., Chen, Y.F., Clemente, L., Holík, L., Hong, C.D., Mayr, R., Vojnar, T.: Simulation subsumption in ramsey-based Büchi automata universality and in-

clusion testing. In: Touili, T., Cook, B., Jackson, P. (eds.) CAV. Lecture Notes in Computer Science, vol. 6174, pp. 132–147. Springer (2010)
2. Abdulla, P.A., Chen, Y.F., Clemente, L., Holík, L., Hong, C.D., Mayr, R., Vojnar, T.: Advanced ramsey-based Büchi automata inclusion testing. In: Katoen, J.P., König, B. (eds.) CONCUR. Lecture Notes in Computer Science, vol. 6901, pp. 187–202. Springer (2011)
3. Clemente, L., Mayr, R.: Multipebble simulations for alternating automata - (extended abstract). In: Gastin, P., Laroussinie, F. (eds.) CONCUR. Lecture Notes in Computer Science, vol. 6269, pp. 297–312. Springer (2010)
4. Dill, D.L., Hu, A.J., Wong-Toi, H.: Checking for language inclusion using simulation preorders. In: Larsen, K.G., Skou, A. (eds.) CAV. Lecture Notes in Computer Science, vol. 575, pp. 255–265. Springer (1991)
5. Doyen, L., Raskin, J.F.: Antichains for the automata-based approach to model-checking. Logical Methods in Computer Science 5(1) (2009)
6. Etessami, K.: A hierarchy of polynomial-time computable simulations for automata. In: Brim, L., Jancar, P., Kretínský, M., Kucera, A. (eds.) CONCUR. Lecture Notes in Computer Science, vol. 2421, pp. 131–144. Springer (2002)
7. Fogarty, S., Vardi, M.Y.: Efficient Büchi universality checking. In: Proc. 16th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'10. LNCS, vol. 6015, pp. 205–220. Springer (2010)
8. Friedmann, O., Lange, M.: Solving parity games in practice. In: Liu, Z., Ravn, A.P. (eds.) ATVA. Lecture Notes in Computer Science, vol. 5799, pp. 182–196. Springer (2009)
9. Grädel, E., Thomas, W., Wilke, T. (eds.): Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001], Lecture Notes in Computer Science, vol. 2500. Springer (2002)
10. Henzinger, T.A., Kupferman, O., Rajamani, S.K.: Fair simulation. Inf. Comput. 173(1), 64–81 (2002)
11. Jurdzinski, M.: Small progress measures for solving parity games. In: Proceedings of the 17th Annual Symposium on Theoretical Aspects of Computer Science. pp. 290–301. STACS '00, Springer-Verlag, London, UK, UK (2000), http://dl.acm.org/citation.cfm?id=646514.695804
12. Kupferman, O., Vardi, M.Y.: Weak alternating automata are not that weak. ACM Transactions on Computational Logic 2(3), 408–429 (2001)
13. Miyano, S., Hayashi, T.: Alternating finite automata on omega-words. Theor. Comput. Sci. 32, 321–330 (1984)
14. Safra, S.: On the complexity of ω-automata. In: Proc. 29th Symp. on Foundations of Computer Science, FOCS'88. pp. 319–327. IEEE (1988)
15. Savitch, W.J.: Relationships between nondeterministic and deterministic tape complexities. Journal of Computer and System Sciences 4, 177–192 (1970)
16. Tabakov, D., Vardi, M.Y.: Experimental evaluation of classical automata constructions. In: Sutcliffe, G., Voronkov, A. (eds.) LPAR. Lecture Notes in Computer Science, vol. 3835, pp. 396–411. Springer (2005)
17. Thomas, W.: Complementation of Büchi automata revisited. In: et al., J.K. (ed.) Jewels are Forever, Contributions on Theoretical Computer Science in Honor of Arto Salomaa, pp. 109–122. Springer (1999)
18. Vardi, M.Y., Wolper, P.: An automata-theoretic approach to automatic program verification (preliminary report). In: Proc. 1st Symp. on Logic in Computer Science, LICS'86, pp. 332–344. IEEE, Washington, DC (1986)
19. Zielonka, W.: Infinite games on finitely coloured graphs with applications to automata on infinite trees. TCS 200(1–2), 135–183 (1998)