Weakly Continuation Closed Homomorphisms on Automata*

Thierry Nicola and Ulrich Ultes-Nitsche

Department of Computer Science, University of Fribourg
Boulevard de Pérolles 90, CH-1700 Fribourg, Switzerland

Abstract. A major limitation of system and program verification is the state space explosion problem. To avoid this problem, there exists several approaches to reduce the size of the system state space. Some methods try to keep the state space small during the verification run, other methods reduce the original state space prior to the verification. One of the later are abstraction homomorphisms. *Weakly Continuation Closed* homomorphisms are abstraction homomorphisms preserving exactly those properties of the original behaviour which are satisfied inherently fair [1]. However, the practical use of WCC homomorphisms is limited by the lack of a reasonably efficient algorithm, checking whether or not a homomorphism is WCC and performing reasonably well. This paper presents a result which allows to develop algorithms for WCC on automata.

1 Introduction

There exists several methods to tackle the state space explosion problem for system verification [2–4]. An optimal solution is to use combinations of known methods. Abstraction homomorphisms try to reduce the original system behaviour state space prior to the verification. As the abstract system model is used for the verification, the abstract model should satisfy exactly those properties the original system model satisfies. Weakly Continuation Closed(WCC) homomorphisms preserve exactly those properties which are satisfied inherently fair [1]. This means the conclusion whether or not the original system satisfies a property follows from the verification of the property on the abstract model.

A problem of abstraction homomorphisms is that the complete state space has to be computed before the abstraction homomorphism can be applied. This leads to a dilemma, it is not possible to apply the abstraction onto the original state space due to its vast state space size. This problem is resolved for WCC homomorphisms, as these abstraction homomorphisms work as well on abstract-compatible trace reductions [1].

The motivation for using WCC homomorphims for model checking is two-folded. Firstly, WCC homomorphims preserve the original properties on the abstract model and secondly, WCC homomorphims can be applied on trace reductions of the original system.

* Supported by the Swiss National Science Foundation under grant # 200021-103985/1

Nicola T. and Ultes-Nitsche U. (2008).
Weakly Continuation Closed Homomorphisms on Automata.
In *Proceedings of the 6th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems*, pages 39-48
DOI: 10.5220/0001736500390048

We consider system behaviours which are represented by a deterministic Büchi automaton. The abstraction homomorphisms are applied onto the original system automata. The result is again a deterministic Büchi Automaton, representing the abstract system behaviour.

The goal of this paper is to present a new starting point for the development of more efficient algorithms for weakly continuation closed homomorphisms.

This paper is structured as follows. After some preliminary definitions, we explain how homomorphisms are applied onto automata. In section 4 the main results of this paper is established. Section 5 gives a rough outline of what a possible algorithm might look like, Section 6 gives an example and Section 7 concludes the paper.

2 Preliminaries

We assume the reader is familiar with the common notions of formal language, ω-languages and automata theory as presented in [5, 6].

For an alphabet Σ, let Σ^* be the set of all finitely long sequences on Σ, let Σ^ω be the set of all infinitely long sequences, and, let $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$. A set $L \subseteq \Sigma^*$ is called a (finitary) language and a set $M \subseteq \Sigma^\omega$ is called an ω-language.

Let L be language. Then the set of all finite prefixes of L is $pre(L) = \{w \in \Sigma^* | \exists x \in \Sigma^\infty : wx \in L\}$. The continuation of a prefix $w \in pre(L)$ in the language L is $cont(w, L) = \{x \in \Sigma^\infty | wx \in L\}$.

The *behaviour* of a system is a set of sequences of actions. Such a sequence is then called a behaviour. It is therefore justified to say that system behaviours are languages. We consider systems which do not terminate and therefore the sequences are infinitely long. We will use Büchi Automaton for representing these system behaviours.

Let $\mathscr{A} = (Q, \Sigma, \delta, q_0, F)$ be a Büchi automaton, where Q is a finite set of states, Σ is an alphabet, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is a set of accepting states and $\delta : Q \times \Sigma \to 2^Q$ is a transition relation. If $p \in \delta(q, a)$ then we call (q, a, p) a transition from state q to state p with symbol a. For the rest of this paper, we assume the transition relation to be extended to $\delta : 2^Q \times \Sigma^* \to 2^Q$ in the usual way.

Let $w = a_1 a_2 \ldots \in \Sigma^\omega$ be an infinite sequence. A run of the automaton $\mathscr{A}$ on w is a sequence of states $\rho(w) = s_0 s_1 \ldots$, such that $s_{i+1} \in \delta(s_i, a_{i+1})$ for $i \geq 0$. A run is called successful if $s_0 = q_0$ and $inf(\rho(w)) \cap F \neq \emptyset$ where $inf(\rho(w))$ is the set of states that occur infinitely often in $\rho(w)$. The automaton $\mathscr{A}$ Büchi-accepts w if and only there is a successful run of $\mathscr{A}$ on w. The infinite-string language accepted by Büchi automaton $\mathscr{A}$ is $L(\mathscr{A}) = \{w \in \Sigma^\omega | \mathscr{A}$ Büchi-accepts $w\}$.

A state $q \in Q$ of automaton $\mathscr{A}$ is called *reachable* if there exists a path from the initial state q_0 to q. A state $q \in Q$ of $\mathscr{A}$ is called *co-reachable* if there exists a run $\rho(w) = q_1 q_2 \ldots$ of $\mathscr{A}$ on an infinite sequence $w \in \Sigma^\omega$ such that $q_1 = q$ and $inf(\rho(w)) \cap F \neq \emptyset$. Let $q \in Q$, the set $r(q, \mathscr{A}) \subseteq Q$ is the set of all reachable states from the state q in the automaton $\mathscr{A}$. We assume that each state is reachable from itself (at least with the empty string), so for all state $q \in Q$, we have that $q \in r(q, \mathscr{A})$. This notation is extended to $r(\{q_1, \ldots, q_n\}, \mathscr{A}) = r(q_1, \mathscr{A}) \cup \ldots \cup r(q_n, \mathscr{A})$.

Let $S \subseteq Q$ be a *Strongly Connected Bottom Component*(SCBC). A SCBC is a set of states such that from each state within S there exists a path to every other state within

S. Further no state within S has a path to a state p such that $p \notin S$ and we require that S is maximal, no more states can be added to S without destroying the property of connectivity of S.

Let $q \in Q$ be a state of the automaton $\mathscr{A}$, we denote by $c(q, \mathscr{A})$ the language accepted by the automaton $\mathscr{A}_q = (Q, \Sigma, \delta, q, F)$.(The automaton $\mathscr{A}$ with q as initial state) It holds that $c(q_0, \mathscr{A}) = L(\mathscr{A})$. Again, we extend the notation to applied to sets of states, let $\{q_1, \ldots, q_n\} \subseteq Q$, then we define the set $c(\{q_1, \ldots, q_n\}, \mathscr{A}) = c(q_1, \mathscr{A}) \cup \ldots \cup c(q_n, \mathscr{A})$.

Let $q \in Q$ be a state from automaton $\mathscr{A}$ and let $b \in pre(L(\mathscr{A}))$ be a prefix leading to state q, in general we have that $c(q, \mathscr{A}) \subseteq cont(b, L(\mathscr{A}))$. If the automaton $\mathscr{A}$ is deterministic then it holds that $cont(b, L(\mathscr{A})) = c(q, \mathscr{A})$, as each prefix b leads to exactly one state q. $\delta(b, q_0)$ is the set of states reached by the prefix b in $\mathscr{A}$, then we have that $c(\delta(b, q_0), \mathscr{A}) = cont(b, L(\mathscr{A}))$

Throughout this paper the automaton $\mathscr{B} = (Q_B, \Sigma, \delta_B, q_B, F_B)$ represents the original system behaviour.[1] We will consider system behaviours where every state of the automaton is accepting, $F_B = Q_B$. It can therefore be assumed that each automaton representing a system behaviour is deterministic, because each Büchi Automaton $\mathscr{A} = (Q, \Sigma, \delta, q_0, Q)$ can be transformed into a deterministic Büchi automaton accepting the same language [7]. Further, we require that each state is reachable and co-reachable, every automaton has no useless states.

Let h be a homomorphism on alphabet Σ, and $w = a_1 a_2 \ldots a_n$ a sequence of symbols from Σ, then we have that $h(w) = h(a_1)h(a_2)\ldots h(a_n)$. An homomorphism h can be applied on a language L, by applying h to each string $w \in L$, $h(L) = \{h(w) | w \in L\}$.

A special class of homomorphisms are *abstraction homomorphism*. Abstraction Homomorphisms are defined as follows:

Definition 1. *$h : \Sigma^\infty \to \Sigma'^\infty$ is an abstraction homomorphism if and only if the following conditions hold:*

- $h(\Sigma) \subseteq \Sigma' \cup \{\epsilon\}$,
- $\forall v, w \in \Sigma^*, x \in \Sigma^\omega : h(vw) = h(v)h(w)$ *and* $h(vx) = h(v)h(x)$,
- $h(\Sigma^\omega) \subseteq \Sigma'^\omega$

Abstraction homomorphisms are partial mappings since they are not defined for ω-words that would be taken to finitely long words, and that they do not increase the length of the words, i.e. $|h(w)| \leq |w|$.

Weakly Continuation Closed homomorphisms are abstraction homomorphisms. In general abstraction homomorphisms do not preserve the properties of the original system. Weakly Continuation Closed homomorphism have been shown to preserve exactly those properties which are satisfied inherently fair [8]. Regarding a model checking algorithm, weakly continuation closed homomorphisms work best with inherently fair linear time verification(IFLTV) [9] and together they provide a promising tool for model checking purposes when fairness [10] is enabled.

The definition of weakly continuation closed homomorphisms is as follows

[1] By referring to the original behaviour, it might also be the system behaviour's trace reduction.

Definition 2. *The homomorphism h is weakly continuation closed on a language L if and only if, for all $w \in \Sigma^*$, there exists $v \in cont(h(w), h(L))$ such that $cont(v, cont(h(w), h(L))) = cont(v, h(cont(w, L)))$.*

In general, we know that $h(cont(v, L)) \subseteq cont(h(w), h(L))$ holds, which implies that $cont(v, h(cont(w, L))) \subseteq cont(v, cont(h(w), h(L)))$ for all $v \in cont(h(w), h(L))$ always holds.

3 Homomorphisms on Automata

Let $\mathscr{B} = (Q_B, \Sigma, \delta_B, q_B, Q_B)$ be an automaton representing a system behaviour and let $h : \Sigma \to \Sigma'$ be an abstraction homomorphism. Applying the homomorphism h onto the automaton $\mathscr{B}$, results in a new automaton. We will refer to this resulting automaton by $\mathscr{H} = (Q_H, \Sigma', \delta_H, q_H, Q_H)$.

The homomorphism h applied on the automaton $\mathscr{B}$, translates the automaton's alphabet Σ to another alphabet Σ'. The homomorphism changes the symbol $a \in \Sigma$ of the transition (q, a, p) of automaton $\mathscr{B}$ to a new symbol $\alpha \in \Sigma'$, $h(a) = \alpha$. The transition becomes (q, α, p) in the automaton $\mathscr{H}$. The automaton $\mathscr{H}$ is identical to the automaton $\mathscr{B}$, except for the transition symbols and the alphabet used. In other words the state set, the initial state and the set of accepting states are equal for both automata, $Q_H = Q_B$ and $q_H = q_B$. The transition relation δ_H contains the transition (q, α, p) if and only if the transition (q, a, p) is in the original automaton's transition relation δ_B and $h(a) = \alpha$.
In general, the automaton $\mathscr{H}$ might be non-deterministic (and might have ϵ-transitions). The automaton $\mathscr{H}$ is therefore not suited as an automaton representing a system behaviour, as automata representing system behaviours are deterministic. To obtain an automaton representing a system behaviour, $\mathscr{H}$ needs to be determinized. By applying the powerset construction the resulting deterministic automaton accepts the same language as the automaton $\mathscr{H}$, as all state of $\mathscr{H}$ are accepting [7]. We will refer to the resulting automaton by $\mathscr{D} = (Q_D, \Sigma', \Delta, q_D, Q_D)$.

As we will work on these automata, it is important to specify certain observations about the relation between these automata. A first observation is that $h(L(\mathscr{B})) = L(\mathscr{H}) = L(\mathscr{D})$. This guarantees that the final resulting automaton $\mathscr{D}$, corresponds again to a system behaviour, as it is deterministic, has only accepting states and is the abstracted automaton to $\mathscr{B}$, i.e. $h(L(\mathscr{B})) = L(\mathscr{D})$.

For the rest of this paper, let $\mathscr{B} = (Q_B, \Sigma, \delta_B, q_B, Q_B)$ be the automaton representing the original system behaviour, refered to as the original automaton, and let $\mathscr{H} = (Q_H, \Sigma, \delta_H, q_H, Q_H)$ and $\mathscr{D} = (Q_D, \Sigma', \Delta, q_D, Q_D)$ be the automaton as described above.

4 Weakly Continuation Closed Homomorphisms on Automata

In Defintion 2, weakly continuation closed homomorphisms are defined for languages. The definition has to be slightly modified for automata. Let $\mathscr{A} = (Q, \Sigma, \delta, q, F)$ be a Büchi automaton then we define WCC homomorphisms on automata as follows

Definition 3. *The homomorphism h is weakly continuation closed on $\mathscr{A}$ if and only if h is weakly continuation closed on $L(\mathscr{A})$.*

We say that a homomorphism is WCC on the automaton $\mathscr{A}$ if and only if h is WCC on the language accepted by this automaton, i.e. $L(\mathscr{A})$.

This definition's naïve algorithm, would need to verify all prefixes of $L(\mathscr{A})$. By verifying prefixes, we understand taking a prefix w and comparing it to all $v \in cont(h(w), h(L(\mathscr{A})))$, whether or not $cont(v, cont(h(w), h(L(\mathscr{A})))) = cont(v, h(cont(w, L(\mathscr{A}))))$ holds. But as we are working with ω-languages and there are infinitely many prefixes, this procedure cannot be regarded as efficient. At best it is possible to reduce the verification to the number of states in $\mathscr{A}$. Our approach reduces the number of such verification and at the same time tries to reduce the amount of possible continuations to choose from, the number of v's.

The first lemma makes the switch from prefixes to states. This step decreases the verification steps from an infinite amount of prefixes to a finite amount of states. But this number may still be too large for an efficient practical application.

Lemma 1. *The homomorphism h is weakly continuation closed on $\mathscr{B}$ if and only if for all $q_i \in Q_B$ and for all $q \in Q_D$, where $q_i \in q$, there exists $\alpha \in c(q, \mathscr{D})$ such that $cont(\alpha, c(q, \mathscr{D}) = cont(\alpha, h(c(q_i, \mathscr{B})))$.*

Proof. '$\Rightarrow$': Let's assume h is weakly continuation closed on $\mathscr{B}$. This means, for all $b \in pre(L(\mathscr{B}))$, there exists $\alpha \in c(h(b), L(\mathscr{D}))$ such that $cont(\alpha, cont(h(b),\ L(\ \mathscr{D}))) = cont(\alpha, h(cont(b, L(\mathscr{B}))))$.

As $\mathscr{B}$ is deterministic, we know that we can replace $cont(b, L(\mathscr{B}))$ by $c(q_i, \mathscr{B})$, where $q_i \in Q_B$ is the state reached with prefix b in $\mathscr{B}$, and, as $\mathscr{D}$ is deterministic, we can replace $cont(h(b), L(\mathscr{D}))$ by $c(q, \mathscr{D})$, where $q \in Q_D$ is the state reached with prefix $h(b) \in pre(L(\mathscr{D}))$ in $\mathscr{D}$.

This holds for all $b \in pre(L(\mathscr{B}))$ and $cont(b, L(\mathscr{B})) = c(q_i, \mathscr{B})$, where q_i is the state reached by b, it follows that it holds for all states which are reachable. In $\mathscr{D}$, $h(b)$ then leads to a state q such that $q_i \in q$. Therefore, for a state $q_i \in Q_B$, to get all all the corresponing $h(b)$, one needs all states $q \in Q_D$, such that $q_i \in q$. This then finally makes

For all $q_i \in Q_B$, and for all $q \in Q_D$, where $q_i \in q$, there exists $\alpha \in c(q, \mathscr{D})$ such that $cont(\alpha, c(q, \mathscr{D})) = cont(\alpha, h(c(q_i, \mathscr{B})))$.

'$\Leftarrow$': Let's assume h is not WCC on $\mathscr{B}$. This implies that there exists $b \in pre(L(\mathscr{B}))$, such that for all $\alpha \in cont(h(b), L(\mathscr{D}))$ we have that $cont(\alpha, cont(h(b),\ L(\mathscr{D}))) \neq cont(\alpha, h(cont(b, L(\mathscr{B}))))$.

Let $q_i \in Q_B$ be the state in $\mathscr{B}$ reached with the prefix b and let $q \in Q_D$ be the state reached with $h(b)$ in $\mathscr{D}$. In this case we know that $q_i \in q$. For all $\alpha \in c(q, \mathscr{D})$, we get $cont(\alpha, c(q, \mathscr{D})) \neq cont(\alpha, h(c(q_i, \mathscr{B})))$.

This is a contradiction, therefore h is weakly continuation closed on $\mathscr{B}$ if for all $q_i \in Q_B$ and for all $q \in Q_D$, where $q_i \in q$, there exists $\alpha \in c(q, \mathscr{D}))$ such that $cont(\alpha, c(q, \mathscr{D})) = cont(\alpha, h(c(q_i, \mathscr{B})))$. □

The following lemma reduces the number of states from the original automaton which needs to be verified to the number of Strongly Connected Bottom Components.

Lemma 2. *For all $q_i \in Q_B$ and for all $q \in Q_D$, where $q_i \in q$, there exists $\alpha \in c(q, \mathscr{D})$ such that $cont(\alpha, c(q, \mathscr{D}) = cont(\alpha, h(c(q_i, \mathscr{B})))$ if and only if for all* Strongly Connected Bottom Components *$S \subseteq Q_B$ of $\mathscr{B}$, there exists $p_s \in S$, and for all states $p \in Q_D$ where $p_s \in p$ there exists $\beta \in c(p, \mathscr{D})$ such that*

$$cont(\beta, c(p, \mathscr{D})) = cont(\beta, h(c(p_s, \mathscr{B})))$$

Proof. '$\Rightarrow$': Let's assume that for all $q_i \in Q_B$ and for all $q \in Q_B$ there exists an $\alpha \in c(q, \mathscr{D})$ such that $cont(\alpha, c(q, \mathscr{D})) = cont(\alpha, h(c(q_i, \mathscr{B})))$.

This implies, as the above holds for all states $q_i \in Q_B$, it holds for all $q_i \in Q_S$, as $Q_S \subseteq Q_B$, where Q_S contains all states within SCBC. Therefore it holds for all states within a SCBC, which implies that for each SCBC S there exists a state $q_S \in S$ such that the hypothesis holds.

'$\Leftarrow$': Let's assume there exists a state $q_i \in Q_B$ and a state $q \in Q_D$, where $q_i \in q$, such that for all $\alpha \in c(q, \mathscr{D})$ we have $cont(\alpha, c(q, \mathscr{D})) \neq cont(\alpha, h(c(q_i, \mathscr{B})))$. This implies that we have $cont(\alpha, h(c(q_i, \mathscr{B}))) \subset cont(\alpha, c(q, \mathscr{D}))$.

Let $x_i \in r(q_i, \mathscr{B})$, be the state reached by $a \in c(q_i, \mathscr{B})$. This makes then that $h(c(x_i, \mathscr{B})) \subseteq cont(h(a), h(c(c_i, \mathscr{B})))$, as we have $cont(a, c(q_i, \mathscr{B})) = c(x_i, \mathscr{B})$ and, in general, $h(cont(a, \mathscr{B})) \subseteq cont(h(a), h(c(c_i, \mathscr{B})))$.

We know that $cont(\alpha, h(c(q_i, \mathscr{B}))) \subseteq cont(\alpha, c(q, \mathscr{D}))$, we get $h(c(x_i, \mathscr{B})) \subset cont(\alpha, c(q, \mathscr{D}))$, where $\alpha = h(a)$. The sequence α leads from state $q \in Q_D$, to a state $x \in r(q, \mathscr{D})$, where $x_i \in x$ and $c(x, \mathscr{D}) = cont(\alpha, c(q, \mathscr{D}))$.

Finally, there exists $q_i \in Q_B$ and $q \in Q_D$, where $q_i \in q$ such that for all $x_i \in r(q_i, \mathscr{B})$ and for all $x \in r(q, \mathscr{D})$, where $x_i \in x$, we have that $h(c(x_i, \mathscr{B})) \subset c(x, \mathscr{D})$, and therefore $h(c(x_i, \mathscr{B})) \neq c(x, \mathscr{D})$.

This implies, there exists $x_s \in r(q_i, \mathscr{B})$ and $x \in r(q, \mathscr{D})$, where $x_s \in x$ and x_s is a state within a Strongly Connected Bottom Component S, such that $h(c(x_s, \mathscr{B})) \neq c(x, \mathscr{D})$.

By repeating the same procedure for x_s as before for state q_i, we find that there exists a Strongly Connected Bottom Component S, where for all states $q_s \in S$ and $q \in Q_D$, $q_s \in q$ such that for all $\alpha \in c(q, \mathscr{D})$, we have that $cont(\alpha, c(q, \mathscr{D}) \neq cont(\alpha, h(c(q_s, \mathscr{B})))$.

This is a contradiction therefore it follows that h is weakly continuation closed on $\mathscr{B}$ if for all *Strongly Connected Bottom Component* S of $\mathscr{B}$, there exists a $q_s \in S$, and for all states $q \in Q_D$ where $q_s \in q$ it holds that $\exists \alpha \in c(q, \mathscr{D})$ such that $cont(\alpha, c(q, \mathscr{D})) = cont(\alpha, c(q_s, \mathscr{B}_h))$. □

Combining Lemma 1 with Lemma 2 leads us to the main result of this paper, namely Theorem 1. The theorem states that a homomorphism h is weakly continuation closed on the automaton $\mathscr{B}$ if and only if for all SCBCs, there exists a state q_i within this component such that for all occurences of q_i within a macrostate q of $\mathscr{D}$, there exists a continuation $\alpha \in c(q, \mathscr{D})$ such that eventually $cont(\alpha, c(q, \mathscr{D})) = cont(\alpha, h(c(q_s, \mathscr{B})))$, note that $h(c(q_s, \mathscr{B})) = c(q_s, \mathscr{H})$.

Theorem 1. *The homomorphism h is weakly continuation closed on $\mathscr{B}$ if and only if for all Strongly Connected Bottom Component $S \subseteq Q_B$, there exists $q_s \in S$ and for all $q \in Q_D$, where $q_s \in q$, there exists $\alpha \in c(q, \mathscr{D})$ such that*

$$cont(\alpha, c(q, \mathscr{D})) = cont(\alpha, h(c(q_s, \mathscr{B})))$$

Proof. The proof of the theorem is immediate with Lemma 1 and Lemma 2.

5 Draft of an Algorithm for WCC Homomorphisms

Theorem 1 reduces the number of states that needs to be verified to the number of Strongly Connected Bottom Components within the original automaton $\mathscr{B}$. This section sketches an algorithm implementing the above result.

By applying the homomorphism h, the automaton $\mathscr{B}$ is first transformed into the automaton $\mathscr{H}$, by changing the transition symbols according to the homomorphism h. Then in a second step, by determinizing the automaton $\mathscr{H}$, the result will be the automaton $\mathscr{D}$ describing the abstract system behaviour. The check whether h is WCC or not, is done in a third step.

For each SCBC S of the original automaton $\mathscr{B}$, we select one state $q_s \in S$ and identify all macrostates q of $\mathscr{D}$ where $q_s \in q$. The next step is to gain access to the prefixes of $c(q_s, \mathscr{H})$, the SCBC S is regarded as an automaton with q_s as initial state. In general this automaton is non-deterministic. A determinization of this automaton results in an automaton called $\mathscr{S}$ and it holds that $L(\mathscr{S}) = c(q_s, \mathscr{H})$.

This means by taking state r of $\mathscr{S}$ and α the sequence leading to state r in $\mathscr{S}$, it holds that $\alpha \in c(q_s, \mathscr{H})$. Then $\alpha \in c(q, \mathscr{D})$, for all q where $q_s \in q$ and let α lead from q to state t in $\mathscr{D}$. Then the following lemma follows

Lemma 3. *for all Strongly Connected Bottom Component $S \subseteq Q_B$, there exists $q_s \in S$ and for all $q \in Q_D$, where $q_s \in q$, there exists $\alpha \in c(q, \mathscr{D})$ such that $cont(\alpha, c(q, \mathscr{D})) = cont(\alpha, h(c(q_s, \mathscr{B})))$ if and only if $c(r, \mathscr{S}) = c(t, \mathscr{D})$, where state r is reached by α in $\mathscr{S}$ and α leads from state q to state t in $\mathscr{D}$.*

Proof. As both $\mathscr{S}$ and $\mathscr{D}$ are deterministic,$cont(\alpha, c(q_s, \mathscr{H})) = c(r, \mathscr{S})$ and $cont(\alpha, c(q, \mathscr{D})) = c(t, \mathscr{D})$, because α leads from state q_s to r in $\mathscr{H}$ and from q to t in $\mathscr{D}$. □

By reducing the number of possible states t in $\mathscr{D}$, the number of necessary language comparisons, $c(r, \mathscr{S}) = c(t, \mathscr{D})$, is reduced too. Such that $c(r, \mathscr{S}) = c(t, \mathscr{D})$ might hold, it must be true that $r \subseteq t$, where r a state of $\mathscr{S}$. This fact must hold as we know that $\mathscr{D}$ is the determinized automaton of $\mathscr{H}$ and that $q_s \in q$ and α leads from q_s to state r in $\mathscr{S}$[2] and α leads from q to t in $\mathscr{D}$.

For checking whether or not $c(t, \mathscr{D}) = c(r, \mathscr{S})$, we will rely on standard algorithms, as language comparison is a quite common task for automata. In fact we only need to verify whether or not $c(t, \mathscr{D})) \subseteq c(r, \mathscr{S})$, as we know that $r \subseteq t$, it always holds that $c(r, \mathscr{S})) \subseteq c(t, \mathscr{D})$.

6 Example

Let $\mathscr{B}$ be the automaton in Figure 1a. Every state is an accepting state, therefore we omit the usual accepting conditions of the states in the figures. The homomorphism

[2] Which means α leads from state q_s to states r in $\mathscr{H}$.

h hides[3] all the symbols except req, rej, res, which are mapped to their identity. By applying the homomorphism h to $\mathscr{B}$, the final result will then be the automaton $\mathscr{D}$, fig 1b.

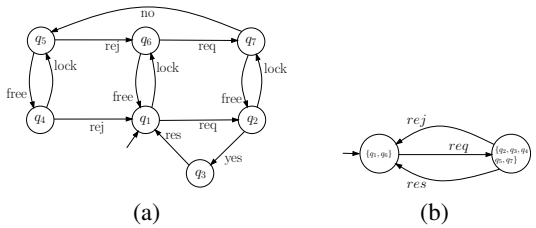

Fig. 1. The automaton $\mathscr{B}$ representing the original system behaviour (a) and the automaton $\mathscr{D}$ representing the abstracted system behaviour (b) reduced by homomorphism h.

To verify that h is weakly continuation closed on $\mathscr{B}$, we will apply the presented procedure. The automaton $\mathscr{B}$ consists of exactly one SCBC. So we select one state of $\mathscr{B}$ and identify all occurences of this state within a macrostate in $\mathscr{D}$. As the automaton $\mathscr{B}$ is a SCBC, we can select any state of it. We choose state q_1 and there is exactly one occurence of the state q_1 within macrostate $\{q_1, q_6\}$.

As the automaton $\mathscr{B}$ is composed of one SCBC, determinization of the SCBC containing state q_1 results in the automaton $\mathscr{S}$, which is identical to $\mathscr{D}$, the determinization of $\mathscr{B}$. This is why a homomorphism applied on an automaton which is strongly connected, will always be weakly continuation closed.

The next step is then to choose a state r of $\mathscr{S}$ and then for each state $t \in r(q, \mathscr{D})$ we verify if $c(t, \mathscr{D}) = c(r, \mathscr{S})$ holds. For our example, we choose the state $\{q_1, q_6\}$ as state r of $\mathscr{S}$. Such that $c(t, \mathscr{D}) = c(\{q_1, q_6\}, \mathscr{S})$ holds, $\{q_1, q_6\} \subseteq t$ must be true. The only state satisfying this property and being reachable from state $q = \{q_1, q_6\}$, is the state $\{q_1, q_6\}$. And we know that $c(\{q_1, q_6\}, \mathscr{D}) = c(\{q_1, q_6\}, \mathscr{S})$ is always true, $\mathscr{D} = \mathscr{S}$.

The homomorphism h is therefore weakly continuation closed on $\mathscr{B}$ and the abstract system automaton $\mathscr{D}$ satisfies the same property as the $\mathscr{B}$ inherently fair.

Let $\mathscr{B}_1$ be automaton describing a different system, see fig 1a. The same homomorphism h will be applied to $\mathscr{B}_1$, which results in the automaton is $\mathscr{D}_1$, fig 1b.

The automaton $\mathscr{B}_1$ has one SCBC, so we select for example state q_4. There is one macrostate of $\mathscr{D}_1$ containing an occurence of q_4. By determinization of the SCBC containing state q_4, an automaton $\mathscr{S}$ is obtained such that $L(\mathscr{S}) = (req \cdot rej)^\omega$. The automaton $\mathscr{S} = (Q_S, \Sigma', \delta_s, q_4, Q_S)$, where Q_S contains two states, namely, q_4 and $\{q_5, q_6\}$ and δ_s has two transitions, $(q_4, req, \{q_5, q_6\})$ and $(\{q_5, q_6\}, rej, q_4)$.

The state $\{q_1, q_4\}$ of $\mathscr{D}_1$ contains the only occurence of q_4. It can easily be seen that $c(\{q_1, q_4\}, \mathscr{D}_1) \neq c(q_4, \mathscr{S})$, because $c(\{q_1, q_4\}, \mathscr{D}_1)$ contains the sequence $(req \cdot res)^\omega$ and $c(q_4, \mathscr{S})$ does not. Then for state $\{q_5, q_6\}$ of $\mathscr{S}$, there exists state $t = \{q_2, q_3, q_5, q_6\}$, where $\{q_5, q_6\} \subseteq t$. But again $c(t, \mathscr{D}_1) \neq c(\{q_5, q_6\}, \mathscr{S})$ because $c(t, \mathscr{D}_1)$ contains the sequence $(res \cdot req)^\omega$ and $c(\{q_5, q_6\}, \mathscr{S})$ does not.

[3] Set a symbol to the empty string ϵ

This implies that for all states r of $\mathscr{S}$ it holds that $c(r, \mathscr{S}) \neq c(t, \mathscr{D}_1)$, where t is reachable from state q and $r \subseteq t$, therefore the homomorphism h is not weakly continuation closed on $\mathscr{B}_1$.

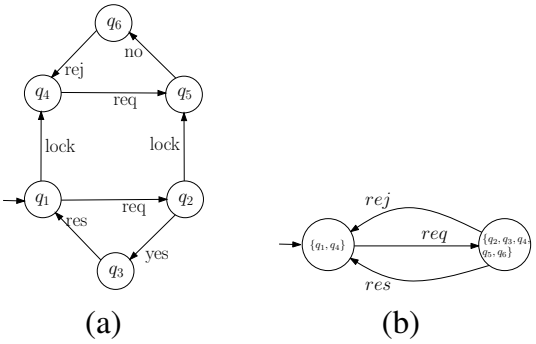

Fig. 2. The automaton $\mathscr{B}_1$ representing the original system behaviour (a) and the automaton $\mathscr{D}_1$ representing the abstracted system behaviour (b) reduced by homomorphism h.

7 Conclusions

This paper is a first step towards an optimized algorithm for checking whether or not a homomorphism is weakly continuation closed on a given automaton. The approach of this paper is to reduce the number of language comparison needed to check for WCC homomorphisms. The task of language comparison is a well-known task, so the authors assume this task to be optimized and therefore the interest to loose complexity of a procedure in a different step. As ω-language comparison presents a complex task, reducing its number results in a great optimization benefit.

We assume that the benefit is sufficient enough to think of a practical implementation of the procedure. As the presented procedure is only a sketch of a possible algorithm, we believe that there is still room for improvements.

Actual work in progress includes he development and implementation of an optimized algorithm for verifying whether or not a homomorphism is weakly continuation closed.

References

1. Ultes-Nitsche, U., James, S.S.: Improved verification of linear-time properties within fairness: weakly contiuation-closed behaviour abstractions computed from trace reductions. Software testing, Verification and Reliability **13** (2003) 241–255
2. Clarke, E.M., Grumberg, O., Long, D.E.: Model checking and abstraction. ACM Transactions on Programming Languages and Systems **16** (1994) 1512–1542
3. Cousot, P., Cousot, R.: Software analysis and model checking. In Brinksma, E., Larsen, K., eds.: Proceedings of the 14th International Conference on Computer Aided Verification, CAV 2002. Copenhagen, Denmark, LNCS 2404, Springer-Verlag Berlin Heidelberg (2002) 37–56

4. Dams, D., Gerth, R., Knaack, B., Kuiper, R.: Partial-order reduction techniques for real-time model checking. Formal Asp. Comput. **10** (1998) 469–482
5. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison Wesley Longman (2001)
6. Thomas, W.: Automata on infinite objects. In van Leeuwen, J., ed.: Formal Models and Semantics. Volume B of Handbook of Theoretical Computer Science., Elsevier (1990) 133–191
7. Ultes-Nitsche, U.: A power-set construction for reducing Büchi automata to non-determinism degree two. Information Processing Letters (IPL) **101** (2007) 107–111
8. Nitsche, U., Ochsenschläger, P.: Approximately satisfied properties of systems and simple language homomorphisms. I Information Processing Letters **60** (1996)
9. Nicola, T., Niessner, F., Ultes-Nitsche, U.: Model-checking inherently fair linear-time properties. In: Proceedings of the 3rd International Workshop on Modelling, Simulation, Verification, and Validation of Enterprise Information Systems (MSVVEIS 2005), Miami, Florida, USA (2005) 3–8
10. Francez, N.: Fairness. Springer Verlag (1986)
11. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. The MIT Press (1999)