Compositional Nonblocking Verification Using Generalised Nonblocking Abstractions

Robi Malik   Ryan Leduc

***Abstract*—This paper proposes a method for compositional verification of the standard and generalised nonblocking properties of large discrete event systems. The method is efficient as it avoids the explicit construction of the complete state space by considering and simplifying individual subsystems before they are composed further. Simplification is done using a set of abstraction rules preserving generalised nonblocking equivalence, which are shown to be correct and computationally feasible. Experimental results demonstrate the suitability of the method to verify several large-scale discrete event systems models both for standard and generalised nonblocking.**

***Index Terms*—Discrete event systems; Automata; Nonblocking; Model/Controller reduction.**

I. Introduction

THIS paper is concerned about model checking of the *generalised nonblocking* property [1] of large discrete event systems. *Standard nonblocking* is a weak liveness property commonly used in supervisory control theory to express the absence of livelocks or deadlocks [2], [3]. Generalised nonblocking adds the ability to restrict the set of states from which the property is checked. While generalised nonblocking includes standard nonblocking as a special case, it has increased expressive powers that make it possible to specify functional properties of software and certain conditions in Hierarchical Interface-Based Supervisory Control [4], [5].

Properties such as generalised nonblocking can be verified using standard state-space exploration or CTL model checking [6], but these approaches are limited by the well-known state-space explosion problem. As an alternative, *compositional verification* [7] uses *abstraction* to simplify components before or during verification, reducing the size of the state space that needs to be explored.

Very specific abstraction methods are needed in order to verify nonblocking-like properties compositionally. A suitable theory for standard nonblocking is laid out in [8], where it is argued that abstractions used in nonblocking verification should preserve a process-algebraic equivalence called *conflict equivalence*. Various abstraction rules preserving conflict equivalence are proposed in [7], [9]–[11]. Although similar to standard nonblocking, generalised nonblocking requires a different abstraction theory. *Generalised nonblocking equivalence* was first proposed in [1], and a process-algebraic standard form for it is presented in [12]. A set of computationally feasible abstraction rules for generalised nonblocking can be found in [13], [14], and a first implementation is described in [15]. Some of these abstraction rules are generalisations of rules for standard nonblocking [7], while others are only applicable to generalised nonblocking.

Robi Malik is with the Department of Computer Science, University of Waikato, Hamilton, New Zealand; robi@waikato.ac.nz

Ryan Leduc is with the Department of Computing and Software, McMaster University, Hamilton, Canada; leduc@mcmaster.ca

As standard nonblocking is a special case of generalised nonblocking [1], all these methods can also be used to verify standard nonblocking. Moreover, after translation from standard to generalised nonblocking, abstraction can produce a true generalised nonblocking verification problem. It then becomes possible to apply abstraction rules not normally available for standard nonblocking [14].

This paper further develops the results of [10], [13]–[15]. It contains modified abstraction rules for *weak observation equivalence* [10], [16] and *marking removal*, and a new α-*determinisation rule*. It also includes full correctness proofs, and experimental results that demonstrate the feasibility of the method to verify large models both for standard and generalised nonblocking.

In the following, Sect. II introduces the necessary background of nondeterministic automata and defines the generalised nonblocking property and generalised nonblocking equivalence. Then Sect. III describes the abstraction rules for generalised nonblocking and proves their correctness, Sect. IV presents the experimental results, and Sect. V adds some concluding remarks.

II. Preliminaries

A. Events and Languages

Event sequences and languages are a simple means to describe discrete system behaviours [2], [3]. Their basic building blocks are *events*, which are taken from a finite *alphabet* Σ. In addition, the *silent event* $\tau \notin \Sigma$ is used, which is not usually included in the event alphabet. An alphabet including τ is denoted by $\Sigma_\tau = \Sigma \cup \{\tau\}$.

Σ^* denotes the set of all finite *traces* of the form $\sigma_1\sigma_2 \dots \sigma_n$ of events from Σ, including the *empty trace* ε. The set $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$ does not include the empty trace. The *concatenation* of two traces $s, t \in \Sigma^*$ is written as st. A subset $\mathcal{L} \subseteq \Sigma^*$ is called a *language*. The *natural projection* $P_\tau \colon \Sigma_\tau^* \to \Sigma^*$ is the operation that deletes all silent (τ) events from traces.

B. Multi-coloured Automata

In this paper, system behaviours are modelled using nondeterministic multi-coloured automata. *Nondeterminism* is essential for the abstraction techniques in this paper. *Multi-coloured* automata extend the traditional concept of *marked states* to multiple marking conditions, by labelling states with different *colours* or *propositions*. The generalised nonblocking property is defined using these propositions. The following definition from [1] is based on similar ideas in [17], [18].

Definition 1: A *multi-coloured automaton* is a tuple $G = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi \rangle$ where Σ is a finite set of *events*, Π is a finite set of *propositions* or *colours*, Q is a set of *states*, $\rightarrow \subseteq Q \times \Sigma_\tau \times Q$ is the *state transition relation*, $Q^\circ \subseteq Q$ is the set of *initial states*, and $\Xi \colon \Pi \rightarrow 2^Q$ defines the set of marked states for each proposition in Π.

The transition relation is written in infix notation $x \xrightarrow{\sigma} y$, and is extended to traces in Σ_τ^* in the standard way. For state sets $X, Y \subseteq Q$, the notation $X \xrightarrow{s} Y$ means $x \xrightarrow{s} y$ for some $x \in X$ and $y \in Y$, and $x \xrightarrow{s} Y$ means $x \xrightarrow{s} y$ for some $y \in Y$. Also, $x \rightarrow y$ denotes the existence of a trace $s \in \Sigma_\tau^*$ such that $x \xrightarrow{s} y$, and $x \xrightarrow{s}$ denotes the existence of a state $y \in Q$ such that $x \xrightarrow{s} y$. Finally, $G \xrightarrow{s} x$ stands for $Q^\circ \xrightarrow{s} x$.

To support silent events, another transition relation $\Rightarrow \subseteq Q \times \Sigma^* \times Q$ is introduced, where $x \overset{s}{\Rightarrow} y$ denotes the existence of a trace $t \in \Sigma_\tau^*$ such that $P_\tau(t) = s$ and $x \xrightarrow{t} y$. That is, $x \xrightarrow{s} y$ denotes a path with *exactly* the events in s, while $x \overset{s}{\Rightarrow} y$ denotes a path with an arbitrary number of τ events shuffled with the events of s. Notations such as $X \overset{s}{\Rightarrow} Y$, $x \Rightarrow y$, $x \overset{s}{\Rightarrow}$, and $G \overset{s}{\Rightarrow} x$ are defined analogously to $\rightarrow$. For $\pi \in \Pi$, the π-marked language of state $x \in Q$ is

$$\mathcal{L}^\pi(x) = \{\, s \in \Sigma^* \mid x \overset{s}{\Rightarrow} \Xi(\pi) \,\} \ . \qquad (1)$$

Synchronous composition models the parallel execution of two or more automata, and is done using lock-step synchronisation in the style of [19].

Definition 2: Let $G_1 = \langle \Sigma, \Pi, Q_1, \rightarrow_1, Q_1^\circ, \Xi_1 \rangle$ and $G_2 = \langle \Sigma, \Pi, Q_2, \rightarrow_2, Q_2^\circ, \Xi_2 \rangle$ be multi-coloured automata. The *synchronous composition* of G_1 and G_2 is

$$G_1 \parallel G_2 = \langle \Sigma, \Pi, Q_1 \times Q_2, \rightarrow, Q_1^\circ \times Q_2^\circ, \Xi \rangle \qquad (2)$$

where

$$\begin{aligned}
(x_1, x_2) &\xrightarrow{\sigma} (y_1, y_2) && \text{if } \sigma \in \Sigma,\ x_1 \xrightarrow{\sigma}_1 y_1,\ x_2 \xrightarrow{\sigma}_2 y_2; \\
(x_1, x_2) &\xrightarrow{\tau} (y_1, x_2) && \text{if } x_1 \xrightarrow{\tau}_1 y_1; \\
(x_1, x_2) &\xrightarrow{\tau} (x_1, y_2) && \text{if } x_2 \xrightarrow{\tau}_2 y_2;
\end{aligned}$$

and $\Xi(\pi) = \Xi_1(\pi) \times \Xi_2(\pi)$ for each $\pi \in \Pi$.

This definition assumes that the two composed automata share the same event and proposition alphabets. Automata with different alphabets can also be composed by lifting them to common alphabets first: when an event σ is added to the alphabet Σ, selfloop transitions $x \xrightarrow{\sigma} x$ are added for all states $x \in Q$, and when a proposition π is added to Π, it is defined that $\Xi(\pi) = Q$.

Hiding is the process-algebraic operation that generalises natural projection of languages when nondeterministic automata are considered. Events that are not of interest are replaced by silent (τ) transitions or *ε-moves* [20].

Definition 3: Let $G = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi \rangle$ be a multi-coloured automaton, and let $\Upsilon \subseteq \Sigma$. The result of *hiding* Υ in G is

$$G \setminus \Upsilon \ = \ \langle \Sigma \setminus \Upsilon, \Pi, Q, \rightarrow \setminus \Upsilon, Q^\circ, \Xi \rangle \ , \qquad (3)$$

where $\rightarrow \setminus \Upsilon$ is obtained from $\rightarrow$ by replacing all events in Υ with the silent event τ.

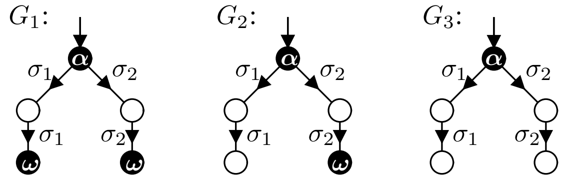

Fig. 1. Generalised nonblocking vs. standard nonblocking.

C. Generalised Nonblocking

It is desirable for control systems to be free from *livelock* and *deadlock*. This is captured by the *nonblocking* property [3], which requires that a terminal state can be reached from every reachable state. In this paper, a proposition $\omega \in \Pi$ is used to designate states as terminal states.

Definition 4: Let $G = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi \rangle$ with $\omega \in \Pi$ be a multi-coloured automaton. G is *ω-nonblocking* or *standard nonblocking* if for all states $x \in Q$ such that $G \Rightarrow x$ it also holds that $x \Rightarrow \Xi(\omega)$. Otherwise, G is *ω-blocking*.

Standard nonblocking is the weak liveness property used in most applications of supervisory control theory, particularly for synthesis [3]. Yet, there are cases where standard nonblocking is insufficient [1], [18], [21], for example in Hierarchical Interface-Based Supervisory Control [1], [4] and in software verification.

When analysing software, it is desirable to verify correct functionality in addition to nonblocking. When a software function is called, it is desirable that *all* specified return values are possible. For example, if a function is specified to return a Boolean value, then it should be possible for both results *true* and *false* to be returned. This amounts to two individual nonblocking checks, one for each possible return value. However, standard nonblocking does not correctly capture the desired property. The software may reach a point during its execution where a decision has been made to return a particular value, e.g. *true*, and from that point on the other result, *false*, is no longer possible. A standard nonblocking check to determine whether a state returning *false* is always reachable, will incorrectly report this behaviour as blocking. The question that really needs to be checked is whether the return value *false* is possible from the state *immediately* after the function call.

Such questions can be expressed using *generalised nonblocking* [1]. Generalised nonblocking uses two propositions, called α and ω, with the intended meaning that ω represents terminal states, while α specifies a set of states from which terminal states are required to be reachable.

Definition 5: Let $G = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi \rangle$ with $\alpha, \omega \in \Pi$ be a multi-coloured automaton. G is *(α,ω)-nonblocking* or *generalised nonblocking*, if for all states $x \in \Xi(\alpha)$ such that $G \Rightarrow x$ it also holds that $x \Rightarrow \Xi(\omega)$. Otherwise, G is (α,ω)-*blocking*.

Example 1: In Fig. 1, automaton G_1 is ω-nonblocking and (α,ω)-nonblocking, G_2 is ω-blocking but (α,ω)-nonblocking, and G_3 is both ω-blocking and (α,ω)-blocking,

Generalised nonblocking requires that, from all reachable states marked α, it is possible to reach a state marked ω. Clearly, if an automaton is ω-nonblocking, it is also (α,ω)-

nonblocking, but the converse is not true in general. The relationship between standard and generalised nonblocking along with some applications is discussed in [1].

D. Generalised Nonblocking Equivalence

The straightforward approach to verify whether a composed system

$$G_1 \parallel G_2 \parallel \cdots \parallel G_n \tag{4}$$

is (α, ω)-nonblocking consists of explicitly constructing the synchronous composition and checking whether a state marked ω can be reached from every state marked α. This can be done using CTL model checking, and models of substantial size can be analysed if the state space is represented symbolically [6]. Yet, the technique remains limited by the amount of memory available to store representations of the synchronous composition.

As an alternative, *compositional* reasoning [7] attempts to rewrite individual components of a composed system such as (4) and, e.g., replace G_1 by a simpler version G'_1, to analyse the simpler system

$$G'_1 \parallel G_2 \parallel \cdots \parallel G_n \ . \tag{5}$$

Such compositional reasoning requires that G_1 and G'_1 are related in some way. An appropriate notion of equivalence has been identified for the verification of standard nonblocking in [8], and adapted to (α, ω)-nonblocking in [1].

Definition 6: Let G_1 and G_2 be two multi-coloured automata with $\alpha, \omega \in \Pi$. Then G_1 and G_2 are called (α, ω)-*nonblocking equivalent*, written $G_1 \simeq_{(\alpha,\omega)} G_2$, if for any multi-coloured automaton T, it holds that $G_1 \parallel T$ is (α, ω)-nonblocking if and only if $G_2 \parallel T$ is (α, ω)-nonblocking.

To be feasible for compositional verification, the equivalence used must be well-behaved with respect to synchronous composition and hiding. These so-called *congruence* properties can easily be shown for (α, ω)-nonblocking equivalence [1], [14].

Proposition 1: Let G_1, G_2, T be multi-coloured automata with $\alpha, \omega \in \Pi$. If $G_1 \simeq_{(\alpha,\omega)} G_2$, then $G_1 \parallel T \simeq_{(\alpha,\omega)} G_2 \parallel T$.

Proposition 2: Let $G = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi \rangle$ be a multi-coloured automaton with $\alpha, \omega \in \Pi$, and let $\Upsilon \subseteq \Sigma$. Then G is (α, ω)-nonblocking if and only if $G \setminus \Upsilon$ is (α, ω)-nonblocking.

Note that, if given two automata G and H such that H does not use any events in alphabet Υ, then $(G \parallel H) \setminus \Upsilon = (G \setminus \Upsilon) \parallel H$. In combination with Prop. 2 this means that abstractions can be applied in a compositional way, as long as only events local to the subsystem considered are subject to hiding. Subsystems can be simplified individually or composed as needed, and the verification and simplification strategies outlined in [7], [8] can be used.

III. Abstractions that Preserve Generalised Nonblocking

Compositional verification relies on algorithms that rewrite a given automaton to a simpler equivalent form. A general way to achieve this is proposed in [12], where it is shown how any given automaton can be replaced by an (α, ω)-nonblocking equivalent *canonical form*. Unfortunately, the proposed algorithm has exponential complexity in the size of the automaton, and the resulting canonical form is not guaranteed to be smaller. This paper uses a weaker method based on [13], [14]. It introduces a set of computationally feasible abstraction rules that, although incomplete, achieve a guaranteed reduction of the state space.

Some of the following results are similar and closely related to similar results about standard nonblocking [7]. Yet, although (α, ω)-nonblocking seems to be more complicated than standard nonblocking at first glance, it is a weaker property and different kinds of abstraction are possible. Markings can be removed from certain states, and some states that are not coreachable can be removed. Furthermore, several of the states encountered in generalised nonblocking are *not* marked α, and these can often be simplified more aggressively than states marked α.

A. Weak Observation Equivalence

One of the strongest known equivalences of nondeterministic automata is *observation equivalence* [22]. Observation equivalence considers two states as equivalent if they have exactly the same structure of nondeterministic future behaviour.

Observation equivalence is known to preserve all temporal properties. It preserves standard conflict equivalence, and it also is finer than and implies (α, ω)-nonblocking equivalence [14]. Observation equivalence comes with efficient simplification algorithms [23] and has been used successfully to simplify automata for the verification of standard nonblocking, where this abstraction alone is responsible for a substantial reduction in the number of states [7].

It is shown in [10] that observation equivalence can be relaxed to *weak* observation equivalence for the purpose of standard nonblocking verification, and this is shown below to hold for generalised nonblocking as well.

Definition 7: Let $G_1 = \langle \Sigma, \Pi, Q_1, \rightarrow_1, Q_1^\circ, \Xi_1 \rangle$ and $G_2 = \langle \Sigma, \Pi, Q_2, \rightarrow_2, Q_2^\circ, \Xi_2 \rangle$ be two multi-coloured automata. A relation $\approx_w \subseteq Q_1 \times Q_2$ is a *weak observation equivalence* relation between G_1 and G_2 if, for all states $x_1 \in Q_1$ and $x_2 \in Q_2$ such that $x_1 \approx_w x_2$,

(i) if $x_1 \overset{s}{\Rightarrow}_1 y_1$ for some $s \in \Sigma^+$, then there exists $y_2 \in Q_2$ such that $y_1 \approx_w y_2$ and $x_2 \overset{s}{\Rightarrow}_2 y_2$;
(ii) if $x_2 \overset{s}{\Rightarrow}_2 y_2$ for some $s \in \Sigma^+$, then there exists $y_1 \in Q_1$ such that $y_1 \approx_w y_2$ and $x_1 \overset{s}{\Rightarrow}_1 y_1$;
(iii) if $x_1 \overset{\varepsilon}{\Rightarrow}_1 \Xi_1(\pi)$ for some $\pi \in \Pi$, then $x_2 \overset{\varepsilon}{\Rightarrow}_2 \Xi_2(\pi)$;
(iv) if $x_2 \overset{\varepsilon}{\Rightarrow}_2 \Xi_2(\pi)$ for some $\pi \in \Pi$, then $x_1 \overset{\varepsilon}{\Rightarrow}_1 \Xi_1(\pi)$;
(v) for all $x_1^\circ \in Q_1$ such that $Q_1^\circ \overset{\varepsilon}{\Rightarrow}_1 x_1^\circ$, there exists $x_2^\circ \in Q_2$ such that $Q_2^\circ \overset{\varepsilon}{\Rightarrow}_2 x_2^\circ$ and $x_1^\circ \approx_w x_2^\circ$;
(vi) for all $x_2^\circ \in Q_2$ such that $Q_2^\circ \overset{\varepsilon}{\Rightarrow}_2 x_2^\circ$, there exists $x_1^\circ \in Q_1$ such that $Q_1^\circ \overset{\varepsilon}{\Rightarrow}_1 x_1^\circ$ and $x_1^\circ \approx_w x_2^\circ$.

G_1 and G_2 are *weakly observation equivalent*, $G_1 \approx_w G_2$, if there exists a weak observation equivalence relation $\approx_w$ between G_1 and G_2.

The difference between weak observation equivalence and observation equivalence is that weak observation equivalence only considers traces containing at least one event. An observation equivalence relation [22] can be defined using the same

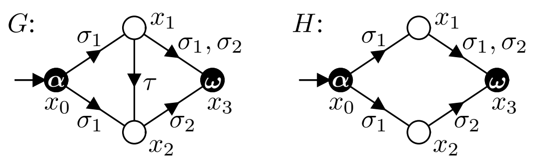

Fig. 2. Example of weak observation equivalence.

conditions (i)–(vi), except that all traces $s \in \Sigma^*$ are considered in conditions (i) and (ii).

Example 2: Automata G and H in Fig. 2 are weakly observation equivalent. Note that these automata are not observation equivalent, because from state x_1 in G, the state x_2 where only σ_2 is enabled can be reached silently, but this is not possible in H. For weak observation equivalence, it is enough that a state like x_2 can be reached via σ_1 from the initial state.

Weak observation equivalence is coarser than observation equivalence and provides for better abstraction [10]. At the same time, it can be computed using almost the same algorithms as observation equivalence, and it implies generalised nonblocking equivalence.

Proposition 3: Let G_1 and G_2 be two multi-coloured automata with $\alpha, \omega \in \Pi$. If $G_1 \approx_w G_2$ then $G_1 \simeq_{(\alpha,\omega)} G_2$.

Proof: Let $\approx_w$ be a weak observation equivalence relation between G_1 and G_2, let T be such that $G_1 \parallel T$ is (α, ω)-nonblocking, and let $G_2 \parallel T \overset{s}{\Rightarrow} (x_2, x_T) \in \Xi_{G_2 \parallel T}(\alpha)$. Then either $s = \varepsilon$ or $s \in \Sigma^+$.

If $s = \varepsilon$, then $Q_2^\circ \overset{\varepsilon}{\Rightarrow}_2 x_2$, so by Def. 7 (vi) there exists $x_1 \in Q_1$ such that $Q_1^\circ \overset{\varepsilon}{\Rightarrow}_1 x_1$ and $x_1 \approx_w x_2$.

If $s \in \Sigma^+$, then let $x_2^\circ \in Q_2^\circ$ such that $x_2^\circ \overset{s}{\Rightarrow}_2 x_2$. Clearly $Q_2^\circ \overset{\varepsilon}{\Rightarrow}_2 x_2^\circ$, so by Def. 7 (vi) there exists $x_1^\circ \in Q_1$ such that $Q_1^\circ \overset{\varepsilon}{\Rightarrow}_1 x_1^\circ$ and $x_1^\circ \approx_w x_2^\circ$. Then since $x_2^\circ \overset{s}{\Rightarrow}_2 x_2$, by Def. 7 (ii) there exists $x_1 \in Q_1$ such that $x_1^\circ \overset{s}{\Rightarrow}_1 x_1$ and $x_1 \approx_w x_2$. Thus, $Q_1^\circ \overset{\varepsilon}{\Rightarrow} x_1^\circ \overset{s}{\Rightarrow} x_1$ and $x_1 \approx_w x_2$.

In both cases, $G_1 \parallel T \overset{s}{\Rightarrow} (x_1, x_T)$ for some $x_1 \in Q_1$ such that $x_1 \approx_w x_2$. Since $x_2 \in \Xi_2(\alpha)$, it follows from Def. 7 (iv) that $x_1 \overset{\varepsilon}{\Rightarrow}_1 \Xi_1(\alpha)$. Thus, there exists $x_1^\alpha \in \Xi_1(\alpha)$ such that $G_1 \parallel T \overset{s}{\Rightarrow} (x_1, x_T) \overset{\varepsilon}{\Rightarrow} (x_1^\alpha, x_T) \in \Xi_{G_1 \parallel T}(\alpha)$. Since $G_1 \parallel T$ is (α, ω)-nonblocking, there exist $t \in \Sigma^*$, $y_1 \in \Xi_1(\omega)$, and $y_T \in \Xi_T(\omega)$ such that

$$G_1 \parallel T \overset{s}{\Rightarrow} (x_1, x_T) \overset{\varepsilon}{\Rightarrow} (x_1^\alpha, x_T) \overset{t}{\Rightarrow} (y_1, y_T) \in \Xi_{G_1 \parallel T}(\omega) \ .$$

Again, either $t = \varepsilon$ or $t \in \Sigma^+$.

If $t = \varepsilon$, then $x_1 \overset{\varepsilon}{\Rightarrow}_1 \Xi_1(\omega)$, and since $x_1 \approx_w x_2$, it follows from Def. 7 (iii) that $x_2 \overset{\varepsilon}{\Rightarrow}_2 \Xi_2(\omega)$.

If $t \in \Sigma^+$, then since $x_1 \overset{t}{\Rightarrow}_1 y_1$ and $x_1 \approx_w x_2$, by Def. 7 (i) there exists $y_2 \in Q_2$ such that $x_2 \overset{t}{\Rightarrow}_2 y_2$ and $y_1 \approx_w y_2$. Also since $y_1 \in \Xi_1(\omega)$, it follows from Def. 7 (iii) that $y_2 \overset{\varepsilon}{\Rightarrow}_2 \Xi_2(\omega)$. Thus, $x_2 \overset{t}{\Rightarrow} y_2 \overset{\varepsilon}{\Rightarrow} \Xi_2(\omega)$.

In both cases, $G_2 \parallel T \overset{s}{\Rightarrow} (x_2, x_T) \overset{t}{\Rightarrow} \Xi_{G_2 \parallel T}(\omega)$. Since such a trace t can be constructed for any $s \in \Sigma^*$, it follows that $G_2 \parallel T$ is (α, ω)-nonblocking.

Analogously it can be shown that, if $G_2 \parallel T$ is (α, ω)-nonblocking, then $G_1 \parallel T$ is (α, ω)-nonblocking. It follows that $G_1 \simeq_{(\alpha,\omega)} G_2$. ■

Prop. 3 confirms that an automaton can be replaced by a weakly observation equivalent version when verifying generalised nonblocking.

Rule 1 (Weak Observation Equivalence Rule): If two automata G_1 and G_2 are weakly observation equivalent, then G_1 can be replaced by G_2 (and vice versa).

Complexity. A coarsest weak observation equivalence relation for a given automaton can be computed using a partition refinement algorithm [23] in $O(|{\Rightarrow}| \log |Q|)$ time. This algorithm requires an explicit representation of the relation $\Rightarrow$, which in turn requires computation of the transitive closure of silent transitions. This step takes $O(|Q|^3)$ time and usually dominates the complexity of observation equivalence [24]. The worst-case time complexity to simplify an automaton based on weak observation equivalence is $O(|Q|^3 + |{\Rightarrow}| \log |Q|)$.

B. Removal of Observation Equivalent Markings

As a special case of (weak) observation equivalence, it is possible to remove markings from certain states with outgoing silent transitions. This is particularly helpful in generalised nonblocking, as it reduces the number of α-marked states, which contribute to the bulk of the verification effort.

Rule 2 (Marking Removal Rule): If an automaton contains a state y marked by proposition $\pi \in \Pi$ and a path $x \overset{\varepsilon}{\Rightarrow} y$, then a marking π can be removed from or added to state x.

Example 3: Automata G_1 and G_2 in Fig. 3 are (α, ω)-nonblocking equivalent. Since state x_1 is marked α, any test that is to be (α, ω)-nonblocking in combination with G_1 needs to be able to execute σ_2 initially. This implicitly includes the condition for state x_0, which says that a test needs to be able to execute σ_1 or σ_2 initially. As the test must satisfy both, the condition simplifies to just executing σ_2. Testing for state x_1 alone is thus sufficient, so the α-marking of state x_0 can be removed as shown in G_2.

Proposition 4: Let $G = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi_G \rangle$ be a multi-coloured automaton with $\alpha, \omega, \pi \in \Pi$ and states $p, q \in Q$ such that $p \overset{\varepsilon}{\Rightarrow} q$, $p \neq q$, and $q \in \Xi_G(\pi)$. Define $H = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi_H \rangle$ where Ξ_H is identical to Ξ_G except $\Xi_H(\pi) = \Xi_G(\pi) \setminus \{p\}$. Then $G \simeq_{(\alpha,\omega)} H$.

Proof: It follows from Def. 7 that $G \approx_w H$. Therefore, the claim follows from Prop. 3. ■

Complexity. Marking removal is best applied to an automaton without any loops of silent transitions. These loops can be found using Tarjan's algorithm [25], and afterwards marking removal can be achieved in a single pass over the source states of the τ-transitions. Both operations can be completed in $O(|Q|^2)$ time.

While the removal of markings does not reduce the number of states of an automaton, it can make it simpler and enable other abstractions. The removal of α-markings can also be considered when verifying standard nonblocking, where all states are marked α initially. After removal of α-markings, other rules for generalised nonblocking may become applicable.

C. Removal of ω-Markings

In addition to the removal of observation equivalent markings, it is possible to remove further ω-markings while still preserving generalised nonblocking equivalence. This is the

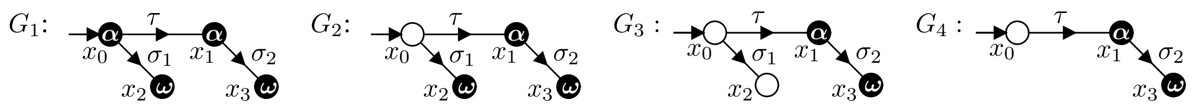

Fig. 3. Example application of Marking Removal Rule, followed by ω-Removal Rule, and Coreachability Rule.

first abstraction rule that extends beyond the scope of (weak) observation equivalence [10] and conflict equivalence [7].

Rule 3 (ω-Removal Rule): If a state x is not reachable from any state marked α, then an ω-marking can be removed from (or added to) state x.

Example 4: Automata G_2 and G_3 in Fig. 3 are (α, ω)-nonblocking equivalent. Only for states marked α, it is required that a state marked ω is reachable, but state x_2 in G_2 cannot be reached from any state marked α. Therefore, the fact that x_2 is marked ω is irrelevant, and this marking can be removed as shown in G_3.

Proposition 5: Let $G = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi_G \rangle$ be a multi-coloured automaton with $\alpha, \omega \in \Pi$, and let $q \in Q$ such that $\Xi_G(\alpha) \rightarrow q$ does not hold. Define $H = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi_H \rangle$ where Ξ_H is identical to Ξ_G except $\Xi_H(\omega) = \Xi_G(\omega) \setminus \{q\}$. Then $G \simeq_{(\alpha,\omega)} H$.

Proof: Let T be an arbitrary multi-coloured automaton. It is sufficient to show that $G \parallel T$ is (α, ω)-nonblocking if and only if $H \parallel T$ is (α, ω)-nonblocking.

First assume that $G \parallel T$ is (α, ω)-nonblocking, and let $H \parallel T \Rightarrow (x, x_T) \in \Xi_{H\parallel T}(\alpha)$. Since the transition relations of G and H are equal, it follows that $G \parallel T \Rightarrow (x, x_T) \in \Xi_{H\parallel T}(\alpha) = \Xi_{G\parallel T}(\alpha)$. Since $G \parallel T$ is (α, ω)-nonblocking, there are states $y \in \Xi_G(\omega)$ and $y_T \in \Xi_T(\omega)$ such that $G \parallel T \Rightarrow (x, x_T) \Rightarrow (y, y_T)$. Again, since the transition relations of G and H are equal, it follows that $H \parallel T \Rightarrow (x, x_T) \Rightarrow (y, y_T)$. Also note $y \neq q$ as $x \in \Xi_G(\alpha)$ and $x \rightarrow y$, and thus $y \in \Xi_G(\omega) \setminus \{q\} = \Xi_H(\omega)$. This implies $H \parallel T \Rightarrow (x, x_T) \Rightarrow (y, y_T) \in \Xi_{H\parallel T}(\omega)$.

Second assume that $H \parallel T$ is (α, ω)-nonblocking, and let $G \parallel T \Rightarrow (x, x_T) \in \Xi_{G\parallel T}(\alpha)$. Since the transition relations of G and H are equal, it follows that $H \parallel T \Rightarrow (x, x_T) \in \Xi_{G\parallel T}(\alpha) = \Xi_{H\parallel T}(\alpha)$. Since $H \parallel T$ is (α, ω)-nonblocking, it follows that $H \parallel T \Rightarrow (x, x_T) \Rightarrow \Xi_{H\parallel T}(\omega)$, and since the transition relations of G and H are equal, also $G \parallel T \Rightarrow (x, x_T) \Rightarrow \Xi_{H\parallel T}(\omega) \subseteq \Xi_{G\parallel T}(\omega)$. ■

Complexity. To apply the ω-Removal Rule to an automaton, it needs to be checked for all states whether they are reachable from an α-marked state. This can be done by a standard graph search visiting each transition at most once. There are at most $|Q|^2|\Sigma_\tau|$ transitions, and this leads to the overall complexity of $O(|Q|^2|\Sigma|)$ to check and apply the ω-Removal Rule to all states where it is applicable.

Again, the removal of ω-markings does not directly reduce the state space, but it can make other rules applicable. In particular, it may increase the number of non-coreachable states, which can be deleted according to the following rule.

D. Removal of Non-coreachable States

Following is the first abstraction that actually removes states from an automaton. The generalised nonblocking property only needs to be checked from states marked α, and from there only traces that can reach a state marked ω are relevant. If it is not possible to reach a state marked α or ω from some state x, then this state x is irrelevant for the generalised nonblocking property. Such states x can be removed.

Rule 4 (Coreachability Rule): States that are not α/ω-coreachable, i.e., from which neither a state marked α nor a state marked ω can be reached, can be removed.

Example 5: Automata G_3 and G_4 in Fig. 3 are (α, ω)-nonblocking equivalent. State x_2 in G_3 is neither α-coreachable nor ω-coreachable, and therefore it is not needed to reach an ω-marked state, nor does it lead to any further conditions (α-marked state) that need to be satisfied. This state can be removed as shown in G_4.

The coreachability rule seems superficially similar to the *Certain Conflicts Rule* [7], yet it is quite different. The Certain Conflicts Rule merges blocking states into a single state when verifying the standard nonblocking property. Here, the coreachability rule allows non-coreachable states to be removed entirely.

Proposition 6: Let $G = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi \rangle$ be a multi-coloured automaton with $\alpha, \omega \in \Pi$, and let C be the set of α/ω-coreachable states for G, namely $C = \{\, x \in Q \mid x \rightarrow \Xi(\alpha) \cup \Xi(\omega) \,\}$. Define $H = \langle \Sigma, \Pi, C, \rightarrow_{|C}, Q^\circ \cap C, \Xi \rangle$ where $\rightarrow_{|C} = \{\, (x, \sigma, y) \in \rightarrow \mid x, y \in C \,\}$. Then $G \simeq_{(\alpha,\omega)} H$.

Proof: Let T be an arbitrary multi-coloured automaton. It is sufficient to show that $G \parallel T$ is (α, ω)-nonblocking if and only if $H \parallel T$ is (α, ω)-nonblocking.

First assume that $G \parallel T$ is (α, ω)-nonblocking, and let $H \parallel T \Rightarrow (x, x_T) \in \Xi_{H\parallel T}(\alpha)$. Obviously, since $\rightarrow_{|C} \subseteq \rightarrow$ and $Q^\circ \cap C \subseteq Q^\circ$, this implies $G \parallel T \Rightarrow (x, x_T) \in \Xi_{H\parallel T}(\alpha) \subseteq \Xi_{G\parallel T}(\alpha)$. Since $G \parallel T$ is (α, ω)-nonblocking, it holds that $G \parallel T \Rightarrow (x, x_T) \overset{t}{\Rightarrow} \Xi_{G\parallel T}(\omega)$ for some $t \in \Sigma^*$. Then there exist events $\sigma_1, \ldots, \sigma_n \in \Sigma_\tau$, $n \geq 0$, such that $t = P_\tau(\sigma_1 \ldots \sigma_n)$ and states $y_0, \ldots, y_n \in Q$ such that

$$G \Rightarrow x = y_0 \xrightarrow{\sigma_1} y_1 \xrightarrow{\sigma_2} \cdots \xrightarrow{\sigma_n} y_n \in \Xi(\omega) \ . \tag{6}$$

Then $y_0, y_1, \ldots, y_n \in C$ by construction of C, and hence $H \parallel T \Rightarrow (x, x_T) \Rightarrow \Xi(\omega) \times \Xi_T(\omega) = \Xi_{H\parallel T}(\omega)$.

Second assume that $H \parallel T$ is (α, ω)-nonblocking, and let $G \parallel T \overset{s}{\Rightarrow} (x, x_T) \in \Xi_{G\parallel T}(\alpha)$. Then there exist events $\sigma_1, \ldots, \sigma_n \in \Sigma_\tau$, $n \geq 0$, such that $s = P_\tau(\sigma_1 \ldots \sigma_n)$ and states $x_0, \ldots, x_n \in Q$ such that $x_0 \in Q^\circ$ and

$$x_0 \xrightarrow{\sigma_1} x_1 \xrightarrow{\sigma_2} \cdots \xrightarrow{\sigma_n} x_n = x \in \Xi(\alpha) \ . \tag{7}$$

Then $x_0, x_1, \ldots, x_n \in C$ by construction of C, and hence $H \parallel T \overset{s}{\Rightarrow} (x, x_T) \in \Xi(\alpha) \times \Xi_T(\alpha) = \Xi_{H\parallel T}(\alpha)$. Since $H \parallel T$ is (α, ω)-nonblocking, it follows that $H \parallel T \overset{s}{\Rightarrow} (x, x_T) \Rightarrow \Xi_{H\parallel T}(\omega)$, and since $\rightarrow_{|C} \subseteq \rightarrow$ also $G \parallel T \overset{s}{\Rightarrow} (x, x_T) \Rightarrow \Xi_{H\parallel T}(\omega) \subseteq \Xi_{G\parallel T}(\omega)$. ■

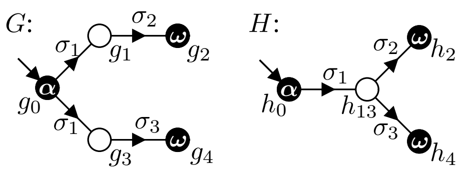

Fig. 4. Example application of Non-α Determinisation Rule.

Complexity. α/ω-coreachability of all states in an automaton can be checked by a standard graph search visiting each transition at most once. There are at most $|Q|^2|\Sigma_\tau|$ transitions, and this leads to the overall complexity of $O(|Q|^2|\Sigma|)$ to check and apply the Coreachability Rule.

E. Determinisation of Non-α States

In generalised nonblocking, there are two different kinds of states. States marked α carry nonblocking requirements, which means that their precise nondeterministic future may be relevant. These states can only be simplified using rules preserving conflict equivalence such as those in [7]. On the other hand, non-α states do not carry nonblocking requirements, and only the language associated with these states is important. These states can be treated using language equivalence, and determinisation algorithms [20] can be used to merge them.

Rule 5 (Non-α Determinisation Rule): Two non-α marked states that are reachable by exactly the same traces from initial states and from each state marked α, can be merged into a single state.

Example 6: Automata G and H in Fig. 4 are (α, ω)-nonblocking equivalent. States g_1 and g_3 are only reachable via trace σ_1 from the initial state or from the only α-marked state and therefore can be merged into a single state h_{13} as shown in H. Note that this simplification is not possible for standard nonblocking, or if one of the two states is marked α, because in this case it is important that the two states have different continuations to states marked ω.

To describe this rule formally, the concept of *quotient automaton* is used. The idea is to identify certain groups of states as equivalent and merge each group into a single state. The following definitions are standard.

Definition 8: Let X be an arbitrary set. A relation $\sim \subseteq X \times X$ is an *equivalence relation*, if $\sim$ is reflexive, symmetric, and transitive. If $\sim$ is an equivalence relation on X, the *equivalence class* of $x \in X$ is $[x] = \{\, y \in X \mid x \sim y \,\}$, and the set of equivalence classes modulo $\sim$ is $X/\!\sim\; = \{\, [x] \mid x \in X \,\}$.

Definition 9: Let $G = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi \rangle$ be a multi-coloured automaton, and let $\sim \subseteq Q \times Q$ be an equivalence relation. The *quotient automaton* of G modulo $\sim$ is

$$G/\!\sim \;=\; \langle \Sigma, \Pi, Q/\!\sim, \rightarrow/\!\sim, \tilde{Q^\circ}, \tilde{\Xi} \rangle \;, \tag{8}$$

where

$$\begin{aligned} \rightarrow/\!\sim \;&=\; \{\, ([x], \sigma, [y]) \mid x \xrightarrow{\sigma} y \,\} \;; \\ \tilde{Q^\circ} \;&=\; \{\, [x^\circ] \mid x^\circ \in Q^\circ \,\} \;; \\ \tilde{\Xi}(\pi) \;&=\; \{\, [x] \mid x \in \Xi(\pi) \,\} \quad \text{for all } \pi \in \Pi \;. \end{aligned}$$

The Non-α Determinisation Rule is described using a particular equivalence relation, namely a *reverse observation equivalence* [26]: two states are considered as equivalent if they can be reached via the same traces from the initial states.

Definition 10: Let $G = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi \rangle$. An equivalence relation $\sim \subseteq Q \times Q$ is a *reverse observation equivalence* on G, if the following conditions hold for all $x_1, x_2 \in Q$ with $x_1 \sim x_2$.

- If $x_1 \in Q^\circ$, then $Q^\circ \overset{\varepsilon}{\Rightarrow} x_2$.
- For all states $w_1 \in Q$ and all events $\sigma \in \Sigma_\tau$ such that $w_1 \xrightarrow{\sigma} x_1$ there exists a state $w_2 \in Q$ such that $w_2 \overset{P_\tau(\sigma)}{\Longrightarrow} x_2$ and $w_1 \sim w_2$.

All equivalent states can be merged at the same time. Therefore, the previously stated Non-α Determinisation Rule is replaced by the following more general version.

Rule 5 (Non-α Determinisation Rule): If $\sim$ is a reverse observation equivalence on an automaton G such that states marked α are only equated to themselves by $\sim$, then G can be replaced by $G/\!\sim$.

To prove the validity of this rule, the relationship between the traces in an automaton G and its abstraction $G/\!\sim$ needs to be established first. It is well-known that every trace in G also has a corresponding trace in $G/\!\sim$. The following result quoted from [7] holds for every equivalence relation.

Lemma 7: Let $G = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi \rangle$, and let $\sim \subseteq Q \times Q$ be an equivalence relation. Then, for all states $w, x \in Q$ and all traces $s \in \Sigma^*$ such that $w \overset{s}{\Rightarrow} x$ in G, it holds that $[w] \overset{s}{\Rightarrow} [x]$ in $G/\!\sim$.

Proof: Let $w \overset{s}{\Rightarrow} x$ in G. Then there exists $t = \sigma_1 \ldots \sigma_n \in \Sigma_\tau^*$ such that $P_\tau(t) = s$ and $w \xrightarrow{t} x$. Also, there exist states $x_0, \ldots, x_n \in Q$ such that $w = x_0 \xrightarrow{\sigma_1} \cdots \xrightarrow{\sigma_n} x_n = x$. By Def. 9, it holds that $[x_{k-1}] \xrightarrow{\sigma_k} [x_k]$ for each $k = 1, \ldots, n$, which implies $[w] \overset{s}{\Rightarrow} [x]$ in $G/\!\sim$. ■

Conversely, for a trace in the quotient automaton $G/\!\sim$, there does not always exist a corresponding trace in the original automaton. This only holds under additional conditions, in this case that a reverse observation equivalence is used.

Lemma 8: Let $G = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi \rangle$, and let $\sim \subseteq Q \times Q$ be a reverse observation equivalence on G. Then, for all states $w, x \in Q$ and all traces $s \in \Sigma^*$ such that $[w] \overset{s}{\Rightarrow} [x]$ in $G/\!\sim$, there exists $w' \in [w]$ such that $w' \overset{s}{\Rightarrow} x$ in G.

Proof: Let $w, x \in Q$ and $s \in \Sigma^*$ such that $[w] \overset{s}{\Rightarrow} [x]$ in $G/\!\sim$. Then there exists $s' \in \Sigma_\tau^*$ such that $[w] \xrightarrow{s'} [x]$ and $P_\tau(s') = s$. It is shown by induction on $n = |s'|$ that for $[w] \xrightarrow{s'} [x]$ there exists $w' \in [w]$ such that $w' \overset{P_\tau(s')}{\Longrightarrow} x$.

Base case: $s' = \varepsilon$. $[w] \xrightarrow{\varepsilon} [x]$ implies $[w] = [x]$, and with $x \in [x] = [w]$ it follows that $x \overset{\varepsilon}{\Rightarrow} x$ in G.

Inductive step: $s' = t'\sigma$. Assume that $[w] \xrightarrow{t'} [y] \xrightarrow{\sigma} [x]$. Since $[y] \xrightarrow{\sigma} [x]$ in $G/\!\sim$, by definition of $\rightarrow/\!\sim$ there exist states $x' \in [x]$ and $y' \in [y]$ such that $y' \xrightarrow{\sigma} x'$. Then $x' \sim x$, and since $\sim$ is a reverse observation equivalence there exists $y'' \in Q$ such that $y'' \overset{P_\tau(\sigma)}{\Longrightarrow} x$ and $y'' \sim y'$. By inductive assumption, since $[w] \xrightarrow{t'} [y] = [y'] = [y'']$, there exists $w' \in [w]$ such that $w' \overset{P_\tau(t')}{\Longrightarrow} y'' \overset{P_\tau(\sigma)}{\Longrightarrow} x$. Since $P_\tau(t'\sigma) = P_\tau(s') = s$, it follows that $w' \overset{s}{\Rightarrow} x$ in G. ■

Having thus established a relationship between the traces in an automaton and its quotient, the validity of the Non-α Determinisation Rule can now be proven.

Proposition 9: Let $G = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi \rangle$ be a multi-coloured automaton with $\alpha, \omega \in \Pi$, and let $\sim\, \subseteq Q \times Q$ be a reverse observation equivalence on G such that $[x] = \{x\}$ for all $x \in \Xi(\alpha)$. Then $G \simeq_{(\alpha,\omega)} G/{\sim}$.

Proof: Let T be an arbitrary multi-coloured automaton. It is sufficient to show that $G \,\|\, T$ is (α,ω)-nonblocking if and only if $(G/{\sim}) \,\|\, T$ is (α,ω)-nonblocking.

First assume that $G \,\|\, T$ is (α,ω)-nonblocking, and let $(G/{\sim}) \,\|\, T \overset{s}{\Rightarrow} ([x], x_T) \in \Xi_{(G/\sim)\|T}(\alpha)$. Then $\tilde{Q}^\circ \overset{s}{\Rightarrow} [x] \in \tilde{\Xi}(\alpha)$, i.e., $[x^\circ] \overset{s}{\Rightarrow} [x]$ for some $x^\circ \in Q^\circ$. As $[x] \in \tilde{\Xi}(\alpha)$, by construction of $\tilde{\Xi}$ there exists $x^\alpha \in [x]$ such that $x^\alpha \in \Xi(\alpha)$. By Lemma 8, there exists $x' \in [x^\circ]$ such that $x' \overset{s}{\Rightarrow} x^\alpha$ in G. Then $x' \sim x^\circ \in Q^\circ$, and thus $Q^\circ \overset{\varepsilon}{\Rightarrow} x'$ since $\sim$ is a reverse observation equivalence. Hence, $G \overset{\varepsilon}{\Rightarrow} x' \overset{s}{\Rightarrow} x^\alpha$ and $G \,\|\, T \overset{s}{\Rightarrow} (x^\alpha, x_T) \in \Xi_{G\|T}(\alpha)$. As $G \,\|\, T$ is (α,ω)-nonblocking, there exist $t \in \Sigma^*$, $y \in Q$ and $y_T \in Q_T$ such that $G \,\|\, T \overset{s}{\Rightarrow} (x^\alpha, x_T) \overset{t}{\Rightarrow} (y, y_T) \in \Xi_{G\|T}(\omega)$. Thus, $x^\alpha \overset{t}{\Rightarrow} y$, so it follows from Lemma 7 that $[x^\alpha] \overset{t}{\Rightarrow} [y]$. Therefore, $(G/{\sim}) \,\|\, T \overset{s}{\Rightarrow} ([x], x_T) = ([x^\alpha], x_T) \overset{t}{\Rightarrow} ([y], y_T) \in \Xi_{(G/\sim)\|T}(\omega)$, i.e., $(G/{\sim}) \,\|\, T$ is (α,ω)-nonblocking.

Second assume $(G/{\sim}) \,\|\, T$ is (α,ω)-nonblocking, and let $G \,\|\, T \overset{s}{\Rightarrow} (x, x_T) \in \Xi_{G\|T}(\alpha)$. Then $x^\circ \overset{s}{\Rightarrow} x$ for some $x^\circ \in Q^\circ$, and by Lemma 7 it follows that $[x^\circ] \overset{s}{\Rightarrow} [x]$, i.e., $(G/{\sim}) \overset{s}{\Rightarrow} [x]$. As $x \in \Xi(\alpha)$, it holds that $[x] \in \tilde{\Xi}(\alpha)$. Thus, $(G/{\sim}) \,\|\, T \overset{s}{\Rightarrow} ([x], x_T) \in \Xi_{(G/\sim)\|T}(\alpha)$. Since $(G/{\sim}) \,\|\, T$ is (α,ω)-nonblocking, there exist $t \in \Sigma^*$, $y \in Q$, and $y_T \in Q_T$ such that $(G/{\sim}) \,\|\, T \overset{s}{\Rightarrow} ([x], x_T) \overset{t}{\Rightarrow} ([y], y_T) \in \Xi_{(G/\sim)\|T}(\omega)$. Then $[y] \in \tilde{\Xi}(\omega)$, and by construction of $\tilde{\Xi}$ there exists $y^\omega \in [y]$ such that $y^\omega \in \Xi(\omega)$. Then $[x] \overset{t}{\Rightarrow} [y] = [y^\omega]$ in $G/{\sim}$, and by Lemma 8, there exists $x^\omega \in [x]$ such that $x^\omega \overset{t}{\Rightarrow} y^\omega$ in G. Since $x \in \Xi(\alpha)$, it holds that $x^\omega = x$ by assumption. It then follows that $G \,\|\, T \overset{s}{\Rightarrow} (x, x_T) = (x^\omega, x_T) \overset{t}{\Rightarrow} (y^\omega, y_T) \in \Xi_{G\|T}(\omega)$. ■

Complexity. A coarsest reverse observation equivalence relation can be computed in the same way as a weak observation equivalence relation using the algorithm in [23], also under the additional constraint that states marked α cannot be merged. Its complexity is the same as for weak observation equivalence, i.e., $O(|Q|^3 + |{\Rightarrow}| \log |Q|)$.

F. Determinisation of α-Marked States

While states not marked α can be merged easily, more care needs to be taken when merging states with an α-marking. States marked α have associated nonblocking requirements. Such states can only be merged if the nonblocking requirements are equal, i.e., if they have the same ω-marked languages.

Rule 6 (α-Determinisation Rule): If $\sim$ is a reverse observation equivalence on an automaton G such that equivalent states also have equal ω-marked languages, then G can be replaced by $G/{\sim}$.

Example 7: Automata G and H in Fig. 5 are (α,ω)-nonblocking equivalent. States g_1 and g_2 in G are reverse observation equivalent and have the same ω-marked languages, so they can be merged into a single state h_{12}. This is possible despite the two states having different α-markings and thus not being (weakly) observation equivalent.

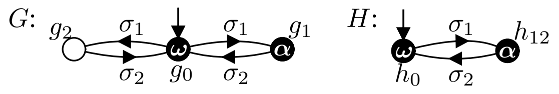

Fig. 5. Example application of α-Determinisation Rule.

Proposition 10: Let $G = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi \rangle$ be a multi-coloured automaton with $\alpha, \omega \in \Pi$, and let $\sim\, \subseteq Q \times Q$ be a reverse observation equivalence on G such that for all $x, y \in Q$, if $x \sim y$ then $\mathcal{L}^\omega(x) = \mathcal{L}^\omega(y)$. Then $G \simeq_{(\alpha,\omega)} G/{\sim}$.

Proof: Let T be an arbitrary multi-coloured automaton. It is sufficient to show that $G \,\|\, T$ is (α,ω)-nonblocking if and only if $(G/{\sim}) \,\|\, T$ is (α,ω)-nonblocking.

If $G \,\|\, T$ is (α,ω)-nonblocking, it follows by the same argument as in the proof of Prop. 9 that $(G/{\sim}) \,\|\, T$ is (α,ω)-nonblocking.

Conversely, assume that $(G/{\sim}) \,\|\, T$ is (α,ω)-nonblocking, and let $G \,\|\, T \overset{s}{\Rightarrow} (x, x_T) \in \Xi_{G\|T}(\alpha)$. Then $x^\circ \overset{s}{\Rightarrow} x$ for some $x^\circ \in Q^\circ$, and by Lemma 7 it follows that $[x^\circ] \overset{s}{\Rightarrow} [x]$; and as $x \in \Xi(\alpha)$, it holds that $[x] \in \tilde{\Xi}(\alpha)$. Thus, $(G/{\sim}) \,\|\, T \overset{s}{\Rightarrow} ([x], x_T) \in \Xi_{(G/\sim)\|T}(\alpha)$. Since $(G/{\sim}) \,\|\, T$ is (α,ω)-nonblocking, there exist $t \in \Sigma^*$, $y \in Q$, and $y_T \in Q_T$ such that $(G/{\sim}) \,\|\, T \overset{s}{\Rightarrow} ([x], x_T) \overset{t}{\Rightarrow} ([y], y_T) \in \Xi_{(G/\sim)\|T}(\omega)$. Then $[y] \in \tilde{\Xi}(\omega)$, and by construction of $\tilde{\Xi}$ there exists $y^\omega \in [y]$ such that $y^\omega \in \Xi(\omega)$. Then $[x] \overset{t}{\Rightarrow} [y] = [y^\omega]$ in $G/{\sim}$, and by Lemma 8, there exists $x^\omega \in [x]$ such that $x^\omega \overset{t}{\Rightarrow} y^\omega \in \Xi(\omega)$. Then $x^\omega \sim x$ and $t \in \mathcal{L}^\omega(x^\omega) = \mathcal{L}^\omega(x)$ by assumption. It follows that $G \,\|\, T \overset{s}{\Rightarrow} (x, x_T) \overset{t}{\Rightarrow} \Xi_{G\|T}(\omega)$. ■

Complexity. The complexity to check for equality of ω-marked languages of states in a nondeterministic automaton is exponential because of the need for subset construction [20]. To avoid this, the implementation in Sect. IV uses weak observation equivalence instead of language equivalence. More precisely, two states are only merged if they are found to be weakly observation equivalent, but only considering $\pi = \omega$ in Def. 7 (iii) and (iv). As weak observation equivalence implies language equivalence [10], this ensures that only states with equal ω-marked languages are merged. It also means that the implementation is only useful for true generalised nonblocking verification problems, where it can merge α-marked states with states not marked α as in example 7. A coarsest weak observation equivalence relation that is also a reverse observation equivalence can be computed in $O(|Q|^3 + |{\Rightarrow}| \log |Q|)$ time as is the case for the other partitioning abstractions.

G. Removal of τ-Transitions Leading to Non-α States

Silent (τ) transitions provide a significant potential for abstraction. If a silent transition links two α-marked states, then the α-Removal Rule can be used to remove the α-marking of the source state. If neither the source nor the target state are marked α, then only the ω-marked languages of these states are relevant, and simplification is often possible by means of the Non-α Determinisation Rule. Additionally, and also in cases where at most one of the two states linked by a silent transition is marked α, the Silent Continuation Rule in this section or the Only Silent Outgoing Rule in the following section may be applicable to remove the transition.

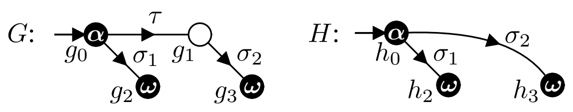

Fig. 6. Example application of Silent Continuation Rule.

Rule 7 (Silent Continuation Rule): A transition $p \xrightarrow{\tau} q$ with $q \notin \Xi(\alpha)$ can be removed if all transitions originating from state q are copied to state p.

Example 8: Automata G and H in Fig. 6 are (α,ω)-nonblocking equivalent. The transition $g_0 \xrightarrow{\tau} g_1$ in G leads to a non-α state, so it can be removed after copying the σ_2-transition originating from the target state g_1 to the source state g_0. As a result, the target state g_1 becomes unreachable and can be removed as shown in H.

This simplification relies on the fact that the target state g_1 is *not* marked α, so there is no nonblocking requirement associated with that state; thus it can be merged into the source state, leading to much stronger simplification than the Silent Continuation Rule for standard nonblocking [7].

The following definition describes the construction of the simplified automaton formally. To prove the validity of the rule, it is again necessary to establish the relationship between traces in the original and reduced automata. This is done in Lemmas 11 and 12 below, and afterwards the validity of the Silent Continuation Rule is established in Prop. 13.

Definition 11: Let $G = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi \rangle$ be a multi-coloured automaton with states $p, q \in Q$ such that $p \xrightarrow{\tau} q$. The *target bypass* of transition $p \xrightarrow{\tau} q$ in G is the automaton $G_{p\curvearrowright q} = \langle \Sigma, \Pi, Q, \rightarrow_{p\curvearrowright q}, Q^\circ, \Xi_{p\curvearrowright q} \rangle$ where

$$\begin{aligned} \rightarrow_{p\curvearrowright q} &= (\rightarrow \setminus \{(p,\tau,q)\}) \cup \{\, (p,\sigma,z) \mid q \xrightarrow{\sigma} z \,\} \; ; \\ \Xi_{p\curvearrowright q}(\pi) &= \begin{cases} \Xi(\pi) \cup \{p\}, & \text{if } q \in \Xi(\pi) \; ; \\ \Xi(\pi), & \text{otherwise .} \end{cases} \end{aligned}$$

Lemma 11: Let $G = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi \rangle$ be a multi-coloured automaton with states $p, q \in Q$ such that $p \xrightarrow{\tau} q$. For all states $x, y \in Q$ and all traces $s \in \Sigma^*$, if $x \overset{s}{\Rightarrow}_{p\curvearrowright q} y$ in $G_{p\curvearrowright q}$, then $x \overset{s}{\Rightarrow} y$ in G.

Proof: Given $x \overset{s}{\Rightarrow}_{p\curvearrowright q} y$, there exist $\sigma_1, \ldots, \sigma_n \in \Sigma_\tau$, $n \geq 0$ such that $s = P_\tau(\sigma_1 \ldots \sigma_n)$ and $x_0, \ldots, x_n \in Q$ such that

$$x = x_0 \xrightarrow{\sigma_1}_{p\curvearrowright q} x_1 \xrightarrow{\sigma_2}_{p\curvearrowright q} \cdots \xrightarrow{\sigma_n}_{p\curvearrowright q} x_n = y \; . \tag{9}$$

It suffices to show $x_{i-1} \overset{P_\tau(\sigma_i)}{\Longrightarrow} x_i$ for $i = 1, \ldots, n$. If $x_{i-1} \xrightarrow{\sigma_i} x_i$, this is trivial. Otherwise, $x_{i-1} = p$ and $q \xrightarrow{\sigma_i} x_i$ by Def. 11. This implies $x_{i-1} = p \xrightarrow{\tau} q \xrightarrow{\sigma_i} x_i$, i.e., $x_{i-1} \overset{P_\tau(\sigma_i)}{\Longrightarrow} x_i$ as required. ■

Lemma 12: Let $G = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi \rangle$ be a multi-coloured automaton with states $p, q \in Q$ such that $p \xrightarrow{\tau} q$. Furthermore, let $x, y \in Q$ and $s \in \Sigma^*$ such that $x \overset{s}{\Rightarrow} y$ in G. Then the following statements hold for $G_{p\curvearrowright q}$.

(i) If $y \neq q$, then $x \overset{s}{\Rightarrow}_{p\curvearrowright q} y$;
(ii) If $y = q$, then $x \overset{s}{\Rightarrow}_{p\curvearrowright q} \{y, p\}$.

Proof: Given $x \overset{s}{\Rightarrow} y$, there exists $s' \in \Sigma_\tau^*$ such that $x \xrightarrow{s'} y$ and $P_\tau(s') = s$. It suffices to show that $x \xrightarrow{s'} y$ implies $x \overset{P_\tau(s')}{\Longrightarrow}_{p\curvearrowright q} F(y)$, where the map $F\colon Q \to 2^Q$ is defined as $F(z) = \{z\}$ for $z \neq q$ and $F(q) = \{p, q\}$. This claim is proven using induction on the length of the trace s'.

Base case: $s' = \varepsilon$. In this case, $P_\tau(s') = \varepsilon$ and $x = y$, and it follows immediately that $x \overset{\varepsilon}{\Rightarrow}_{p\curvearrowright q} x = y \in F(y)$.

Inductive step: $s' = \sigma t'$ for $\sigma \in \Sigma_\tau$ and $t' \in \Sigma_\tau^*$. In this case, there is a state $z \in Q$ such that $x \xrightarrow{\sigma} z \xrightarrow{t'} y$.

If $x \xrightarrow{\sigma} z$ is *not* the transition $p \xrightarrow{\tau} q$, then $x \xrightarrow{\sigma}_{p\curvearrowright q} z$ by Def. 11, and it follows from the inductive assumption that $z \overset{P_\tau(t')}{\Longrightarrow}_{p\curvearrowright q} F(y)$. Thus,

$$x \overset{P_\tau(\sigma)}{\Longrightarrow}_{p\curvearrowright q} z \overset{P_\tau(t')}{\Longrightarrow}_{p\curvearrowright q} F(y) \; , \tag{10}$$

i.e., $x \overset{P_\tau(s')}{\Longrightarrow}_{p\curvearrowright q} F(y)$.

Otherwise, if $x \xrightarrow{\sigma} z$ is the transition $p \xrightarrow{\tau} q$, i.e., $x = p$, $z = q$, and $\sigma = \tau$, two more cases need to be considered.

If $t' = \varepsilon$, then $P_\tau(s') = P_\tau(\sigma t') = P_\tau(\tau\varepsilon) = \varepsilon$ and $y = z = q$. In this case, the claim follows because $x = p \overset{\varepsilon}{\Rightarrow}_{p\curvearrowright q} p \in \{p, q\} = F(q) = F(y)$.

If $t' \neq \varepsilon$, let $t' = \sigma' u'$ for $\sigma' \in \Sigma_\tau$ and $u' \in \Sigma_\tau^*$. Then $q = z \xrightarrow{\sigma'} r' \xrightarrow{u'} y$ for some state $r' \in Q$. By Def. 11, $q \xrightarrow{\sigma'} r'$ implies that $p \xrightarrow{\sigma'}_{p\curvearrowright q} r'$, and by the inductive assumption, $r' \xrightarrow{u'} y$ implies $r' \overset{P_\tau(u')}{\Longrightarrow}_{p\curvearrowright q} F(y)$. Thus,

$$x = p \xrightarrow{\sigma'}_{p\curvearrowright q} r' \overset{P_\tau(u')}{\Longrightarrow}_{p\curvearrowright q} F(y) \; , \tag{11}$$

and given $P_\tau(\sigma' u') = P_\tau(t') = P_\tau(\tau t') = P_\tau(s')$, it follows that $x \overset{P_\tau(s')}{\Longrightarrow}_{p\curvearrowright q} F(y)$. ■

Proposition 13: Let $G = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi \rangle$ be a multi-coloured automaton with $\alpha, \omega \in \Pi$ and states $p, q \in Q$ such that $p \xrightarrow{\tau} q$, and $q \notin \Xi(\alpha)$. Then $G \simeq_{(\alpha,\omega)} G_{p\curvearrowright q}$.

Proof: Let T be an arbitrary multi-coloured automaton. It is sufficient to show that $G \parallel T$ is (α,ω)-nonblocking if and only if $G_{p\curvearrowright q} \parallel T$ is (α,ω)-nonblocking.

First assume that $G \parallel T$ is (α,ω)-nonblocking, and let $G_{p\curvearrowright q} \parallel T \overset{s}{\Rightarrow} (y, y_T) \in \Xi_{G_{p\curvearrowright q} \parallel T}(\alpha)$. Then $Q^\circ \overset{s}{\Rightarrow}_{p\curvearrowright q} y \in \Xi_{p\curvearrowright q}(\alpha)$. By Lemma 11 it holds that $Q^\circ \overset{s}{\Rightarrow} y$ in G, and by Def. 11 it holds that $\Xi_{p\curvearrowright q}(\alpha) = \Xi(\alpha)$ as $q \notin \Xi(\alpha)$. Thus,

$$G \parallel T \overset{s}{\Rightarrow} (y, y_T) \in \Xi_{p\curvearrowright q}(\alpha) \times \Xi_T(\alpha) = \Xi_{G \parallel T}(\alpha) \; . \tag{12}$$

Since $G \parallel T$ is (α,ω)-nonblocking, there exist $t \in \Sigma^*$, $z \in Q$, and $z_T \in Q_T$ such that $G \parallel T \overset{s}{\Rightarrow} (y, y_T) \overset{t}{\Rightarrow} (z, z_T) \in \Xi_{G \parallel T}(\omega)$. Hence, $G \overset{s}{\Rightarrow} y \overset{t}{\Rightarrow} z$. By Lemma 12, it follows that either $y \overset{t}{\Rightarrow}_{p\curvearrowright q} z$ or $y \overset{t}{\Rightarrow}_{p\curvearrowright q} p$ with $z = q$. In the first case, note $z \in \Xi(\omega) \subseteq \Xi_{p\curvearrowright q}(\omega)$. In the second case, note that for $q = z \in \Xi(\omega)$, it also holds that $p \in \Xi_{p\curvearrowright q}(\omega)$ by Def. 11. Hence, in both cases $y \overset{t}{\Rightarrow}_{p\curvearrowright q} \Xi_{p\curvearrowright q}(\omega)$. It follows that $G_{p\curvearrowright q} \parallel T \overset{s}{\Rightarrow} (y, y_T) \overset{t}{\Rightarrow} \Xi_{p\curvearrowright q}(\omega) \times \Xi_T(\omega)$.

Second assume that $G_{p\curvearrowright q} \parallel T$ is (α,ω)-nonblocking, and let $G \parallel T \overset{s}{\Rightarrow} (y, y_T) \in \Xi_{G \parallel T}(\alpha)$. Then $Q^\circ \overset{s}{\Rightarrow} y \in \Xi(\alpha)$ in G. As $y \in \Xi(\alpha)$ and $q \notin \Xi(\alpha)$ by assumption, it holds that $y \neq q$. Then it follows from Lemma 12 that $Q^\circ \overset{s}{\Rightarrow}_{p\curvearrowright q} y \in \Xi(\alpha) \subseteq \Xi_{p\curvearrowright q}(\alpha)$ by Def. 11. Thus,

$$G_{p\curvearrowright q} \parallel T \overset{s}{\Rightarrow} (y, y_T) \in \Xi_{p\curvearrowright q}(\alpha) \times \Xi_T(\alpha) \; . \tag{13}$$

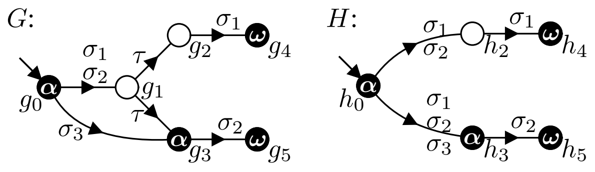

Fig. 7. Example application of Only Silent Outgoing Rule.

Since $G_{p\frown q} \parallel T$ is (α,ω)-nonblocking, there exist $t \in \Sigma^*$, $z \in Q$, and $z_T \in Q_T$ such that

$$G_{p\frown q} \parallel T \overset{s}{\Rightarrow} (y, y_T) \overset{t}{\Rightarrow} (z, z_T) \in \Xi_{p\frown q}(\omega) \times \Xi_T(\omega) \ . \quad (14)$$

Then $y \overset{t}{\Rightarrow}_{p\frown q} z$, and by Lemma 11 it follows that $y \overset{t}{\Rightarrow} z$ in G. If $z \in \Xi(\omega)$, it follows immediately that $y \overset{t}{\Rightarrow} \Xi(\omega)$. Otherwise, if $z \notin \Xi(\omega)$, note that $z \in \Xi_{p\frown q}(\omega)$, which means that $z = p$ and $q \in \Xi(\omega)$ by Def. 11. This implies $y \overset{t}{\Rightarrow} z = p \xrightarrow{\tau} q \in \Xi(\omega)$, and it again follows that $y \overset{t}{\Rightarrow} \Xi(\omega)$.

It follows that $G \parallel T \overset{s}{\Rightarrow} (y, y_T) \overset{t}{\Rightarrow} \Xi_{G\parallel T}(\omega)$. ∎

Complexity. The Silent Continuation Rule can be applied at most once to every τ-transition in an automaton, i.e., at most $|Q|^2$ applications. Each application involves the copying of all transitions from the target state to the source state, and there may be up to $|\Sigma_\tau||Q|$ transitions outgoing from every state. Therefore, the overall complexity to check the applicability of this rule and apply it to all applicable transitions is $O(|Q|^3|\Sigma|)$.

The removal of a τ-transition alone does not necessarily lead to a reduction in state space or complexity, and indeed careless use of the Silent Continuation Rule can substantially increase the number of transitions. The implementation in Sect. IV avoids this by only replacing τ-transitions leading to states that have no other incoming transitions except τ-transitions. Then the target state becomes unreachable, and the number of states is reduced when applying the Silent Continuation Rule.

H. Removal of τ-Transitions Originating from Non-α States

The final rule considers the case of a silent transition originating from a non-α state. This case is more difficult, and the following rule is more restrictive than its companion for standard nonblocking [7], because α-markings need to be taken into account in addition to other conditions.

Rule 8 (Only Silent Outgoing Rule): A state p that is not marked α or ω can be removed, if $p \xrightarrow{\tau}$, and p has only τ-transitions outgoing. Incoming transitions to p must be redirected to all the τ-successor states of p.

Example 9: Automata G and H in Fig. 7 are (α,ω)-nonblocking equivalent. State g_1 in G is not marked α or ω and has only τ-transitions outgoing, so it can be bypassed and removed as shown in H. This simplification is only possible because state g_1 is not marked α or ω. If the state is marked, the nonblocking conditions associated with it needs to be retained, and there is no easy way to merge these into one or both of the successor states.

Following is a formal description of the Only Silent Outgoing Rule. Again, the relationship between traces in the original and the simplified automata is established before proving the validity of the rule.

Definition 12: Let $G = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi \rangle$ be a multi-coloured automaton, and let $p \in Q$. The *silent outgoing bypass* of state p in G is the automaton $G_{p\curvearrowright} = \langle \Sigma, \Pi, Q, \rightarrow_{p\curvearrowright}, Q^\circ_{p\curvearrowright}, \Xi \rangle$ where

$$\begin{aligned} \rightarrow_{p\curvearrowright} \ &= \ (\rightarrow \setminus \{\, (w,\sigma,p) \mid w \xrightarrow{\sigma} p \,\}) \ \cup \\ &\quad\ \{\, (w,\sigma,x) \mid w \xrightarrow{\sigma} p \xrightarrow{\tau} x \,\} \ ; \\ Q^\circ_{p\curvearrowright} \ &= \ \begin{cases} (Q^\circ \setminus \{p\}) \cup \{\, x \in Q \mid p \xrightarrow{\tau} x \,\}, & \text{if } p \in Q^\circ \ , \\ Q^\circ, & \text{otherwise} \ . \end{cases} \end{aligned}$$

No state is explicitly removed in this construction. However, the bypassed state p becomes unreachable and can be removed, provided that $p \xrightarrow{\tau} p$ does not hold. If $p \xrightarrow{\tau} p$, then p remains reachable (consider $w \xrightarrow{\sigma} p \xrightarrow{\tau} p$ in the definition of $\rightarrow_{p\curvearrowright}$), but such τ-selfloops can be removed first using observation equivalence.

Lemma 14: Let $G = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi \rangle$, and let $p \in Q$. For all states $x, y \in Q$ and all traces $s \in \Sigma^*$, if $x \overset{s}{\Rightarrow}_{p\curvearrowright} y$ in $G_{p\curvearrowright}$, then $x \overset{s}{\Rightarrow} y$ in G.

Proof: For $x \overset{s}{\Rightarrow}_{p\curvearrowright} y$, there exist $\sigma_1, \ldots, \sigma_n \in \Sigma_\tau$, $n \geq 0$ such that $s = P_\tau(\sigma_1 \ldots \sigma_n)$ and $x_0, \ldots, x_n \in Q$ such that

$$x = x_0 \xrightarrow{\sigma_1}_{p\curvearrowright} x_1 \xrightarrow{\sigma_2}_{p\curvearrowright} \cdots \xrightarrow{\sigma_n}_{p\curvearrowright} x_n = y \ . \quad (15)$$

It suffices to show $x_{i-1} \overset{P_\tau(\sigma_i)}{\Longrightarrow} x_i$ for $i = 1, \ldots, n$. If $x_{i-1} \xrightarrow{\sigma_i} x_i$, this is trivial. Otherwise, $x_{i-1} \xrightarrow{\sigma_i} p \xrightarrow{\tau} x_i$ by Def. 12, which also implies $x_{i-1} \overset{P_\tau(\sigma_i)}{\Longrightarrow} x_i$. ∎

Lemma 15: Let $G = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi \rangle$, and let $p \in Q$ be a state with $p \xrightarrow{\tau}$, which has only τ-transitions outgoing, i.e., $p \xrightarrow{\sigma}$ implies $\sigma = \tau$. Furthermore, let $x, y \in Q$ and $s \in \Sigma^*$ such that $x \overset{s}{\Rightarrow} y$ in G and $y \neq p$. Then $x \overset{s}{\Rightarrow}_{p\curvearrowright} y$ in $G_{p\curvearrowright}$.

Proof: For $x \overset{s}{\Rightarrow} y$, there exists $s' \in \Sigma_\tau^*$ such that $x \xrightarrow{s'} y$ and $P_\tau(s') = s$. It is shown by induction on the length of s' that $x \xrightarrow{s'} y$ with $y \neq p$ in G implies $x \overset{P_\tau(s')}{\Longrightarrow}_{p\curvearrowright} y$ in $G_{p\curvearrowright}$.

Base case: $s' = \varepsilon$. In this case, $P_\tau(s') = \varepsilon$ and $x = y$. It follows immediately that $x \overset{\varepsilon}{\Rightarrow}_{p\curvearrowright} x = y$.

Inductive step: $s' = \sigma t'$ *for* $\sigma \in \Sigma_\tau$ *and* $t' \in \Sigma_\tau^*$. In this case, there is a state $z \in Q$ such that $x \xrightarrow{\sigma} z \xrightarrow{t'} y$.

If $z \neq p$, then $x \xrightarrow{\sigma}_{p\curvearrowright} z$ by Def. 12, and it follows from the inductive assumption that $z \overset{P_\tau(t')}{\Longrightarrow}_{p\curvearrowright} y$. Thus $x \overset{P_\tau(\sigma)}{\Longrightarrow}_{p\curvearrowright} z \overset{P_\tau(t')}{\Longrightarrow}_{p\curvearrowright} y$, and this implies $x \overset{P_\tau(s')}{\Longrightarrow}_{p\curvearrowright} y$.

Otherwise, if $z = p$, note that $p = z \xrightarrow{t'} y \neq p$ and hence $t' \neq \varepsilon$. Then let $t' = \sigma' u'$ for $\sigma' \in \Sigma_\tau$ and $u' \in \Sigma_\tau^*$. Then $z \xrightarrow{\sigma'} z' \xrightarrow{u'} y$ for some state $z' \in Q$. Also $\sigma' = \tau$ since $p = z$ has only τ-transitions outgoing, i.e., $x \xrightarrow{\sigma} z \xrightarrow{\tau} z'$. By Def. 12, it follows that $x \xrightarrow{\sigma}_{p\curvearrowright} z'$. By inductive assumption, $z' \xrightarrow{u'} y$ implies $z' \overset{P_\tau(u')}{\Longrightarrow}_{p\curvearrowright} y$. Thus, $x \xrightarrow{\sigma}_{p\curvearrowright} z' \overset{P_\tau(u')}{\Longrightarrow}_{p\curvearrowright} y$, and given $P_\tau(\sigma u') = P_\tau(\sigma\tau u') = P_\tau(\sigma\sigma' u') = P_\tau(\sigma t') = P_\tau(s')$, it follows that $x \overset{P_\tau(s')}{\Longrightarrow}_{p\curvearrowright} y$. ∎

Proposition 16: Let $G = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi \rangle$, and let $p \in Q$ be a state with $p \xrightarrow{\tau}$ and $p \notin \Xi(\alpha) \cup \Xi(\omega)$, which has only τ-transitions outgoing, i.e., $p \xrightarrow{\sigma}$ implies $\sigma = \tau$. Then $G \simeq_{(\alpha,\omega)} G_{p\curvearrowright}$.

Proof: Let T be an arbitrary multi-coloured automaton. It is sufficient to show that $G \parallel T$ is (α,ω)-nonblocking if and only if $G_{p\curvearrowright} \parallel T$ is (α,ω)-nonblocking.

First assume that $G \| T$ is (α,ω)-nonblocking, and let $G_{p\frown} \| T \overset{s}{\Rightarrow} (y, y_T) \in \Xi_{G_{p\frown}\|T}(\alpha)$. Then $Q^\circ_{p\frown} \overset{s}{\Rightarrow}_{p\frown} y \in \Xi(\alpha)$. Then there exists $x^\circ \in Q^\circ_{p\frown}$ such that $x^\circ \overset{s}{\Rightarrow}_{p\frown} y$. Applying Lemma 14 it follows that $x^\circ \overset{s}{\Rightarrow} y$. Also note that, if $x^\circ \notin Q^\circ$, then $x^\circ \in Q^\circ_{p\frown}$ means that $p \in Q^\circ$ and $p \xrightarrow{\tau} x^\circ$ by Def. 12. Thus, $G \overset{s}{\Rightarrow} y$ and therefore $G \parallel T \overset{s}{\Rightarrow} (y, y_T) \in \Xi_{G\|T}(\alpha)$. Since $G \parallel T$ is (α,ω)-nonblocking, there exist $t \in \Sigma^*$, $z \in Q$, and $z_T \in Q_T$ such that $G \parallel T \overset{s}{\Rightarrow} (y, y_T) \overset{t}{\Rightarrow} (z, z_T) \in \Xi_{G\|T}(\omega)$. Hence, $G \overset{s}{\Rightarrow} y \overset{t}{\Rightarrow} z$. Since $z \in \Xi(\omega)$ and $p \notin \Xi(\omega)$, it holds that $z \neq p$. Then $y \overset{t}{\Rightarrow}_{p\frown} z \in \Xi(\omega)$ by Lemma 15, and this means $G_{p\frown} \parallel T \overset{s}{\Rightarrow} (y, y_T) \overset{t}{\Rightarrow} \Xi_{G_{p\frown}\|T}(\omega)$.

Second assume that $G_{p\frown} \| T$ is (α,ω)-nonblocking, and let $G \parallel T \overset{s}{\Rightarrow} (y, y_T) \in \Xi_{G\|T}(\alpha)$. Then there exists $x^\circ \in Q^\circ$ such that $x^\circ \overset{s}{\Rightarrow} y \in \Xi(\alpha)$ in G. As $y \in \Xi(\alpha)$ and $p \notin \Xi(\alpha)$ by assumption, it holds that $y \neq p$, which implies $x^\circ \overset{s}{\Rightarrow}_{p\frown} y$ by Lemma 15. If $x^\circ \neq p$, it also holds that $x^\circ \in Q^\circ_{p\frown}$ by Def. 12. If $x^\circ = p$, note that p has only τ-transitions outgoing, and $y \neq p$, so the path $p = x^\circ \overset{s}{\Rightarrow} y$ is not empty and has the form $p \xrightarrow{\tau} x \overset{s}{\Rightarrow} y$ with $x \in Q^\circ_{p\frown}$. Thus, $Q^\circ_{p\frown} \overset{s}{\Rightarrow}_{p\frown} y \in \Xi(\alpha)$ and

$$G_{p\frown} \parallel T \overset{s}{\Rightarrow} (y, y_T) \in \Xi_{G\|T}(\alpha) = \Xi_{G_{p\frown}\|T}(\alpha) \ . \tag{16}$$

Since $G_{p\frown} \| T$ is (α,ω)-nonblocking, there exists $t \in \Sigma^*$ such that $G_{p\frown} \parallel T \overset{s}{\Rightarrow} (y, y_T) \overset{t}{\Rightarrow} \Xi_{G_{p\frown}\|T}(\omega)$. Then there exists a state $z \in \Xi(\omega)$ such that $y \overset{t}{\Rightarrow}_{p\frown} z$, and by Lemma 14 also $y \overset{t}{\Rightarrow} z$ in G. Hence, $G \parallel T \overset{s}{\Rightarrow} (y, y_T) \overset{t}{\Rightarrow} \Xi_{G\|T}(\omega)$. ■

Complexity. To check whether the Only Silent Outgoing Rule is applicable to a state, it must be confirmed that it is not marked and has at least one and only τ-transitions outgoing. Using appropriate data structures, this can be done in constant complexity. Applying the rule requires all incoming transitions to be copied to all τ-successor states. There can be up to $|Q||\Sigma_\tau|$ incoming transitions and up to $|Q|$ τ-successors per state. Then the complexity to check and apply the Only Silent Outgoing Rule to all states of an automaton is $O(|Q|^3|\Sigma|)$.

IV. Experimental Results

The compositional nonblocking verification algorithm has been implemented in the DES software tool Supremica [27] and tested on a number of models of reactive and control systems. The software is an improved version of [15], which includes new implementations of weak observation equivalence and the standard nonblocking abstractions of [7]. These are compared to the approach proposed in this paper.

The test suite includes complex industrial models and case studies taken from various application areas such as manufacturing systems, communication protocols, and automotive electronics. Included are all models used in [7] with at least 10^7 reachable states. The smaller models have been replaced by more complex models that can be solved by the new implementation. The following list gives some details about each group of models in the test suite.

agv Automated Guided Vehicle Coordination system based on a Petri Net model [28].

aip Model of the automated manufacturing system of the Atelier Inter-établissement de Productique [29]. The tests consider three early versions (**aip0**) based on [30], and a more detailed version (**aip1**) according to [31], which has been modified for a parametrisable number of pallets.

big_bmw Automotive Window Lift Controller model according to [21]. The model used here is an extended version with four individual windows.

fencaiwon09 Model of a production cell in a metal-processing plant from [32]. The supervisors in this model are handwritten and differ slightly from the synthesised original.

ftechnik Flexible production cell model based on [33].

profisafe PROFIsafe field bus protocol model [34]. The task considered here is to verify nonblocking of the communication partners and the network in input-slave configuration with sequence numbers ranging up to 4, 5, and 6.

tbed Model of a toy railroad system described in [35]. There are three versions representing different designs.

tip3 Model of the interaction between a mobile client and event-based servers of a Tourist Information System [36].

verriegel Central locking system of a BMW car, originally from the KorSys project [37].

6link Models of a cluster tool for wafer processing [16].

All these models have been checked for standard nonblocking using compositional verification, and the **aip1** models have been checked for generalised nonblocking in addition.

Compositional verification repeatedly chooses a small set of automata, composes them, applies abstraction rules to the synchronous composition, and replaces the composed automata with the result. This is repeated until the remaining automata are considered too large, or there are only two automata left. The final automata are not simplified, because it is easier to check them for generalised nonblocking directly. This is done by explicitly constructing and exploring the synchronous composition—the present implementation does not use BDDs or other symbolic representations [6]. To provide the user with diagnostic information when the model is blocking, the counterexample obtained from the final check is expanded to produce a counterexample for the original model by propagating it back through all abstraction steps [15].

A key aspect for a compositional verification algorithm is the way how automata are selected to be composed. The implementation considered here follows a two-step approach [7]. In the first step, some *candidate* sets of automata are formed, and in the second a most promising candidate is selected. For each event σ in the model, a candidate is formed consisting of all automata with σ in their alphabet; this strategy is called **MustL** [7], [15]. Other ways of forming candidates [7], [15] have been found to perform poorly for the larger models considered in this paper, and therefore are not considered in the following.

After forming a set of candidates, a most promising candidate is identified heuristically. The following heuristics are used for this purpose.

MinS Chooses the candidate with the smallest estimated number of states after abstraction. The estimate is obtained by multiplying the product of the state numbers of the automata forming the candidate with the ratio of the numbers of events in the synchronous composition of the candidate after and before hiding [15].

MinS$^{\alpha}$ Like **MinS**, but estimates the number of α-marked states.

MinSync Computes the synchronous composition of the automata in each candidate and chooses the candidate with the fewest states in the synchronous composition.

MinSync$^{\alpha}$ Like **MinSync**, but chooses the candidate with the fewest α-marked states in the synchronous composition.

After identification of a candidate, its automata are composed, and then a sequence of abstraction rules is applied. The Generalised Nonblocking Abstraction Sequence (**GNB**) first uses Tarjan's algorithm [25] to remove loops of τ-transitions from the automaton. This special case of observation equivalence is applied first because it is fast, and other abstractions can be implemented more efficiently for τ-loop free automata. After τ-Loop Removal, markings are removed by applying the Marking Removal and ω-Removal Rules. Next come the relatively fast Coreachability Rule, the Silent Incoming Rule, and the Only Silent Outgoing Rule, followed by the partitioning rules, namely Weak Observation Equivalence, Non-α Determinisation, and α-Determinisation. Finally, markings are added back into the automaton by applying the Marking Removal Rule in reverse, to enable early termination in trivial cases where all states are marked.

The **GNB** Abstraction Sequence is also used to verify standard nonblocking, simply by first adding α-markings to all states. When verifying standard nonblocking, the Coreachability Rule and the α-Determinisation Rule are not used, because they can be shown to have no effect or no effect beyond weak observation equivalence when starting from a standard nonblocking verification problem. Also, the **MinS**$^{\alpha}$ and **MinSync**$^{\alpha}$ selection heuristics only make sense for generalised nonblocking.

For the standard nonblocking test cases, the **GNB** Abstraction Sequence is compared to a Standard Nonblocking Abstraction Sequence (**NB**) based on [7]. After τ-Loop and Marking Removal, this sequence applies standard nonblocking versions of the Silent Incoming and Only Silent Outgoing Rules. Next are the Silent Continuation Rule, the Active Events Rules, and the Certain Conflicts Rule [7], which only work for standard nonblocking. Afterwards, the sequence completes with Weak Observation Equivalence and adding markings back in.

The results are furthermore compared to abstraction using only Weak Observation Equivalence (**WOEQ**) as proposed in [10]. This abstraction sequence only consists of τ-Loop Removal followed by Weak Observation Equivalence.

Table I shows experimental results for standard nonblocking verification using different abstraction sequences and candidate selection heuristics. Table II shows experimental results for verifying some Hierarchical Interface-Based Supervisory Control properties cast as generalised nonblocking verification problems [4], [5]. For each model, the tables show the total number of reachable states (Size) if known, and whether or not the model is nonblocking (Res). Then they show for each abstraction sequence and heuristic, the number of states in the largest automaton encountered during abstraction (Peak States), the number of states in the synchronous composition explored after abstraction (Final States), and the total verification time (Time). In case of early termination (no states marked α or all states marked ω), the final synchronous composition is not constructed and its size is shown as 0.

All experiments are run on a standard desktop computer using a single 3.3 GHz CPU and 8 GB of RAM. The experiments are controlled by state limits and timeouts. If during abstraction the synchronous composition of a candidate has more than 100,000 states, it is discarded and another candidate is chosen instead. The state limit for the final synchronous composition after abstraction is $5 \cdot 10^7$ states. If this limit is exceeded, or if the total runtime exceeds 15 minutes, the run is aborted and the corresponding table entries are left blank.

The tables show that compositional verification is highly sensitive to the selection heuristics. There is no clearly best strategy. For the standard nonblocking problems, **MinSync** usually is more effective, but it is outperformed by **MinS** and **MinS**$^{\alpha}$ for generalised nonblocking.

The Generalised Nonblocking Abstraction Sequence (**GNB**) tends to be more effective than the Standard Nonblocking (**NB**) and Weak Observation Equivalence (**WOEQ**) sequences. It can solve more problems within the set state limits, and usually produces smaller abstractions. Yet, the runtimes are not always better. Abstraction tends to take more time than synchronous composition exploration, so small models can often be solved faster with a weaker abstraction sequence.

It is interesting that the **GNB** sequence seems to work better than **NB** for standard nonblocking. Although **NB** includes the Silent Continuation, Active Events, and Certain Conflicts Rules, which do not work for generalised nonblocking, this is offset by the Non-α Determinisation Rule only present in **GNB**. It is to be noted, however, that the Certain Conflicts Rule does achieve abstractions not possible for generalised nonblocking, particularly for blocking models, which in some cases lead to early termination. On the other hand, the **NB** abstraction sequence suffers from the more complicated counterexample computation algorithm when the Certain Conflicts Rule is used.

The generalised nonblocking problems are more challenging. The tests in Table II represent different ways of verifying Serial Interface Consistency properties V and VI (**sic5** and **sic6**) for a subsystem of the Multiple-AIP model [31]. This version of the model can be parametrised, and the number in angle brackets in Table II indicates the number of pallets in each case. Serial Interface Consistency is transformed into generalised nonblocking according to [5]. This results in the addition of a *test automaton* to the model, which includes a large number of events, and which makes it difficult for the selection heuristics to identify suitable candidates. Nevertheless, the model has been successfully verified.

Fig. 8 shows a breakdown of the performance of the individual abstraction rules in the **GNB** sequence. The piecharts display the total numbers of states removed by abstraction and the total runtimes over all tests in the standard and generalised nonblocking test suites, using the **MinSync** heuristic. The runtimes only include abstractions and do not add up to the times in the tables, which also include candidate evaluation, synchronous composition, and counterexample computation.

It is clear that the partitioning abstraction rules (Weak Ob-

TABLE I
STANDARD NONBLOCKING EXPERIMENTS

Model	Size	Res	GNB/MinS			GNB/MinSync			NB/MinS			NB/MinSync			WOEQ/MinS			WOEQ/MinSync		
			Peak states	Final states	Time [s]	Peak states	Final states	Time [s]	Peak states	Final states	Time [s]	Peak states	Final states	Time [s]	Peak states	Final states	Time [s]	Peak states	Final states	Time [s]
agv	$2.57 \cdot 10^{7}$	true	319	472	0.2	133	290	0.4	319	503	0.3	133	348	0.4	319	503	0.2	133	348	0.3
agvb	$2.29 \cdot 10^{7}$	false	319	33	0.4	417	268	0.7	319	28	0.4	241	0	0.6	319	34	0.3	377	451	0.5
aip0aip	$1.02 \cdot 10^{9}$	true	933	5	1.0	226	66	2.5	1289	5	1.1	188	74	2.5	302	5	0.8	129	74	2.4
aip0alps	$3.00 \cdot 10^{8}$	false	172	115	0.4	134	65	0.8	18	16	0.4	13	9	0.7	244	144	0.4	244	144	0.8
aip0tough	$1.02 \cdot 10^{10}$	false	97849	21589427	74.3	73920	805068	14.7				177	0	2.3				15460	4471150	9.4
aip1efa$\langle 3 \rangle$	$6.88 \cdot 10^{8}$	true	40800	1619644	12.5	10734	2887502	10.4	40290	2382871	14.5	10872	3634908	12.6	57120	2081228	14.0	10872	3647013	12.2
aip1efa$\langle 16 \rangle$	$9.50 \cdot 10^{12}$	false	74100	14204058	21.9	74100	20072368	32.4	65520	15432924	26.0	74100	21123323	37.0	77220	15699764	22.8	77220	21015187	32.7
aip1efa$\langle 24 \rangle$	$1.83 \cdot 10^{13}$	false	5628	14231936	17.7	10734	20105881	27.7	6384	15472607	19.5	10872	21147356	28.0	6636	15699764	19.1	10872	21015187	27.2
big_bmw	$3.14 \cdot 10^{7}$	true	80	5	0.3	22	2	0.4	80	5	0.3	22	2	0.5	80	5	0.3	22	2	0.4
fencaiwon09	$1.03 \cdot 10^{8}$	true	11099	167	1.7	495	46	0.8	10566	105	2.0	495	41	0.8	11277	197	1.6	495	47	0.7
fencaiwon09b	$8.93 \cdot 10^{7}$	false	11099	96	2.3	495	96	1.1	10566	118	3.3	495	118	1.6	11277	211	1.9	495	211	1.0
ftechnik	$1.21 \cdot 10^{8}$	false	13160	0	2.9	152	0	0.9	8476	0	6.1	152	0	1.0	19538	0	3.1	226	0	0.7
profisafe_i4		true	49152	48205492	270.9	74088	409	46.3	49152	49223799	341.0	49152	49223799	365.3						
profisafe_i5		true	6144	237344	20.8	98304	57888	85.1	98304	57888	149.3	98304	57888	166.2						
profisafe_i6		true	55296	148284	53.7	18432	303353	33.5	55296	148284	114.0	18432	303353	45.6	55296	148284	53.4			
tbed_ctct	$3.94 \cdot 10^{13}$	false	36277	0	8.4	15612	0	5.9	66151	0	346.2	15039	0	289.7	36277	0	6.2	15612	0	5.2
tbed_hisc	$5.99 \cdot 10^{12}$	true	4140	167	2.5	788	216	3.7	2089	182	2.3	875	94	4.0	4188	33	2.1	846	33	3.7
tbed_valid	$3.01 \cdot 10^{12}$	true	62584	3286	9.5	4648	3286	3.4	84337	4004	14.6	4640	3374	3.4	71228	4368	10.8	4648	4344	3.0
tip3	$2.27 \cdot 10^{11}$	true	6399	211	3.4	576	128	1.0	14142	306	6.2	768	128	1.0	90132	210064	573.7	77588	210064	348.7
tip3_bad	$5.25 \cdot 10^{10}$	false	1293	719	1.9	784	366	1.6	1568	26	1.4	784	366	1.5	12385	23646	7.0	12385	23646	7.5
verriegel3	$9.68 \cdot 10^{8}$	true	3540	2	1.7	635	2	0.9	6084	2	2.2	2943	2	1.6	7115	2	1.8	3790	2	1.6
verriegel3b	$1.32 \cdot 10^{9}$	false	3753	0	1.8	672	39	1.7	4650	0	2.2	27	0	1.1	9048	4	2.1	925	245	1.7
verriegel4	$4.59 \cdot 10^{10}$	true	2724	2	1.3	635	2	1.1	6084	2	2.3	4125	2	1.9	7115	2	1.8	3790	2	1.7
verriegel4b	$6.26 \cdot 10^{10}$	false	3753	0	2.3	668	39	1.8	4650	0	3.1	27	0	1.3	9048	4	2.4	925	245	1.9
6linka	$2.45 \cdot 10^{14}$	false	559	33	1.2	8379	8	6.0	64	0	0.5	61	0	0.6	9836	60	13.8	12998	61	801.9
6linki	$2.75 \cdot 10^{14}$	false	636	65	1.2	11664	5	9.5	61	0	0.4	32	0	0.5	11595	1567	533.2			
6linkp	$4.43 \cdot 10^{14}$	false	88	0	0.3	30	0	0.5	32	0	0.4	16	0	0.5	12629	122	714.6	12629	122	776.3
6linkre	$6.21 \cdot 10^{13}$	false	362	678	1.1	584	424	1.4	118	44	1.0	29	13	1.0	669	684	1.2	737	211	1.3

TABLE II
GENERALISED NONBLOCKING EXPERIMENTS

Model	Size	Res	GNB/MinS			GNB/MinS$^{\alpha}$			GNB/MinSync			GNB/MinSync$^{\alpha}$			WOEQ/MinS			WOEQ/MinS$^{\alpha}$			WOEQ/MinSync			WOEQ/MinSync$^{\alpha}$		
			Peak states	Final states	Time [s]	Peak states	Final states	Time [s]	Peak states	Final states	Time [s]	Peak states	Final states	Time [s]	Peak states	Final states	Time [s]	Peak states	Final states	Time [s]	Peak states	Final states	Time [s]	Peak states	Final states	Time [s]
aip1efa_sic5a$\langle 3 \rangle$	$1.38 \cdot 10^{9}$	true	50904	3349340	29.4	5628	3214900	10.1	10734	5335633	16.2	10734	5335633	16.2	67032	4610864	44.8	6636	4662042	12.8	10872	7294014	20.1	10872	7294014	20.5
aip1efa_sic5a$\langle 4 \rangle$	$1.36 \cdot 10^{10}$	false	8112	11835409	16.3	50904	9394977	35.0	10734	18722528	26.5	10734	18276252	26.4	8424	15300064	18.7	67032	11053503	46.7	10872	20681543	27.1	10872	20681543	27.3
aip1efa_sic5a$\langle 6 \rangle$	$3.29 \cdot 10^{11}$	false	17100	23904005	28.8	50904	19946564	44.6	17100	36907733	48.8	17100	36907733	49.0	17784	32127224	34.9	67032	22307169	57.9	17784	42116470	51.4	17784	42116470	52.0
aip1efa_sic5a$\langle 8 \rangle$	$2.11 \cdot 10^{12}$	false	29400	23929636	29.4	29400	23929636	29.1	29400	36926515	49.6	29400	36926515	49.8	30600	32183217	35.8	30600	32183217	35.6	30600	42149449	52.3	30600	42149449	52.8
aip1efa_sic5a$\langle 10 \rangle$	$5.81 \cdot 10^{12}$	false	45012	23929636	29.9	45012	23929636	30.1	45012	36926515	50.5	45012	36926515	50.7	46872	32183217	36.7	46872	32183217	36.9	46872	42149449	53.4	46872	42149449	53.9
aip1efa_sic5a$\langle 16 \rangle$	$1.90 \cdot 10^{13}$	false	5628	28693973	32.5	5628	28693973	32.3	10734	40404808	52.7	10734	40404808	51.8	6636	32183217	34.9	6636	32183217	34.5	10872	42149449	50.8	10872	42149449	51.6
aip1efa_sic5b$\langle 3 \rangle$	$1.38 \cdot 10^{9}$	true	50904	3349340	29.7	5628	2886892	9.3	36160	5516756	26.4	10734	4896286	15.3	67032	4610864	44.2	6636	4662042	12.5	49280	7229794	33.5	10872	7294014	20.7
aip1efa_sic5b$\langle 4 \rangle$	$1.36 \cdot 10^{10}$	true	50904	23809874	75.4	7776	18511198	44.5	36160	46989160	140.5	10734	37979398	102.9	67032	34478924	111.2	8100	35644052	83.0						
aip1efa_sic5c$\langle 3 \rangle$	$6.88 \cdot 10^{8}$	true	26292	1643376	11.3	26292	1643376	10.6	10734	3244243	11.9	10734	3244243	12.5	31332	2972172	12.5	31332	2972172	12.9	10872	4362906	13.7	10872	4362906	14.3
aip1efa_sic5c$\langle 4 \rangle$	$6.82 \cdot 10^{9}$	true	26292	11475543	31.6	26292	11475543	32.0	10734	28422025	80.1	10734	18989723	53.5	31332	23854178	59.5	31332	23854178	59.2	10872	40945156	108.4	10872	33592307	89.9
aip1efa_sic6$\langle 3 \rangle$	$1.38 \cdot 10^{9}$	true	40320	916915	11.4	5628	1443446	6.5	10734	2448155	9.7	10734	2448155	10.4	60480	2081228	14.1	6636	2331031	8.1	10872	3647013	12.2	10872	3647013	13.1
aip1efa_sic6$\langle 4 \rangle$	$1.36 \cdot 10^{10}$	true	50904	11904937	48.4	69120	4784638	24.2	10734	18989711	53.5	10734	18989699	53.5	67032	17239472	72.2	67032	17239472	71.5	10872	33592295	89.7	10872	33592295	90.4
aip1efa_sic6$\langle 5 \rangle$	$8.18 \cdot 10^{10}$	true	8001	38110689	95.4																					

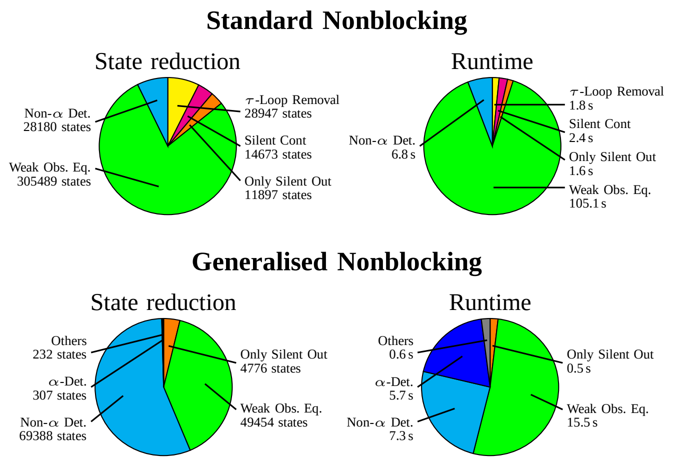

Fig. 8. Performance of individual rules.

servation Equivalence and Non-α-Determinisation) contribute to most of the runtime, but also achieve most of the state space reduction. Non-α-Determinisation has a significant impact when verifying generalised nonblocking with about the same effort as Weak Observation Equivalence—its runtime is shorter mainly due to the fact that Non-α-Determinisation is invoked later and therefore receives smaller input automata. The effect of α-Determinisation is small compared to the effort, probably due to its implementation that does not allow it to do much more than Weak Observation Equivalence. Other rules such as τ-Loop Removal and the Only Silent Outgoing Rule achieve a small reduction of states, but they run so fast that their application is still worthwhile.

V. CONCLUSIONS

A compositional method for verification of standard and generalised nonblocking has been described. The approach mitigates state-space explosion by simplifying individual components of a system before or while composing them. Eight abstraction rules preserving generalised nonblocking equivalence have been proposed, which substantially reduce the number of states of automata encountered during verification. The rules are chosen to be computationally feasible, while still covering a wide range of situations encountered in nondeterministic automata. The method has been implemented, and experimental results demonstrate its usefulness for model checking large discrete event systems models. While originally developed specifically for generalised nonblocking, the results are also applicable to standard nonblocking, and the new implementation brings improvements over previous compositional methods to verify standard nonblocking.

References

[1] R. Malik and R. Leduc, "Generalised nonblocking," in *Proc. 9th Int. Workshop on Discrete Event Systems, WODES '08*, Göteborg, Sweden, May 2008, pp. 340–345.

[2] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Kluwer, Sept. 1999.

[3] P. J. G. Ramadge and W. M. Wonham, "The control of discrete event systems," *Proc. IEEE*, vol. 77, no. 1, pp. 81–98, Jan. 1989.

[4] R. J. Leduc, B. A. Brandin, M. Lawford, and W. M. Wonham, "Hierarchical interface-based supervisory control—part I: Serial case," *IEEE Trans. Autom. Control*, vol. 50, no. 9, pp. 1322–1335, Sept. 2005.

[5] R. Leduc and R. Malik, "A compositional approach for verifying hierarchical interface-based supervisory control," in *Proc. 10th Int. Workshop on Discrete Event Systems, WODES '10*, Berlin, Germany, 2010, pp. 114–120.

[6] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT Press, 2008.

[7] H. Flordal and R. Malik, "Compositional verification in supervisory control," *SIAM J. Control and Optimization*, vol. 48, no. 3, pp. 1914–1938, 2009.

[8] R. Malik, D. Streader, and S. Reeves, "Conflicts and fair testing," *Int. J. Found. Comput. Sci.*, vol. 17, no. 4, pp. 797–813, 2006.

[9] P. N. Pena, J. E. R. Cury, and S. Lafortune, "Verification of nonconflict of supervisors using abstractions," *IEEE Trans. Autom. Control*, vol. 54, no. 12, pp. 2803–2815, 2009.

[10] R. Su, J. H. van Schuppen, J. E. Rooda, and A. T. Hofkamp, "Nonconflict check by using sequential automaton abstractions based on weak observation equivalence," *Automatica*, vol. 46, no. 6, pp. 968–978, June 2010.

[11] S. Ware and R. Malik, "Compositional nonblocking verification using annotated automata," in *Proc. 10th Int. Workshop on Discrete Event Systems, WODES '10*, Berlin, Germany, 2010, pp. 374–379.

[12] ——, "A process-algebraic semantics for generalised nonblocking," in *Proc. CATS 2011—Computing: The Australasian Theory Symposium*, Perth, Australia, 2011, pp. 75–84.

[13] R. Malik and R. Leduc, "A compositional approach for verifying generalised nonblocking," in *Proc. 7th Int. Conf. Control and Automation, ICCA '09*, Christchurch, New Zealand, Dec. 2009, pp. 448–453.

[14] ——, "Seven abstraction rules preserving generalised nonblocking," Working Paper 07/2009, Dept. of Computer Science, University of Waikato, Hamilton, New Zealand, 2009. [Online]. Available: http://hdl.handle.net/10289/2908

[15] R. Francis, "An implementation of a compositional approach for verifying generalised nonblocking," Working Paper 04/2011, Dept. of Computer Science, University of Waikato, Hamilton, New Zealand, 2011. [Online]. Available: http://hdl.handle.net/10289/5312

[16] R. Su, J. H. van Schuppen, and J. E. Rooda, "Aggregative synthesis of distributed supervisors based on automaton abstraction," *IEEE Trans. Autom. Control*, vol. 55, no. 7, pp. 1267–1640, July 2010.

[17] E. M. Clarke, Jr., O. Grumberg, and D. A. Peled, *Model Checking*. MIT Press, 1999.

[18] M. H. de Queiroz amd J. E. R. Cury and S. Lafortune, "Multitasking supervisory control of discrete-event systems," *Discrete Event Dyn. Syst.*, vol. 15, no. 4, pp. 375–395, 2005.

[19] C. A. R. Hoare, *Communicating Sequential Processes*. Prentice-Hall, 1985.

[20] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 2001.

[21] P. Malik, "From supervisory control to nonblocking controllers for discrete event systems," Ph.D. dissertation, University of Kaiserslautern, Kaiserslautern, Germany, 2003.

[22] R. Milner, *A Calculus of Communicating Systems*, ser. LNCS. Springer, 1980, vol. 92.

[23] J.-C. Fernandez, "An implementation of an efficient algorithm for bisimulation equivalence," *Science of Computer Programming*, vol. 13, pp. 219–236, 1990.

[24] T. Bolognesi and S. A. Smolka, "Fundamental results for the verification of observational equivalence: a survey," in *Protocol Specification, Testing and Verification VII: Proc. IFIP WG6.1 7th Int. Conf. Protocol Specification, Testing and Verification*, H. Rudin and C. H. West, Eds. Amsterdam, The Netherlands: North Holland, 1987, pp. 165–179.

[25] E. Nuutila and E. Soisalon-Soininen, "On finding the strongly connected components in a directed graph," *Information Processing Letters*, vol. 49, no. 1, pp. 9–14, Jan. 1994.

[26] Y. Wen, J. Wang, and Z. Qi, "Reverse observation equivalence between labelled state transition systems," in *Proc. 1st Int. Colloquium on Theoretical Aspects of Computing, ICTAC '04*, Guiyang, China, Sept. 2004, pp. 204–219.

[27] K. Åkesson, M. Fabian, H. Flordal, and R. Malik, "Supremica—an integrated environment for verification, synthesis and simulation of discrete event systems," in *Proc. 8th Int. Workshop on Discrete Event Systems, WODES '06*, Ann Arbor, MI, USA, July 2006, pp. 384–385.

[28] J. O. Moody and P. J. Antsaklis, *Supervisory Control of Discrete Event Systems Using Petri Nets*. Kluwer, 1998.

[29] B. Brandin and F. Charbonnier, "The supervisory control of the automated manufacturing system of the AIP," in *Proc. Rensselaer's 4th Int. Conf. Computer Integrated Manufacturing and Automation Technology*, Troy, NY, USA, 1994, pp. 319–324.

[30] R. J. Leduc, "Hierarchical interface-based supervisory control," Ph.D. dissertation, Dept. of Electrical Engineering, University of Toronto, ON, Canada, 2002. [Online]. Available: http://www.cas.mcmaster.ca/∼leduc

[31] R. Song, "Symbolic synthesis and verification of hierarchical interface-based supervisory control," Master's thesis, Dept. of Computing and Software, McMaster University, Hamilton, ON, Canada, 2006. [Online]. Available: http://www.cas.mcmaster.ca/∼leduc

[32] L. Feng, K. Cai, and W. M. Wonham, "A structural approach to the non-blocking supervisory control of discrete-event systems," *Int. J. Adv. Manuf. Technol.*, vol. 41, pp. 1152–1168, 2009.

[33] C. Lewerentz and T. Linder, *Case Study "Production Cell"*, ser. LNCS. Springer, 1995, vol. 891.

[34] R. Malik and R. Mühlfeld, "A case study in verification of UML statecharts: the PROFIsafe protocol," *J. Universal Computer Science*, vol. 9, no. 2, pp. 138–151, Feb. 2003.

[35] R. J. Leduc, "PLC implementation of a DES supervisor for a manufacturing testbed: An implementation perspective," Master's thesis, Dept. of Electrical Engineering, University of Toronto, ON, Canada, 1996. [Online]. Available: http://www.cas.mcmaster.ca/∼leduc

[36] A. Hinze, P. Malik, and R. Malik, "Interaction design for a mobile context-aware system using discrete event modelling," in *Proc. 29th Australasian Computer Science Conf., ACSC '06*, Hobart, Australia, 2006, pp. 257–266.

[37] KorSys Project. [Online]. Available: http://www4.in.tum.de/proj/korsys/

Robi Malik received the M.S. and Ph.D. degree in computer science from the University of Kaiserslautern, Germany, in 1993 and 1997, respectively. From 1998 to 2002, he worked in a research and development group at Siemens Corporate Research in Munich, Germany, where he was involved in the development and application of modelling and analysis software for discrete event systems. Since 2003, he is lecturing at the Department of Computer Science at the University of Waikato in Hamilton, New Zealand. He is participating in the development of the Supremica software for modelling and analysis of discrete event systems. His current research interests are in the area of model checking and synthesis of large discrete event systems and other finite-state machine models.

Ryan Leduc (M'02) received the B.Eng degree in electrical engineering from the University of Victoria in 1993, and the M.A.Sc and Ph.D. in electrical engineering from the University of Toronto in 1996 and 2002.

In 1997 and 1998, he was a Guest Scientist at Siemens Corporate Technology, Munich, Germany. In 2001, he joined McMaster University where he is currently an Associate Professor in the Department of Computing and Software.

Leduc was the Chair of the IEEE Control Systems Society Technical Committee on Discrete Event Systems from 2005–2011. He is the project leader for the DES software research tool, DESpot. His research interests include Supervisory Control of discrete-event systems (DES), hierarchical structure, concurrency and implementation issues, and DES as software and hardware. He is also interested in hierarchical approaches to formal verification of software and hardware. He is a licensed Professional Engineer in the province of Ontario, Canada.