Advanced Selfloop Removal in Compositional Nonblocking Verification of Discrete Event Systems

Robi Malik

***Abstract*— This paper investigates possible improvements of abstraction to simplify finite-state machines during compositional nonblocking verification of large discrete event systems. Current methods to simplify finite-state machines depend on the absence of transitions from the states to be simplified, and selfloop transitions, i.e., transitions with the same source and target state, are a common culprit that prevents simplification. Some methods to remove such selfloops are known, but they require events that appear on selfloops in the entire finite-state machine to be simplified. The methods described in this paper improve on this, because they allow for the removal of individual selfloop transitions from a finite-state machine while preserving conflict equivalence. This makes it possible to remove more transitions, thus reducing the computational effort of compositional nonblocking verification. Two abstraction rules are proposed, and experimental results show the potential of improvement over previously used methods.**

***Index Terms*— Discrete event systems, finite-state machines, nonblocking, model checking, compositional verification.**

I. Introduction

The *nonblocking property* is a weak liveness property commonly used in *supervisory control theory* of discrete event systems to express the absence of livelocks or deadlocks [1], [2]. This is a crucial property of safety-critical control systems, and with the increasing size and complexity of these systems, there is an increasing need to verify the nonblocking property automatically. The standard method to check whether a system is nonblocking involves the explicit composition of all the components involved, and is limited by the well-known *state-space explosion* problem. *Symbolic model checking* has been used successfully to reduce the amount of memory required by representing the state space symbolically rather than enumerating it explicitly [3].

Compositional verification [4], [5] is an effective alternative that can be used independently of or in combination with symbolic methods. Compositional verification works by *abstracting* or *simplifying* individual components of a large system, gradually reducing the state space and allowing larger systems to be verified in the end. When applied to the nonblocking property, compositional verification requires specific abstraction methods [6], [7]. A suitable theory is laid out in [8], where *conflict equivalence* is shown to be the most general process-algebraic equivalence to preserve the nonblocking property. Various abstraction rules preserving conflict equivalence have been proposed [6], [7], [9], [10]. Generalisations beyond conflict equivalence that take context information into account have also been investigated [11].

The author is with the Department of Computer Science, The University of Waikato, Hamilton, New Zealand (robi@waikato.ac.nz).

Several conflict-preserving abstraction rules have specific requirements about the presence or absence of transitions in the states to be simplified, and the presence of *selfloop* transitions, i.e., transitions with the same source and target states, has often been observed to prevent simplification. *Selfloop Removal* [11] can remove an event entirely from a system or a component, provided that it appears only on selfloops. This paper proposes two further abstraction rules, referred to as the Selfloop Subsumption and Weak Active Events Rules, which can remove individual selfloops from a finite-state machine, even if the event also appears on non-selfloop transitions.

In the following, Section II introduces the background of finite-state machines, the nonblocking property, and compositional verification. Next, Section III describes the Selfloop Subsumption Rule, and Section IV describes the Weak Active Events Rule, which are the contributions of this paper. Small examples highlight the benefits of these rules and show how they work. Afterwards, Section V presents experimental results, and Section VI adds concluding remarks.

II. Preliminaries

A. Events and Languages

Event sequences and languages are a simple means to describe discrete system behaviours [1], [2]. Their basic building blocks are *events*, which are taken from a finite *alphabet* Σ. The *silent event* τ labels transitions that are only taken by the component under consideration, and the *termination event* ω shows completion of a task. These special events are never included in an alphabet Σ unless mentioned explicitly using notation such as $\Sigma_\omega = \Sigma \cup \{\omega\}$ or $\Sigma_{\tau,\omega} = \Sigma \cup \{\tau, \omega\}$.

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

B. Finite-State Machines

System behaviours are modelled using finite-state machines. Typically, system models are deterministic, but abstraction may result in nondeterminism.

Definition 1: A (nondeterministic) *finite-state machine (FSM)* is a tuple $G = \langle \Sigma, Q, \to, Q^\circ \rangle$ where Σ is a set of *events*, Q is a finite set of *states*, $\to \subseteq Q \times \Sigma_{\tau,\omega} \times Q$ is the *state transition relation*, and $Q^\circ \subseteq Q$ is the set of *initial states*.

The transition relation is written in infix notation $x \xrightarrow{\sigma} y$, and is extended to traces $s \in \Sigma^*_{\tau,\omega}$ 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 $x \in X$. Also, $X \xrightarrow{s}$ for a state or state set X denotes the existence of a state $y \in Q$ such that $X \xrightarrow{s} y$.

The termination event $\omega \notin \Sigma$ denotes completion of a task and does not appear anywhere else but to mark such completions. States reached by ω do not have any outgoing transitions, i.e., if $x \xrightarrow{\omega} y$ then there does not exist $\sigma \in \Sigma_{\tau,\omega}$ such that $y \xrightarrow{\sigma}$. The traditional set of *terminal* states is $Q^\omega = \{\, x \in Q \mid x \xrightarrow{\omega} \,\}$ in this notation. For graphical simplicity, states in Q^ω are coloured black in the figures of this paper instead of explicitly showing ω-transitions.

To support silent events, another transition relation $\Rightarrow \subseteq Q \times \Sigma^*_\omega \times Q$ is introduced, where $x \overset{s}{\Rightarrow} y$ denotes the existence of a trace $t \in \Sigma^*_{\tau,\omega}$ such that $P(t) = s$ 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 $X \overset{s}{\Rightarrow} Y$ and $x \overset{s}{\Rightarrow}$ are defined analogously to $\rightarrow$.

An FSM is *τ-loop free* if, for all states $x \in Q$ and all traces $t \in \{\tau\}^*$ such that $x \xrightarrow{t} x$, it holds that $t = \varepsilon$. As every FSM can be transformed into an observation equivalent τ-loop free FSM [7], it is enough to consider τ-loop free FSMs for abstraction in this paper.

Definition 2: Let $G = \langle \Sigma_G, Q_G, \rightarrow_G, Q^\circ_G \rangle$ and $H = \langle \Sigma_H, Q_H, \rightarrow_H, Q^\circ_H \rangle$ be two FSMs. The *synchronous composition* of G and H is

$$G \parallel H \;=\; \langle \Sigma_G \cup \Sigma_H, Q_G \times Q_H, \rightarrow, Q^\circ_G \times Q^\circ_H \rangle \;, \tag{1}$$

where

- $(x_G, x_H) \xrightarrow{\sigma} (y_G, y_H)$ if $\sigma \in (\Sigma_G \cap \Sigma_H) \cup \{\omega\}$, $x_G \xrightarrow{\sigma}_G y_G$, and $x_H \xrightarrow{\sigma}_H y_H$;
- $(x_G, x_H) \xrightarrow{\sigma} (y_G, x_H)$ if $\sigma \in (\Sigma_G \setminus \Sigma_H) \cup \{\tau\}$ and $x_G \xrightarrow{\sigma}_G y_G$;
- $(x_G, x_H) \xrightarrow{\sigma} (x_G, y_H)$ if $\sigma \in (\Sigma_H \setminus \Sigma_G) \cup \{\tau\}$ and $x_H \xrightarrow{\sigma}_H y_H$.

FSMs are synchronised using lock-step synchronisation [12]. Shared events (including ω) must be executed by all FSMs synchronously, while other events (including τ) are executed independently.

A common method to simplify an FSM is to construct its *quotient* modulo an equivalence relation on the state set. An *equivalence relation* is a binary relation that is reflexive, symmetric and transitive. Given an equivalence relation $\sim$ on a set Q, the *equivalence class* of $x \in Q$ with respect to $\sim$, denoted $[x]$, is defined as $[x] = \{\, x' \in Q \mid x' \sim x \,\}$. An equivalence relation on a set Q partitions Q into the set $Q/\!\sim\; = \{\, [x] \mid x \in Q \,\}$ of its equivalence classes.

Definition 3: Let $G = \langle \Sigma, Q, \rightarrow, Q^\circ \rangle$ be an FSM, and let $\sim \subseteq Q \times Q$ be an equivalence relation. The *quotient* of G modulo $\sim$ is $G/\!\sim\; = \langle \Sigma, Q/\!\sim, \rightarrow/\!\sim, \tilde{Q}^\circ \rangle$, where $\rightarrow/\!\sim\; = \{\, ([x], \sigma, [y]) \mid x \xrightarrow{\sigma} y \,\}$ and $\tilde{Q}^\circ = \{\, [x^\circ] \mid x^\circ \in Q^\circ \,\}$.

C. Compositional Nonblocking Verification

The key liveness property in supervisory control theory is the *nonblocking* property [2]. An FSM is nonblocking if, from every reachable state, a terminal state can be reached.

Definition 4: [8] An FSM $G = \langle \Sigma, Q, \rightarrow, Q^\circ \rangle$ is *nonblocking* if, for every state $x \in Q$ and every trace $s \in \Sigma^*$ such that $Q^\circ \overset{s}{\Rightarrow} x$, there exists a trace $t \in \Sigma^*$ such that $x \overset{t\omega}{\Rightarrow}$; otherwise G is *blocking*.

To reason about this property in a compositional way, the notion of *conflict equivalence* is used [8]. According to process-algebraic testing theory, two FSMs are considered as equivalent if they both respond in the same way to tests [13]. For *conflict equivalence*, a *test* is an arbitrary FSM, and the *response* is the observation whether the test composed with the FSM in question is nonblocking or not.

Definition 5: [8] Two FSMs G and H are *conflict equivalent*, written $G \simeq_{\mathrm{conf}} H$, if, for any FSM T, $G \parallel T$ is nonblocking if and only if $H \parallel T$ is nonblocking.

When verifying whether a composed system of FSMs

$$G_1 \parallel G_2 \parallel \cdots \parallel G_n \;, \tag{2}$$

is nonblocking, compositional methods [6], [7] avoid building the full synchronous composition immediately. Instead, individual FSMs G_i are simplified and replaced by smaller conflict equivalent FSMs $H_i \simeq_{\mathrm{conf}} G_i$. If no simplification is possible, a subsystem $(G_j)_{j \in J}$ is selected and replaced by its synchronous composition, which then may be simplified.

The soundness of this approach is justified by the *congruence* properties [8] of conflict equivalence. For example, if G_1 in (2) is replaced by $H_1 \simeq_{\mathrm{conf}} G_1$, then by considering $T = G_2 \parallel \cdots \parallel G_n$ in Def. 5, it follows that the abstracted system $H_1 \parallel T = H_1 \parallel G_2 \parallel \cdots \parallel G_n$ is nonblocking if and only if the original system (2) is.

A component G_1 in a system such as (2) typically contains events that appear only in G_1 and not in the remainder $T = G_2 \parallel \cdots \parallel G_n$ of the system. These so-called *local* events are abstracted using *hiding*, i.e., they are replaced by the silent event τ. This paper is concerned about methods to simplify FSMs with silent transitions in such a way that conflict equivalence is preserved.

III. Selfloop Subsumption

Several abstraction rules to simplify FSMs while preserving conflict equivalence are known [6], [7], [9], [10]. One of these rules, called the *Only Silent Outgoing Rule* [6], allows for the removal of a state that has no outgoing transitions except transitions labelled by the silent event τ.

Example 1: Consider FSM G_1 in Fig. 1. State 1 has only two outgoing transitions, which are both labelled by the silent event τ. This state can be removed after redirecting its incoming transition $0 \xrightarrow{\alpha} 1$ to both its τ-successor states, i.e., after adding the transitions $0 \xrightarrow{\alpha} 2$ and $0 \xrightarrow{\alpha} 3$. Fig. 1 shows the resultant abstraction H_1, which is known to be conflict equivalent to G_1.

The *Only Silent Outgoing Rule* in Example 1, while useful for conflict-preserving abstraction [6], requires states with no

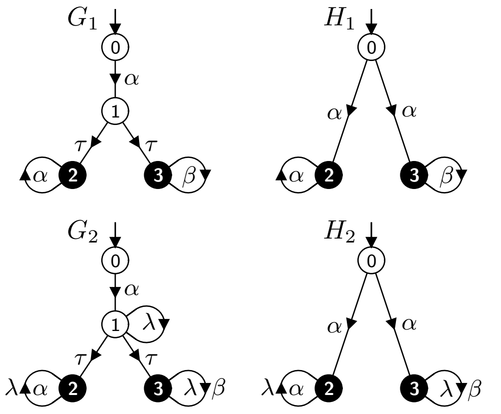

Fig. 1. Only Silent Outgoing abstraction.

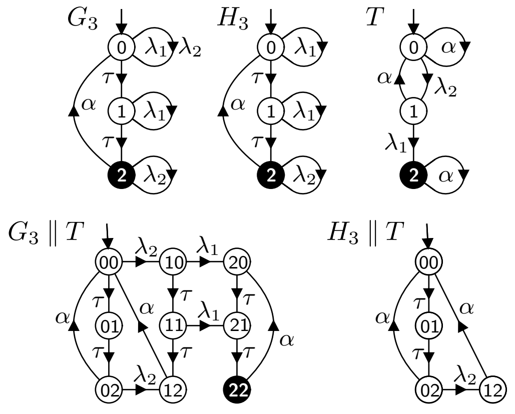

Fig. 2. Counterexample to selfloop subsumption.

other outgoing transitions except τ-transitions, which is an unnecessarily restrictive requirement.

Example 2: Consider FSM G_2 in Fig. 1. State 1 has the outgoing transition $1 \xrightarrow{\lambda} 1$, which is *not* silent, and therefore the Only Silent Outgoing Rule is not applicable to this state.

Nevertheless, FSM G_2 is conflict equivalent to H_2, which would result from removing state 1 using the Only Silent Outgoing Rule. The selfloop in state 1 is irrelevant for the purpose of conflict equivalence, because the successor states 2 or 3 also have selfloops labelled λ. Therefore, any continuation from state 1 to a terminal state can be rearranged by first using the silent transition to state 2 or 3 and executing any λ events needed for termination in that state. This idea is captured by the following definition.

Definition 6: Let $G = \langle \Sigma, Q, \rightarrow, Q^\circ \rangle$, and let $\Lambda \subseteq \Sigma$ and $p \in Q$. Then G is said to have *subsumed* Λ*-selfloops* at state p, if the following conditions hold.

(SS1) $p \xrightarrow{\lambda} p$ for all $\lambda \in \Lambda$.
(SS2) If $p \xrightarrow{\sigma} x$ for some $\sigma \in \Sigma_\omega$ and $x \in Q$, then $\sigma \in \Lambda$ and $x = p$.
(SS3) For every path $p \xrightarrow{\tau} p_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} p_k$ with $p_k \xrightarrow{\sigma}$ for some $\sigma \in \Sigma_\omega \setminus \Lambda$ or $p_k \xrightarrow{\lambda} r$ for some $\lambda \in \Lambda$ and $p_k \neq r$, there exists $1 \leq j \leq k$ such that $p_j \overset{\lambda}{\Rightarrow} p_j$ for all $\lambda \in \Lambda$.

Conditions (SS1) and (SS2) require that the state p has the selfloops by the events in Λ as its only outgoing transitions, apart from silent τ-transitions. Condition (SS3) requires that these selfloops also appear together on every path of τ-transitions originating from state p, before any other event becomes possible. Under these conditions, the following operation of selfloop subsumption is used to remove the selfloops from the start state p.

Definition 7: Let $G = \langle \Sigma, Q, \rightarrow, Q^\circ \rangle$, and let $\Lambda \subseteq \Sigma$ and $p \in Q$. The result of *selfloop subsumption* is $G \ominus (p; \Lambda) = \langle \Sigma, Q, \rightarrow_\ominus, Q^\circ \rangle$ where $\rightarrow_\ominus = \rightarrow \setminus (\{p\} \times \Lambda \times \{p\})$.

Example 3: FSM G_2 in Fig. 1 has subsumed $\{\lambda\}$-selfloops at state 1, because both τ-successor states 2 and 3 also have $\{\lambda\}$-selfloops. The result of selfloop subsumption, $G_2 \ominus (1, \{\lambda\})$, is obtained by deleting the transition $1 \xrightarrow{\lambda} 1$ from G_2, and the result can be further simplified using the Only Silent Outgoing Rule to obtain the abstraction H_2.

It is important to note in Def. 6 that the events of *all* selfloops on state p are collected in the set Λ: either all these selfloops are removed together, or none can be removed.

Example 4: FSM G_3 in Fig. 2 does not have subsumed $\{\lambda_1, \lambda_2\}$-selfloops at state 1. Condition (SS3) is not satisfied, because there is no state τ-reachable from state 1 with both events λ_1 and λ_2 on selfloops, except for state 1 itself.

Note that removing only the λ_2-selfloop from state 1 results in H_3, which is *not* conflict equivalent to G_3. Fig. 2 shows an FSM T and the synchronous composition $G_3 \parallel T$, which is nonblocking, and $H_3 \parallel T$, which is blocking.

The following proposition confirms that selfloop subsumption as defined above indeed results in a conflict equivalent abstraction.

Proposition 1 (Selfloop Subsumption Rule): Let $G = \langle \Sigma, Q, \rightarrow, Q^\circ \rangle$ have subsumed Λ-selfloops at $p \in Q$. Then $G \simeq_{\text{conf}} G \ominus (p; \Lambda)$.

The selfloop subsumption rule is implemented by checking each state of the FSM with only selfloops and τ-transitions as outgoing transitions. For such states p, the set Λ of selfloops to be checked for removal is determined as the set of non-τ events enabled in the state. Then a search of the τ-successors is performed. For each τ-successor x, it is checked whether $x \overset{\lambda}{\Rightarrow} x$ for all $\lambda \in \Lambda$. If this is the case, the state x passes the check. Otherwise, it is determined whether $x \xrightarrow{\sigma}$ for some $\sigma \in \Sigma_\omega \setminus \Lambda$ or $x \xrightarrow{\lambda} y \neq x$ for some $\lambda \in \Lambda$, in which case state p does not have subsumed Λ-selfloops. If no such transition is found, the τ-successors of x must all be checked. If all checked states pass the test, then state p has subsumed Λ-selfloops.

As the removal of selfloops at a state may enable selfloop subsumption at its τ-predecessors, the order in which states are analysed can be important. If the FSM is τ-loop free, maximum effectiveness is achieved by first analysing the states with the fewest other states reachable by τ-transitions.

Complexity: The above algorithm checks $|Q|$ states of an FSM, each time visiting up to $|Q| - 1$ proper τ-successors. For each τ-successor, it has to explore up to $|Q|$ outgoing τ-transitions, and up $|\Sigma||Q|$ outgoing transitions with events from Σ. This gives a total of $O(|Q|^3|\Sigma|)$ operations.

The most complicated check, whether $x \overset{\lambda}{\Rightarrow} x$, needs to be performed only once per state x and event λ, as the result will not change for a τ-loop free FSM even when selfloops are removed. This requires computation of the transitive closure of τ-transitions ahead of time, which can be done in $O(|Q|^3)$ time [14], and using this each test for $x \overset{\lambda}{\Rightarrow} x$ is possible in $O(|Q|^2)$ time. In the worst case, the check is performed for each pair of state x and event λ, with total time complexity $O(|Q|^3|\Sigma|)$. This is the same as the above complexity to analyse the states, so it is the worst-case time complexity of the Selfloop Subsumption Rule.

IV. Active Events Abstraction

The *Active Events Rule* is another conflict-preserving abstraction [6], which merges states with exactly the same enabled events, if they are reached after some nondeterministic choice. This is justified by the fact that, to preserve the nonblocking property, only the traces leading to terminal states are important, which makes it possible to postpone a nondeterministic choice by one step. The requirement for states to be reached after a nondeterministic choice is expressed by the relation of *incoming equivalence*.

Definition 8: [6] Let $G = \langle \Sigma, Q, \rightarrow, Q^\circ \rangle$ be an FSM. The *incoming equivalence* relation $\sim_{\text{inc}} \subseteq Q \times Q$ is defined such that $y_1 \sim_{\text{inc}} y_2$ if the following conditions hold.

(IE1) $Q^\circ \overset{\varepsilon}{\Rightarrow} y_1$ if and only if $Q^\circ \overset{\varepsilon}{\Rightarrow} y_2$.

(IE2) For all states $x \in Q$ and all events $\sigma \in \Sigma_\omega$ it holds that $x \overset{\sigma}{\Rightarrow} y_1$ if and only if $x \overset{\sigma}{\Rightarrow} y_2$.

Two states are incoming equivalent, if either both or none are silently reachable from initial states (IE1), and if they have exactly the same incoming transitions, i.e., transitions with the same events and the same source states (IE2). Based on this, the Active Events Rule merges states that are incoming equivalent and have the same sets of active events.

Proposition 2 (Active Events Rule): [6] Let $G = \langle \Sigma, Q, \rightarrow, Q^\circ \rangle$ be an FSM, and let $\sim \subseteq \sim_{\text{inc}}$ be an equivalence relation on Q such that $x \sim y$ implies $x \overset{\sigma}{\Rightarrow}$ if and only if $y \overset{\sigma}{\Rightarrow}$ for all $\sigma \in \Sigma_\omega$. Then $G \simeq_{\text{conf}} G/{\sim}$.

Example 5: Consider FSM G_1 in Fig. 3. States 1 and 2 both only have incoming transitions from state 0 with event α, which is enough to establish incoming equivalence, and they both have the same active event, β. The Active Events Rule can be applied to merge states 1 and 2 into a single state 12 as shown in H_1.

While some simplification is achieved by the Active Events Rule, the condition of incoming equivalence is restrictive as it requires incoming transitions from exactly the same states. This rules out some desirable simplification, particularly when selfloops are involved.

Example 6: Consider FSM G_2 in Fig. 3. States 1 and 2 are *not* incoming equivalent, because $1 \overset{\alpha}{\Rightarrow} 1$ while $1 \overset{\alpha}{\Rightarrow} 2$ does not hold. Therefore, the Active Events Rule cannot be used to merge states 1 and 2 and obtain H_2.

Although H_2 cannot be obtained from G_2 by the Active Events Rule, FSMs G_2 and H_2 are conflict equivalent: similarly to Example 5, the nondeterministic choice between states 1 and 2 can be deferred as both states only have got

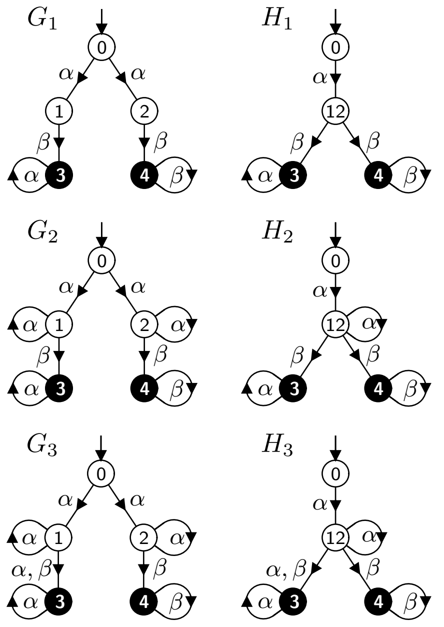

Fig. 3. Active Events abstraction.

outgoing transitions by the same events, and the selfloops with event α do not change the way how termination is possible only via β in both states.

As incoming equivalence is too restrictive, it is desirable to relax it by allowing for *equivalent* rather than identical source states of incoming transitions. This leads to *reverse weak bisimulation*, which has been used in compositional verification of the generalised nonblocking property [7].

Definition 9: [15] Let $G = \langle \Sigma, Q, \rightarrow, Q^\circ \rangle$. An equivalence relation $\sim \subseteq Q \times Q$ is a *reverse weak bisimulation* on G, if the following conditions hold for all $y_1, y_2 \in Q$ with $y_1 \sim y_2$.

(RB1) If $Q^\circ \overset{\varepsilon}{\Rightarrow} y_1$ then also $Q^\circ \overset{\varepsilon}{\Rightarrow} y_2$.

(RB2) For all states $x_1 \in Q$ and all $s \in \Sigma_\omega \cup \{\varepsilon\}$ such that $x_1 \overset{s}{\Rightarrow} y_1$ there exists a state $x_2 \in Q$ such that $x_2 \overset{s}{\Rightarrow} y_2$ and $x_1 \sim x_2$.

Condition (RB1) is the equivalent to (IE1) for incoming equivalence, while condition (RB2) differs in that it considers states to be equivalent if they have got incoming transitions with the same events, originating from states that are *equivalent* under the same relation. This is weaker than condition (IE2) for incoming equivalence, which requires *identical* source states. Reverse weak bisimulation allows for states 1 and 2 in FSM G_2 in Fig. 3 to be considered as equivalent.

Unfortunately, the Active Events Rule cannot be relaxed by replacing incoming equivalence with reverse weak bisimulation, as the following example shows.

Example 7: A relation that considers states 1 and 2 as equivalent is a reverse weak bisimulation on FSM G_3 in Fig. 3, and both states have the same active events α and β. Merging these states results in FSM H_3, which is *not* conflict equivalent to G_3. For example, if T is an FSM that always enables α and always disables β, then $G_3 \parallel T$ is blocking while $H_3 \parallel T$ is nonblocking.

The problem in Example 7 is caused by the transition $1 \xrightarrow{\alpha} 3$, which makes it possible to reach a different equivalence class by event α from state 1 but not from the equivalent state 2. This observation suggests the need to distinguish active events that lead to the same equivalence class only from active events that lead to a different equivalence class.

Definition 10: Let $G = \langle \Sigma, Q, \rightarrow, Q^\circ \rangle$. An equivalence relation $\sim \subseteq Q \times Q$ is a *weak active events equivalence* on G if for all $y_1, y_2 \in Q$ such that $y_1 \sim y_2$ the following conditions hold.

(WA1) If $Q^\circ \overset{\varepsilon}{\Rightarrow} y_1$ then also $Q^\circ \overset{\varepsilon}{\Rightarrow} y_2$.
(WA2) If $x \overset{\sigma}{\Rightarrow} y_1$, and $x \sim y_1$ does not hold, then $x \overset{\sigma}{\Rightarrow} y_2$.
(WA3) If $x_1 \overset{\sigma}{\Rightarrow} y_1$ and $x_1 \sim y_1$, then there exists $x_2 \sim x_1$ such that $x_2 \overset{\sigma}{\Rightarrow} y_2$.
(WA4) If $y_1 \overset{\sigma}{\Rightarrow}$ then $y_2 \overset{\sigma}{\Rightarrow}$.
(WA5) If $y_1 \overset{\sigma}{\Rightarrow} z_1$ for some $z_1 \notin [y_1]$ then $y_2 \overset{\sigma}{\Rightarrow} z_2$ for some $z_2 \notin [y_2]$

Condition (WA1) is equivalent to the first condition (IE1) for incoming equivalence, while the second condition (IE2) is replaced by two conditions (WA2) and (WA3). Condition (WA2) requires identical predecessor states in the same way as (IE2), but only for transitions originating from a different equivalence class. For transitions originating from the same equivalence class, condition (WA3) requires an equivalent predecessor state as reverse weak bisimulation does. Condition (WA4) requires equivalent states to have exactly the same active events, as it is also required in Prop. 2 for the Active Events Rule. In addition to that, condition (WA5) requires any active event that leads from a state to a different equivalence class to do so also from any equivalent state.

These conditions allow for states 1 and 2 of FSM G_2 in Example 6 to be equivalent, because for the transition $1 \xrightarrow{\alpha} 1$ there exists the transition $2 \xrightarrow{\alpha} 2$ that originates from an equivalent state, $1 \sim 2$. On the other hand, states 1 and 2 of FSM G_3 in Example 7 cannot be equivalent because of condition (WA5), since $1 \xrightarrow{\alpha} 3$, where state 3 is *not* equivalent to state 1, while the only α-transition from state 2, $2 \xrightarrow{\alpha} 2$, leads to a state in the same equivalence class.

The following result confirms that the merging of weakly active events equivalent states results in a conflict equivalent abstraction.

Proposition 3 (Weak Active Events Rules): Let $G = \langle \Sigma, Q, \rightarrow, Q^\circ \rangle$, and let $\sim \subseteq Q \times Q$ be a weak active events equivalence on G. Then $G \simeq_{\text{conf}} G/{\sim}$.

The Weak Active Events Rule is implemented by a partition refinement algorithm similar to [16]. In a first step, all states with equal active events and equal silent reachability from initial states are grouped together, producing a partition satisfying conditions (WA1) and (WA4). This partition is then refined to satisfy (WA2) by exploring the successors for each state and event and splitting the other equivalence classes by separating the states that can be reached from those that cannot. Afterwards, each pair of class and event is checked to see which states in the same class can be reached, if necessary splitting the classes to satisfy (WA3). Finally, each class and event is checked again to separate states that can reach a different class from those that cannot, ensuring that the partition satisfies (WA5). Whenever a class is split, all states of the two classes resulting from the split must be processed again to check for further splits according to (WA2), (WA3), and (WA5).

Unlike incoming equivalence and reverse weak bisimulation, there does not necessarily exist a coarsest weak active events equivalence for a given FSM. This is because condition (WA5) can be satisfied by splitting classes in different ways. The above algorithm results in a weak active events equivalence, but not necessarily in an optimal solution.

If some states have been found to be equivalent by the algorithm, the FSM quotient is computed to simplify the FSM. Afterwards, the algorithm starts over to compute another weak active events equivalence. This is because the merging of states can cause other states to satisfy condition (WA2), making it possible to merge more states.

Complexity: The computation of a weak active events equivalence involves splitting classes up to $|Q|$ times. After each split, in the worst case, all pairs of state and event need to be checked for conditions (WA2), (WA3), and (WA5). These are up to $3|Q||\Sigma|$ checks, each of which can be performed by exploring up to $|Q|$ successor states, assuming the relation $\Rightarrow$ is computed in advance and stored. This gives a worst-case time complexity of $O(|Q|^3|\Sigma|)$ to compute a weak active events equivalence. The complexity to compute the relation $\Rightarrow$ in advance is the same, the initial partitioning according to (WA1) and (WA4), the splitting of the classes, and the construction of the quotient FSM can be done in lower time complexity.

As the algorithm iterates and computes up to $|Q|$ weak active events equivalence relations, the worst-case time complexity to apply the Weak Active Events Rule to an FSM is $O(|Q|^4|\Sigma|)$.

V. Experimental Results

The abstraction rules proposed in this paper have been integrated in a compositional nonblocking verification algorithm in the discrete event systems tool Supremica [17]. Compositional nonblocking verification receives as input a system of FSMs (2), and repeatedly applies abstraction to individual components, or composes a few components if no abstraction is possible, until a final simplified system is verified using a standard monolithic nonblocking check [1]. The abstraction sequence [11] in the implementation [17] has been modified by inserting the Selfloop Subsumption and Weak Active Events Rules at appropriate positions.

The compositional algorithm without and with the new rules has been used to check the nonblocking property of 26 benchmark examples [6], [7], [11]. The test suite includes complex industrial models and case studies from application areas such as manufacturing systems, communication protocols, and automotive electronics.

Table I shows the results of the experiments. It shows for each test case the number of FSMs in the model (Aut),

TABLE I

EXPERIMENTAL RESULTS

				Previous Work [11]			Advanced Selfloop Treatment				
Model	Aut	State space	Res	Peak	Final	Time	Peak	Final	Time	SS	WA
aip0aip	117	$1.0\cdot10^{9}$	yes	226	66	1.7 s	226	66	1.7 s	316	0
aip0alps	35	$3.0\cdot10^{8}$	no	13	9	0.4 s	13	9	0.4 s	8	0
aip0tough	60	$1.0\cdot10^{10}$	no	177	0	1.3 s	177	0	1.5 s	44	16
aip1efa ⟨3⟩	50	$6.9\cdot10^{8}$	yes	10734	2887502	11.3 s	10759	2540103	11.3 s	157	1158
aip1efa ⟨16⟩	50	$9.5\cdot10^{12}$	no	74100	20072368	34.1 s	57395	7108974	119.1 s	2760	27063
aip1efa ⟨24⟩	50	$1.8\cdot10^{13}$	no	10734	20105881	28.3 s	10759	13867554	20.8 s	103	198
fencaiwon09	32	$1.0\cdot10^{8}$	yes	279	41	0.8 s	279	41	0.9 s	7	0
fencaiwon09b	31	$8.9\cdot10^{7}$	no	279	68	1.2 s	279	62	1.2 s	10	1
ftechnik	36	$1.2\cdot10^{8}$	no	152	0	1.2 s	152	0	1.2 s	129	3
profisafe_i4	82		yes	37044	4440	16.7 s	37044	2974	17.6 s	677	12
profisafe_i5	88		yes	98304	5414	58.7 s	98304	3643	63.2 s	948	14
profisafe_i6	94		yes	52224	300735	73.7 s	47736	179	36.0 s	1011	16
tbed_ctct	84	$3.9\cdot10^{13}$	no	15039	0	6.2 s	15039	0	6.5 s	727	774
tbed_hisc0	196	$6.0\cdot10^{12}$	yes	766	50	3.5 s	766	33	3.6 s	114	34
tbed_hisc1	184	$2.9\cdot10^{17}$	no	19	0	0.4 s	19	0	0.4 s	2	0
tbed_valid	84	$3.0\cdot10^{12}$	yes	4640	3019	3.6 s	4398	3019	3.8 s	456	126
tip3	58	$2.3\cdot10^{11}$	yes	192	0	1.1 s	192	0	1.2 s	46	0
tip3_bad	54	$5.2\cdot10^{10}$	no	192	0	1.0 s	192	0	1.1 s	45	0
verriegel3	53	$9.7\cdot10^{8}$	yes	636	2	1.6 s	774	2	1.6 s	134	27
verriegel3b	52	$1.3\cdot10^{9}$	no	27	0	0.6 s	27	0	0.6 s	1	1
verriegel4	65	$4.6\cdot10^{10}$	yes	636	2	1.7 s	774	2	1.7 s	209	33
verriegel4b	64	$6.3\cdot10^{10}$	no	27	0	0.7 s	27	0	0.7 s	2	2
6linka	53	$2.4\cdot10^{14}$	no	61	0	0.8 s	61	0	0.8 s	0	0
6linki	53	$2.7\cdot10^{14}$	no	32	0	0.5 s	32	0	0.6 s	0	0
6linkp	48	$4.4\cdot10^{14}$	no	16	0	0.4 s	16	0	0.4 s	0	0
6linkre	59	$6.2\cdot10^{13}$	no	29	0	0.8 s	29	0	0.8 s	0	0

the number of reachable states in the synchronous composition (State space) if known, and whether or not the model is nonblocking (Res). Then it shows, for the case with and without the Selfloop Subsumption and Weak Active Events Rules, the number of states of the largest FSM simplified (Peak), the number of states processed by monolithic nonblocking verification after simplification (Final), and the total verification time (Time). A final states number of 0 indicates that the algorithm has terminated early without a monolithic nonblocking check [11].

The table also shows the number of selfloops removed by Selfloop Subsumption (SS) and the number of states removed by the Weak Active Events Rule (WA). The latter are in addition to the original Active Events Rule [6], which is also contained in the abstraction sequence and executed before the Weak Active Events Rule.

The data shows that the Selfloop Subsumption and Active Events Rules can achieve additional simplification over the abstraction sequence [11], particularly for the large aip1 and profisafe models. The runtime is not always better, because the Weak Active Events Rule can be slow and its implementation is not optimised as much as the other abstractions. Yet, the reduction in the peak and final states numbers is encouraging, as these can be the bottleneck for compositional verification.

VI. CONCLUSIONS

Two conflict-preserving abstraction rules have been proposed that allow for simplification of FSMs during compositional nonblocking verification, particularly in cases where previously used methods [6], [7], [11] fail due to the presence of selfloops. Specifically, the Selfloop Subsumption Rule allows for the removal of individual selfloops from an FSM, and the Weak Active Events Rule allows for the merging of states with the same enabled events even though they are not incoming equivalent due to selfloops. Experimental results demonstrate that these rules provide for better abstraction during compositional nonblocking verification of large discrete event systems. In future work, it is of interest to generalise these rules beyond conflict equivalence, by taking into account how shared events are used in the rest of the system [11], and to develop similar rules for extended finite-state machines [18].

REFERENCES

[1] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd ed. New York, NY, USA: Springer Science & Business Media, 2008.

[2] 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.

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

[4] S. Graf and B. Steffen, "Compositional minimization of finite state systems," in *Proc. 1990 Workshop on Computer-Aided Verification*, ser. LNCS, vol. 531. New Brunswick, NJ, USA: Springer, June 1990, pp. 186–196.

[5] A. Valmari, "Compositionality in state space verification methods," in *Proc. 18th Int. Conf. Application and Theory of Petri Nets*, ser. LNCS, vol. 1091. Osaka, Japan: Springer, June 1996, pp. 29–56.

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

[7] R. Malik and R. Leduc, "Compositional nonblocking verification using generalised nonblocking abstractions," *IEEE Trans. Autom. Control*, vol. 58, no. 8, pp. 1–13, Aug. 2013.

[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] C. Pilbrow and R. Malik, "Compositional nonblocking verification with always enabled events and selfloop-only events," in *Proc. 2nd Int. Workshop on Formal Techniques for Safety-Critical Systems, FTSCS 2013*, Queenstown, New Zealand, 2013, pp. 147–162.

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

[13] R. De Nicola and M. C. B. Hennessy, "Testing equivalences for processes," *Theoretical Comput. Sci.*, vol. 34, no. 1–2, pp. 83–133, Nov. 1984.

[14] E. Nuutila, *Efficient Transitive Closure Computation in Large Digraphs*, ser. Acta Polytechnica Scandinavica, Mathematics and Computing in Engineering Series. Helsinki, Finland: Finnish Academy of Technology, 1995, vol. 74.

[15] 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.

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

[17] 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: IEEE, July 2006, pp. 384–385.

[18] S. Mohajerani, R. Malik, and M. Fabian, "An algorithm for compositional nonblocking verification of extended finite-state machines," in *Proc. 12th Int. Workshop on Discrete Event Systems, WODES '14*, Paris, France, May 2014, pp. 376–382.