A Process-Algebraic Semantics for Generalised Nonblocking

Simon Ware **Robi Malik**

Department of Computer Science
The University of Waikato
Private Bag 3105, Hamilton 3240, New Zealand
Email: {siw4,robi}@cs.waikato.ac.nz

Abstract

Generalised nonblocking is a weak liveness property to express the ability of a system to terminate under given preconditions. This paper studies the notions of equivalence and refinement that preserve generalised nonblocking and proposes a semantic model that characterises generalised nonblocking equivalence. The model can be constructed from the transition structure of an automaton, and has a finite representation for every finite-state automaton. It is used to construct a unique automaton representation for all generalised nonblocking equivalent automata. This gives rise to effective decision procedures to verify generalised nonblocking equivalence and refinement, and to a method to simplify automata while preserving generalised nonblocking equivalence. The results of this paper provide for better understanding of nonblocking in a compositional framework, with possible applications in compositional verification.

1 Introduction

Blocking or *conflicts* are common faults in the design of concurrent programs that can be very subtle and hard to detect (Dietrich et al. 2002, Wong et al. 2000). They have long been studied in the field of *discrete-event systems* (Cassandras & Lafortune 1999, Ramadge & Wonham 1989), which is applied to the modelling of complex, safety-critical systems. To improve the reliability of such systems, techniques are needed to detect the presence or verify the absence of blocking in models of an ever increasing size.

In discrete-events theory, the absence of blocking is formalised using the *nonblocking* property, which is used very successfully for *synthesis* (Cassandras & Lafortune 1999, Ramadge & Wonham 1989). A lot of research has been conducted to study the compositional semantics (Kumar & Shayman 1994, Malik et al. 2006) of nonblocking and its verification (Flordal & Malik 2009, Su et al. 2010). Despite its widespread use, the expressive powers of nonblocking are limited. To overcome its weaknesses, nonblocking has been modified and extended in several ways (Fabian & Kumar 1997, de Queiroz et al. 2004, Malik & Leduc 2008).

This paper is concerned about *generalised nonblocking* (Malik & Leduc 2008), which adds to standard nonblocking the ability to restrict the set of states from which blocking is checked. This is useful for the verification of software components and of certain conditions in Hierarchical Interface-Based Supervisory Control (Leduc et al. 2005, Leduc & Malik 2010).

In (Malik & Leduc 2009), a set of abstraction rules is proposed to simplify automata in such a way that generalised nonblocking equivalence is preserved. Although the approach provides a useful means for compositional verification, the abstraction rules are incomplete and limited in their reduction potential. In an attempt to develop more effective abstraction, this paper analyses generalised nonblocking using process-algebraic testing theory (Hennessy 1988, De Nicola & Hennessy 1984). It characterises generalised nonblocking equivalence and refinement using a process-algebraic semantic model, and provides an algorithm to construct a canonical automaton representation that can be used as a unique abstraction for all generalised nonblocking equivalent automata. These results pave the way towards more general means of abstraction than the local abstraction rules of (Malik & Leduc 2009), and in addition make it possible to reason about refinement.

The paper uses similar methods and ideas as previously used for *standard nonblocking* (Malik et al. 2006) and *fair testing* (Brinksma et al. 1995, Natarajan & Cleaveland 1995), yet the results are quite different. Generalised nonblocking semantics requires no interdependency between the possible completions associated with different states of an automaton. This leads to a simpler semantic model than in the case of standard nonblocking or fair testing, with finite representations and more straightforward algorithms.

This paper is organised as follows. Sect. 2 introduces the necessary background of nondeterministic automata and defines generalised nonblocking. Then Sect. 3 introduces a testing equivalence and preorder for generalised nonblocking, presents a semantic model, and proves results about its adequacy and finiteness. Afterwards, Sect. 4 describes the canonical automaton as a standardised normal form with respect to generalised nonblocking, and proposes an algorithm to construct it. Finally, Sect. 5 adds some concluding remarks.

2 Preliminaries

This section introduces the notations used throughout this paper. Dynamic systems are modelled using multi-coloured automata, with the possibility of nondeterminism, which naturally arises from abstraction and hiding (Hoare 1985, Roscoe 1997). System behaviour is described using languages, with notations taken from the background of discrete event systems and automata theory (Ramadge & Wonham 1989, Hopcroft et al. 2001).

2.1 Events and Languages

Event sequences and languages are a simple means to describe discrete system behaviours (Ramadge & Wonham 1989, Cassandras & Lafortune 1999). Their basic building blocks are *events*, which are taken from a finite *alphabet* Σ. In addition, the *silent event* $\tau \notin \Sigma$ is used, with the notation $\Sigma_\tau = \Sigma \cup \{\tau\}$.

Σ^* denotes the set of all finite *traces* or *strings* of the form $\sigma_1\sigma_2 \dots \sigma_n$ of events from Σ, including the *empty trace* ε. A subset $L \subseteq \Sigma^*$ is called a *language*. The *concatenation* of two traces $s, t \in \Sigma^*$ is written as st. Traces and languages can also be catenated, for example $sL = \{\, st \in \Sigma^* \mid t \in L \,\}$. The *prefix-closure* of a language L is $\overline{L} = \{\, s \in \Sigma^* \mid st \in L \text{ for some } t \in \Sigma^* \,\}$. *Natural projection* $P_\tau \colon \Sigma_\tau^* \to \Sigma^*$ is the operation that deletes all silent (τ) events from traces.

2.2 Multi-coloured Automata

Nondeterministic multi-coloured automata are used to model dynamic system behaviours. *Nondeterminism* is essential for the abstraction techniques in this paper. *Multi-coloured* automata extend the traditional concept of *marked states* to multiple simultaneous marking conditions, by labelling states with different *colours* or *propositions*. The generalised nonblocking condition is defined using these propositions. The following definition appears in (Malik & Leduc 2008), and is based on similar ideas in (Clarke et al. 1999, de Queiroz et al. 2004).

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

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 a state set $Q_1 \subseteq Q$, the notation $Q_1 \xrightarrow{s} x_2$ means that $x_1 \xrightarrow{s} x_2$ for some $x_1 \in Q_1$, and likewise $Q_1 \xrightarrow{s} Q_2$ means $x_1 \xrightarrow{s} x_2$ for some $x_1 \in Q_1$ and $x_2 \in Q_2$. Also, $x \to y$ denotes that $x \xrightarrow{s} y$ for some trace $s \in \Sigma_\tau^*$, and $x \xrightarrow{s}$ means $x \xrightarrow{s} y$ for some state $y \in Q$. Finally, $G \xrightarrow{s} x$ stands for $Q^\circ \xrightarrow{s} x$.

To support hiding of 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 $Q_1 \overset{s}{\Rightarrow} Q_2$, $x \Rightarrow y$, and $x \overset{s}{\Rightarrow}$ are defined analogously to $\to$.

For a state or state set x, the *continuation language* is defined as

$$\mathcal{L}(x) \;=\; \{\, s \in \Sigma^* \mid x \overset{s}{\Rightarrow} \,\} \;, \tag{1}$$

and likewise for $\pi \in \Pi$ the π-*marked language* is

$$\mathcal{L}^\pi(x) = \{\, s \in \Sigma^* \mid x \overset{s}{\Rightarrow} \Xi(\pi) \,\} \;. \tag{2}$$

The language and the π-marked language of the automaton G are $\mathcal{L}(G) = \mathcal{L}(Q^\circ)$ and $\mathcal{L}^\pi(G) = \mathcal{L}^\pi(Q^\circ)$.

An automaton $G = \langle \Sigma, \Pi, Q, \to, Q^\circ, \Xi \rangle$ is *deterministic* if it has at most one initial state, i.e., $|Q^\circ| \le 1$, if $x \xrightarrow{\sigma} y_1$ and $x \xrightarrow{\sigma} y_2$ always implies $y_1 = y_2$, and if G has no transitions labelled τ. Given a possibly nondeterministic automaton G, the well-known *subset construction* can be used to obtain a language-equivalent deterministic automaton (Hopcroft et al. 2001). More precisely,

$$\det(G) = \langle \Sigma, \Pi, \mathbb{P}Q, \to_{\det}, Q^\circ_{\det}, \Xi_{\det} \rangle \;, \tag{3}$$

where

- $X \xrightarrow{\sigma}_{\det} Y$ for $X, Y \subseteq Q$ and $\sigma \in \Sigma$ if and only if $Y = \{\, y \in Q \mid X \overset{\sigma}{\Rightarrow} y \,\}$ and $Y \neq \emptyset$;
- $Q^\circ_{\det} = \{\{\, x \in Q \mid Q^\circ \overset{\varepsilon}{\Rightarrow} x \,\}\} \setminus \{\emptyset\}$;
- $\Xi_{\det}(\pi) = \{\, X \subseteq Q \mid X \cap \Xi(\pi) \neq \emptyset \,\}$.

The automaton $\det(G)$ is deterministic and satisfies $\mathcal{L}(\det(G)) = \mathcal{L}(G)$ and $\mathcal{L}^\pi(\det(G)) = \mathcal{L}^\pi(G)$ for each $\pi \in \Pi$.

2.3 Operations

The process-algebraic operations of synchronous composition and hiding are used in this paper to compose automata. *Synchronous composition* models the parallel execution of two or more automata, and is done using lock-step synchronisation in the style of (Hoare 1985).

Definition 2 Let $G = \langle \Sigma, \Pi, Q_G, \to_G, Q^\circ_G, \Xi_G \rangle$ and $H = \langle \Sigma, \Pi, Q_H, \to_H, Q^\circ_H, \Xi_H \rangle$ be multi-coloured automata. The *synchronous product* of G and H is

$$G \parallel H = \langle \Sigma, \Pi, Q_G \times Q_H, \to, Q^\circ_G \times Q^\circ_H, \Xi \rangle \tag{4}$$

where

$$\begin{array}{lll} (x_G, x_H) \xrightarrow{\sigma} (y_G, y_H) & \text{if } \sigma \in \Sigma,\ x_G \xrightarrow{\sigma}_G y_G, \text{ and} \\ & x_H \xrightarrow{\sigma}_H y_H; \\ (x_G, x_H) \xrightarrow{\tau} (y_G, x_H) & \text{if } x_G \xrightarrow{\tau}_G y_G; \\ (x_G, x_H) \xrightarrow{\tau} (x_G, y_H) & \text{if } x_H \xrightarrow{\tau}_H y_H; \end{array}$$

and $\Xi(\pi) = \Xi_G(\pi) \times \Xi_H(\pi)$ for each $\pi \in \Pi$.

This definition assumes that the two composed automata share the same event and proposition alphabets. This is sufficient for the purpose of this paper. 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 $\Xi(\pi) = Q$.

It is easily confirmed that synchronous composition is a commutative and associative operation.

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* (Hopcroft et al. 2001).

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

$$G \setminus \Upsilon \;=\; \langle \Sigma \setminus \Upsilon, \Pi, Q, \to \setminus\, \Upsilon, Q^\circ, \Xi \rangle \;, \tag{5}$$

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

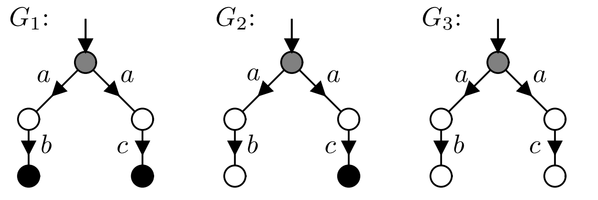

Figure 1: Generalised nonblocking vs. standard nonblocking.

2.4 Generalised Nonblocking

It is a desirable for control systems to be free from *livelock* and *deadlock*. This is typically expressed and checked by designating certain states of an automaton as *success* or *terminal* states and checking their reachability. In discrete event systems theory, this idea is called the nonblocking or nonconflicting property, which requires that a terminal state be reachable from every reachable system state (Ramadge & Wonham 1989).

Nonblocking is generalised in (Malik & Leduc 2008), using two propositions α and ω. The intended meaning is that ω represents terminal states, while α specifies a set of states from which terminal states are required to be reachable.

Definition 4 Let $G = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi \rangle$ be a multi-coloured automaton with $\alpha, \omega \in \Pi$.

- 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.*
- 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 Consider the automata in Fig. 1. States marked α are grey, and states marked ω are black. Automaton G_1 is both ω-nonblocking and (α,ω)-nonblocking, G_2 is ω-blocking and (α,ω)-nonblocking, and G_3 is both ω-blocking and (α,ω)-blocking.

Clearly, if an automaton is ω-nonblocking, it is also (α,ω)-nonblocking, but the converse is not true in general. The relationship between generalised nonblocking and standard nonblocking along with some applications is discussed in (Malik & Leduc 2008).

3 Generalised Nonblocking Equivalence

The straightforward approach to verify whether a composed system

$$G_1 \parallel G_2 \parallel \cdots \parallel G_n \tag{6}$$

is (α,ω)-nonblocking consists of explicitly constructing the synchronous product and checking for each state marked α whether it has a reachable state marked ω. This can be done using CTL model checking, and models of substantial size can be analysed if the state space is represented symbolically (Clarke et al. 1999). Yet, the technique remains limited by the amount of memory available to store representations of the synchronous product.

In an attempt to alleviate this state-space explosion problem, *compositional* verification (Flordal & Malik 2009) seeks to rewrite individual system components and, e.g., replace G_1 in (6) by a simpler version G_1', to analyse the simpler system

$$G_1' \parallel G_2 \parallel \cdots \parallel G_n \ . \tag{7}$$

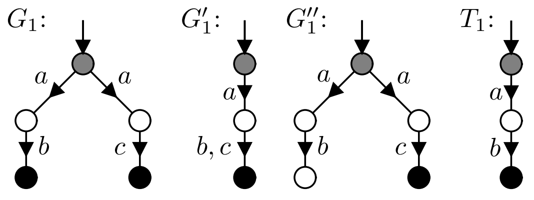

Figure 2: Generalised nonblocking equivalence.

This reasoning requires that G_1 and G_1' are related in some way. For example, automaton G_1 in Fig. 2 may be replaced by G_1' while preserving the generalised nonblocking property of the system (6). If the remainder $G_2 \parallel \cdots \parallel G_n$ of the system has an α-marked initial state, the composed system is (α,ω)-nonblocking if and only if it can reach an ω-marked state after executing the trace ab or ac, regardless of whether G_1 or G_1' is used.

On the other hand, generalised nonblocking is not preserved if G_1 is replaced by G_1'' in Fig. 2. If $G_2 \parallel \cdots \parallel G_n$ has an α-marked initial state and can only reach an ω-marked state after executing the trace ab, like automaton T_1 in Fig. 2, then (6) is (α,ω)-nonblocking while (7) is (α,ω)-blocking.

3.1 The Generalised Nonblocking Preorder

A notion of process *equivalence* to perform abstractions preserving generalised nonblocking is described in (Malik & Leduc 2008). This section generalises these definitions and introduces a *preorder*, which makes it possible to reason not only about equivalence but also about *refinement.* The definitions are based on the traditional testing framework (Hennessy 1988, De Nicola & Hennessy 1984) that defines preorders and equivalences relating processes based on their responses to *tests.* In the context of generalised nonblocking, a test can be an arbitrary automaton, and the test's response is the observation whether the test is (α,ω)-nonblocking in combination with the given automaton or not. Two automata are considered as equivalent, if the responses of all tests are equal.

Definition 5 Let G and H be two multi-coloured automata with $\alpha, \omega \in \Pi$.

- G is *less (α,ω)-conflicting* than H, written $G \lesssim_{(\alpha,\omega)} H$, if for every multi-coloured automaton T such that $H \| T$ is (α,ω)-nonblocking, $G \| T$ also is (α,ω)-nonblocking.
- G and H are *(α,ω)-conflict equivalent*, written $G \simeq_{(\alpha,\omega)} H$, if $G \lesssim_{(\alpha,\omega)} H$ and $H \lesssim_{(\alpha,\omega)} G$.

The relation $\lesssim_{(\alpha,\omega)}$ defines the *generalised nonblocking* preorder. An automaton G is less (α,ω)-conflicting than H if there are fewer tests T that are (α,ω)-blocking in combination with G than in combination with H. Two automata are (α,ω)-conflict equivalent if they are (α,ω)-blocking in combination with exactly the same tests. If $G_1 \simeq_{(\alpha,\omega)} G_1'$, then G_1 can be replaced by G_1' in (6) without affecting the generalised nonblocking property of the composition.

Example 2 Automata G_1 and G_1' in Fig. 2 are (α,ω)-conflict equivalent, while G_1 and G_1'' are not, because $G_1 \parallel T_1$ is (α,ω)-nonblocking and $G_1'' \parallel T_1$ is (α,ω)-blocking. Furthermore, it can be shown that $G_1 \lesssim_{(\alpha,\omega)} G_1''$.

3.2 Congruence Properties

An important question concerning preorders such as $\lesssim_{(\alpha,\omega)}$ is their relationship to process-algebraic operations. For compositional verification, the equivalence used must be well-behaved with respect to synchronous composition and hiding. These so-called *congruence* properties have been established in (Malik et al. 2006) for standard nonblocking and in (Malik & Leduc 2008) for generalised nonblocking equivalence, and can easily be extended to the generalised nonblocking preorder.

Definition 6 Let $\lesssim$ be a preorder on the set of multi-coloured automata.

- $\lesssim$ is a *pre-congruence* with respect to $\|$ if, for all multi-coloured automata G, H, and T such that $G \lesssim H$, it follows that $G \parallel T \lesssim H \parallel T$.
- $\lesssim$ *respects* (α,ω)*-nonblocking* if, for all multi-coloured automata G and H such that $G \lesssim H$, if H is (α,ω)-nonblocking then G also is (α,ω)-nonblocking.

Proposition 1 $\lesssim_{(\alpha,\omega)}$ is a pre-congruence with respect to $\|$.

Proof. Let G, H, and T be such that $G \lesssim_{(\alpha,\omega)} H$, and let T' be an arbitrary multi-coloured automaton such that $(H \parallel T) \parallel T'$ is (α,ω)-nonblocking. Then clearly, $H\|(T\|T') = (H\|T)\|T'$ is (α,ω)-nonblocking, and since $G \lesssim_{(\alpha,\omega)} H$ it follows that $(G \parallel T) \parallel T' = G\|(T\|T')$ is (α,ω)-nonblocking. Since T' was chosen arbitrarily, it follows that $G \parallel T \lesssim_{(\alpha,\omega)} H \parallel T$. □

Proposition 2 $\lesssim_{(\alpha,\omega)}$ respects (α,ω)-nonblocking.

Proof. Note that there exists a multi-coloured automaton U such that $G \parallel U = G$ for every multi-coloured automaton G. Let $G \lesssim_{(\alpha,\omega)} H$, and let H be (α,ω)-nonblocking. Then $H\|U = H$ is (α,ω)-nonblocking. Since $G \lesssim_{(\alpha,\omega)} H$, it follows that $G = G\|U$ is (α,ω)-nonblocking. □

Thus, the generalised nonblocking equivalence is a congruence with respect to synchronous composition and respects (α,ω)-nonblocking. This is enough to justify the correctness of a compositional verification approach such as the one outlined at the beginning of Sect. 3.

Similarly to standard nonblocking (Malik et al. 2006), the generalised nonblocking preorder turns out to be the coarsest pre-congruence with respect to synchronous composition that respects (α,ω)-nonblocking. In other words, any preorder that relates multi-coloured automata according to their generalised nonblocking behaviour and preserves synchronous composition is contained in the generalised nonblocking preorder. Therefore, the generalised nonblocking preorder is the best possible process refinement for reasoning about generalised nonblocking.

Proposition 3 Let $\lesssim$ be a pre-congruence with respect to $\|$ which respects (α,ω)-nonblocking. Then $G \lesssim H$ implies $G \lesssim_{(\alpha,\omega)} H$.

Proof. Let $G \lesssim H$, and let T be a multi-coloured automaton such that $H\|T$ is (α,ω)-nonblocking. Then $G \parallel T \lesssim H \parallel T$ since $\lesssim$ is a pre-congruence with respect to $\|$. Since $\lesssim$ respects blocking it follows that $G \parallel T$ is (α,ω)-nonblocking. Since G, H, and T were chosen arbitrarily, it follows that $G \lesssim_{(\alpha,\omega)} H$. □

3.3 Characterising the Preorder

In addition to the test-based definition of a process preorder, it is desirable to have a characterisation that can be derived from the state structure of an automaton (van Glabbeek 2001). This section introduces the generalised nonconflicting completion semantics as an algebraic model of the generalised nonblocking preorder and equivalence, which can be derived from the state and transitions of a multi-coloured automaton in such a way that the model can be represented finitely for every finite-state automaton. This model will be used in the following section to construct a canonical automaton.

The following definition restates the generalised nonblocking preorder as a state-based criterion. To check whether an automaton G is less (α,ω)-conflicting than another automaton H, it is enough to collect the ω-marked languages of all α-marked states of G and check whether H contains larger languages associated with the same α-markings. This idea is formalised by the concept of being state-wise less (α,ω)-conflicting, which turns out to be equivalent to the generalised nonblocking preorder.

Definition 7 Let $G = \langle \Sigma, \Pi, Q_G, \rightarrow_G, Q_G^\circ, \Xi_G \rangle$ and $H = \langle \Sigma, \Pi, Q_H, \rightarrow_H, Q_H^\circ, \Xi_H \rangle$ be multi-coloured automata with $\alpha,\omega \in \Pi$. G is said to be *state-wise less* (α,ω)*-conflicting* than H if the following property holds for every $s \in \Sigma^*$: for every $x_G \in \Xi_G(\alpha)$ such that $G \overset{s}{\Rightarrow} x_G$ there exists $x_H \in \Xi_H(\alpha)$ such that $H \overset{s}{\Rightarrow} x_H$ and $\mathcal{L}^\omega(x_H) \subseteq \mathcal{L}^\omega(x_G)$.

Proposition 4 Let $G = \langle \Sigma, \Pi, Q_G, \rightarrow_G, Q_G^\circ, \Xi_G \rangle$ and $H = \langle \Sigma, \Pi, Q_H, \rightarrow_H, Q_H^\circ, \Xi_H \rangle$ be multi-coloured automata with $\alpha,\omega \in \Pi$. G is state-wise less (α,ω)-conflicting than H if and only if G is less (α,ω)-conflicting than H.

Proof. First assume that G is state-wise less (α,ω)-conflicting than H, and let $T = \langle \Sigma, \Pi, Q_T, \rightarrow_T, Q_T^\circ, \Xi_T \rangle$ be an automaton such that $H \parallel T$ is (α,ω)-nonblocking. Let $G \parallel T \overset{s}{\Rightarrow} (x_G, x_T) \in \Xi_G(\alpha) \times \Xi_T(\alpha)$. Clearly $G \overset{s}{\Rightarrow} x_G \in \Xi_G(\alpha)$, and since G is state-wise less (α,ω)-conflicting than H, there exists a state $x_H \in \Xi_H(\alpha)$ such that $H \overset{s}{\Rightarrow} x_H$ and $\mathcal{L}^\omega(x_H) \subseteq \mathcal{L}^\omega(x_G)$. Thus, $H \parallel T \overset{s}{\Rightarrow} (x_H, x_T) \in \Xi_H(\alpha) \times \Xi_T(\alpha)$, and since $H \parallel T$ is (α,ω)-nonblocking, there exists a trace $t \in \Sigma^*$ such that $(x_H, x_T) \overset{t}{\Rightarrow} \Xi_H(\omega) \times \Xi_T(\omega)$. Then, $t \in \mathcal{L}^\omega(x_H) \subseteq \mathcal{L}^\omega(x_G)$, which implies $x_G \overset{t}{\Rightarrow}_G \Xi_G(\omega)$, and therefore $(x_G, x_T) \overset{t}{\Rightarrow} \Xi_G(\omega) \times \Xi_T(\omega)$. Since s, x_G, and x_T were chosen arbitrarily, it follows that $G \parallel T$ is (α,ω)-nonblocking.

Second, assume that G is less (α,ω)-conflicting than H. Let $s \in \Sigma^*$ and $G \overset{s}{\Rightarrow} x_G \in \Xi_G(\alpha)$. Construct a deterministic automaton $T = \langle \Sigma, \Pi, Q_T, \rightarrow_T, Q_T^\circ, \Xi_T \rangle$ such that $\mathcal{L}(T) = \Sigma^*$, $\mathcal{L}^\alpha(T) = \{s\}$, and $\mathcal{L}^\omega(T) = \Sigma^* \setminus s\mathcal{L}^\omega(x_G)$. Since T is deterministic, there exists a unique state $x_T \in Q_T$ such that $T \overset{s}{\Rightarrow} x_T$, which satisfies $x_T \in \Xi_T(\alpha)$ and $\mathcal{L}^\omega(x_T) = \Sigma^* \setminus \mathcal{L}^\omega(x_G)$. Then $G \parallel T$ is (α,ω)-blocking, because $G \parallel T \overset{s}{\Rightarrow} (x_G, x_T) \in \Xi_G(\alpha) \times \Xi_T(\alpha)$ and $\mathcal{L}^\omega(x_G) \cap \mathcal{L}^\omega(x_T) = \emptyset$. Since G is less (α,ω)-conflicting than H, it follows that $H \parallel T$ is (α,ω)-blocking. This means that there exist $u \in \Sigma^*$, $y_H \in Q_H$, and $y_T \in Q_T$ such that $H\|T \overset{u}{\Rightarrow} (y_H, y_T) \in \Xi_H(\alpha) \times \Xi_T(\alpha)$ and $\mathcal{L}^\omega(y_H) \cap \mathcal{L}^\omega(y_T) = \emptyset$. Then $y_T \in \Xi_T(\alpha)$, and by construction of T it follows that $u = s$ and $y_T = x_T$. This implies $H \overset{s}{\Rightarrow} y_H \in \Xi_H(\alpha)$ and $\mathcal{L}^\omega(y_H) \cap (\Sigma^* \setminus \mathcal{L}^\omega(x_G)) = \mathcal{L}^\omega(y_H) \cap \mathcal{L}^\omega(x_T) =$

$\mathcal{L}^\omega(y_H) \cap \mathcal{L}^\omega(y_T) = \emptyset$, i.e., $\mathcal{L}^\omega(y_H) \subseteq \mathcal{L}^\omega(x_G)$. Thus, y_H satisfies the requirements given for x_H in Def. 7, so G is state-wise less (α, ω)-conflicting than H □

Prop. 4 is the key to constructing a process-algebraic model of generalised nonblocking. Essentially, generalised nonblocking can be characterised by the sets of ω-marked languages associated with the α-marked states or, more precisely, with the traces leading to α-marked states.

Definition 8 Let $G = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi \rangle$ be a multi-coloured automaton with $\alpha, \omega \in \Pi$. The *generalised nonconflicting completion semantics* for G is defined as

$$\mathcal{CC}_{(\alpha,\omega)}(G) = \{\, (c, C) \in \Sigma^* \times \mathbb{P}\Sigma^* \mid \text{There exists } x \in \Xi(\alpha) \text{ such that } G \overset{c}{\Rightarrow} x \text{ and } \mathcal{L}^\omega(x) \subseteq C \,\} \ . \tag{8}$$

If $(c, C) \in \mathcal{CC}_{(\alpha,\omega)}(G)$, then C is called a *nonconflicting completion* for c in G.

Assume G contains an α-marked state x reachable via trace $c \in \Sigma^*$, i.e., $G \overset{c}{\Rightarrow} x \in \Xi(\alpha)$. Then the marked language $\mathcal{L}^\omega(x)$ of x clearly is a nonconflicting completion for c in G, i.e.,

$$(c, \mathcal{L}^\omega(x)) \in \mathcal{CC}_{(\alpha,\omega)}(G) \ . \tag{9}$$

Furthermore, all superlanguages of $\mathcal{L}^\omega(x)$ are also nonconflicting completions,

$$(c, C) \in \mathcal{CC}_{(\alpha,\omega)}(G) \quad \text{for all } C \supseteq \mathcal{L}^\omega(x) \ . \tag{10}$$

If G is finite-state, then there exists only a finite number of α-states x and thus only a finite number of associated ω-marked languages $\mathcal{L}^\omega(x)$. This means that all nonconflicting completions can be obtained as supersets of the ω-marked language of some state x, of which there are only finitely many. Therefore, the following closure operations are used.

Definition 9 For $\mathrm{CC} \subseteq \Sigma^* \times \mathbb{P}\Sigma^*$, the *upward closure* $\mathrm{CC}^\uparrow$ and the *reduced form* $\mathrm{CC}^\downarrow$ are

$$\mathrm{CC}^\uparrow = \{\, (c, C') \in \Sigma^* \times \mathbb{P}\Sigma^* \mid \text{There exists } (c, C) \in \mathrm{CC} \text{ such that } C \subseteq C' \,\} \ ; \tag{11}$$

$$\mathrm{CC}^\downarrow = \{\, (c, C) \in \mathrm{CC} \mid \text{For all } (c, C') \in \mathrm{CC} \text{ with } C' \subseteq C \text{ it holds that } C' = C \,\} \ . \tag{12}$$

Example 3 The generalised nonconflicting completion semantics of automaton G_1 in Fig. 2 is

$$\mathcal{CC}_{(\alpha,\omega)}(G_1) = \{(\varepsilon, \{ab, ac\})\}^\uparrow \ . \tag{13}$$

Example 4 The generalised nonconflicting completion semantics of automaton G_4 in Fig. 3 is

$$\mathcal{CC}_{(\alpha,\omega)}(G_4) = \{\, (a^n, a^+b) \mid n \geq 0 \,\}^\uparrow \ . \tag{14}$$

The ω-marked language of the α-marked state q_0 is $\mathcal{L}^\omega(q_0) = a^+b$, and since this state can be reached after any number of a events, this language is associated with all traces a^n for $n \geq 0$. The ω-marked language of the second α-marked state q_1 is $\mathcal{L}^\omega(q_1) = a^*b \supseteq \mathcal{L}^\omega(q_0)$, and as a superlanguage of the already listed language, it is automatically included in the upward closure.

Not every nonconflicting completion semantics CC can be reconstructed from its reduced form $\mathrm{CC}^\downarrow$. In infinite structures, it is not guaranteed for $(c, C) \in \mathrm{CC}$ that there exists a minimal subset $C' \subseteq C$ such that $(c, C') \in \mathrm{CC}$. However, if the set of nonconflicting completions C that appear in CC is finite, then the existence of minimal subsets is guaranteed. Thus, if G is a finite-state automaton, then it indeed holds that

$$\mathcal{CC}_{(\alpha,\omega)}(G)^{\downarrow\uparrow} = \mathcal{CC}_{(\alpha,\omega)}(G) \ . \tag{15}$$

The following main result of this section states that the generalised nonconflicting completion semantics indeed characterises the generalised nonblocking preorder. If an automaton G is less (α, ω)-conflicting than automaton H, then the generalised nonconflicting completion semantics of G is contained in that of H.

Proposition 5 Let $G = \langle \Sigma, \Pi, Q_G, \rightarrow_G, Q_G^\circ, \Xi_G \rangle$ and $H = \langle \Sigma, \Pi, Q_H, \rightarrow_H, Q_H^\circ, \Xi_H \rangle$ be multi-coloured automata with $\alpha, \omega \in \Pi$. Then $G \lesssim_{(\alpha,\omega)} H$ if and only if $\mathcal{CC}_{(\alpha,\omega)}(G) \subseteq \mathcal{CC}_{(\alpha,\omega)}(H)$.

Proof. First let $G \lesssim_{(\alpha,\omega)} H$ and $(c, C) \in \mathcal{CC}_{(\alpha,\omega)}(G)$. Then there exists $x_G \in \Xi_G(\alpha)$ such that $G \overset{c}{\Rightarrow} x_G$ and $\mathcal{L}^\omega(x_G) \subseteq C$. By Prop. 4, G is state-wise less (α, ω)-conflicting than H, so there exists $x_H \in \Xi_H(\alpha)$ such that $H \overset{c}{\Rightarrow} x_H$ and $\mathcal{L}^\omega(x_H) \subseteq \mathcal{L}^\omega(x_G) \subseteq C$. This already implies $(c, C) \in \mathcal{CC}_{(\alpha,\omega)}(H)$.

Second let $\mathcal{CC}_{(\alpha,\omega)}(G) \subseteq \mathcal{CC}_{(\alpha,\omega)}(H)$. By Prop. 4, it is sufficient to show that G is state-wise less (α, ω)-conflicting than H. Therefore, let $s \in \Sigma^*$ and $x_G \in \Xi_G(\alpha)$ such that $G \overset{s}{\Rightarrow} x_G$. Then $(s, \mathcal{L}^\omega(x_G)) \in \mathcal{CC}_{(\alpha,\omega)}(G) \subseteq \mathcal{CC}_{(\alpha,\omega)}(H)$. By definition of $\mathcal{CC}_{(\alpha,\omega)}(H)$, there exists $x_H \in \Xi_H(\alpha)$ such that $H \overset{s}{\Rightarrow} x_H$ and $\mathcal{L}^\omega(x_H) \subseteq \mathcal{L}^\omega(x_G)$. Thus, x_H satisfies the conditions of Def. 7, so G is state-wise less (α, ω)-conflicting than H. □

3.4 Relationship to Standard Nonblocking

In (Malik et al. 2006), the nonconflicting completion semantics is introduced as an algebraic model for standard nonblocking. The model is similar in structure to the generalised nonconflicting completion semantics introduced above, however it cannot easily be constructed out of the states and transitions of an automaton, so tests are referred to instead.

Definition 10 (Malik et al. 2006) Let $G = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi \rangle$ be a multi-coloured automaton with $\omega \in \Pi$. The *nonconflicting completion semantics* of G is

$$\mathcal{CC}(G) = \{\, (c, C) \in \Sigma^* \times \mathbb{P}\Sigma^* \mid \text{For every automaton } T \text{ such that } G \parallel T \text{ is } \omega\text{-nonblocking and } T \overset{c}{\Rightarrow} x, \text{ there exists } t \in C \text{ with } x \overset{t}{\Rightarrow}_T \Xi_T(\omega) \,\} \ . \tag{16}$$

The idea of the nonconflicting completion semantics of an automaton G is that each nonconflicting completion represents a requirement that needs to be satisfied by any test that is to be nonblocking in combination with G. If the test can execute the trace c associated with a nonconflicting completion C, then, in order to be nonblocking in combination with G, the test must be able to terminate with at least one of the traces $t \in C$.

The following result shows that the generalised nonconflicting completion semantics can be explained in the same way: if a pair (c, C) is contained in the

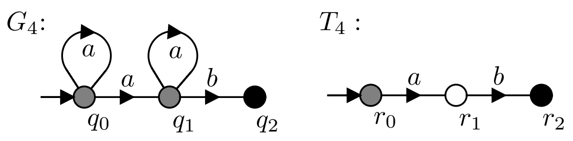

Figure 3: Standard nonconflicting completion semantics may be not well-founded.

semantics, then every test that can enter an α-marked state after trace c must be able to terminate with at least one of the traces in C, in order to be (α,ω)-nonblocking in combination with G.

Proposition 6 Let $G = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi\rangle$ be a multi-coloured automaton with $\alpha, \omega \in \Pi$. The generalised nonconflicting completion semantics can be alternatively characterised as

$$\mathcal{CC}_{(\alpha,\omega)}(G) = \{\, (c, C) \in \Sigma^* \times \mathbb{P}\Sigma^* \mid \text{For every automaton } T \text{ such that } G \parallel T \text{ is } (\alpha,\omega)\text{-nonblocking and } T \overset{c}{\Rightarrow} x \in \Xi_T(\alpha), \text{ there exists } t \in C \text{ with } x \overset{t}{\Rightarrow}_T \Xi_T(\omega) \,\} \ . \tag{17}$$

Proof. Let $(c, C) \in \mathcal{CC}_{(\alpha,\omega)}(G)$ and $T = \langle \Sigma, \Pi, Q_T, \rightarrow_T, Q^\circ_T, \Xi_T\rangle$ such that $G \parallel T$ is (α,ω)-nonblocking and $T \overset{c}{\Rightarrow} x_T \in \Xi_T(\alpha)$. Since $(c, C) \in \mathcal{CC}_{(\alpha,\omega)}(G)$, there exists $x \in \Xi(\alpha)$ such that $G \overset{c}{\Rightarrow} x$ and $\mathcal{L}^\omega(x) \subseteq C$. Then $G \parallel T \overset{c}{\Rightarrow} (x, x_T) \in \Xi(\alpha) \times \Xi_T(\alpha)$, and since $G \parallel T$ is (α,ω)-nonblocking there exists $t \in \Sigma^*$ such that $(x, x_T) \overset{t}{\Rightarrow} \Xi(\omega) \times \Xi_T(\omega)$. This implies $x_T \overset{t}{\Rightarrow}_T \Xi_T(\omega)$ and $t \in \mathcal{L}^\omega(x) \subseteq C$.

Now let $(c, C) \in \Sigma^* \times \mathbb{P}\Sigma^*$, and assume that for every automaton $T = \langle \Sigma, \Pi, Q_T, \rightarrow_T, Q^\circ_T, \Xi_T\rangle$ such that $G \parallel T$ is (α,ω)-nonblocking and $T \overset{c}{\Rightarrow} x \in \Xi_T(\alpha)$, there exists $t \in C$ such that $x \overset{t}{\Rightarrow}_T \Xi_T(\omega)$. Consider a deterministic automaton $T = \langle \Sigma, \Pi, Q_T, \rightarrow_T, Q^\circ_T, \Xi_T\rangle$ such that $\mathcal{L}(T) = \Sigma^*$, $\mathcal{L}^\alpha(T) = \{c\}$, and $\mathcal{L}^\omega(T) = c(\Sigma^* \setminus C)$. There exists exactly one state $x_T \in \Xi_T(\alpha)$, which also satisfies $T \overset{c}{\Rightarrow} x_T$ and $\mathcal{L}^\omega(x_T) = \Sigma^* \setminus C$, so there does not exist $t \in C$ such that $x_T \overset{t}{\Rightarrow} \Xi_T(\omega)$. By assumption it follows that $G \parallel T$ is (α,ω)-blocking. Then there exists a state $y \in \Xi(\alpha) \times \Xi_T(\alpha)$ such that $G \parallel T \Rightarrow y$ and $\mathcal{L}^\omega(y) = \emptyset$. By construction of T, there exists $x \in \Xi(\alpha)$ such that $y = (x, x_T)$ and $G \parallel T \overset{c}{\Rightarrow} y = (x, x_T)$, and furthermore $\emptyset = \mathcal{L}^\omega(y) = \mathcal{L}^\omega(x) \cap \mathcal{L}^\omega(x_T) = \mathcal{L}^\omega(x) \cap (\Sigma^* \setminus C)$, which implies $\mathcal{L}^\omega(x) \subseteq C$. It follows that $(c, C) \in \mathcal{CC}_{(\alpha,\omega)}(G)$ by definition. □

This shows that the standard and generalised nonconflicting completion semantics are closely related to each other. Yet, there are also important differences. While the generalised nonconflicting completion semantics only is closed via upward closure, in standard nonblocking there are interdependencies between states that lead to further closure properties.

Example 5 (Malik et al. 2006) In order to be ω-nonblocking in combination with automaton G_4 in Fig. 3, a test must initially be able to accept at least one of the traces $ab, aab, aaab, \ldots$ Therefore, $\mathcal{CC}(G_4)$ contains the pair $(\varepsilon, \{a^+b\})$. Furthermore, any such test must be able to execute a in its initial state, and any test executing a initially must also be able to cope with G_4 being put back to its initial state q_0 by executing the selfloop in q_0. Therefore, such a test also has to accept at least one of the traces $aab, aaab, aaaab, \ldots$ in its initial state. It follows that $\mathcal{CC}(G_4)$ contains all the pairs $(\varepsilon, \{a^n a^* b\})$ for $n \geq 1$.

This example shows that, even for a finite-state automaton, the standard nonconflicting completion semantics is not necessarily *well-founded*, and in general cannot be described by listing a finite set of minimal nonconflicting completions. For generalised nonblocking, this is possible. Due to the presence of α-markings, there always is the possibility for a test to be not α-marked for certain states.

Example 6 Consider automaton G_4 in Fig. 3 in combination with test T_4. Clearly, $G_4 \parallel T_4$ is (α,ω)-nonblocking, because the only reachable α-marked state of the synchronous product $G_4 \parallel T_4$ is the initial state, from which both automata can terminate by executing trace ab. However, the test T_4 cannot execute any trace $t \in \{a^n a^* b\}$ for $n > 1$, so unlike the case of standard nonblocking, $(\varepsilon, \{aaa^*b\}) \notin \mathcal{CC}_{(\alpha,\omega)}(G_4)$.

The presence of α-markings makes the nonconflicting completions for different traces independent from each other. This leads to a simpler semantic model with a finite characterisation. It also means that some abstractions possible for standard nonblocking are not applicable to generalised nonblocking.

4 Canonical Automaton

For compositional reasoning, it is necessary to modify automata in such a way that generalised nonblocking equivalence is preserved. This is facilitated by the fact that the generalised nonconflicting completion semantics can be represented finitely. This section explains how the generalised nonconflicting completion semantics can be used to construct a canonical form for any given finite-state automaton, which is generalised nonblocking equivalent to the original automaton, and such that the canonical forms of any two generalised nonblocking equivalent automata are equal.

4.1 Construction from Semantics

To ensure uniqueness, the canonical form is constructed directly from the generalised nonconflicting completion semantics. More precisely, it is shown in the following how to construct a *canonical automaton* $\mathcal{CA}(\mathrm{CC})$ for any given model

$$\mathrm{CC} \subseteq \Sigma^* \times \mathbb{P}\Sigma^* \ . \tag{18}$$

Afterwards, an algorithm will be given to compute the canonical automaton for any given multi-coloured automaton G.

The canonical automaton consists of two parts, called the *upper* and *lower automaton*. The upper automaton of CC essentially is a minimal deterministic recogniser of the language covered by CC,

$$\mathcal{L}(\mathrm{CC}) = \{\, c \in \Sigma^* \mid \text{There exists } C \subseteq \Sigma^* \text{ such that } (c, C) \in \mathrm{CC} \,\} \ . \tag{19}$$

The lower automaton consists of minimal deterministic recognisers of all the nonconflicting completions in CC, which are linked to transitions from the corresponding states in the upper automaton.

To ensure uniqueness, the upper automaton needs to be minimised in such a way that traces leading to equal nonconflicting completions in the future are mapped to the same state of the upper automaton. The following definition provides the necessary equality for any given model CC.

Definition 11 Let $\mathrm{CC} \subseteq \Sigma^* \times \mathbb{P}\Sigma^*$. Two traces $c_1, c_2 \in \Sigma^*$ are said to be *equivalent modulo* CC, written $c_1 \equiv_{\mathrm{CC}} c_2$, if for all $t \in \Sigma^*$ and all $C \subseteq \Sigma^*$, it holds that $(c_1 t, C) \in \mathrm{CC}$ if and only if $(c_2 t, C) \in \mathrm{CC}$.

Given this definition, the state set of the upper automaton is

$$U_{\mathrm{CC}} = \overline{\mathcal{L}(\mathrm{CC})}/{\equiv_{\mathrm{CC}}} \ , \tag{20}$$

and the transitions of the upper automaton are

$$[s]_{\mathrm{CC}} \xrightarrow{\sigma}_{U,\mathrm{CC}} [s\sigma]_{\mathrm{CC}} \quad \text{for all } s\sigma \in \overline{\mathcal{L}(\mathrm{CC})}. \tag{21}$$

Here, $[s]_{\mathrm{CC}} = \{\, s' \in \overline{\mathcal{L}(\mathrm{CC})} \mid s \equiv_{\mathrm{CC}} s' \,\}$ denotes the *equivalence class* of s modulo $\equiv_{\mathrm{CC}}$, and for $L \subseteq \Sigma^*$, the notation $L/{\equiv_{\mathrm{CC}}} = \{\, [s]_{\mathrm{CC}} \mid s \in L \,\}$ represents its partition into equivalence classes.

The lower automaton consists of deterministic recognisers for all the nonconflicting completions. It includes states accepting each of the following languages,

$$V_{\mathrm{CC}} = \{\, C\omega/t \mid \text{There exists } c \in \Sigma^* \text{ such that } (c, C) \in \mathrm{CC}, \text{ and } t \in \overline{C} \,\} \ . \tag{22}$$

Here, $L/s = \{\, t \in \Sigma^* \mid st \in L \,\}$ denotes the continuation language of $L \subseteq \Sigma^*$ after $s \in \Sigma^*$. To ensure minimality and thus uniqueness, it is convenient to identify the states of the lower automaton with the languages in V_{CC}. Accordingly, the transitions of the lower automaton are

$$L \xrightarrow{\sigma}_{V,\mathrm{CC}} L/\sigma \quad \text{for all } L \in V_{\mathrm{CC}} \text{ and } \sigma \in \Sigma \cap \overline{L}. \tag{23}$$

A lower-automaton state in $L \in V_{\mathrm{CC}}$ is marked ω if and only if $\omega \in L$. This ensures that the ω-marked languages of these states are equal to the languages they represent, i.e.,

$$\mathcal{L}^\omega(L\omega) = L \quad \text{for each } L\omega \in V_{\mathrm{CC}} \ . \tag{24}$$

To complete the lower automaton, each nonconflicting completion in CC is associated with its own α-marked state. The α-marked states may only be accessed from the upper automaton and therefore need to be distinct from any lower-automaton state. Therefore, the following additional states are used,

$$V^\alpha_{\mathrm{CC}} = \{\, (C, \alpha) \mid \text{There exists } c \in \Sigma^* \text{ such that } (c, C) \in \mathrm{CC} \,\} \ . \tag{25}$$

Given these state sets and transitions, the *canonical automaton* for CC is constructed as follows,

$$\mathcal{CA}(\mathrm{CC}) = \langle \Sigma, \{\alpha, \omega\}, Q_{\mathrm{CA}}, \rightarrow_{\mathrm{CA}}, Q^\circ_{\mathrm{CA}}, \Xi_{\mathrm{CA}} \rangle \tag{26}$$

where

- $Q_{\mathrm{CA}} = U_{\mathrm{CC}} \cup V_{\mathrm{CC}} \cup V^\alpha_{\mathrm{CC}}$;
- $\rightarrow_{\mathrm{CA}} = \ \rightarrow_{U,\mathrm{CC}} \cup \rightarrow_{V,\mathrm{CC}} \cup \{\, ([c]_{\mathrm{CC}}, \tau, (C, \alpha)) \mid (c, C) \in \mathrm{CC} \,\} \cup \{\, ((C, \alpha), \tau, C\omega) \mid (C, \alpha) \in V^\alpha_{\mathrm{CC}} \,\}$;
- $Q^\circ_{\mathrm{CA}} = \{[\varepsilon]_{\mathrm{CC}}\} \setminus \{\emptyset\}$;
- $\Xi_{\mathrm{CA}}(\alpha) = V^\alpha_{\mathrm{CC}}$;
- $\Xi_{\mathrm{CA}}(\omega) = \{\, C \in V_{\mathrm{CC}} \mid \omega \in C \,\}$.

The canonical automaton has a simple regular form, but it is not necessarily minimal. For example, the α-marked states can be merged into their successors, if those successors do not have other incoming transitions. The potential for reduction becomes clear in Example 7 below.

The following result confirms that the canonical automaton construction preserves generalised nonblocking in that the generalised nonconflicting completion semantics of the canonical automaton is equal to the upwards closure of the model CC, from which the automaton was constructed.

Proposition 7 Let $\mathrm{CC} \subseteq \Sigma^* \times \mathbb{P}\Sigma^*$. Then

$$\mathcal{CC}_{(\alpha,\omega)}(\mathcal{CA}(\mathrm{CC})) = \mathrm{CC}^{\uparrow} \ . \tag{27}$$

Proof. First, let $(c, C) \in \mathcal{CC}_{(\alpha,\omega)}(\mathcal{CA}(\mathrm{CC}))$. Then there exists $x \in \Xi_{\mathrm{CA}}(\alpha)$ such that $\mathcal{CA}(\mathrm{CC}) \overset{c}{\Rightarrow} x$ and $\mathcal{L}^\omega(x) \subseteq C$. By construction, this means that $x \in V^\alpha_{\mathrm{CC}}$, so $x = (C', \alpha)$ for some $(c', C') \in \mathrm{CC}$. Also by construction of the upper automaton, since $\mathcal{CA}(\mathrm{CC}) \overset{c}{\Rightarrow} x = (C', \alpha)$, it follows that $\mathcal{CA}(\mathrm{CC}) \xrightarrow{c} [c]_{\mathrm{CC}} \xrightarrow{\tau} (C', \alpha)$, which implies $(c, C') \in \mathrm{CC}$. Furthermore by construction of the lower automaton, $C' = \mathcal{L}^\omega(C'\omega) = \mathcal{L}^\omega((C', \alpha)) = \mathcal{L}^\omega(x) \subseteq C$, so it follows from $(c, C') \in \mathrm{CC}$ that $(c, C) \in \mathrm{CC}^{\uparrow}$.

Second, let $(c, C) \in \mathrm{CC}^{\uparrow}$. Then there exists $C' \subseteq C$ such that $(c, C') \in \mathrm{CC}$. By construction of the upper automaton, $\mathcal{CA}(\mathrm{CC}) \xrightarrow{c} [c]_{\mathrm{CC}} \xrightarrow{\tau} (C', \alpha) \in \Xi_{\mathrm{CA}}(\alpha)$, and by construction of the lower automaton, $(C', \alpha) \xrightarrow{\tau} C'\omega$ and $\mathcal{L}^\omega((C', \alpha)) = \mathcal{L}^\omega(C'\omega) = C' \subseteq C$. Thus, given $\mathcal{CA}(\mathrm{CC}) \overset{c}{\Rightarrow} (C', \alpha) \in \Xi_{\mathrm{CA}}(\alpha)$ and $\mathcal{L}^\omega((C', \alpha)) \subseteq C$, it follows by definition of $\mathcal{CC}_{(\alpha,\omega)}$ that $(c, C) \in \mathcal{CC}_{(\alpha,\omega)}(\mathcal{CA}(\mathrm{CC}))$. □

The canonical automaton can be constructed for any model CC, but the result is only finite-state if the set of nonconflicting completions in CC is finite, and the upper automaton has a finite-state representation. These conditions can be ensured when CC is obtained from the generalised nonconflicting completion semantics of a finite-state automaton. In this case, the upper automaton is finite-state because of the finite number of α-states from which nonconflicting completions can originate, and although the set of nonconflicting completions is typically infinite due to upwards closure, it is enough to construct the canonical automaton using only minimal nonconflicting completions.

Definition 12 The *canonical form* of a finite-state multi-coloured automaton G is

$$\mathcal{CA}(G) = \mathcal{CA}(\mathcal{CC}_{(\alpha,\omega)}(G)^{\downarrow}) \ . \tag{28}$$

As explained above, the canonical form of an automaton G is finite-state as long as G is finite-state. Given the previous results, it is not difficult to show that the canonical form is unique for all generalised nonblocking equivalent automata.

Proposition 8 Let G and H be two finite-state multi-coloured automata. Then

$$G \simeq_{(\alpha,\omega)} H \quad \text{if and only if} \quad \mathcal{CA}(G) = \mathcal{CA}(H) \ . \tag{29}$$

Proof. First assume that $G \simeq_{(\alpha,\omega)} H$. It follows that $\mathcal{CC}_{(\alpha,\omega)}(G) = \mathcal{CC}_{(\alpha,\omega)}(H)$ by Prop. 5, which implies $\mathcal{CA}(G) = \mathcal{CA}(\mathcal{CC}_{(\alpha,\omega)}(G)^{\downarrow}) = \mathcal{CA}(\mathcal{CC}_{(\alpha,\omega)}(H)^{\downarrow}) = \mathcal{CA}(H)$ by definition.

Second assume that $\mathcal{CA}(G) = \mathcal{CA}(H)$. From the fact that G is finite-state and Prop. 7, it follows that

$$\begin{aligned}\mathcal{CC}_{(\alpha,\omega)}(G) &= \mathcal{CC}_{(\alpha,\omega)}(G)^{\downarrow\uparrow} \\ &= \mathcal{CC}_{(\alpha,\omega)}(\mathcal{CA}(\mathcal{CC}_{(\alpha,\omega)}(G)^{\downarrow})) \\ &= \mathcal{CC}_{(\alpha,\omega)}(\mathcal{CA}(G)) \\ &= \mathcal{CC}_{(\alpha,\omega)}(\mathcal{CA}(H)) \\ &= \mathcal{CC}_{(\alpha,\omega)}(\mathcal{CA}(\mathcal{CC}_{(\alpha,\omega)}(H)^{\downarrow})) \\ &= \mathcal{CC}_{(\alpha,\omega)}(H)^{\downarrow\uparrow} \\ &= \mathcal{CC}_{(\alpha,\omega)}(H) \ . \end{aligned} \tag{30}$$

By Prop. 5, this implies $G \simeq_{(\alpha,\omega)} H$. □

Prop. 8 shows that the canonical automaton can be used for identification of generalised nonblocking equivalent automata. To determine whether two finite-state automata are generalised nonblocking equivalent, it is enough to construct their canonical automata and check whether they are equal.

Canonical automata can also be used to test the generalised nonblocking preorder. To check whether $G \lesssim_{(\alpha,\omega)} H$, it is possible to inspect all α-marked states of the synchronous product of the canonical forms of G and H and compare the associated languages. For every ω-marked language of an α-marked state of G, there needs to be a sublanguage associated with some corresponding α-marked state of H. The languages can be compared polynomially since they are represented deterministically in the canonical automata. However, the test for language inclusion requires only a deterministic representation for one of the two languages compared, and it is enough to construct only the canonical automaton of H to check whether $G \lesssim_{(\alpha,\omega)} H$.

4.2 Algorithmic Construction

In the previous section, the canonical automaton has been constructed from a semantic model CC, and its uniqueness has been established. This section proposes an algorithm that, given a finite-state multi-coloured automaton $G = \langle \Sigma, \Pi, Q, \rightarrow, Q^\circ, \Xi \rangle$, computes its canonical form $\mathcal{CA}(G)$.

The first step in the computation of the canonical automaton is the construction of the lower automaton, because it contains the languages associated with all α-marked states, which are needed to ensure minimality of the upper automaton.

The lower automaton consists of the minimal deterministic recognisers of all the ω-marked languages of all α-marked states of G. To construct it, the first step is to remove from G all states from where no ω-marked can be reached, that is, its state set is restricted to

$$R_\omega = \{\, x \in Q \mid x \rightarrow \Xi(\omega) \,\} \ . \tag{31}$$

Then subset construction (Hopcroft et al. 2001) is used to construct a deterministic recogniser V^{det} of all nonconflicting completion languages of G. The subset construction starts with initial state sets corresponding to each α-marked state and continues until all reachable state sets have been explored. More precisely,

$$V^{\text{det}} = \langle \Sigma, \{\omega\}, \mathbb{P}R_\omega, \rightarrow_V, Q^\circ_V, \Xi_V \rangle \tag{32}$$

where

- $X \xrightarrow{\sigma}_V Y$ for $X, Y \subseteq R_\omega$ and $\sigma \in \Sigma$ if and only if $Y = \{\, y \in R_\omega \mid X \overset{\sigma}{\Rightarrow} y \,\}$ and $Y \neq \emptyset$;
- $X \in Q^\circ_V$ if and only if $X = \{\, x \in R_\omega \mid x_\alpha \overset{\varepsilon}{\Rightarrow} x \,\}$ for some $x_\alpha \in \Xi(\alpha)$;
- $\Xi_V(\omega) = \{\, X \subseteq R_\omega \mid X \cap \Xi(\omega) \neq \emptyset \,\}$.

This automaton is then minimised using Hopcroft's algorithm (Hopcroft 1971) to obtain a unique and minimal lower automaton V. For each initial state x° of the minimised lower automaton, a new α-marked state x^α is created and linked via a τ-transition to x°. These α-marked state comprise the state set V^α. In order to link this automaton to the upper automaton later, a map is kept that links the α-marked states of G to their corresponding states in V^α.

Next, the upper automaton is constructed. In order to ensure that it accepts precisely the language $\mathcal{L}(\mathcal{CC}_{(\alpha,\omega)}(G)) = \mathcal{L}^\alpha(G)$, the state set of G is restricted to states from where an α-marked can be reached, i.e., to

$$R_\alpha = \{\, x \in Q \mid x \rightarrow \Xi(\alpha) \,\} \ . \tag{33}$$

Then a second subset construction is used to obtain a deterministic recogniser U^{det} of $\mathcal{L}^\alpha(G)$.

In order to establish uniqueness with respect to $\equiv_{\mathcal{CC}_{(\alpha,\omega)}(G)}$, for each state set $X \subseteq R_\alpha$ in this subset construction, the associated set of minimal nonconflicting completions,

$$\begin{aligned}\mathcal{CC}_{(\alpha,\omega)}(X) = \{\, C \subseteq \Sigma^* \mid{} & \text{There exists } c \in \Sigma^* \\ & \text{such that } G \overset{c}{\Rightarrow} X \text{ and } (c, C) \in \\ & \mathcal{CC}_{(\alpha,\omega)}(G)^{\downarrow} \,\} \ , \end{aligned} \tag{34}$$

needs to be determined. Therefore, each state set X in the subset construction is associated with the set of all initial states of the lower automaton V that have been associated with some α-marked state contained in X. The ω-marked languages of these states are checked for language inclusion, and the initial states associated with non-minimal languages are removed from the set of languages associated with X. The ω-marked languages of the remaining states make up the set $\mathcal{CC}_{(\alpha,\omega)}(X)$.

Now the automaton U^{det} is minimised subject to an initial partition based on the sets (34). Two subset states $X, Y \subseteq R_\alpha$ can only be merged if

$$\mathcal{CC}_{(\alpha,\omega)}(X) = \mathcal{CC}_{(\alpha,\omega)}(Y) \ . \tag{35}$$

This is done using Hopcroft's algorithm (Hopcroft 1971) with an initial partition based on the minimised sets of α-marked states, which satisfies (35). The result is a unique minimal upper automaton with states partitioned in the coarsest possible way that respects $\equiv_{\text{CC}}$.

The final step in the construction of the canonical automaton is to link the upper and lower automata. Each state $[X]$ of the minimised upper automaton is linked via a τ-transition to all the α-marked states in V^α that have been associated with some α-marked state of G contained in one of the state sets associated with the merged state $[X]$.

Example 7 Fig. 4 demonstrates the process of construction of the canonical form $\mathcal{CA}(G)$ of automaton G.

The first step is to apply subset construction starting from the three α-marked states q_4, q_8, and q_{11}. This results in the deterministic automaton V^{det} also shown in Fig. 4. Its three initial states $\{q_8\}$, $\{q_{11}\}$, and $\{q_4, q_5, q_8\}$ correspond to the three α-marked states of G, from which the subset construction originates—the α-marked state q_4 is expanded to $\{q_4, q_5, q_8\}$ because of its outgoing τ-transitions.

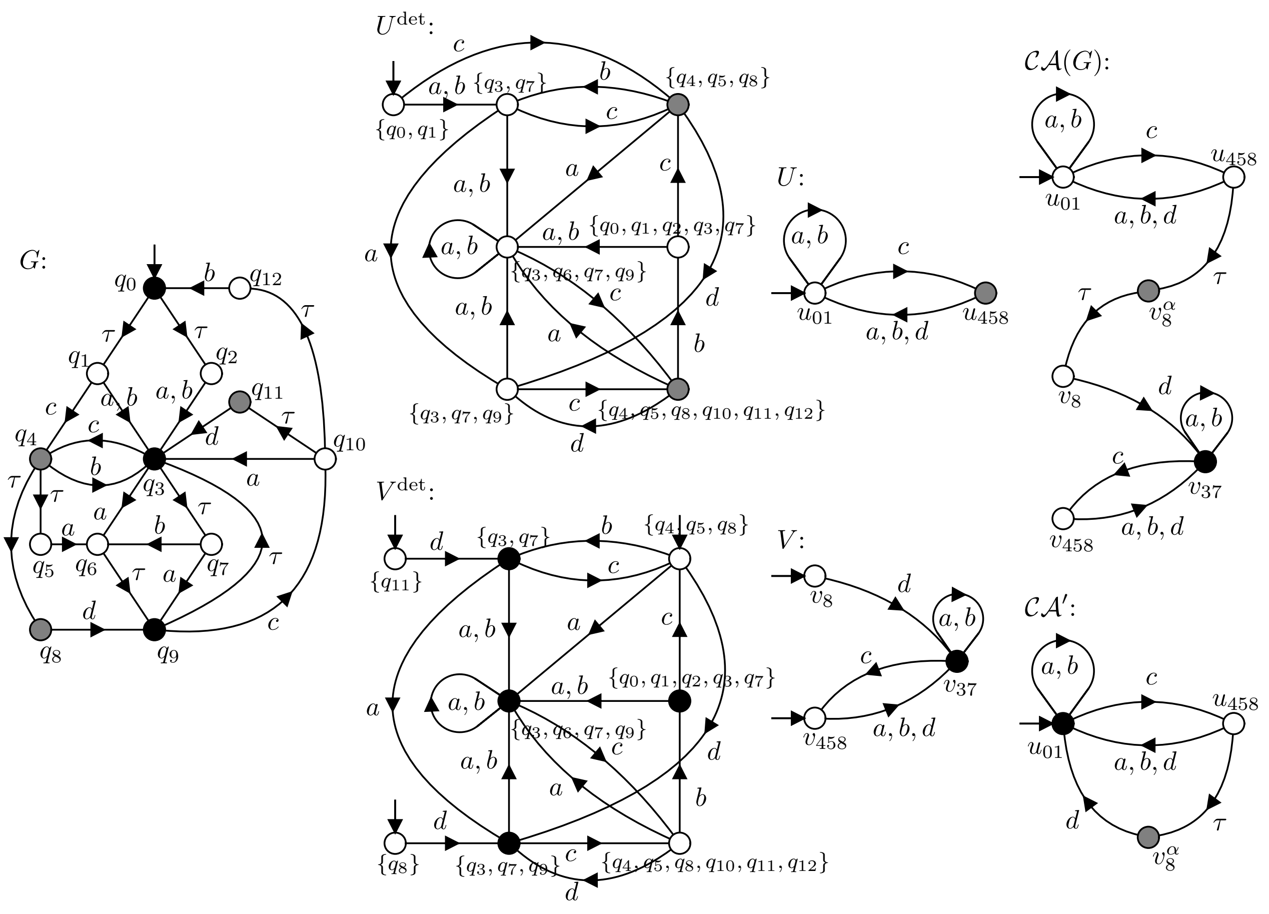

Figure 4: Example construction of canonical automaton.

Next, the intermediate lower automaton V^{det} is minimised using Hopcroft's algorithm, resulting in the lower automaton V. After merging, this automaton has only two initial states: state v_8 corresponds to the original α-marked states q_8 and q_{11}, while v_{458} corresponds to the original α-marked state q_4. It can already be seen that the ω-marked language of v_8 is contained in the ω-marked language of v_{458}.

Next, to construct the upper automaton, subset construction is applied to G to obtain its deterministic form U^{det}. Owing to the fact that α- and ω-marked states are reachable from all states of G, this automaton is very similar to the intermediate lower automaton V^{det}. The α-marked states of U^{det} are $\{q_4, q_5, q_8\}$ and $\{q_4, q_5, q_8, q_{10}, q_{11}, q_{12}\}$. These states are both associated with the lower-automaton initial states v_8 and v_{458}, however since $\mathcal{L}^\omega(v_8) \subseteq \mathcal{L}^\omega(v_{458})$, only v_8 is considered. Both α-marked states are associated with equal sets of lower-automaton initial states, so they may be merged during minimisation. And indeed, minimisation results in the automaton U with only one α-marked state u_{458}.

Finally, the upper and lower automata are linked, resulting in the canonical automaton $\mathcal{CA}(G)$. The only α-marked state of the upper automaton is u_{458}, which is to be associated with v_8 in the lower automaton. Therefore, the new α-marked state v_8^α is created and linked via silent transitions to u_{458} and v_8.

It becomes clear that the canonical automaton, although unique, is not minimal. Since v_8^α has only one outgoing τ-transition that leads to state v_8 with no other incoming transitions, states v_8^α and v_8 can be merged while preserving generalised nonblocking equivalence. Furthermore, the language of lower-automaton state v_{37} is equal to the language of upper-automaton state u_{01}, and since for lower-automaton states only the language is relevant, v_{37} can be replaced by u_{01}. This results in the automaton $\mathcal{CA}'$, which is generalised nonblocking equivalent to $\mathcal{CA}(G)$ and to G.

The algorithm to construct the canonical automaton is exponential. The upper and lower automaton are obtained through subset construction, and the number of states of the canonical automaton is bounded by

$$\begin{aligned} |U_{\mathrm{CC}}| + |V_{\mathrm{CC}}| + |V_{\mathrm{CC}}^\alpha| &\leq 2^{|Q|} + 2^{|Q|} + |\Xi(\alpha)| \\ &= O(2^{|Q|}) \ . \end{aligned} \tag{36}$$

To estimate the number of transitions, note that the upper and lower automaton are deterministic automata linked by two τ-transitions for each α-marked state. Thus, the number of transitions of the canonical automaton is bounded by

$$|\Sigma||U_{\mathrm{CC}}| + |\Sigma||V_{\mathrm{CC}}| + 2|V_{\mathrm{CC}}^\alpha| = O(|\Sigma|2^{|Q|}) \ . \tag{37}$$

The construction of the upper automaton requires tests for language inclusion to see whether languages associated to different α-marked states are contained in each other. There are up to $\frac{1}{2}|\Xi(\alpha)|(|\Xi(\alpha)| - 1)$ pairs of α-marked states that need to be compared, and each test in the worst case requires construction of a synchronous product of two deterministic automata with $2^{|Q|}$ states each. The time complexity of the language inclusion check is determined by the number of transitions of the synchronous product, which is bounded by $|\Sigma|(2^{|Q|})^2 = |\Sigma|4^{|Q|}$. In practice, the test can often be completed much faster, because identical states of G can be recognised in the subset construction, and because the test can stop early when language inclusion is not satisfied. Still, the worst-case time complexity of the algorithm to construct the canonical form is

$$O(|\Sigma||\Xi(\alpha)|^2 4^{|Q|}) = O(|\Sigma||Q|^2 4^{|Q|}) \ . \tag{38}$$

Despite its exponential complexity, subset construction is known to be well-behaved in many practical cases. In (Ware & Malik 2008), subset construction has been used for compositional verification of safety properties of very large discrete-event systems models. Such results suggest that the canonical automaton may be a useful tool for compositional verification of generalised nonblocking.

5 Conclusions

The *generalised nonconflicting completion semantics* has been presented as a process-algebraic model that characterises automata according to their generalised nonblocking behaviour. The semantics can be constructed from the transition structure and has a finite representation for any given finite-state automaton. The generalised nonconflicting completion semantics has been used to define a *canonical form* of automata, a unique automaton that is the same for all generalised nonblocking equivalent automata. An algorithm to construct the canonical form has been given.

The results presented in this paper provide an algorithmic means to identify generalised nonblocking equivalent automata and to perform refinement with respect generalised nonblocking. In future work, the authors would like to study possible applications in the area of compositional verification, and to extend the results to standard nonblocking.

References

Brinksma, E., Rensink, A. & Vogler, W. (1995), Fair testing, *in* I. Lee & S. A. Smolka, eds, 'Proc. 6th Int. Conf. Concurrency Theory, CONCUR '95', Vol. 962 of *LNCS*, Springer, Philadelphia, PA, USA, pp. 313–327.

Cassandras, C. G. & Lafortune, S. (1999), *Introduction to Discrete Event Systems*, Kluwer.

Clarke, Jr., E. M., Grumberg, O. & Peled, D. A. (1999), *Model Checking*, MIT Press.

De Nicola, R. & Hennessy, M. C. B. (1984), 'Testing equivalences for processes', *Theoretical Comput. Sci.* **34**(1–2), 83–133.

de Queiroz, M. H., Cury, J. E. R. & Wonham, W. M. (2004), Multi-tasking supervisory control of discrete-event systems, *in* 'Proc. 7th Int. Workshop on Discrete Event Systems, WODES '04', Reims, France, pp. 175–180.

Dietrich, P., Malik, R., Wonham, W. M. & Brandin, B. A. (2002), Implementation considerations in supervisory control, *in* B. Caillaud, P. Darondeau, L. Lavagno & X. Xie, eds, 'Synthesis and Control of Discrete Event Systems', Kluwer, pp. 185–201.

Fabian, M. & Kumar, R. (1997), Mutually nonblocking supervisory control of discrete event systems, *in* 'Proc. 36th IEEE Conf. Decision and Control, CDC '97', San Diego, CA, USA, pp. 2970–2975.

Flordal, H. & Malik, R. (2009), 'Compositional verification in supervisory control', *SIAM J. Control and Optimization* **48**(3), 1914–1938.

Hennessy, M. (1988), *Algebraic Theory of Processes*, MIT Press.

Hoare, C. A. R. (1985), *Communicating Sequential Processes*, Prentice-Hall.

Hopcroft, J. E. (1971), An $n \log n$ algorithm for minimizing states in a finite automaton, *in* Z. Kohavi & A. Paz, eds, 'Theory of Machines and Computations', Academic Press, New York, NY, USA, pp. 189–196.

Hopcroft, J. E., Motwani, R. & Ullman, J. D. (2001), *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley.

Kumar, R. & Shayman, M. A. (1994), Non-blocking supervisory control of nondeterministic discrete event systems, *in* 'Proc. American Control Conf.', Baltimore, MD, USA, pp. 1089–1093.

Leduc, R. J., Brandin, B. A., Lawford, M. & Wonham, W. M. (2005), 'Hierarchical interface-based supervisory control—part I: Serial case', *IEEE Trans. Automat. Contr.* **50**(9), 1322–1335.

Leduc, R. & Malik, R. (2010), A compositional approach for verifying hierarchical interface-based supervisory control, *in* 'Proc. 10th Int. Workshop on Discrete Event Systems, WODES '10', Berlin, Germany.

Malik, R. & Leduc, R. (2008), Generalised nonblocking, *in* 'Proc. 9th Int. Workshop on Discrete Event Systems, WODES '08', Göteborg, Sweden, pp. 340–345.

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

Malik, R., Streader, D. & Reeves, S. (2006), 'Conflicts and fair testing', *Int. J. Found. Comput. Sci.* **17**(4), 797–813.

Natarajan, V. & Cleaveland, R. (1995), Divergence and fair testing, *in* 'Proc. 22nd Int. Colloquium on Automata, Languages, and Programming, ICALP '95', pp. 648–659.

Ramadge, P. J. G. & Wonham, W. M. (1989), 'The control of discrete event systems', *Proc. IEEE* **77**(1), 81–98.

Roscoe, A. W. (1997), *The Theory and Practice of Concurrency*, Prentice-Hall.

Su, R., van Schuppen, J. H., Rooda, J. E. & Hofkamp, A. T. (2010), 'Nonconflict check by using sequential automaton abstractions based on weak observation equivalence', *Automatica* **46**(6), 968–978.

van Glabbeek, R. J. (2001), The linear time — branching time spectrum I: The semantics of concrete, sequential processes, *in* J. A. Bergstra, A. Ponse & S. A. Smolka, eds, 'Handbook of Process Algebra', Elsevier, pp. 3–99.

Ware, S. & Malik, R. (2008), The use of language projection for compositional verification of discrete event systems, *in* 'Proc. 9th Int. Workshop on Discrete Event Systems, WODES '08', Göteborg, Sweden, pp. 322–327.

Wong, K. C., Thistle, J. G., Malhame, R. P. & Hoang, H.-H. (2000), 'Supervisory control of distributed systems: Conflict resolution', *Discrete Event Dyn. Syst.* **10**, 131–186.