A State-Based Characterisation of the Conflict Preorder

Simon Ware   Robi Malik
Department of Computer Science, University of Waikato, Hamilton, New Zealand
{siw4,robi}@waikato.ac.nz

This paper proposes a way to effectively compare the potential of processes to cause *conflict*. In discrete event systems theory, two concurrent systems are said to be in conflict if they can get trapped in a situation where they are both waiting or running endlessly, forever unable to complete their common task. The *conflict preorder* is a process-algebraic pre-congruence that compares two processes based on their possible conflicts in combination with other processes. This paper improves on previous theoretical descriptions of the conflict preorder by introducing *less conflicting pairs* as a concrete state-based characterisation. Based on this characterisation, an effective algorithm is presented to determine whether two processes are related according to the conflict preorder.

1 Introduction

A key question in process algebra is how processes can be composed and compared [4, 6]. An understanding of what makes processes equivalent is important for several applications, ranging from comparison and minimisation in model checking to program construction using abstraction and refinement. Several equivalence relations have been studied, most notably *observation equivalence* [12], *failures equivalence* [7], and *trace equivalence* [7]. Each equivalence has its own properties, making it suitable for particular applications and verification tasks [6].

This paper focuses on *conflict equivalence*, which compares processes based on which other processes they can come into conflict [3, 14] with. Two processes are in conflict, if they can reach a state from which termination is no longer possible. This can be because of *deadlock* where neither process is capable of doing anything, or *livelock* where the system continues to run without ever terminating.

It is difficult to reason about conflicts in a modular way. If two processes are free from conflict individually, they may well be involved in a conflict when running together, and vice versa [18]. This makes it difficult to apply most methods of abstraction common in model checking [1] to verify systems to be free from conflict, and standard process-algebraic equivalences [6] are not applicable either.

Conflict equivalence is introduced in [11] as the best possible process equivalence to reason compositionally about conflicts. Conflict equivalence is coarser than observation equivalence [12] and different from failures and trace equivalence [7]. The process-algebraic theory most closely related to conflict equivalence is *fair testing* [2, 13, 15]. The essential difference between conflict equivalence and fair testing lies in the capability to compare processes that exhibit blocking behaviour, as expressed by the *set of certain conflicts* [9, 10, 11].

In [5, 16, 17], various conflict-preserving rewrite rules are used to simplify processes and check whether or not large systems of concurrent finite-state automata are free from conflict. While of good use in practice, the rewrite rules are incomplete, and it remains an open question how processes can be normalised or compared for conflict equivalence.

This paper improves on previous results about conflict equivalence and the associated conflict preorder [11], and fair testing [15], by providing a state-based characterisation of the conflict preorder. It proposes *less conflicting pairs* as a more concrete way to compare processes for their conflicting behaviour than the abstract test-based characterisation using *nonconflicting completions* in [11] and the *refusal trees* of [15]. Less conflicting pairs give a means to directly compare processes based on their reachable state sets, which leads to an alternative algorithm to test the conflict preorder. While still linear exponential, this algorithm is simpler and has better time complexity than the decision procedure for fair testing [15].

In the following, Section 2 briefly reviews the needed terminology of languages, automata, and conflict equivalence. Then Section 3 introduces less conflicting pairs and shows how they can be used to describe certain conflicts and the conflict preorder. Afterwards, Section 4 proposes an algorithm to calculate less conflicting pairs for finite-state automata, and Section 5 adds some concluding remarks.

2 Preliminaries

2.1 Languages and Automata

Event sequences and languages are a simple means to describe process behaviours. Their basic building blocks are *events*, which are taken from a finite *alphabet* Σ. Two special events are used, the *silent event* τ and the *termination event* ω. These are never included in an alphabet Σ unless mentioned explicitly.

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 *length* of trace s is denoted by $|s|$. A subset $L \subseteq \Sigma^*$ is called a *language*. The *concatenation* of two traces $s, t \in \Sigma^*$ is written as st, and a trace s is called a *prefix* of t, written $s \sqsubseteq t$, if $t = su$ for some trace u. A language $L \subseteq \Sigma^*$ is *prefix-closed*, if $s \in L$ and $r \sqsubseteq s$ implies $r \in L$.

In this paper, process behaviour is modelled using nondeterministic *labelled transitions systems* or *automata* $A = \langle \Sigma, Q, \rightarrow, Q^\circ \rangle$, where Σ is a finite alphabet of *events*, Q is a set of *states*, $\rightarrow \subseteq Q \times (\Sigma \cup \{\tau, \omega\}) \times Q$ is the *state transition relation*, and $Q^\circ \subseteq Q$ is the set of *initial states*. The automaton A is called *finite-state* if its state set Q is finite.

The transition relation is written in infix notation $x \xrightarrow{\sigma} y$, and is extended to traces by letting $x \xrightarrow{\varepsilon} x$ for all $x \in Q$, and $x \xrightarrow{s\sigma} y$ if $x \xrightarrow{s} z \xrightarrow{\sigma} y$ for some $z \in Q$. The transition relation must satisfy the additional requirement that, whenever $x \xrightarrow{\omega} y$, there does not exist any outgoing transition from y. The automaton A is called *deterministic* if $|Q^\circ| \leq 1$ and the transition relation contains no transitions labelled τ, and if $x \xrightarrow{\sigma} y_1$ and $x \xrightarrow{\sigma} y_2$ always implies $y_1 = y_2$.

To support silent transitions, $x \overset{s}{\Rightarrow} y$, with $s \in (\Sigma \cup \{\omega\})^*$, denotes the existence of a trace $t \in (\Sigma \cup \{\omega, \tau\})^*$ such that $x \xrightarrow{t} y$, and s is obtained from t by deleting all τ events. For a state set $X \subseteq Q$ and a state $y \in Q$, the expression $X \overset{s}{\Rightarrow} y$ denotes the existence of $x \in X$ such that $x \overset{s}{\Rightarrow} y$, and $A \overset{s}{\Rightarrow} y$ means that $Q^\circ \overset{s}{\Rightarrow} y$. Furthermore, $x \Rightarrow y$ denotes the existence of a trace s such that $x \overset{s}{\Rightarrow} y$, and $x \overset{s}{\Rightarrow}$ denotes the existence of a state $y \in Q$ such that $x \overset{s}{\Rightarrow} y$. For a state, state set, or automaton $\mathbf{X}$, the *language* and the *marked language* are

$$\mathbf{L}(\mathbf{X}) = \{\, s \in (\Sigma \cup \{\omega\})^* \mid \mathbf{X} \overset{s}{\Rightarrow} \,\} \qquad \text{and} \qquad \mathbf{L}^\omega(\mathbf{X}) = \mathbf{L}(\mathbf{X}) \cap \Sigma^*\omega \ . \tag{1}$$

Every prefix-closed language L is recognised by an automaton A such that $\mathbf{L}(A) = L$, but only *regular* languages are recognised by a finite-state automaton [8].

When two automata are running in parallel, lock-step synchronisation in the style of [7] is used. The *synchronous composition* of $A = \langle \Sigma_A, Q_A, \rightarrow_A, Q_A^\circ \rangle$ and $B = \langle \Sigma_B, Q_B, \rightarrow_B, Q_B^\circ \rangle$ is

$$A \,\|\, B = \langle \Sigma_A \cup \Sigma_B, Q_A \times Q_B, \rightarrow, Q_A^\circ \times Q_B^\circ \rangle \tag{2}$$

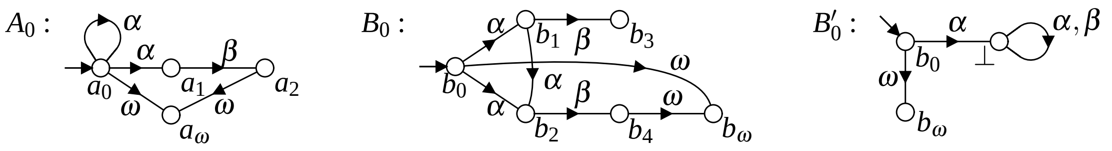

Figure 1: Examples of blocking and nonblocking automata.

where

$$
\begin{aligned}
(x_A, x_B) &\xrightarrow{\sigma} (y_A, y_B) &&\text{if } \sigma \in (\Sigma_A \cap \Sigma_B) \cup \{\omega\},\ x_A \xrightarrow{\sigma}_A y_A, \text{ and } x_B \xrightarrow{\sigma}_B y_B\ ;\\
(x_A, x_B) &\xrightarrow{\sigma} (y_A, x_B) &&\text{if } \sigma \in (\Sigma_A \setminus \Sigma_B) \cup \{\tau\} \text{ and } x_A \xrightarrow{\sigma}_A y_A\ ;\\
(x_A, x_B) &\xrightarrow{\sigma} (x_A, y_B) &&\text{if } \sigma \in (\Sigma_B \setminus \Sigma_A) \cup \{\tau\} \text{ and } x_B \xrightarrow{\sigma}_B y_B\ .
\end{aligned}
$$

In synchronous composition, shared events (including ω) must be executed by all automata together, while events used by only one of the composed automata and silent (τ) events are executed independently.

2.2 Conflict Equivalence

The key liveness property in supervisory control theory [14] is the *nonblocking* property. Given an automaton A, it is desirable that every trace in $\mathbf{L}(A)$ can be completed to a trace in $\mathbf{L}^\omega(A)$, otherwise A may become unable to terminate. A process that may become unable to terminate is called *blocking*. This concept becomes more interesting when multiple processes are running in parallel—in this case the term *conflicting* is used instead.

Definition 1. An automaton $A = \langle \Sigma, Q, \rightarrow, Q^\circ \rangle$ is *nonblocking* if for every state $x \in Q$, $Q^\circ \Rightarrow x$ implies that $\mathbf{L}^\omega(x) \neq \emptyset$. Otherwise A is *blocking*. Two automata A and B are *nonconflicting* if $A \parallel B$ is nonblocking, otherwise they are *conflicting*.

Example 1. Automaton A_0 in Figure 1 is nonblocking, as it is always possible to reach state a_2 and terminate. Automaton B_0 on the other hand is blocking, because it can enter state b_3 after execution of $\alpha\beta$, from where it is no longer possible to reach a state where the termination event ω is enabled.

For an automaton to be nonblocking, it is enough that a terminal state *can* be reached from *every* reachable state. There is no requirement for termination to be guaranteed. For example, automaton A_0 in Figure 1 is nonblocking despite the presence of a possibly infinite loop of α-transitions in state a_0. Nonblocking is also different from "may"-testing [15], which only requires the possibility of termination from the initial state. The testing semantics most similar to nonblocking is "should"-testing, which is also known as *fair testing* [15].

To reason about nonblocking in a compositional way, the notion of *conflict equivalence* is developed in [11]. According to process-algebraic testing theory, two automata are considered as equivalent if they both respond in the same way to all tests of a certain type [4]. For conflict equivalence, a *test* is an arbitrary automaton, and the *response* is the observation whether or not the test is conflicting with the automaton in question.

Definition 2. Let A and B be two automata. A is *less conflicting* than B, written $A \lesssim_{\text{conf}} B$, if, for every automaton T, if $B \parallel T$ is nonblocking then $A \parallel T$ also is nonblocking. A and B are *conflict equivalent*, $A \simeq_{\text{conf}} B$, if $A \lesssim_{\text{conf}} B$ and $B \lesssim_{\text{conf}} A$.

Example 2. Consider automata A_1 and B_1 in Figure 2. A_1 is *not* less conflicting than B_1, since $A_1 \parallel T_1$ is blocking while $B_1 \parallel T_1$ is nonblocking. This is because $A_1 \parallel T_1$ can enter the blocking state (a_2, q_1) after executing of α, whereas after executing α in B_1, it eventually becomes possible to continue using either the β- or γ-transition of T_1. It can also be shown that $B_1 \lesssim_{\text{conf}} A_1$ does not hold.

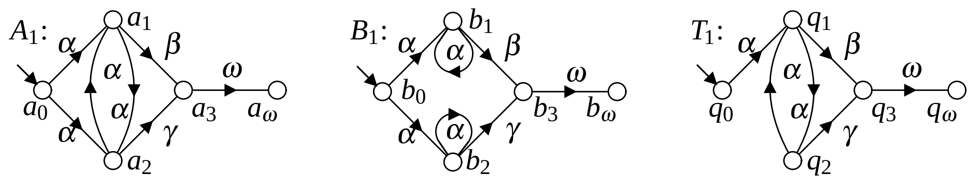

Figure 2: Two automata that are not conflict equivalent.

The properties of the conflict preorder $\lesssim_{\rm conf}$ and of conflict equivalence and their relationship to other process-algebraic relations are studied in [11]. It is enough to consider deterministic tests in Definition 2, and conflict equivalence is is the coarsest possible congruence with respect to synchronous composition that respects blocking, making it an ideal equivalence for use in compositional verification [5, 17].

2.3 The Set of Certain Conflicts

Every automaton can be associated with a language of *certain conflicts*, which plays an important role in conflict semantics [9].

Definition 3. For an automaton $A = \langle \Sigma, Q, \rightarrow, Q^\circ \rangle$, write

$$\textsc{Conf}(A) = \{\, s \in \Sigma^* \mid \text{For every automaton } T \text{ such that } T \overset{s}{\Rightarrow}, A \parallel T \text{ is blocking} \,\} ; \tag{3}$$

$$\textsc{NConf}(A) = \{\, s \in \Sigma^* \mid \text{There exists an automaton } T \text{ such that } T \overset{s}{\Rightarrow} \text{ and } A \parallel T \text{ is nonblocking} \,\} . \tag{4}$$

$\textsc{Conf}(A)$ is the set of *certain conflicts* of A. It contains all traces that, when possible in the environment, necessarily cause blocking. Its complement $\textsc{NConf}(A)$ is the most general behaviour of processes that are to be nonconflicting with A. If A is nonblocking, then $\textsc{Conf}(A) = \emptyset$ and $\textsc{NConf}(A) = \Sigma^*$, because in this case $A \parallel U$ is nonblocking, where U is a deterministic automaton such that $\mathbf{L}^\omega(U) = \Sigma^*\omega$. The set of certain conflicts becomes more interesting for blocking automata.

Example 3. Consider again automaton B_0 in Figure 1. Clearly $\alpha\beta \in \textsc{Conf}(B_0)$ as B_0 can enter the deadlock state b_3 by executing $\alpha\beta$, and therefore every test T that can execute $\alpha\beta$ is conflicting with B_0. But also $\alpha \in \textsc{Conf}(B_0)$, because B_0 can enter state b_2 by executing α, from where the only possibility to terminate is by executing $\beta\omega$. So any test that can execute α also needs to be able to execute $\alpha\beta$ if it is to be nonconflicting with B_0; but such a test is conflicting with B_0 as explained above. It can be shown that $\textsc{Conf}(B_0) = \alpha\Sigma^*$.

The set of certain conflicts is introduced in [9], and its properties and its relationship to conflict equivalence are studied in [11]. Even if an automaton is nondeterministic, its set of certain conflicts is a *language*, but as shown in Example 3, it is not necessarily a subset of the language $\mathbf{L}(A)$ of its automaton. If a trace s is a trace of certain conflicts, then so is any extension st. An algorithm to compute the set of certain conflicts for a given finite-state automaton is presented in [10].

Certain conflicts constitute the main difference between conflict equivalence and *fair testing* [15]. In fair testing, processes are not allowed to synchronise on the termination event ω, so termination is determined solely by the test. This can be expressed as conflict equivalence by requiring that ω be enabled in all states of the automata compared [11].

Conversely, it is possible to factor out certain conflicts from any given automaton, by redirecting all traces of certain conflicts to a single state [9, 10]. For example, automaton B_0 in Figure 1 can be replaced by the conflict equivalent automaton B_0', which uses the single deadlock state $\perp$. Two automata A and B are conflict equivalent if and only if their normalised forms A' and B' are fair testing equivalent. The decision procedure for fair testing [15] can be used to test the conflict preorder, and vice versa.

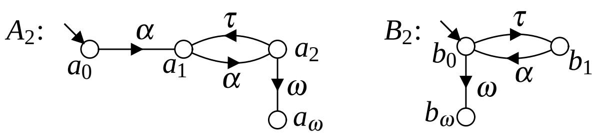

Figure 3: Two automata that are conflict equivalent.

3 Characterising the Conflict Preorder

This section is concerned about characterising two automata A and B as conflict equivalent, or characterising A as less conflicting than B, in a state-based way. First, 3.1 explains the crucial properties of conflict equivalence using examples. *Less conflicting pairs* are introduced in 3.2, and they are used to characterise certain conflicts in 3.3 and the conflict preorder in 3.4.

3.1 Understanding Conflict Equivalence

Every reachable state of an automaton A carries a *nonblocking requirement* (also known as a *nonconflicting completion* [11]) that needs to be satisfied by tests that are to be nonconflicting with A. For example, if $A \overset{s}{\Rightarrow} x_A$, then every test T that can execute s needs to be able to continue with at least one trace $t \in \mathbf{L}^\omega(x_A)$, or T is conflicting with A. An automaton A is less conflicting than another automaton B, if every nonblocking requirement associated with A also is a nonblocking requirement associated with B.

Example 4. Consider again automata A_1 and B_1 in Figure 2. They have the same marked languages. Thus, if the initial state a_0 of A_1 is blocking in combination with some test T, then so is the initial state b_0 of B_1. But this is not the case when $A_1 \parallel T$ enters a state (a_1, x_T) after execution of α. State a_1 requires x_T to be capable of performing at least one trace from the language $\mathbf{L}^\omega(a_1) = (\alpha\alpha)^*\beta\omega + (\alpha\alpha)^*\alpha\gamma\omega$, whereas the states b_1 and b_2, which can both be entered after executing α, require a trace from the language $\alpha^*\beta\omega$ and $\alpha^*\gamma\omega$, respectively. Both of these languages contain traces outside of the language $\mathbf{L}^\omega(a_1)$. Automaton T_1 in Figure 2 is in conflict with A_1 but not with B_1.

In general, it is not enough to compare only the marked languages of states reached by equal traces. Not every nonblocking requirements is a marked language of some state of its automaton. The following example shows one of the problems.

Example 5. Consider automata A_2 and B_2 in Figure 3. The marked language of the initial state of A_2 is $\mathbf{L}^\omega(a_0) = \alpha\alpha^+\omega$, while the marked languages of the two states in B_2 that can be entered initially are $\mathbf{L}^\omega(b_0) = \alpha^*\omega$ and $\mathbf{L}^\omega(b_1) = \alpha^+\omega$. Although the marked languages are different, for any automaton T, if $B_2 \parallel T$ is nonblocking, then $A_2 \parallel T$ must also be nonblocking. If T is to be nonconflicting in combination with B_2, since B_2 may initially enter state b_1, there must be the possibility to continue with event α. However, after executing α, automaton B_2 may again silently enter state b_1, which means that α must be possible again. This is enough to ensure that $A_2 \parallel T$ is nonblocking. Using this argument, it can be shown that A_2 and B_2 are conflict equivalent.

3.2 Less Conflicting Pairs

In order to compare two nondeterministic automata according to conflicts, it is necessary to identify sets of states the two automata may reach under the same input. This is done using the well-known *subset construction* [8]. To capture termination, the usual powerset state space is extended by a special state ω entered only after termination.

Definition 4. The *deterministic state space* of automaton $A = \langle \Sigma, Q, \rightarrow, Q^\circ \rangle$ is

$$Q_A^{\text{det}} = 2^Q \cup \{\omega\} \; , \tag{5}$$

and the *deterministic transition function* $\delta_A^{\text{det}} \colon Q^{\text{det}} \times (\Sigma \cup \{\omega\}) \rightarrow Q^{\text{det}}$ for A is defined as

$$\delta_A^{\text{det}}(X, \sigma) = \begin{cases} \omega, & \text{if } \sigma = \omega \text{ and } X \overset{\omega}{\Rightarrow}; \\ \{\, y \in Q \mid X \overset{\sigma}{\Rightarrow} y \,\}, & \text{otherwise}. \end{cases} \tag{6}$$

The deterministic transition function δ_A^{det} is extended to traces $s \in \Sigma^* \cup \Sigma^*\omega$ in the standard way. Note that $\delta_A^{\text{det}}(X, s)$ is defined for every trace $s \in \Sigma^* \cup \Sigma^*\omega$; if none of the states in X accepts the trace s, this is indicated by $\delta_A^{\text{det}}(X, s) = \emptyset$. This is also true for termination: if ω is enabled in some state in X, then $\delta_A^{\text{det}}(X, \omega) = \omega$, otherwise $\delta_A^{\text{det}}(X, \omega) = \emptyset$.

In order to compare two automata A and B with respect to possible conflicts, *pairs* of state sets of the subset construction of A and B need to be considered. Therefore, the deterministic transition function is also applied to pairs $\mathbf{X} = (X_A, X_B)$ of state sets $X_A \subseteq Q_A$ and $X_B \subseteq Q_B$,

$$\delta_{A,B}^{\text{det}}(\mathbf{X}, s) = \delta_{A,B}^{\text{det}}(X_A, X_B, s) = (\delta_A^{\text{det}}(X_A, s), \delta_B^{\text{det}}(X_B, s)) \; . \tag{7}$$

To determine whether $A \lesssim_{\text{conf}} B$, it is necessary to check all states $x_A \in Q_A$ against matching state sets $X_B \subseteq Q_B$ and determine whether all possible conflicts of x_A are also present in X_B. For example, when automaton A_2 in Figure 3 is in state a_1, then B_2 may be in b_0 or b_1. In state a_1, at least one of the traces in $\alpha^+\omega$ needs to be enabled to avert blocking, and the same requirement to avert blocking is seen in state b_1. When state a_1 is entered with some test T, blocking occurs if none of the traces in $\alpha^+\omega$ is enabled, and such a test T is also blocking when combined with a system that may be in b_0 or b_1. Therefore, a_1 is considered in the following as *less conflicting* (**LC**) than $\{b_0, b_1\}$.

It cannot always be determined directly whether a state $x_A \in Q_A$ is less conflicting than a state set $X_B \subseteq Q_B$. In some cases, it is necessary also to consider the deterministic successors of x_A and X_B. Therefore, the following definition considers pairs (X_A, X_B) of state sets.

Definition 5. Let $A = \langle \Sigma, Q_A, \rightarrow_A, Q_A^\circ \rangle$ and $B = \langle \Sigma, Q_B, \rightarrow_B, Q_B^\circ \rangle$ be automata. The set $\mathbf{LC}(A,B) \subseteq Q_A^{\text{det}} \times Q_B^{\text{det}}$ of *less conflicting pairs* for A and B is inductively defined by

$$\mathbf{LC}^0(A,B) = \{\omega\} \times Q_B^{\text{det}} \cup \{\, (X_A, X_B) \mid X_B \subseteq Q_B \text{ and there exists } x_B \in X_B \text{ with } \mathbf{L}^\omega(x_B) = \emptyset \,\} \; ; \tag{8}$$

$$\mathbf{LC}^{n+1}(A,B) = \{\, (X_A, X_B) \mid \text{there exists } x_B \in X_B \text{ such that for all } t \in \Sigma^*, \text{ if } x_B \overset{t\omega}{\Rightarrow} \text{ then there exists } r \sqsubseteq t\omega \text{ such that } \delta_{A,B}^{\text{det}}(X_A, X_B, r) \in \mathbf{LC}^i(A,B) \text{ for some } i \leq n \,\} \; ; \tag{9}$$

$$\mathbf{LC}(A,B) = \bigcup_{n \geq 0} \mathbf{LC}^n(A,B) \; . \tag{10}$$

Remark 1. If $(X_A, X_B) \notin \mathbf{LC}(A,B)$, then according to (9), for every state $x_B \in X_B$, there exists $t \in \Sigma^*$ such that $x_B \overset{t\omega}{\Rightarrow}$, and $\delta^{\text{det}}(X_A, X_B, r) \notin \mathbf{LC}(A,B)$ for all prefixes $r \sqsubseteq t\omega$.

The idea of Definition 5 is to classify a pair (X_A, X_B) as less conflicting, if the marked language of X_A is a *nonconflicting completion* [11] for the process with initial states X_B. That is, every test that is nonconflicting in combination with each of the states in X_B can terminate with at least one trace from the marked language of X_A. Or conversely, every test that cannot terminate using any of the traces in the marked language of X_A also is conflicting with X_B (see Lemma 1 below).

The first state set X_A of a pair (X_A, X_B) is just used to represent a *language* of possible completions. If state sets X_A and Y_A have the same languages, then all pairs (X_A, X_B) and (Y_A, X_B) have exactly the same less conflicting status. For the second state set X_B on the other hand, the complete nondeterministic behaviour is relevant.

A pair (ω, X_B) is considered as "less conflicting" (8), since termination has already been achieved in A. If X_B contains a state x_B such that $\mathbf{L}^\omega(x_B) = \emptyset$, then (X_A, X_B) also is less conflicting (8), because conflict is guaranteed in X_B. For other pairs (X_A, X_B), it must be checked whether X_B contains a requirement to avert blocking matching that given by the language of X_A (9).

Example 6. Consider again automata A_0 and B_0 in Figure 1. It holds that $(\{a_0\},\{b_0\}) \in \mathbf{LC}^1(A_0, B_0)$. There are three ways to terminate from b_0, by executing ω or $\alpha\beta\omega$ or $\alpha\alpha\beta\omega$. All three traces are possible in a_0, each taking the pair $(\{a_0\},\{b_0\})$ to the deterministic successor $(\omega,\omega) \in \mathbf{LC}^0(A_0, B_0)$. This is enough to confirm that (9) is satisfied.

On the other hand, $(\{a_0\},\{b_2\}) \notin \mathbf{LC}^1(A_0, B_0)$. From state a_0, blocking occurs with a test T that can only execute $\beta\omega$, but this test is nonblocking with b_2. It holds that $b_2 \xrightarrow{\beta\omega}$, where trace $\beta\omega$ has the prefixes ε, β, and $\beta\omega$, but $\delta^{\mathrm{det}}_{A_0,B_0}(\{a_0\},\{b_2\},\varepsilon) = (\{a_0\},\{b_2\}) \notin \mathbf{LC}^0(A_0, B_0)$, $\delta^{\mathrm{det}}_{A_0,B_0}(\{a_0\},\{b_2\},\beta) = (\emptyset,\{b_4\}) \notin \mathbf{LC}^0(A_0, B_0)$, and $\delta^{\mathrm{det}}_{A_0,B_0}(\{a_0\},\{b_2\},\beta\omega) = (\emptyset,\omega) \notin \mathbf{LC}^0(A_0, B_0)$. Therefore, (9) is not satisfied and $(\{a_0\},\{b_2\}) \notin \mathbf{LC}^1(A_0, B_0)$. It can also be shown that $(\{a_0\},\{b_2\}) \notin \mathbf{LC}(A_0, B_0)$.

For a *level-1* less conflicting pair $(X_A, X_B) \in \mathbf{LC}^1(A,B)$, if X_B does not contain blocking states, then there must exist a state $x_B \in X_B$ such that $\mathbf{L}^\omega(x_B) \subseteq \mathbf{L}^\omega(X_A)$. This is not the case for every less conflicting pair, as some nonblocking requirements are only implicitly contained in the automaton. To show that (X_A, X_B) is a less conflicting pair, it is enough to find a state in $x_B \in X_B$ that can cover an initial segment of $\mathbf{L}^\omega(X_A)$, as long as a less conflicting pair of a *lower level* is reached afterwards.

Example 7. Consider again automata A_2 and B_2 in Figure 3. By definition, $(\omega,\omega) \in \mathbf{LC}^0(A_2, B_2)$, and following from this, $(\{a_1\},\{b_0,b_1\}) \in \mathbf{LC}^1(A_2, B_2)$, because the marked language of a_1 is $\alpha^+\omega$, which also is the marked language of b_1.

Now consider the pair $(\{a_0\},\{b_0,b_1\})$. State a_0 has the marked language $\alpha\alpha^+\omega$, i.e., to avert blocking from a_0, a test must be able to execute at least one of the traces in $\alpha\alpha^+\omega$. Although this language is not directly associated with any state in B_2, the nonblocking requirement is implicitly present in state b_1. If blocking is to be averted from state b_1, event α must be possible. After executing α, state b_0 is entered, from where it is always possible to silently return to state b_1 with marked language $\alpha^+\omega$. Therefore, in order to avert blocking from state b_1, it is necessary to execute α and afterwards be able to terminate using one of the traces in $\alpha^+\omega$. This amounts to the implicit nonblocking requirement to execute a trace from $\alpha\alpha^+\omega$ in state b_1.

Therefore $(\{a_0\},\{b_0,b_1\}) \notin \mathbf{LC}^1(A_2, B_2)$, but $(\{a_0\},\{b_0,b_1\}) \in \mathbf{LC}^2(A_2, B_2)$ according to (9): every trace that leads to a terminal state from state b_1 has the prefix α, and $\delta^{\mathrm{det}}_{A_2,B_2}(\{a_0\},\{b_0,b_1\},\alpha) = (\{a_1\},\{b_0,b_1\}) \in \mathbf{LC}^1(A_2, B_2)$.

As shown in the example, some nonblocking requirements have to be constructed using a saturation operation that combines two previously found nonblocking requirements. The level n of a less conflicting pair $(X_A, X_B) \in \mathbf{LC}^n(A,B)$ represents the nesting depth of applications of this saturation operation.

The following two lemmas relate the state-based definition of less conflicting pairs to possible tests and thus to the conflict preorder. A pair (X_A, X_B) is a less conflicting pair, if every test T such that $\mathbf{L}^\omega(X_A) \cap \mathbf{L}^\omega(T) = \emptyset$ also is conflicting with X_B.

Lemma 1. Let $A = \langle \Sigma, Q_A, \rightarrow_A, Q_A^\circ \rangle$, $B = \langle \Sigma, Q_B, \rightarrow_B, Q_B^\circ \rangle$, and $T = \langle \Sigma, Q_T, \rightarrow_T, Q_T^\circ \rangle$ be automata, and let $x_T \in Q_T$ be a (possibly unreachable) state. For every less conflicting pair $(X_A, X_B) \in \mathbf{LC}(A,B)$, at least one of the following conditions holds.

(i) $X_A = \omega$, or $X_A \subseteq Q_A$ and there exists $x_A \in X_A$ such that $\mathbf{L}^\omega(x_A, x_T) \neq \emptyset$.

(ii) There exist states $x_B \in X_B$, $y_B \in Q_B$, and $y_T \in Q_T$ such that $(x_B, x_T) \Rightarrow (y_B, y_T)$ and $\mathbf{L}^\omega(y_B, y_T) = \emptyset$.

(Here and in the following, notation $\mathbf{L}^\omega(x_A, x_T)$ is abused to be a shorthand for $\mathbf{L}^\omega((x_A, x_T))$.)

Proof. As (X_A, X_B) is a less conflicting pair, it holds that $(X_A, X_B) \in \mathbf{LC}^n(A, B)$ for some $n \in \mathbb{N}_0$. The claim is shown by induction on n.

If $(X_A, X_B) \in \mathbf{LC}^0(A, B)$ then by (8) it holds that $X_A = \omega$, or $X_B \subseteq Q_B$ and there exists $x_B \in X_B$ such that $\mathbf{L}^\omega(x_B) = \emptyset$. In the first case (i) holds, and in the second case (ii) holds as $(x_B, x_T) \xrightarrow{\varepsilon} (x_B, x_T)$ and $\mathbf{L}^\omega(x_B, x_T) = \mathbf{L}^\omega(x_B) \cap \mathbf{L}^\omega(x_T) = \emptyset$.

Now assume the claim holds for all $i \leq n$, i.e., for all $(X_A, X_B) \in \mathbf{LC}^i(A, B)$, one of the conditions (i) or (ii) holds, and consider $(X_A, X_B) \in \mathbf{LC}^{n+1}(A, B)$. By (9), there exists $x_B \in X_B$ such that for all $t \in \Sigma^*$, if $x_B \overset{t\omega}{\Rightarrow}$ then there exists a prefix $r \sqsubseteq t\omega$ such that $\delta^{\text{det}}_{A,B}(X_A, X_B, r) \in \mathbf{LC}^i(A, B)$ for some $i \leq n$. If $\mathbf{L}^\omega(x_B, x_T) = \emptyset$, (ii) follows immediately as $(x_B, x_T) \xrightarrow{\varepsilon} (x_B, x_T)$. Therefore assume that $\mathbf{L}^\omega(x_B, x_T) \neq \emptyset$, i.e., there exists $t \in \Sigma^*$ such that $(x_B, x_T) \overset{t\omega}{\Rightarrow}$. Then $x_B \overset{t\omega}{\Rightarrow}$, so there exists $r \sqsubseteq t\omega$ such that $\delta^{\text{det}}_{A,B}(X_A, X_B, r) \in \mathbf{LC}^i(A, B)$ for some $i \leq n$. As $r \sqsubseteq t\omega$ and $x_T \overset{t\omega}{\Rightarrow}$, it also holds that $x_T \overset{r}{\Rightarrow} y_T$ for some $y_T \in Q_T$. Let $\delta^{\text{det}}_{A,B}(X_A, X_B, r) = (Y_A, Y_B)$. By inductive assumption, (i) or (ii) holds for $(Y_A, Y_B) \in \mathbf{LC}^i(A, B)$ and y_T.

(i) In this case, either $Y_A = \omega$, or $Y_A \subseteq Q_A$ and there exists $y_A \in Y_A$ and $u \in \Sigma^*$ such that $(y_A, y_T) \overset{u\omega}{\Rightarrow}$. If $Y_A = \omega$, then $\delta^{\text{det}}_A(X_A, r) = Y_A = \omega$ and according to Definition 4 there exists $r_A \in \Sigma^*$ such that $r = r_A\omega$, and there exist states $x_A \in X_A$ and $y_A \in Q_A$ such that $x_A \overset{r_A}{\Rightarrow} y_A \overset{\omega}{\Rightarrow}$, i.e., $(x_A, x_T) \overset{r_A\omega}{\Longrightarrow}$. If there exists $y_A \in Y_A$ and $u \in \Sigma^*$ such that $(y_A, y_T) \overset{u\omega}{\Rightarrow}$, then since $\delta^{\text{det}}_A(X_A, r) = Y_A$, there exists $x_A \in X_A$ such that $x_A \overset{r}{\Rightarrow} y_A$, i.e., $(x_A, x_T) \overset{r}{\Rightarrow} (y_A, y_T) \overset{u\omega}{\Rightarrow}$. In both cases, (i) holds for (X_A, X_B) and x_T.

(ii) If there exists a state $y_B \in Y_B$ such that $(y_B, y_T) \Rightarrow (z_B, z_T)$ where $\mathbf{L}^\omega(z_B, z_T) = \emptyset$, then since $\delta^{\text{det}}_B(X_B, r) = Y_B$, there exists $x_B \in X_B$ such that $x_B \overset{r}{\Rightarrow} y_B$, which implies $(x_B, x_T) \overset{r}{\Rightarrow} (y_B, y_T) \Rightarrow (z_B, z_T)$ with $\mathbf{L}^\omega(z_B, z_T) = \emptyset$. Thus, (ii) holds for (X_A, X_B) and x_T. ☐

Conversely, if a pair of state sets is *not* a less conflicting pair for A and B, then this pair gives rise to a test automaton to show that A is not less conflicting than B. This test exhibits blocking behaviour in combination with A but not with B.

Lemma 2. Let $A = \langle \Sigma, Q_A, \rightarrow_A, Q^\circ_A \rangle$ and $B = \langle \Sigma, Q_B, \rightarrow_B, Q^\circ_B \rangle$ be automata. For every pair $\mathbf{X} = (X_A, X_B) \notin \mathbf{LC}(A, B)$, there exists a deterministic automaton $T_\mathbf{X} = \langle \Sigma, Q_T, \rightarrow_T, \{x^\circ_T\} \rangle$ such that both the following conditions hold.

(i) For all states $x_A \in X_A$, it holds that $\mathbf{L}^\omega(x_A, x^\circ_T) = \emptyset$.

(ii) For all states $x_B \in X_B$, $y_B \in Q_B$, $y_T \in Q_T$ such that $(x_B, x^\circ_T) \Rightarrow (y_B, y_T)$, it holds that $\mathbf{L}^\omega(y_B, y_T) \neq \emptyset$.

Proof. Construct the deterministic automaton $T_\mathbf{X} = \langle \Sigma, Q_T, \rightarrow_T, \{x^\circ_T\} \rangle$ such that

$$\mathbf{L}(T_\mathbf{X}) = \{\, s \in \Sigma^* \cup \Sigma^*\omega \mid \delta^{\text{det}}_{A,B}(\mathbf{X}, r) \notin \mathbf{LC}(A, B) \text{ for all } r \sqsubseteq s \,\} \ . \tag{11}$$

This language is prefix-closed by construction and nonempty because $\mathbf{X} \notin \mathbf{LC}(A, B)$. Therefore, $T_\mathbf{X}$ is a well-defined automaton.

(i) Let $x_A \in X_A$. If $x_A \overset{t\omega}{\Rightarrow}$ for some $t \in \Sigma^*$, then $\delta^{\text{det}}_{A,B}(\mathbf{X}, t\omega) = (\omega, Y_B) \in \mathbf{LC}^0(A, B) \subseteq \mathbf{LC}(A, B)$ for some $Y_B \in Q^{\text{det}}_B$ by Definition 4 and 5. It follows from (11) that $t\omega \notin \mathbf{L}(T_\mathbf{X})$, and thus $(x_A, x^\circ_T) \overset{t\omega}{\Rightarrow}$ does not hold. Since $t \in \Sigma^*$ was chosen arbitrarily, it follows that $\mathbf{L}^\omega(x_A, x^\circ_T) = \emptyset$.

(ii) Let $x_B \in X_B$, $y_B \in Q_B$, $y_T \in Q_T$, and $s \in \Sigma^*$ such that $(x_B, x_T^\circ) \overset{s}{\Rightarrow} (y_B, y_T)$. Clearly $s \in \mathbf{L}(T_\mathbf{X})$, and by (11) it follows that $\delta^{\det}_{A,B}(\mathbf{X}, r) \notin \mathbf{LC}(A,B)$ for all prefixes $r \sqsubseteq s$. Let $\delta^{\det}_{A,B}(\mathbf{X}, s) = \mathbf{Y}$. Then $\mathbf{Y} \notin \mathbf{LC}(A,B)$, so there exists a trace $t \in \Sigma^*$ such that $y_B \overset{t\omega}{\Rightarrow}$ and for all $r \sqsubseteq t$ it holds that $\delta^{\det}_{A,B}(\mathbf{Y}, r) \notin \mathbf{LC}(A,B)$ (see Remark 1). Thus $x_B \overset{s}{\Rightarrow} y_B \overset{t\omega}{\Rightarrow}$ and for all prefixes $u \sqsubseteq st\omega$, it holds that $\delta^{\det}_{A,B}(\mathbf{X}, u) \notin \mathbf{LC}(A,B)$. Then $st\omega \in \mathbf{L}(T_\mathbf{X})$ according to (11), and since $T_\mathbf{X}$ is deterministic, it follows that $y_T \overset{t\omega}{\Rightarrow}$. Therefore, $(y_B, y_T) \overset{t\omega}{\Rightarrow}$, i.e., $\mathbf{L}^\omega(y_B, y_T) \neq \emptyset$. □

3.3 Less Conflicting Pairs and Certain Conflicts

Less conflicting pairs can be used to characterise the set of *certain conflicts* of an automaton as defined in 2.3. This shows the close link between the conflict preorder and the set of certain conflicts. If a pair $(\emptyset, X_B)$ is a less conflicting pair then, since termination is impossible from $\emptyset$, conflict must be also present in X_B. In this case, every trace leading to X_B must be a trace of certain conflicts. This observation leads to the following alternative characterisation of the set of certain conflicts.

Theorem 3. The set of certain conflicts of $B = \langle \Sigma, Q, \rightarrow, Q^\circ \rangle$ can also be written as

$$\textsc{Conf}(B) = \{\, s \in \Sigma^* \mid (\emptyset, \delta^{\det}_B(Q^\circ, r)) \in \mathbf{LC}(O,B) \text{ for some prefix } r \sqsubseteq s \,\} \,, \tag{12}$$

where $O = \langle \Sigma, \emptyset, \emptyset, \emptyset \rangle$ stands for the empty automaton.

Proof. First let $s \in \Sigma^*$ such that $(\emptyset, \delta^{\det}_B(Q^\circ, r)) \in \mathbf{LC}(O,B)$ for some $r \sqsubseteq s$, and let $T = \langle \Sigma, Q_T, \rightarrow_T, Q_T^\circ \rangle$ be an automaton such that $T \overset{s}{\Rightarrow}$. It is to be shown that $B \,\|\, T$ is blocking. Since $T \overset{s}{\Rightarrow}$ and $r \sqsubseteq s$, it holds that $T \overset{r}{\Rightarrow} x_T$ for some state $x_T \in Q_T$. Since $(\emptyset, \delta^{\det}_B(Q^\circ, r)) \in \mathbf{LC}(O,B)$, either (i) or (ii) in Lemma 1 holds. However, (i) is impossible as the first state set of the pair is empty, so (ii) must be true. Thus, there exists a state $x \in \delta^{\det}_B(Q^\circ, r)$ such that $(x, x_T) \Rightarrow (y, y_T)$ where $\mathbf{L}^\omega(y, y_T) = \emptyset$. Then $B \,\|\, T$ is blocking as $B \,\|\, T \overset{r}{\Rightarrow} (x, x_T) \Rightarrow (y, y_T)$.

Conversely, let $s \in \Sigma^*$ such that $(\emptyset, \delta^{\det}_B(Q^\circ, r)) \notin \mathbf{LC}(O,B)$ for every prefix $r \sqsubseteq s$. It is to be shown that $s \in \textsc{NConf}(B)$. Consider the deterministic automaton T such that

$$\mathbf{L}(T) = \{\, t \in \Sigma^* \mid (\emptyset, \delta^{\det}_B(Q^\circ, r)) \notin \mathbf{LC}(O,B) \text{ for all } r \sqsubseteq t \,\} \,. \tag{13}$$

T is a well-defined automaton as $\mathbf{L}(T)$ is prefix-closed by construction. It remains to be shown that $B \,\|\, T$ is nonblocking. Let $B \,\|\, T \overset{t}{\Rightarrow} (x, x_T)$. Then $t \in \mathbf{L}(T)$, and by definition of T (13), it holds that $(\emptyset, \delta^{\det}_B(Q^\circ, t)) \notin \mathbf{LC}(O,B)$, and the same holds for all prefixes of t. Also $x \in \delta^{\det}_B(Q^\circ, t)$, so there exists a trace $u \in \Sigma^*$ such that $x \overset{u\omega}{\Rightarrow}$, and for every prefix $r \sqsubseteq u\omega$, it holds that $\delta^{\det}_{O,B}(\emptyset, \delta^{\det}_B(Q^\circ, t), r) \notin \mathbf{LC}(O,B)$ (see Remark 1). By definition (13), it follows that $tu\omega \in \mathbf{L}(T)$, and since T is deterministic also $x_T \overset{u\omega}{\Rightarrow}$. Therefore, $B \,\|\, T \overset{t}{\Rightarrow} (x, x_T) \overset{u\omega}{\Rightarrow}$, i.e., $B \,\|\, T$ is nonblocking. □

The result of Theorem 3 shows how less conflicting pairs generalise certain conflicts for the case when two automata are compared, and in combination with the algorithm in Section 4, less conflicting pairs lead to an alternative presentation of the algorithm [10] to compute the set of certain conflicts.

3.4 Testing the Conflict Preorder

Given the less conflicting pairs for two automata A and B, it is possible to determine whether $A \lesssim_{\text{conf}} B$. Automaton A is less conflicting than B if every test T that is nonconflicting in combination with B also is nonconflicting with A. To check this condition, it is enough to consider traces $B \,\|\, T \overset{s}{\Rightarrow} (x_B, x_T)$, and check whether termination is also possible for every state x_A of A such that $A \,\|\, T \overset{s}{\Rightarrow} (x_A, x_T)$. This amounts to checking whether $(\{x_A\}, X_B) \in \mathbf{LC}(A,B)$ when $A \overset{s}{\Rightarrow} x_A$ and $\delta_B^{\det}(Q_B^\circ, s) = X_B$.

However, this condition does not apply to traces of certain conflicts. If $s \in \textsc{Conf}(B)$, then every test T that can execute s is in conflict with B. In this case, A can still be less conflicting than B, no matter whether A can or cannot execute the trace s and terminate afterwards. This observation leads to the following result.

Theorem 4. Let $A = \langle \Sigma, Q_A, \rightarrow_A, Q_A^\circ \rangle$ and $B = \langle \Sigma, Q_B, \rightarrow_B, Q_B^\circ \rangle$ be two automata. A is less conflicting than B if and only if for all $s \in \textsc{NConf}(B)$ and all $x_A \in Q_A$ such that $A \overset{s}{\Rightarrow} x_A$ it holds that $(\{x_A\}, X_B) \in \mathbf{LC}(A,B)$, where $\delta_B^{\det}(Q_B^\circ, s) = X_B$.

Proof. First assume that for all $s \in \textsc{NConf}(B)$ and all $x_A \in Q_A$ such that $A \overset{s}{\Rightarrow} x_A$ it holds that $(\{x_A\}, X_B) \in \mathbf{LC}(A,B)$, where $\delta_B^{\det}(Q_B^\circ, s) = X_B$. Let $T = \langle \Sigma, Q_T, \rightarrow_T, Q_T^\circ \rangle$ such that $B \,\|\, T$ is nonblocking, and assume that $A \,\|\, T \overset{s}{\Rightarrow} (x_A, x_T)$. Since $B \,\|\, T$ is nonblocking and $T \overset{s}{\Rightarrow}$, it follows that $s \in \textsc{NConf}(B)$. Therefore by assumption $(\{x_A\}, X_B) \in \mathbf{LC}(A,B)$, so (i) or (ii) in Lemma 1 must be true. However, (ii) cannot hold, because for all $x_B \in X_B = \delta_B^{\det}(Q^\circ, s)$ it holds that $B \,\|\, T \overset{s}{\Rightarrow} (x_B, x_T)$, and since $B \,\|\, T$ is nonblocking, there cannot exist any state (y_B, y_T) such that $(x_B, x_T) \Rightarrow (y_B, y_T)$ and $\mathbf{L}^\omega(y_B, y_T) = \emptyset$. Thus, (i) must be true, and this means that $\mathbf{L}^\omega(x_A, x_T) \neq \emptyset$. Since T and s such that $A \,\|\, T \overset{s}{\Rightarrow} (x_A, x_T)$ were chosen arbitrarily, it follows that $A \lesssim_{\text{conf}} B$.

Second assume that there exists $s \in \textsc{NConf}(B)$ and $x_A \in Q_A$ such that $A \overset{s}{\Rightarrow} x_A$ and $\mathbf{X} = (\{x_A\}, X_B) \notin \mathbf{LC}(A,B)$, where $X_B = \delta_B^{\det}(Q_B^\circ, s)$. Let $N_B = \langle \Sigma, Q_N, \rightarrow_N, \{x_N^\circ\} \rangle$ be a deterministic recogniser of the language $\textsc{NConf}(B)$, and let $T_{\mathbf{X}} = \langle \Sigma, Q_T, \rightarrow_T, \{x_T^\circ\} \rangle$ be the deterministic automaton that exists according to Lemma 2. Since $s \in \textsc{NConf}(B)$, there exists a unique state $x_s \in Q_N$ such that $N_B \xrightarrow{s} x_s$. Then construct the automaton

$$T = \langle \Sigma, Q_N \,\dot\cup\, Q_T, \rightarrow_N \cup \rightarrow_T \cup \{(x_s, \tau, x_T^\circ)\}, \{x_N^\circ\} \rangle \ . \tag{14}$$

Clearly, $A \,\|\, T \overset{s}{\Rightarrow} (x_A, x_s) \xrightarrow{\tau} (x_A, x_T^\circ)$, and $\mathbf{L}^\omega(x_A, x_T^\circ) = \emptyset$ by Lemma 2 (i). Thus, $A \,\|\, T$ is blocking.

On the other hand, $B \,\|\, T$ is nonblocking. To see this, consider $B \,\|\, T \overset{t}{\Rightarrow} (y_B, y_T)$. If $y_T \in Q_N$, then it follows from the fact that $B \,\|\, N_B$ is nonblocking [11] that there exists $u \in \Sigma^*$ such that $(y_B, y_T) \overset{u\omega}{\Rightarrow}$. Otherwise $y_T \in Q_T$, which means that $t = su$ and $T \xrightarrow{s} x_s \xrightarrow{\tau} x_T^\circ \xrightarrow{u} y_T$. Also since $B \overset{t}{\Rightarrow} y_B$, it follows that $y_B \in \delta_B^{\det}(Q_B^\circ, t) = \delta^{\det}(Q_B^\circ, su) = \delta_B^{\det}(\delta_B^{\det}(Q_B^\circ, s), u) = \delta_B^{\det}(X_B, u)$, i.e., there exists $x_B \in X_B$ such that $x_B \overset{u}{\Rightarrow} y_B$. Thus $(x_B, x_T^\circ) \overset{u}{\Rightarrow} (y_B, y_T)$, and by Lemma 2 (ii), it holds that $\mathbf{L}^\omega(y_B, y_T) \neq \emptyset$.

Thus, $A \,\|\, T$ is blocking and $B \,\|\, T$ is nonblocking, so $A \lesssim_{\text{conf}} B$ cannot hold. $\square$

Example 8. Consider again automata A_0 and B_0 in Figure 1. Recall that $\textsc{Conf}(B_0) = \alpha\Sigma^*$ from Example 3, so the only state in A_0 that can be reached by a trace $s \notin \textsc{Conf}(B_0)$ is a_0. Therefore, it is enough to check the pair $(\{a_0\}, \{b_0\})$ according to Theorem 4, and it has been shown in Example 6 that $(\{a_0\}, \{b_0\}) \in \mathbf{LC}^1(A_0, B_0)$. It follows that $A_0 \lesssim_{\text{conf}} B_0$. This conclusion is made despite the fact that $(\{a_0\}, \{b_2\}) \notin \mathbf{LC}(A_0, B_0)$, because $(\{a_0\}, \{b_2\})$ is only reachable by traces $\alpha^n \in \textsc{Conf}(B_0)$, $n \geq 2$.

When using Theorem 4 to determine whether an automaton A is less conflicting than some blocking automaton B, the set of certain conflicts of B must be known first. This can be achieved using Theorem 3, which makes it possible to classify state sets in the subset construction of B as certain conflicts. If a state set $X_B \subseteq Q_B$ is found to represent certain conflicts, i.e., $(\emptyset, X_B) \in \mathbf{LC}(O, B)$ according to Theorem 3, then $(X_A, X_B) \in \mathbf{LC}(A,B)$ for every state set $X_A \subseteq Q_A$. Successors reached only from such pairs are also certain conflicts of B and should not be considered when testing whether $A \lesssim_{\text{conf}} B$ according to Theorem 4.

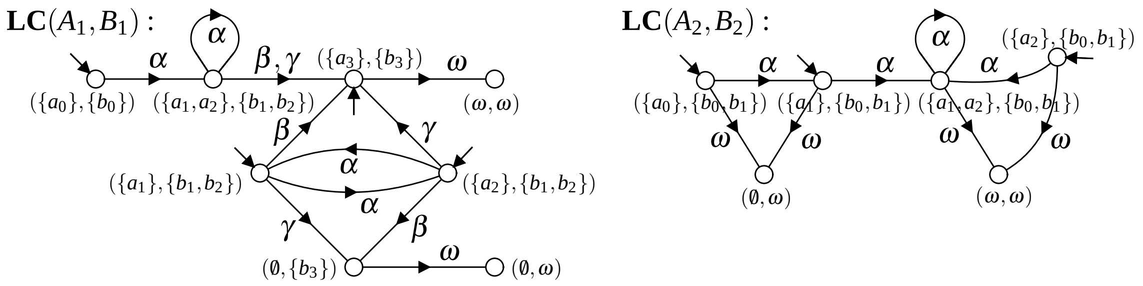

Figure 4: Less conflicting pairs for the automata pairs in Figure 2 and 3.

Example 9. Consider again automata A_1 and B_1 in Figure 2. Composing A_1 with a deterministic version of B_1 results in the following four pairs of states in A_1 and sets of states in B_1 that should be tested according to Theorem 4 to determine whether $A_1 \lesssim_{\mathrm{conf}} B_1$:

$$(\{a_0\},\{b_0\}) \quad (\{a_1\},\{b_1,b_2\}) \quad (\{a_2\},\{b_1,b_2\}) \quad (\{a_3\},\{b_3\})\ . \tag{15}$$

All four pairs need to be considered as B_1 is nonblocking and thus $\textsc{Conf}(B_1) = \emptyset$.

The graph to the left in Figure 4 shows these four pairs and their deterministic successors. The four pairs (15) are marked as initial states, and the arrows in the graph represent the deterministic transition function. Although the deterministic transition function is defined for all state set pairs and events, arrows to $(\emptyset,\emptyset)$ are suppressed for clarity of presentation.

The following less conflicting pairs to compare A_1 to B_1 are determined from the graph:

$$(\omega,\omega) \in \mathbf{LC}^0(A_1,B_1)\ ; \tag{16}$$

$$(\{a_0\},\{b_0\}),\ (\{a_1,a_2\},\{b_1,b_2\}),\ (\{a_3\},\{b_3\}) \in \mathbf{LC}^1(A_1,B_1)\ . \tag{17}$$

For example, $(\{a_1,a_2\},\{b_1,b_2\}) \in \mathbf{LC}^1(A_1,B_1)$, because all the ways to reach termination from state b_1, i.e., all traces in $\mathbf{L}^\omega(b_1) = \alpha^*\beta\omega$ take the pair $(\{a_1,a_2\},\{b_1,b_2\})$ to $(\omega,\omega) \in LC^0(A_1,B_1)$. No further pairs are found in $\mathbf{LC}^2(A_1,B_1)$, so $\mathbf{LC}(A_1,B_1)$ consists only of the pairs listed above. For example, $(\{a_1\},\{b_1,b_2\}) \notin \mathbf{LC}^2(A_1,B_1)$, because the traces $\alpha\beta\omega \in \mathbf{L}^\omega(b_1)$ and $\gamma\omega \in \mathbf{L}^\omega(b_2)$ do not have any prefixes that reach a pair in $\mathbf{LC}^1(A_1,B_1)$.

As $(\{a_1\},\{b_1,b_2\}) \notin \mathbf{LC}(A_1,B_1)$, it follows from Theorem 4 that A_1 is *not* less conflicting than B_1.

Example 10. Consider again automata A_2 and B_2 in Figure 3. Again note that $\textsc{Conf}(B_2) = \emptyset$. By composing A_2 with a deterministic version of B_2, it becomes clear that the only pairs that need to be tested to determine whether $A_2 \lesssim_{\mathrm{conf}} B_2$ according to Theorem 4 are $(\{a_0\},\{b_0,b_1\})$ reached after ε, $(\{a_1\},\{b_0,b_1\})$ reached after α^+, and $(\{a_2\},\{b_0,b_1\})$ reached after $\alpha\alpha^+$.

The graph with these pairs and their deterministic successors is shown to the right in Figure 4, with the three crucial pairs marked as initial. The following less conflicting pairs are discovered (see Example 7):

$$(\omega,\omega) \in \mathbf{LC}^0(A_2,B_2)\ ; \tag{18}$$

$$(\{a_1\},\ \{b_0,b_1\}),\ (\{a_1,a_2\},\{b_0,b_1\}),\ (\{a_2\},\{b_0,b_1\}) \in \mathbf{LC}^1(A_2,B_2)\ ; \tag{19}$$

$$(\{a_0\},\{b_0,b_1\}) \in \mathbf{LC}^2(A_2,B_2)\ . \tag{20}$$

As the three crucial pairs are all in $\mathbf{LC}(A_2,B_2)$, it follows from Theorem 4 that $A_2 \lesssim_{\mathrm{conf}} B_2$.

The result of Theorem 4 is related to the decision procedure for fair testing [15]. The fair testing decision procedure starts by composing the automaton A with a determinised form of B, which gives rise to the same state set combinations that need to be considered as in Theorem 4. From this point on, the two methods differ. The fair testing decision procedure annotates each state of the synchronous product of A and the determinised form of B with automata representing the associated refusal trees, and searches for matching automata (or more precisely, for matching *productive subautomata*) within these annotations. The method based on less conflicting pairs avoids some of the resulting complexity by performing the complete decision on the flat state space of the synchronous product of the determinised forms of A and B.

4 Algorithm to Compute Less Conflicting Pairs

This section proposes a method to effectively compute the less conflicting pairs for two given finite-state automata A and B. This is done in a nested iteration. Assuming that the set $\mathbf{LC}^n(A,B)$ is already known, the set $\mathbf{LC}^{n+1}(A,B)$ is computed in a secondary iteration based on *more conflicting triples*.

Definition 6. Let $A = \langle \Sigma, Q_A, \to_A, Q_A^\circ \rangle$ and $B = \langle \Sigma, Q_B, \to_B, Q_B^\circ \rangle$ be automata. The set $\mathbf{MC}^n(A,B) \subseteq Q_A^{\det} \times Q_B^{\det} \times Q_B$ of nth level *more conflicting triples* for A and B is defined inductively as follows.

$$\mathbf{MC}_0^n(A,B) = \{ (\emptyset, \omega, x_B) \mid x_B \in Q_B \} \; ; \tag{21}$$

$$\mathbf{MC}_{m+1}^n(A,B) = \{ (X_A, X_B, x_B) \mid (X_A, X_B) \notin \mathbf{LC}^n(A,B) \text{ and } x_B \in X_B \text{ and there exists } (Y_A, Y_B, y_B) \in \mathbf{MC}_m^n(A,B) \text{ and } \sigma \in \Sigma \text{ such that } \delta_{A,B}^{\det}(X_A, X_B, \sigma) = (Y_A, Y_B) \text{ and } x_B \overset{\sigma}{\Rightarrow} y_B \} \; ; \tag{22}$$

$$\mathbf{MC}^n(A,B) = \bigcup_{m \geq 0} \mathbf{MC}_m^n(A,B) \; . \tag{23}$$

For a pair (X_A, X_B) to be a less conflicting pair, according to Definition 5 there must be a state $x_B \in X_B$ such that every trace that takes x_B to termination in B has a prefix that leads to another less conflicting pair. A triple (X_A, X_B, x_B) is considered “more conflicting” if (X_A, X_B) is not yet known to be a less conflicting pair, and the state $x_B \in X_B$ cannot be used to confirm the above property. Therefore, Lemma 5 shows that a triple (X_A, X_B, x_B) is nth-level “more conflicting” if and only if the state $x_B \in X_B$ can reach termination without passing through a pair in $\mathbf{LC}^n$.

If (X_A, X_B, x_B) is “more conflicting” for all $x_B \in X_B$, then the pair (X_A, X_B) cannot be a less conflicting pair. Otherwise, if there exists at least one state $x_B \in X_B$ such that (X_A, X_B, x_B) is not “more conflicting”, then (X_A, X_B) is added to set of less conflicting pairs in the next iteration. Theorem 6 below confirms the correctness of this approach.

Lemma 5. Let $A = \langle \Sigma, Q_A, \to_A, Q_A^\circ \rangle$ and $B = \langle \Sigma, Q_B, \to_B, Q_B^\circ \rangle$ be automata, let $n \in \mathbb{N}_0$ and $(X_A, X_B, x_B) \in Q_A^{\det} \times Q_B^{\det} \times Q_B$. The following statements are equivalent.

(i) $(X_A, X_B, x_B) \in \mathbf{MC}^n(A,B)$;

(ii) There exists a trace $s \in \Sigma^* \omega \cup \{\varepsilon\}$ such that $\delta_{A,B}^{\det}(X_A, X_B, s) = (\emptyset, \omega)$ and $x_B \overset{s}{\Rightarrow}$, and $\delta_{A,B}^{\det}(X_A, X_B, r) \notin \mathbf{LC}^n(A,B)$ for all prefixes $r \sqsubseteq s$.

Proof. First let $(X_A, X_B, x_B) \in \mathbf{MC}^n(A,B)$, i.e., $(X_A, X_B, x_B) \in \mathbf{MC}_m^n(A,B)$ for some $m \in \mathbb{N}_0$. It is shown by induction on m that (ii) holds.

In the base case, $m = 0$, and by definition $(X_A, X_B, x_B) \in \mathbf{MC}_0^n(A,B)$ means that $(X_A, X_B) = (\emptyset, \omega)$. Then consider $s = \varepsilon$, and note $\delta_{A,B}^{\det}(X_A, X_B, \varepsilon) = (X_A, X_B) = (\emptyset, \omega)$ and $x_B \overset{\varepsilon}{\Rightarrow}$. Clearly $r \sqsubseteq \varepsilon$ implies $r = \varepsilon$, and $\delta_{A,B}^{\det}(X_A, X_B, \varepsilon) = (\emptyset, \omega) \notin \mathbf{LC}(A,B) \supseteq \mathbf{LC}^n(A,B)$ by Lemma 1.

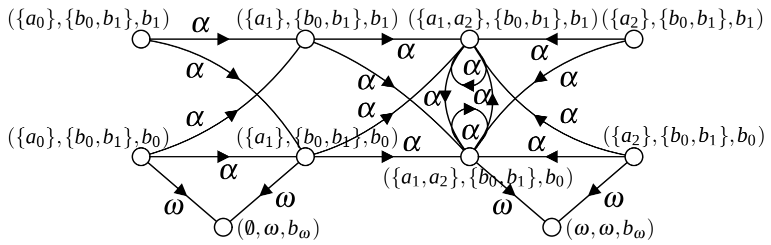
$(\{a_0\},\{b_0,b_1\},b_1)$
$(\{a_1\},\{b_0,b_1\},b_1)$
$(\{a_1,a_2\},\{b_0,b_1\},b_1)$
$(\{a_2\},\{b_0,b_1\},b_1)$
$(\{a_0\},\{b_0,b_1\},b_0)$
$(\{a_1\},\{b_0,b_1\},b_0)$
$(\{a_1,a_2\},\{b_0,b_1\},b_0)$
$(\{a_2\},\{b_0,b_1\},b_0)$
$(\emptyset,\omega,b_\omega)$
(ω,ω,b_ω)
α
ω

Figure 5: Calculating more conflicting triples for automata A_2 and B_2 in Figure 3.

Now consider $(X_A,X_B,x_B) \in \mathbf{MC}^n_{m+1}(A,B)$. It follows from Definition 6 that $(X_A,X_B) \notin LC^n(A,B)$ and $x_B \in X_B$, and there exists $(Y_A,Y_B,y_B) \in \mathbf{MC}^n_m(A,B)$ and $\sigma \in \Sigma$ such that $\delta^{\det}_{A,B}(X_A,X_B,\sigma) = (Y_A,Y_B)$ and $x_B \stackrel{\sigma}{\Rightarrow} y_B$. By inductive assumption, there exists a trace $s \in \Sigma^*\omega \cup \{\varepsilon\}$ such that $\delta^{\det}_{A,B}(Y_A,Y_B,s) = (\emptyset,\omega)$ and $y_B \stackrel{s}{\Rightarrow}$, and for all $r \sqsubseteq s$ it holds that $\delta^{\det}_{A,B}(Y_A,Y_B,r) \notin \mathbf{LC}^n(A,B)$. Then $\delta^{\det}_{A,B}(X_A,X_B,\sigma s) = \delta^{\det}_{A,B}(Y_A,Y_B,s) = (\emptyset,\omega)$ and $x_B \stackrel{\sigma}{\Rightarrow} y_B \stackrel{s}{\Rightarrow}$, and for all $r \sqsubseteq \sigma s$ it holds that $\delta^{\det}_{A,B}(X_A,X_B,r) \notin \mathbf{LC}^n(A,B)$.

Conversely, let $s \in \Sigma^*\omega \cup \{\varepsilon\}$ such that (ii) holds. This means that $\delta^{\det}_{A,B}(X_A,X_B,s) = (\emptyset,\omega)$ and $x_B \stackrel{s}{\Rightarrow}$, and $\delta^{\det}_{A,B}(X_A,X_B,r) \notin \mathbf{LC}^n(A,B)$ for all $r \sqsubseteq s$. It is shown by induction on $m = |s|$ that $(X_A,X_B,x_B) \in \mathbf{MC}^n_m(A,B)$.

In the base case, $m = 0$ and $s = \varepsilon$, it holds by definition that $(X_A,X_B) = \delta^{\det}_{A,B}(X_A,X_B,\varepsilon) = (\emptyset,\omega) \in \mathbf{MC}^n_0(A,B)$.

Now let $s = \sigma t$ such that $|t| = m$, and $\delta^{\det}_{A,B}(X_A,X_B,s) = (\emptyset,\omega)$ and $x_B \stackrel{s}{\Rightarrow}$, and $\delta^{\det}_{A,B}(X_A,X_B,r) \notin \mathbf{LC}^n(A,B)$ for all prefixes $r \sqsubseteq s$. Write $\delta^{\det}_{A,B}(X_A,X_B,\sigma) = (Y_A,Y_B)$ and $x_B \stackrel{\sigma}{\Rightarrow} y_B \stackrel{t}{\Rightarrow}$. Then $y_B \stackrel{t}{\Rightarrow}$ and $\delta^{\det}_{A,B}(Y_A,Y_B,t) = \delta^{\det}_{A,B}(X_A,X_B,\sigma t) = \delta^{\det}_{A,B}(X_A,X_B,s) = (\emptyset,\omega)$ and $\delta^{\det}_{A,B}(Y_A,Y_B,r) \notin \mathbf{LC}^n(A,B)$ for all $r \sqsubseteq t$. Then $(Y_A,Y_B,y_B) \in \mathbf{MC}^n_m(A,B)$ by inductive assumption, and by Definition 6 it follows that $(X_A,X_B,x_B) \in \mathbf{MC}^n_{m+1}(A,B)$. □

Theorem 6. Let $A = \langle \Sigma, Q_A, \rightarrow_A, Q^\circ_A \rangle$ and $B = \langle \Sigma, Q_B, \rightarrow_B, Q^\circ_B \rangle$ be automata, and let $n \in \mathbb{N}_0$. Then

$$\mathbf{LC}^{n+1}(A,B) = \{\, (X_A,X_B) \in Q^{\det}_A \times Q^{\det}_B \mid (X_A,X_B,x_B) \notin \mathbf{MC}^n(A,B) \text{ for some } x_B \in X_B \,\} \,. \tag{24}$$

Proof. Let $(X_A,X_B) \in LC^{n+1}(A,B)$. Then by Definition 5, there exists $x_B \in X_B$ such that for all $t \in \Sigma^*$ such that $x_B \stackrel{t\omega}{\Rightarrow}$, there exists $r \sqsubseteq t\omega$ such that $\delta^{\det}_{A,B}(X_A,X_B,r) \in \mathbf{LC}^i(A,B)$ for some $i \leq n$. Equivalently, this means that there does not exist a trace $t \in \Sigma^*$ such that $x_B \stackrel{t\omega}{\Rightarrow}$ and for all prefixes $r \sqsubseteq t\omega$ it holds that $\delta^{\det}_{A,B}(X_A,X_B,r) \notin \mathbf{LC}^n(A,B)$. Then $(X_A,X_B,x_B) \notin \mathbf{MC}^n(A,B)$ because otherwise such a trace would exist by Lemma 5.

Conversely, let $x_B \in X_B$ such that $(X_A,X_B,x_B) \notin \mathbf{MC}^n(A,B)$. To check the condition in Definition 5 (9), consider $t \in \Sigma^*$ such that $x_B \stackrel{t\omega}{\Rightarrow}$. Then clearly $\delta^{\det}_B(X_B,t\omega) = \omega$. By Definition 4, it holds that either $\delta^{\det}_A(X_A,t\omega) = \omega$ or $\delta^{\det}_A(X_A,t\omega) = \emptyset$. If $\delta^{\det}_A(X_A,t\omega) = \omega$, then $\delta^{\det}_{A,B}(X_A,X_B,t\omega) = (\omega,\omega) \in \mathbf{LC}^0(A,B)$. Otherwise $\delta^{\det}_A(X_A,t\omega) = \emptyset$ and thus $\delta^{\det}_{A,B}(X_A,X_B,t\omega) = (\emptyset,\omega)$, and by Lemma 5 there must exist $r \sqsubseteq t\omega$ such that $\delta^{\det}_{A,B}(X_A,X_B,r) \in \mathbf{LC}^n(A,B)$ as otherwise $(X_A,X_B,x_B) \in \mathbf{MC}^n(A,B)$. In both cases, $\delta^{\det}_{A,B}(X_A,X_B,r) \in \mathbf{LC}^i(A,B)$ for some $r \sqsubseteq t\omega$ and $i \leq n$. Since $t \in \Sigma^*$ with $x_B \stackrel{t\omega}{\Rightarrow}$ was chosen arbitrarily, it follows from Definition 5 (9) that $(X_A,X_B) \in \mathbf{LC}^{n+1}(A,B)$. □

Example 11. Figure 5 shows a graph representing the more conflicting triples to check whether $A_2 \lesssim_{\text{conf}} B_2$ in Figure 3. The arrows in the graph represent the deterministic transition function in combination with the transition relation of B_2. An arrow $(X_A, X_B, x_B) \xrightarrow{\sigma} (Y_A, Y_B, y_B)$ indicates that $\delta^{\det}_{A_2,B_2}(X_A, X_B, \sigma) = (Y_A, Y_B)$ and $x_B \overset{\sigma}{\Rightarrow} y_B$.

In the first iteration to compute $\mathbf{MC}^0(A_2, B_2)$, first the triple $(\emptyset, \omega, b_\omega)$ is added to $\mathbf{MC}^0_0(A_2, B_2)$. Next, the triples $(\{a_0\}, \{b_0, b_1\}, b_0)$ and $(\{a_1\}, \{b_0, b_1\}, b_0)$ are added to $\mathbf{MC}^0_1(A_2, B_2)$ as they can immediately reach $(\emptyset, \omega, b_\omega)$. Finally, $(\{a_0\}, \{b_0, b_1\}, b_1)$ is also added to $\mathbf{MC}^0_2(A_2, B_2)$ as it reaches $(\{a_1\}, \{b_0, b_1\}, b_0) \in \mathbf{MC}^0_1(A_2, B_2)$. No further triples are found to be in $\mathbf{MC}^0_3(A_2, B_2)$. Therefore, $(\{a_1\}, \{b_0, b_1\}, b_1) \notin \mathbf{MC}^0(A_2, B_2)$, so it follows from Theorem 6 that $(\{a_1\}, \{b_0, b_1\}) \in \mathbf{LC}^1(A_2, B_2)$, and likewise $(\{a_1, a_2\}, \{b_0, b_1\})$, $(\{a_2\}, \{b_0, b_1\}) \in \mathbf{LC}^1(A_2, B_2)$.

In the next iteration to compute $\mathbf{MC}^1(A_2, B_2)$, note that $(\{a_1\}, \{b_0, b_1\}, b_0) \notin \mathbf{MC}^1_1(A_2, B_2)$ because $(\{a_1\}, \{b_0, b_1\}) \in \mathbf{LC}^1(A_2, B_2)$. Still, $(\{a_0\}, \{b_0, b_1\}, b_0) \in \mathbf{MC}^1_1(A_2, B_2)$ because of the transition to $(\emptyset, \omega, b_\omega) \in \mathbf{MC}^1_0(A_2, B_2)$, but $(\{a_0\}, \{b_0, b_1\}, b_1) \notin \mathbf{MC}^1_2(A_2, B_2)$ because now $(\{a_1\}, \{b_0, b_1\}, b_0) \notin \mathbf{MC}^1_1(A_2, B_2)$. Accordingly, the pair $(\{a_0\}, \{b_0, b_1\})$ is added to $\mathbf{LC}^2(A_2, B_2)$.

In a final iteration to compute $\mathbf{MC}^2(A_2, B_2)$, only one more conflicting triple is found, $(\emptyset, \omega, b_\omega) \in \mathbf{MC}^2_0(A_2, B_2)$. No further pairs are added in $\mathbf{LC}^3(A_2, B_2)$. At this point, the iteration terminates, having found exactly the four less conflicting pairs given in Example 10, (19) and (20).

To determine whether an automaton A is less conflicting than automaton B, it is first needed to determine the set of certain conflicts of B, and then to find all the state-set pairs for A and B that are reachable from a pair like $(\{x_A\}, X_B)$ associated with some trace that is not a certain conflict of B. The more conflicting triples can be constructed as they are discovered during the backwards search from the terminal states.

The complexity of each iteration of the more conflicting triples computation is determined by the number of arrows in the graph, which is bounded by $|\Sigma| \cdot |Q_B|^2 \cdot 2^{|Q_A|} \cdot 2^{|Q_B|}$, because the powerset transitions are deterministic, which is not the case for the transitions of B. Each iteration except the last adds at least one less conflicting pair, so the number of iterations is bounded by $2^{|Q_A|} \cdot 2^{|Q_B|}$. The complexity of this loop dominates all other tasks of the computation. Therefore, the worst-case time complexity to determine whether $A \lesssim_{\mathrm{conf}} B$ using less conflicting pairs is

$$O(|\Sigma| \cdot |Q_B|^2 \cdot 4^{|Q_A|} \cdot 4^{|Q_B|}) \;=\; O(|\Sigma| \cdot |Q_B|^2 \cdot 2^{2|Q_A|+2|Q_B|}) \,. \tag{25}$$

This shows that the conflict preorder can be tested in linear exponential time, as it is the case for the fair testing preorder. Yet, the complexity (25) is better than the time complexity of the decision procedure for fair testing, which is $O(|Q_A| \cdot |Q_B| \cdot 2^{3|Q_A|+5|Q_B|})$ [15].

5 Conclusions

Less conflicting pairs provide a concrete state-based means to characterise the extent by which one process is or is not less conflicting than another. The characterisation generalises and includes previous results about certain conflicts, and it gives rise to a direct way to test the conflict preorder and the related fair testing preorder by inspecting sets of reachable states. Based on the characterisation, an effective algorithm is presented to test whether a finite-state automaton is less conflicting than another. The algorithm, while still linear exponential, has better time complexity than the previously known decision procedure for fair testing.

In the future, the authors would like to apply the theoretic results of this paper to compute abstractions and improve the performance of compositional model checking algorithms. The more thorough understanding of the conflict preorder will make it possible to better simplify processes with respect to conflict equivalence and other related liveness properties.

References

[1] Christel Baier & Joost-Pieter Katoen (2008): *Principles of Model Checking*. MIT Press.

[2] Ed Brinksma, Arend Rensink & Walter Vogler (1995): *Fair Testing*. In Insup Lee & Scott A. Smolka, editors: *Proc. 6th Int. Conf. Concurrency Theory, CONCUR '95*, *LNCS* 962, Springer, Philadelphia, PA, USA, pp. 313–327.

[3] C. G. Cassandras & S. Lafortune (1999): *Introduction to Discrete Event Systems*. Kluwer.

[4] R. De Nicola & M. C. B. Hennessy (1984): *Testing Equivalences for Processes*. *Theoretical Comput. Sci.* 34(1–2), pp. 83–133, doi:10.1016/0304-3975(84)90113-0.

[5] Hugo Flordal & Robi Malik (2009): *Compositional Verification in Supervisory Control*. *SIAM J. Control and Optimization* 48(3), pp. 1914–1938, doi:10.1137/070695526.

[6] R. J. van Glabbeek (2001): *The Linear Time — Branching Time Spectrum I: The Semantics of Concrete, Sequential Processes*. In J. A. Bergstra, A. Ponse & S. A. Smolka, editors: *Handbook of Process Algebra*, Elsevier, pp. 3–99.

[7] C. A. R. Hoare (1985): *Communicating Sequential Processes*. Prentice-Hall.

[8] John E. Hopcroft, Rajeev Motwani & Jeffrey D. Ullman (2001): *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

[9] Robi Malik (2004): *On the Set of Certain Conflicts of a Given Language*. In: *Proc. 7th Int. Workshop on Discrete Event Systems, WODES '04*, Reims, France, pp. 277–282.

[10] Robi Malik (2010): *The Language of Certain Conflicts of a Nondeterministic Process*. Working Paper 05/2010, Dept. of Computer Science, University of Waikato, Hamilton, New Zealand.

[11] Robi Malik, David Streader & Steve Reeves (2006): *Conflicts and Fair Testing*. *Int. J. Found. Comput. Sci.* 17(4), pp. 797–813.

[12] Robin Milner (1989): *Communication and concurrency*. Series in Computer Science, Prentice-Hall.

[13] V. Natarajan & Rance Cleaveland (1995): *Divergence and Fair Testing*. In: *Proc. 22nd Int. Colloquium on Automata, Languages, and Programming, ICALP '95*, pp. 648–659.

[14] Peter J. G. Ramadge & W. Murray Wonham (1989): *The Control of Discrete Event Systems*. *Proc. IEEE* 77(1), pp. 81–98.

[15] Arend Rensink & Walter Vogler (2007): *Fair testing*. *Information and Computation* 205(2), pp. 125–198, doi:10.1016/j.ic.2006.06.002.

[16] Rong Su, Jan H. van Schuppen, Jacobus E. Rooda & Albert T. Hofkamp (2010): *Nonconflict check by using sequential automaton abstractions based on weak observation equivalence*. *Automatica* 46(6), pp. 968–978, doi:10.1016/j.automatica.2010.02.025.

[17] Simon Ware & Robi Malik (2010): *Compositional Nonblocking Verification Using Annotated Automata*. In: *Proc. 10th Int. Workshop on Discrete Event Systems, WODES '10*, Berlin, Germany, pp. 374–379.

[18] K. C. Wong, J. G. Thistle, R. P. Malhame & H.-H. Hoang (2000): *Supervisory Control of Distributed Systems: Conflict Resolution*. *Discrete Event Dyn. Syst.* 10, pp. 131–186.