CONFLICTS AND PROJECTIONS

Robi Malik [*], **Hugo Flordal** [**], **Patrícia N. Pena** [***]

[*] *The University of Waikato, Hamilton, New Zealand*
[**] *Chalmers University of Technology, Göteborg, Sweden*
[***] *Federal University of Santa Catarina, Florianópolis, Brazil*

Abstract: This paper studies abstraction methods suitable to verify very large models of discrete-event systems to be nonconflicting. It compares the *observer property* to methods known from process algebra, namely to *conflict equivalence* and *observation equivalence*. The observer property is shown to be the property that corresponds to conflict equivalence in the case where natural projection is used for abstraction. In this case, the observer property turns out to be the least restrictive condition that can be imposed on natural projection to enable compositional reasoning about conflicts. The observer property is also shown to be closely related to observation equivalence. Several examples and propositions are presented to relate different aspects of these methods of abstraction.

Keywords: Discrete-event systems, large-scale systems, formal verification.

1. INTRODUCTION

Since discrete-event systems are used to model complex safety-critical systems, tools to detect *blocking* and *conflicts* in very large discrete-event models are needed. Yet, the automatic detection of conflicts remains difficult because of the *state-space explosion problem*. *Compositional* or *modular* verification offers a promising solution to this problem: significant advances have been made using *abstraction* to simplify intermediate results before composing too many components.

Natural projection is a simple and common way of calculating abstractions of finite-state automata. However, it is not guaranteed to preserve the potential of conflict with other automata. To overcome this problem, additional constraints on the projection can be imposed. The *observer property* (Wong and Wonham, 1996) ensures that natural projection preserves conflicts and, together with constraints on the subset of events kept by the projection, has recently been applied for compositional nonblocking verification (Pena *et al.*, 2006*b*). The observer property is also used, among other conditions, for modular synthesis (Feng and Wonham, 2006; Hill and Tilbury, 2006).

Independently of this, an approach based on *conflict equivalence* has been investigated (Malik *et al.*, 2006). Here, the complete branching structures of nondeterministic automata is analysed using process-algebraic testing theory. An algorithm for nonblocking verification based on this framework is proposed in (Flordal and Malik, 2006).

Both methods have been used to verify large modular systems to be nonconflicting. This paper establishes a relationship between them. It shows that the observer property is related to conflict equivalence, and that conflict equivalence has a greater potential of abstraction if one does not rely on projection. To put the results in a wider context, the observer property is also compared to the well established *observation equivalence* (Milner, 1989), to which it is closely related.

Section 2 provides formal notation and definitions for the observer property, conflict equivalence,

and observation equivalence. Section 3 discusses the close relationship between projections based on the observer property on the one hand, and conflict and observation equivalence on the other hand. Section 4 points out some fundamental differences between abstractions obtained by projection and process equivalence, followed by concluding remarks in section 5.

2. PRELIMINARIES

2.1 Languages and Automata

Event sequences and languages are a simple means to describe discrete system behaviours. Their basic building blocks are *events*, taken from a finite *alphabet* Σ. Then, Σ^* denotes the set of all finite *strings* of the form $\sigma_1\sigma_2\cdots\sigma_k$ of events from Σ, including the *empty string* ε. The *catenation* of two strings $s, t \in \Sigma^*$ is written as st.

The special event $\omega \in \Sigma$ is used to indicate successful *termination*. Another special event $\tau \notin \Sigma$ denotes *silent* transitions. In contrast to ω, the event τ is assumed *not* to be included in Σ; if it is to be included, $\Sigma_\tau = \Sigma \cup \{\tau\}$ is used instead.

A *language* over Σ is any subset $L \subseteq \Sigma^*$. A language L is called *prefix-closed* if for all $s, t \in \Sigma^*$, it holds that $st \in L$ implies $s \in L$.

This paper considers models expressed as *nondeterministic automata* $G = \langle Q, \Sigma, \rightarrow, q^i\rangle$ where Q is the set of *states*, Σ is the alphabet, $\rightarrow \subseteq Q \times \Sigma_\tau \times Q$ is the *transition relation*, and $q^i \in Q$ is the *initial state*. G is called *deterministic* if it does not contain any τ transitions, and the transition relation can be written as a function, i.e., if $p \xrightarrow{\sigma} q_1$ and $p \xrightarrow{\sigma} q_2$ always implies $q_1 = q_2$.

The transition relation is extended in the natural way to accept strings $s \in \Sigma_\tau^*$. Also, $Q \xrightarrow{s} Q'$ denotes the existence of states $q \in Q$ and $q' \in Q'$ such that $q \xrightarrow{s} q'$. The expression $q \xrightarrow{s}$ denotes the existence of $q' \in Q$ such that $q \xrightarrow{s} q'$.

The transition relation of an automaton must satisfy the additional requirement that, whenever $p \xrightarrow{\omega} q$, there does not exist any outgoing transition from q. That is, the termination event ω marks states as terminal states. The traditional set of *marked* states of G can be defined as

$$Q^\omega \;=\; \{\, q \in Q \mid q \xrightarrow{\omega} \,\}\;. \tag{1}$$

For the sake of graphical simplicity, states in Q^ω are shaded in the figures of this paper instead of explicitly showing ω transitions.

Another transition relation $\Rightarrow$ is introduced that includes possible interleavings with τ events. For example, $q \overset{\sigma_1\sigma_2}{\Longrightarrow} q'$ denotes the existence of a string $s \in \tau^*\sigma_1\tau^*\sigma_2\tau^*$ such that $q \xrightarrow{s} q'$. Furthermore, $G \overset{s}{\Rightarrow} q$ is a shorthand for $q^i \overset{s}{\Rightarrow} q$.

Given this notation, the *language* or *visible behaviour* of a nondeterministic automaton is defined as $\mathcal{L}(G) = \{\, s \in \Sigma^* \mid G \overset{s}{\Rightarrow} \,\}$. This prefix-closed language does not include any τ events, but it may include strings ending with ω.

It is possible to construct a minimal automaton from a given language. To this end, two strings $s_1, s_2 \in \Sigma^*$ are said to be *Nerode equivalent* with respect to $L \subseteq \Sigma^*$, denoted $s_1 \equiv_L s_2$, if they have the same continuations in L. Formally, $s_1 \equiv_L s_2$ if, for every $t \in \Sigma^*$, it holds that $s_1 t \in L$ if and only if $s_2 t \in L$. Nerode equivalence is a *right congruence* with respect to string catenation, i.e., $s_1 \equiv_L s_2$ implies $s_1 t \equiv_L s_2 t$ for any $t \in \Sigma^*$.

Clearly, $\equiv_L$ is an equivalence relation on the strings in Σ^*. Furthermore, L can be partitioned into the set of *equivalence classes* imposed by $\equiv_L$,

$$[s]_L \;=\; \{\, s' \in \Sigma^* \mid s' \equiv_L s \,\}\;. \tag{2}$$

Given a prefix-closed language $L \subseteq \Sigma^*$, a minimal deterministic automaton G_L recognising L can be constructed as $G_L = \langle L/\equiv_L, \Sigma, \rightarrow, [\varepsilon]_L\rangle$ where

$$L/\equiv_L \;=\; \{\, [s]_L \mid s \in L \,\}\;. \tag{3}$$

and $[s] \xrightarrow{\sigma} [s\sigma]$ whenever $s\sigma \in L$.

When two automata are running in parallel, lockstep synchronisation in the style of (Hoare, 1985) is used. The *synchronous composition* of $G_1 = \langle Q_1, \Sigma, \rightarrow_1, q_1^i\rangle$ and $G_2 = \langle Q_2, \Sigma, \rightarrow_2, q_2^i\rangle$ is

$$G_1 \parallel G_2 \;=\; \langle Q_1 \times Q_2, \Sigma, \rightarrow, (q_1^i, q_2^i)\rangle \tag{4}$$

where

$$\begin{aligned}
&(p,q) \xrightarrow{\sigma} (p',q') \text{ if } \sigma \neq \tau, p \xrightarrow{\sigma}_1 p', \text{and } q \xrightarrow{\sigma}_2 q' \;;\\
&(p,q) \xrightarrow{\tau} (p',q) \text{ if } p \xrightarrow{\tau}_1 p' \;;\\
&(p,q) \xrightarrow{\tau} (p,q') \text{ if } q \xrightarrow{\tau}_2 q' \;.
\end{aligned}$$

In synchronous composition, shared events (including ω) must be executed by all automata together, while silent (i.e., τ) transitions are executed independently. The marking defined by ω works just like "ordinary" marking: a state in a composition is marked if all composed automata are marked in their respective states.

The notion of conflict is extended to nondeterministic automata as follows. A state q is *nonblocking* if $q \overset{s\omega}{\Rightarrow}$ for some $s \in \Sigma^*$. An automaton G is nonblocking if every *reachable* state, i.e., every state q such that $G \overset{s}{\Rightarrow} q$ for some $s \in (\Sigma\backslash\{\omega\})^*$, is nonblocking. Otherwise G is *blocking*. Two automata G_1 and G_2 are *nonconflicting* if $G_1 \parallel G_2$ is nonblocking.

2.2 Projection and Hiding

In large systems composed of several automata, there typically exist some events that are used exclusively by only one automaton or can be abstracted away for other reasons.

To this end, the alphabet Σ is partitioned into the set Υ of events to be abstracted away and the set Ω of events to be retained. Typically, Υ consists of the events used exclusively by the automaton considered, but other choices are possible (Pena *et al.*, 2006*a*; Pena *et al.*, 2006*b*). *Projection*

$$P_\Omega \colon \Sigma^* \to \Omega^* \tag{5}$$

is the operation that removes all events not in Ω (i.e., all events in Υ) from a string.

When considering nondeterministic automata, it is more appropriate to use τ transitions instead of language projection. This makes it possible to use different ways of selecting events or individual transitions for abstraction. In the following, it is assumed that all transitions to be abstracted away are labelled τ. In terms of projection, it then only remains to remove the event τ. This is done by *natural projection*

$$\theta \colon \Sigma_\tau^* \to \Sigma^* \ , \tag{6}$$

which deletes all τ events from a string. This operation is also applied to automata G: $\theta(G)$ denotes the minimal deterministic recogniser of the language $\mathcal{L}(G) = \{\, s \in \Sigma^* \mid G \stackrel{s}{\Rightarrow} \,\}$.

The *observer property* is known as a key property for when language projection can be used as conflict-preserving abstraction. The following extends the original definition (Wong and Wonham, 1996) to the nondeterministic case.

Definition 1. An automaton G satisfies the *observer property* if, for all strings $s, t \in \Sigma^*$ and all states q such that $G \stackrel{s}{\Rightarrow} q$ and $G \stackrel{st\omega}{\Longrightarrow}$ it holds that $q \stackrel{t\omega}{\Rightarrow}$. In this case, the natural projection $\theta \colon \Sigma_\tau^* \to \Sigma^*$ is called an *observer projection* for G.

The observer property requires that the possibility of continuation to a terminal state is independent of the particular state. If an automaton can terminate after executing a string s, it has to be able to terminate in the same way from all states reachable via s. This ensures that all states reached by a given string have the same conflicting properties in all possible contexts.

2.3 Process Equivalences

Effective abstractions are the key to compositional verification of large systems. The idea is to replace a component by a simpler *equivalent* one, such that crucial properties of the whole system are preserved. One of the finest known equivalences is *observation equivalence* (Milner, 1989).

Definition 2. Let $G_1 = \langle Q_1, \Sigma, \to_1, q_1^i \rangle$ and $G_2 = \langle Q_2, \Sigma, \to_2, q_2^i \rangle$ be two automata. A relation $\approx \subseteq Q_1 \times Q_2$ is a *weak bisimulation* between G_1 and G_2 if, for all states $q_1 \in Q_1$ and $q_2 \in Q_2$ such that $q_1 \approx q_2$ and for all $s \in \Sigma^*$,

- if $q_1 \stackrel{s}{\Rightarrow} q_1'$ then there exists $q_2' \in Q_2$ such that $q_2 \stackrel{s}{\Rightarrow} q_2'$ and $q_1' \approx q_2'$;
- if $q_2 \stackrel{s}{\Rightarrow} q_2'$ then there exists $q_1' \in Q_1$ such that $q_1 \stackrel{s}{\Rightarrow} q_1'$ and $q_1' \approx q_2'$.

G_1 and G_2 are *observation equivalent*, denoted $G_1 \approx G_2$, if there exists a weak bisimulation $\approx$ between G_1 and G_2 such that $q_1^i \approx q_2^i$.

Observation equivalent automata have the same branching structure of their nondeterministic behaviour. Observation equivalence preserves all behavioural properties, and therefore has been suggested for modular synthesis in some contexts of discrete-event systems (Su and Thistle, 2006).

Observation equivalence is unnecessarily fine for compositional reasoning about conflicts—it sometimes distinguishes automata that could be considered as equivalent. In (Malik *et al.*, 2006), the coarsest possible equivalence is identified:

Definition 3. Automata G_1 and G_2 are said to be *conflict equivalent*, denoted $G_1 \simeq_{\text{conf}} G_2$, if, for any automaton T, it holds that $G_1 \parallel T$ is nonblocking if and only if $G_2 \parallel T$ is nonblocking.

This definition extends the process-algebraic *fair testing* equivalence (Brinksma *et al.*, 1995) to cover blocking processes. It is inspired by testing theory (De Nicola and Hennessy, 1984), where processes are considered as equivalent if their responses to all *tests* are the same. A *test* here is an arbitrary automaton; it represents the unknown remainder of the system to be verified. The test outcome is the result whether the automaton is nonconflicting together with the test or not.

3. ABSTRACTION BY PROJECTION

Natural projection can be used to simplify automata. If the observer property holds, then this abstraction is suitable for compositional nonblocking verification (Pena *et al.*, 2006*b*). This section explores the relationship between abstractions obtained in this way on the one hand, and conflict and observation equivalence on the other hand.

3.1 Relation to Conflict Equivalence

By the results of (Malik *et al.*, 2006), conflict equivalence is coarser than any abstraction suitable for compositional nonblocking verification, including observer projection. The following proposition gives a direct proof of this fact.

Proposition 4. Let θ be an observer projection for G. Then $G \simeq_{\text{conf}} \theta(G)$.

Proof. Let T be an arbitrary automaton.

First, assume that $G \parallel T$ is nonblocking, and let $\theta(G)\|T \stackrel{s}{\Rightarrow} (q_\theta, q_T)$. Then $\theta(G) \stackrel{s}{\Rightarrow} q_\theta$ and therefore $G \stackrel{s}{\Rightarrow} q$ for some state $q \in Q$. Thus, $G\|T \stackrel{s}{\Rightarrow} (q, q_T)$. Since $G \parallel T$ is nonblocking, $(q, q_T) \stackrel{t\omega}{\Rightarrow}$ for some $t \in \Sigma^*$. Hence, $st\omega \in \mathcal{L}(G) = \mathcal{L}(\theta(G))$, and since $\theta(G)$ is deterministic, this implies $q_\theta \stackrel{t\omega}{\Rightarrow}$. It follows that $(q_\theta, q_T) \stackrel{t\omega}{\Rightarrow}$, i.e., $\theta(G) \parallel T$ is nonblocking.

Second, assume that $\theta(G) \parallel T$ is nonblocking, and let $G \parallel T \stackrel{s}{\Rightarrow} (q, q_T)$. Then $G \stackrel{s}{\Rightarrow} q$, and therefore $\theta(G) \stackrel{s}{\Rightarrow} [s]$, i.e., $\theta(G)\|T \stackrel{s}{\Rightarrow} ([s], q_T)$. Since $\theta(G)\|T$ is nonblocking, $([s], q_T) \stackrel{t\omega}{\Rightarrow}$ for some $t \in \Sigma^*$. Hence, $st\omega \in \mathcal{L}(\theta(G)) = \mathcal{L}(G)$, i.e., $G \stackrel{st\omega}{\Longrightarrow}$. Then $q \stackrel{t\omega}{\Rightarrow}$ because of the observer property, and therefore $(q, q_T) \stackrel{t\omega}{\Rightarrow}$, i.e., $G \parallel T$ is nonblocking. □

Thus, if the observer property holds, then the result of projection is conflict equivalent to the original automaton. The converse of proposition 4 is also true. If natural projection yields a conflict equivalent automaton, then the observer property holds.

Proposition 5. Let $G \simeq_{\text{conf}} \theta(G)$. Then θ is an observer projection for G.

Proof. Let $G = \langle Q, \Sigma, \rightarrow, q^i \rangle$ be such that $G \simeq_{\text{conf}} \theta(G)$. To show the observer property, let $s, t \in \Sigma^*$ and $q \in Q$ such that $G \stackrel{s}{\Rightarrow} q$ and $G \stackrel{st\omega}{\Longrightarrow}$, and assume $t = \alpha_1 \dots \alpha_n$. Recall that $\theta(G)$ is a deterministic automaton that can be written as $\theta(G) = \langle Q_\theta, \Sigma, \rightarrow_\theta, [\varepsilon] \rangle$ where $Q_\theta = \mathcal{L}(G)/\equiv_{\mathcal{L}(G)}$. Construct a test T from $\theta(G)$ by first removing all blocking states and second adding the following states and transitions:

$$[s] \stackrel{\tau}{\rightarrow} p_0 \stackrel{\alpha_1}{\rightarrow} p_1 \stackrel{\alpha_2}{\rightarrow} \dots \stackrel{\alpha_n}{\rightarrow} p_n \stackrel{\omega}{\rightarrow} p_\omega \ . \qquad (7)$$

Note that $[s]$ is not a blocking state in $\theta(G)$ because $st\omega \in \mathcal{L}(G) = \mathcal{L}(\theta(G))$.

$\theta(G) \parallel T$ is nonblocking. To see this, let $u' \in (\Sigma_\tau \backslash \{\omega\})^*$ such that $\theta(G) \parallel T \stackrel{u'}{\rightarrow} (q, p)$. Then $q = [u]$ where $u = \theta(u')$. If u' does not include any τ event, then by construction $q = [u] = p$ is not a blocking state in $\theta(G)$. Thus, $q \stackrel{v\omega}{\rightarrow}$ for some $v \in \Sigma^*$, and again by construction $(q, p) \stackrel{v\omega}{\rightarrow}$. If, on the other hand, u' includes τ, then $u' = s\tau\alpha_1 \dots \alpha_k$ for some $k \leq n$, and $p = p_k \stackrel{\alpha_{k+1} \dots \alpha_n \omega}{\longrightarrow}$. Furthermore, $q = [u] \stackrel{\alpha_{k+1} \dots \alpha_n \omega}{\longrightarrow}$ because $s\alpha_1 \dots \alpha_n\omega = st\omega \in \mathcal{L}(G) = \mathcal{L}(\theta(G))$ and $\theta(G)$ is deterministic.

Hence, $\theta(G) \parallel T$ is nonblocking. Since $G \simeq_{\text{conf}} \theta(G)$, this implies that $G \parallel T$ is nonblocking. By construction of T, it holds that $G \parallel T \stackrel{s}{\Rightarrow} (q, p_0)$. Then the only way how $G \parallel T$ can be nonblocking is when $(q, p_0) \stackrel{\alpha_1 \dots \alpha_n \omega}{\Longrightarrow}$. This means $q \stackrel{t\omega}{\Rightarrow}$, i.e., the observer property holds. □

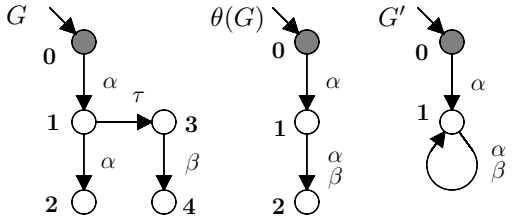

Fig. 1. Equivalence issues with blocking automata.

These results show that conflict equivalence and the observer property are equivalent when using natural projection. The observer property is the weakest possible restriction that can be imposed on natural projection to ensure that conflicts are preserved in all contexts. If projection is to be used for compositional reasoning about conflicts, then the observer property is the best possible choice.

3.2 Relation to Observation Equivalence

Given a nondeterministic automaton G, natural projection produces a deterministic automaton $\theta(G)$, the state set of which is the set of Nerode equivalence classes of $\mathcal{L}(G)$. This automaton may be very different in structure or size from G (Wong, 1998). In particular,

$$G \approx \theta(G) \qquad (8)$$

may or may not hold. This section explores the relationship between (8) and the observer property. Firstly, if an automaton is observation equivalent to its natural projection, then the observer property holds.

Proposition 6. Let $G \approx \theta(G)$. Then θ is an observer projection for G.

Proof. By the results of (Malik *et al.*, 2006) it is known that $G \approx \theta(G)$ implies $G \simeq_{\text{conf}} \theta(G)$. Then the claim follows from proposition 5. □

The converse of proposition 6 does not hold. Observer projection can produce a better abstraction than observation equivalence.

Example 7. The natural projection is an observer projection for automaton G in figure 1. To check this, only state **0** needs to be considered, because no terminal state can be reached from the other states. Therefore, $\theta(G)$ can be used as conflict-preserving abstraction of G. However, $\theta(G)$ is not observation equivalent to G, because $\theta(G)$ does not contain any state equivalent to state **3** of G, where only β is enabled.

The observer property only imposes restrictions on strings that can be extended to a terminal state, so it can equate blocking states with different futures. This is the only way how observer projection differs from observation equivalence.

Proposition 8. Let G be nonblocking, and let θ be an observer projection for G. Then $G \approx \theta(G)$.

Proof. Let $G = \langle Q, \Sigma, \rightarrow, q^i \rangle$ be nonblocking. Recall that the state set Q_θ of $\theta(G)$ can be written as $Q_\theta = \mathcal{L}(G)/\equiv_{\mathcal{L}(G)}$. Define the relation $\sim \subseteq Q \times Q_\theta$ such that $q \sim [s]$ if and only if $q^i \stackrel{\tilde{s}}{\Rightarrow} q$ for some $\tilde{s} \equiv_{\mathcal{L}(G)} s$. Clearly, $\sim$ is a well-defined relation independent of the particular representatives chosen for equivalence classes $[s]$. To see that $\sim$ establishes a weak bisimulation between the states of G and $\theta(G)$, let $q \in Q$ and $[s] \in Q_\theta$ such that $q \sim [s]$. Note that this means $q^i \stackrel{\tilde{s}}{\Rightarrow} q$ for some $\tilde{s} \equiv_{\mathcal{L}(G)} s$.

First, let $q \stackrel{t}{\Rightarrow} p$ for some $t \in \Sigma^*$ and $p \in Q$. Then $q^i \stackrel{\tilde{s}}{\Rightarrow} q \stackrel{t}{\Rightarrow} p$. Since $\tilde{s} \equiv_{\mathcal{L}(G)} s$, it follows that $\tilde{s}t \equiv_{\mathcal{L}(G)} st$. Hence, $p \sim [st]$ by definition of $\sim$.

Second, let $[s] \stackrel{t}{\Rightarrow} [st]$ for some $t \in \Sigma^*$. Then $st \in \mathcal{L}(G)$. Therefore, $q^i \stackrel{s}{\Rightarrow} q' \stackrel{t}{\Rightarrow} p'$ for some states $q', p' \in Q$. Since G is nonblocking, $p' \stackrel{u\omega}{\Rightarrow}$ for some $u \in \Sigma^*$, i.e., $stu\omega \in \mathcal{L}(G)$. Since $\tilde{s} \equiv_{\mathcal{L}(G)} s$, it follows that $\tilde{s}tu\omega \in \mathcal{L}(G)$. Thus, $q^i \stackrel{\tilde{s}}{\Rightarrow} q$ and $q^i \stackrel{\tilde{s}tu\omega}{\Longrightarrow}$. By the observer property, $q \stackrel{tu\omega}{\Longrightarrow}$. This implies $q^i \stackrel{\tilde{s}}{\Rightarrow} q \stackrel{t}{\Rightarrow} p$ for some state $p \in Q$. Since $\tilde{s} \equiv_{\mathcal{L}(G)} s$, it follows that $\tilde{s}t \equiv_{\mathcal{L}(G)} st$, i.e., $p \sim [st]$.

Finally, note that the initial state of $\theta(G)$ is $[\varepsilon]$. The correspondence of initial states follows immediately since $q^i \stackrel{\varepsilon}{\Rightarrow} q^i$ and thus $q^i \sim [\varepsilon]$. □

A nonblocking automaton is observation equivalent to its natural projection if and only if the observer property holds. This very close relationship between the observer property and observation equivalence explains the good computational properties of observer projection. It can be computed in polynomial time, and the result never has more states than the original automaton (Wong, 1998).

4. ON THE EFFECTIVENESS OF NATURAL PROJECTION

The results presented so far discuss the relationships between an automaton and its natural projection. Process equivalence can equate arbitrary processes that are not necessarily obtained by projection, and this gives them more possibilities of abstraction.

Example 9. Automata G_1 and G_2 in figure 2 are conflict equivalent. Every test that is nonconflicting with G_1 or G_2 needs to be able to execute β while α is an "optional" way of terminating. However, G_2 is not the natural projection of G_1 or vice versa. Also, θ is not an observer projection for G_1,

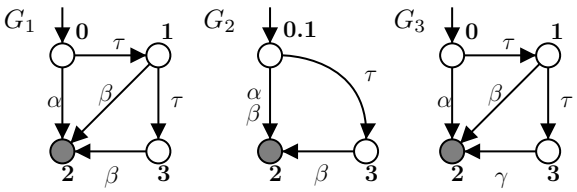

Fig. 2. Conflict equivalence versus projection.

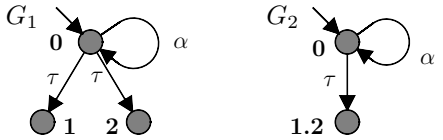

Fig. 3. G_1 and G_2 are observation equivalent, yet the observer property is not satisfied.

because α is enabled in state **0** but not in states **1** and **3**.

Thus, two automata can be conflict equivalent even though the observer property is not satisfied. In this case, this is because conflict equivalence can operate on individual transitions as opposed to projecting all τ transitions together or none.

In fact, if the τ transition between states 0 and 1 in G_1 is relabelled as τ', where τ' is not a silent event, then the observer property holds, and the automaton can be simplified to yield a result like G_2. This interesting observation can enhance the effectiveness of projections, yet not all conflict equivalent automata can be recognised in this way.

Example 10. States **0** and **1** in automaton G_3 of figure 2 are conflict equivalent and can be merged as in example 9. A test only needs to be able to execute γ to be nonconflicting with G_3, while α and β are "optional" ways of terminating. Yet, θ is not an observer projection for G_3, because α is enabled in state **0** but not in state **1**. Likewise, β is enabled in state **1** but not in state **3**. There is no nonempty subset of the τ transitions in G_3 that would make the observer property hold.

This example shows how conflict equivalence, which operates on the branching behaviour of a nondeterministic automaton, can have completely different abstraction qualities from projection. Even observation equivalence can yield better abstractions than projection, for similar reasons.

Example 11. Automata G_1 and G_2 in figure 3 are observation equivalent. Yet, the observer property is not satisfied, because α is enabled in state 0 but not in states **1**, **2**, or **1.2**, so these two automata cannot be equated by observer projection.

In this example, observation equivalence can reduce the number of states of an automaton while observer projection cannot. As shown in example 7, there are also cases where observer projection can perform better than observation equiv-

alence, because it allows for some simplification of the blocking states. Yet, observer projection is restricted to a more limited degree of abstraction for blocking states than conflict equivalence.

Example 12. Automaton G' in figure 1 is conflict equivalent to G and $\theta(G)$: since blocking is inevitable once α has occurred, the precise set of possible continuations after α is immaterial. However, G' has a different language from G and therefore cannot be obtained by projection.

The improved abstraction potential of process equivalence comes at a price. Projections can be computed by straightforward algorithms, in the case of the observer property even in polynomial time (Wong, 1998). While there also are good algorithms for observation equivalence (Milner, 1989), in the case of conflict equivalence, a minimal equivalent automaton is not guaranteed to exist, and the only algorithms available so far are based on incomplete reduction rules and heuristics (Flordal and Malik, 2006).

5. CONCLUSIONS

This paper compares two methods of abstraction for compositional nonblocking verification, namely projection with the observer property and conflict equivalence.

The observer property is shown to be the least restrictive possible condition that can be imposed on natural projection to enable compositional reasoning about conflicts. The observer property is very closely related to observation equivalence, except for blocking automata where observer projection can produce better abstractions.

Process equivalence is not restricted to language projection, and this fact gives conflict equivalence a better potential of abstraction and state reduction. This is partly due to the fact that conflict equivalence can treat the *transitions* of an automaton individually, as opposed to projections, which operate on all transitions with the same *event* in the same way. On the other hand, projections are much easier to compute than abstractions based on conflict equivalence.

It is of great interest to see how the two methods of abstraction can be combined to produce more efficient algorithms for nonblocking verification. In future research, the authors would like to see observer projection with its better computational properties incorporated into their conflict equivalence-based algorithm, and to investigate how observer projection can be enhanced by incorporating some of the ideas from process and conflict equivalence presented in this paper.

REFERENCES

Brinksma, Ed, Arend Rensink and Walter Vogler (1995). Fair testing. In: *Proc. 6th Int. Conf. Concurrency Theory, CONCUR '95* (Insup Lee and Scott A. Smolka, Eds.). Vol. 962 of *LNCS*. Springer. pp. 313–327.

De Nicola, Rocco and Matthew C. B. Hennessy (1984). Testing equivalences for processes. *Theor. Comp. Sci.* **34**(1–2), 83–133.

Feng, Lei and W. Murray Wonham (2006). Computationally efficient supervisor design: Abstraction and modularity. In: *Proc. 8th Int. Workshop on Discrete Event Systems, WODES '06*. pp. 3–8.

Flordal, Hugo and Robi Malik (2006). Modular nonblocking verification using conflict equivalence. In: *Proc. 8th Int. Workshop on Discrete Event Systems, WODES '06*. pp. 100–106.

Hill, Richard C. and Dawn M. Tilbury (2006). Modular supervisory control of discrete-event systems with abstractions and incremental hierarchical construction. In: *Proc. 8th Int. Workshop on Discrete Event Systems, WODES '06*. pp. 399–406.

Hoare, Charles A. R. (1985). *Communicating sequential processes*. Series in Computer Science. Prentice-Hall.

Malik, Robi, David Streader and Steve Reeves (2006). Conflicts and fair testing. *Int. J. Foundations of Comp. Sci.* **17**(4), 797–813.

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

Pena, Patrícia N., José E. R. Cury and Stéphane Lafortune (2006*a*). New results on testing modularity of local supervisors using abstractions. In: *Proc. 11th IEEE Int. Conf. Emerging Technologies and Factory Automation, ETFA '06*. pp. 950–956.

Pena, Patrícia N., José E. R. Cury and Stéphane Lafortune (2006*b*). Testing modularity of local supervisors: An approach based on abstractions. In: *Proc. 8th Int. Workshop on Discrete Event Systems, WODES '06*. pp. 107–112.

Su, Rong and John Thistle (2006). A distributed supervisor synthesis approach based on weak bisimulation. In: *Proc. 8th Int. Workshop on Discrete Event Systems, WODES '06*. pp. 64–69.

Wong, K. (1998). On the complexity of projections of discrete-event systems. In: *Proc. 4th Int. Workshop on Discrete Event Systems, WODES '98*. pp. 201–206.

Wong, Kai C. and W. Murray Wonham (1996). Hierarchical control of discrete-event systems. *Discrete Event Dynamic Systems* **6**(3), 241–273.