Discrete Event Dynamic Systems manuscript No.
(will be inserted by the editor)

Conflict-Preserving Abstraction of Discrete Event Systems Using Annotated Automata

Simon Ware · **Robi Malik**

the date of receipt and acceptance should be inserted later

Abstract This paper proposes to enhance compositional verification of the nonblocking property of discrete event systems by introducing *annotated automata*. Annotations store nondeterministic branching information, which would otherwise be stored in extra states and transitions. This succinct representation makes it easier to simplify automata and enables new efficient means of *abstraction*, reducing the size of automata to be composed and thus the size of the synchronous product state space encountered in verification. The abstractions proposed are of polynomial complexity, and they have been successfully applied for nonblocking verification of the same set of large-scale industrial examples as used in related work.

1 Introduction

With the continuously increasing size and complexity of reactive systems software, the automatic verification of large reactive systems is and remains a challenging problem. This paper focuses on the verification of the *nonblocking* property, which is of great interest in supervisory control of discrete event systems [2, 17]. Nonblocking is the question whether the composed behaviour of a set of automata is under all circumstances capable of reaching a terminal state.

The standard method to check whether a system is nonblocking involves the explicit composition of all the automata involved and the construction of the complete state space. This approach is limited by the well-known *state-space explosion* problem. *Symbolic model checking* has been used successfully to reduce the amount of memory required by representing the state space symbolically rather than enumerating it explicitly [3].

As an alternative, *compositional* verification tries to avoid constructing large state spaces by progressively composing automata and using *abstraction* to simplify intermediate results. This idea has been pursued with notable success in recent years. Automata can be simplified for nonblocking verification using *observer projection* [6, 16] or *weak observation equivalence* [19]. These well-known general-purpose abstractions are more restrictive than necessary for nonblocking verification. A possible alternative is to consider *trajectory*

Department of Computer Science, University of Waikato, Hamilton, New Zealand
E-mail: {siw4,robi}@waikato.ac.nz

nonblocking [11], while *conflict equivalence* is known to be the most general method of abstraction that preserves the nonblocking property in all contexts [12]. Conflict equivalence can be used to implement heuristic simplification rules, making it possible to verify discrete event systems models of industrial complexity [9].

This paper seeks to combine the advantages of bisimulation-based abstractions [19] with the benefits of conflict-preserving simplification [9]. Using *annotations*, certain aspects of the branching structure of nondeterministic automata can be unified. This makes it possible to overcome some limitations of the previous approach based on heuristics, and makes more aspects of conflict-preserving abstraction amenable to global reduction algorithms such as bisimulation.

This paper is an extended version of [20], including more detailed descriptions of annotated automata and full proofs of all results. Section 2 briefly introduces the needed terminology of languages, automata, and conflict equivalence. Then Section 3 presents annotated automata and the rules to construct and simplify them, which are explained using an example. Section 4 contains formal proofs of the correctness of the abstraction rules. Afterwards, Section 5 presents experimental results, and Section 6 adds some concluding remarks.

2 Preliminaries

2.1 Events and Traces

Event sequences and languages are a simple means to describe discrete system 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. For this, $\Sigma_\tau = \Sigma \cup \{\tau\}$, $\Sigma_\omega = \Sigma \cup \{\omega\}$, and $\Sigma_{\tau,\omega} = \Sigma \cup \{\tau, \omega\}$ are used.

Σ^* denotes the set of all finite *traces* of the form $\sigma_1\sigma_2\cdots\sigma_n$ of events from Σ, including the *empty trace* ε. The *concatenation* of two traces $s,t \in \Sigma^*$ is written as st. A subset $L \subseteq \Sigma^*$ is called a *language*. Given two alphabets Σ_1 and $\Sigma_2 \subseteq \Sigma_1$, the *natural projection* $P\colon \Sigma_1^* \to \Sigma_2^*$ is the operation that deletes from traces over Σ_1 all events not in Σ_2.

2.2 Nondeterministic Automata

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

Definition 1 A (nondeterministic) *finite-state automaton* is a 4-tuple $G = \langle \Sigma, Q, \to, Q^\circ \rangle$ where Σ is a finite alphabet of *events*, Q is a finite set of *states*, $\to \subseteq Q \times \Sigma_{\tau,\omega} \times Q$ is the *state transition relation*, and $Q^\circ \subseteq Q$ is the set of *initial states*.

The transition relation is written in infix notation $x \xrightarrow{\sigma} y$, and is extended to traces in $\Sigma_{\tau,\omega}^*$ 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 $z \in Q$. For state sets $X, Y \subseteq Q$, the expression $X \xrightarrow{s} Y$ denotes the existence of $x \in X$ and $y \in Y$ such that $x \xrightarrow{s} y$. Furthermore, $x \to y$ denotes the existence of a trace $s \in \Sigma_\omega^*$ such that $x \xrightarrow{s} y$, and $x \xrightarrow{s}$ denotes the existence of a state $y \in Q$ such that $x \xrightarrow{s} y$. Finally, $G \xrightarrow{s}$ and $G \xrightarrow{s} X$ stand for $Q^\circ \xrightarrow{s}$ and $Q^\circ \xrightarrow{s} X$, respectively.

The transition relation must satisfy the additional requirement that, whenever $x \xrightarrow{\omega} y$, there does not exist any outgoing transition from y. That is, the termination event ω marks states (such as x) as terminal states. The traditional set of *marked* or *terminal* states of G can be defined as $Q^\omega = \{ x \in Q \mid x \xrightarrow{\omega} \}$. For the sake of graphical simplicity, states in Q^ω are shaded in the figures of this paper instead of explicitly showing ω transitions.

To support silent transitions, $x \stackrel{s}{\Rightarrow} y$, with $s \in \Sigma_\omega^*$, denotes the existence of a trace $t \in \Sigma_{\tau,\omega}^*$ such that $x \xrightarrow{t} y$ and $P(t) = s$. That is, $\xrightarrow{s}$ denotes a path with *exactly* the events in s, while $\stackrel{s}{\Rightarrow}$ denotes a path with an arbitrary number of τ shuffled with the events in s. Notations such as $X \stackrel{s}{\Rightarrow} Y$ for state sets, $x \Rightarrow y$, $G \stackrel{s}{\Rightarrow}$, etc., are defined analogously to $\rightarrow$. In addition, for a state $x \in Q$, the set of *active* or *eligible events* is $\mathrm{Elig}_G(x) = \{ \sigma \in \Sigma_\omega \mid x \stackrel{\sigma}{\Rightarrow} \}$.

When two automata are running in parallel, lock-step synchronisation in the style of [10] is used.

Definition 2 Let $G_1 = \langle \Sigma_1, Q_1, \rightarrow_1, Q_1^\circ \rangle$ and $G_2 = \langle \Sigma_2, Q_2, \rightarrow_2, Q_2^\circ \rangle$ be two automata. The *synchronous composition* of G_1 and G_2 is

$$G_1 \parallel G_2 \;=\; \langle \Sigma_1 \cup \Sigma_2, Q_1 \times Q_2, \rightarrow, Q_1^\circ \times Q_2^\circ \rangle \tag{1}$$

where

$$\begin{aligned}
(x,y) &\xrightarrow{\sigma} (x',y') \text{ if } \sigma \in (\Sigma_1 \cap \Sigma_2) \cup \{\omega\},\ x \xrightarrow{\sigma}_1 x',\ y \xrightarrow{\sigma}_2 y' ; \\
(x,y) &\xrightarrow{\sigma} (x',y) \text{ if } \sigma \in (\Sigma_1 \setminus \Sigma_2) \cup \{\tau\},\ x \xrightarrow{\sigma}_1 x' ; \\
(x,y) &\xrightarrow{\sigma} (x,y') \text{ if } \sigma \in (\Sigma_2 \setminus \Sigma_1) \cup \{\tau\},\ y \xrightarrow{\sigma}_2 y' .
\end{aligned}$$

In synchronous composition, shared events (including ω) must be executed by all automata synchronously, while other events (including τ) are executed independently. In the notation of this paper,

$$G_1 \parallel G_2 \stackrel{s}{\Rightarrow} (x_1, x_2) \qquad \text{if and only if} \qquad G_i \stackrel{P_i(s)}{\Longrightarrow} x_i \quad \text{for } i = 1,2 , \tag{2}$$

where $P_i \colon \Sigma \to \Sigma_i$ denotes the natural projection.

2.3 Conflict Equivalence

The key liveness property in supervisory control theory is the *nonblocking* property. An automaton is nonblocking if, from every reachable state, a terminal state can be reached; otherwise it is called *blocking*. When more than one automaton is involved, it also is common to use the terms *nonconflicting* and *conflicting*, respectively.

Definition 3 An automaton $G = \langle \Sigma, Q, \rightarrow, Q^\circ \rangle$ is *nonblocking* if, for every state $x \in Q$ and every trace $s \in \Sigma^*$ such that $Q^\circ \stackrel{s}{\Rightarrow} x$, there exists a trace $t \in \Sigma^*$ such that $x \stackrel{t\omega}{\Rightarrow}$. Two automata G_1 and G_2 are *nonconflicting* if $G_1 \parallel G_2$ is nonblocking.

To reason about conflicts in a compositional way, a notion of *conflict equivalence* is developed in [12]. 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 the test composed with the automaton in question is nonblocking or not.

Definition 4 Two automata G_1 and G_2 are said to be *conflict equivalent*, written $G_1 \simeq_{\mathrm{conf}} G_2$, if, for any automaton T, $G_1 \parallel T$ is nonblocking if and only if $G_2 \parallel T$ is nonblocking.

Conflict equivalence is the coarsest possible congruence with respect to synchronous composition that preserves nonblocking [12]. There are exponential algorithms to determine whether two given automata are conflict equivalent [18, 21]. However, in general there is no unique minimal conflict equivalent representation of a given automaton [8].

When verifying whether a composed system of automata

$$G_1 \parallel G_2 \parallel \cdots \parallel G_n \ , \tag{3}$$

is nonblocking, the compositional method [9] avoids building the complete synchronous product immediately. Typically, some of the components G_i have *local* events, i.e., events used only by G_i. These local events are abstracted using hiding, i.e., they are replaced by the silent event τ. The resultant automaton can then be simplified in various ways, and G_i is replaced by a typically smaller conflict equivalent automaton G_i'. Once no further simplification is possible, a subsystem of automata $(G_j)_{j\in J}$ is selected and replaced by its synchronous composition, and the procedure starts over.

3 Annotated Automata

This section shows how annotations are used to bring automata in a more regular form to make simplification with respect to conflict equivalence more effective. Using the running example in Fig. 1, methods to construct an annotated automaton are described in 3.1 and 3.2, and three abstraction rules to simplify annotated automata are presented in 3.3–3.5. In 3.6, the complete abstraction procedure to simplify automata using annotations is presented. Proofs of the propositions stated in this section can be found in Section 4.

3.1 Annotation

The states in a nondeterministic automaton carry several implicit requirements characterising their blocking or nonblocking behaviour in composition with other automata. For illustration, consider state q_0 in automaton G in Fig. 1. Its eligible event set is $\mathrm{Elig}_G(q_0) = \{\alpha, \beta, \gamma\}$; note that β is included because of the silent transition to q_4. Blocking will occur if state q_0 is composed with a state that does not enable at least one of the events α, β, or γ. Moreover, due to the silent transitions to states q_3 and q_4, any state composed with q_0 also needs to enable at least one event from their sets of eligible events, $\mathrm{Elig}_G(q_4) = \{\alpha, \beta\}$ and $\mathrm{Elig}_G(q_3) = \{\alpha\}$. In order to capture these nonblocking requirements in a more concise manner, the three eligible event sets are associated with state q_0 as *annotations*.

Definition 5 An *annotated automaton* is a 5-tuple $A = \langle \Sigma, Q, \rightarrow, Q^\circ, Ann \rangle$ such that $\langle \Sigma, Q, \rightarrow, Q^\circ \rangle$ is an ordinary automaton without τ-transitions, and $Ann \subseteq Q \times 2^{\Sigma_\omega}$ is the *annotation relation*, which satisfies the following conditions:

(i) for every $x \in Q$, there exists $a \subseteq \Sigma_\omega$ such that $(x, a) \in Ann$;
(ii) for every $(x, a) \in Ann$, it holds that $a \subseteq \mathrm{Elig}_A(x)$.

An annotation is a set of events $a \subseteq \Sigma_\omega$ associated with a state $x \in Q$. The intended meaning of $(x, a) \in Ann$ is that, if the automaton is in state x, at least one of the events in a must be enabled in the synchronous composition of the entire system in order to avert blocking. The empty set of events can also serve as an annotation, which is used to characterise

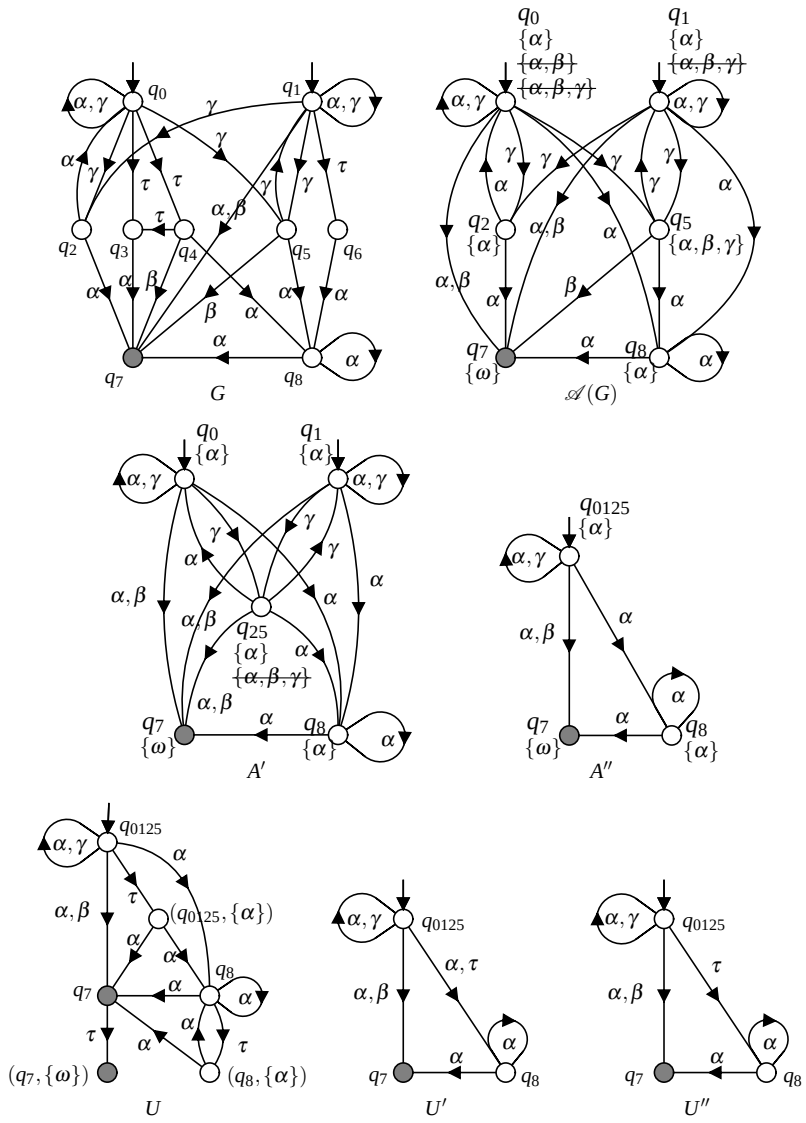

Fig. 1 Simplification of automaton G using annotations gives $G \simeq_{\text{conf}} U''$.

deadlock states. Annotations are similar to *ready sets* [15] or the complements of *failure sets* [10], but they can only be used to partially characterise conflict equivalence.

The two requirements (i) and (ii) ensure that annotations capture the idea of nonblocking requirements correctly. Each state must have at least one annotation, and all annotations must be subsets of the eligible event set of their state. When annotating automata in practice, every state can be associated with its own eligible event set as an annotation, and this "maximal" annotation does not need to be stored explicitly in an annotated automaton as it can be inferred from the transitions.

The following definition shows how to transform an arbitrary nondeterministic automaton into an annotated automaton, replacing silent (τ) transitions by annotations to represent the associated nonblocking requirements.

Definition 6 Let $G = \langle \Sigma, Q, \rightarrow, Q^\circ \rangle$ be an automaton. The *annotated form* of G is

$$\mathscr{A}(G) \;=\; \langle \Sigma, Q, \rightarrow_A, Q^\circ, Ann \rangle \;, \tag{4}$$

where

$$\rightarrow_A = \{\, (x, \sigma, y) \in Q \times \Sigma_\omega \times Q \mid x \stackrel{\varepsilon}{\Rightarrow} z \stackrel{\sigma}{\rightarrow} y \text{ for some } z \in Q \,\} \;; \tag{5}$$

$$Ann = \{\, (x, \mathrm{Elig}_G(y)) \mid x \stackrel{\varepsilon}{\Rightarrow} y \,\} \;. \tag{6}$$

The annotated form clearly satisfies the two conditions (i) and (ii) in Def. 5, because $(x, \mathrm{Elig}_G(x)) \in Ann$ for every $x \in Q$, and $x \stackrel{\varepsilon}{\Rightarrow} y$ implies $\mathrm{Elig}_G(y) \subseteq \mathrm{Elig}_G(x)$.

The annotated form is obtained from the original automaton by replacing all silent transitions by the transitions originating from the silent successor states: if state z can be reached silently from state x, then all transitions originating from z are copied to x. Due to this removal of silent transitions, some states may become unreachable and then can be removed. To retain the nonblocking conditions associated with the originally silently reached states, their eligible event sets are added as annotations to the start states of the removed transitions.

Example 1 Fig. 1 shows an automaton G and its annotated form $\mathscr{A}(G)$. As each state can be reached from itself after 0 silent transitions, it is associated with its own eligible event set as an annotation. The state q_0 collects all the outgoing transitions of q_3 and q_4, because it is connected to these two states by silent transitions, and annotations are added to q_0 for each of these two states. Similarly, q_1 has all the outgoing transitions and the annotation $\{\alpha\}$ of q_6. The states q_3, q_4, and q_6 have been deleted because they become unreachable after the removal of silent transitions.

Complexity The annotated form $\mathscr{A}(G)$ of $G = \langle \Sigma, Q, \rightarrow, Q^\circ \rangle$ has $|Q|$ states, up to $|Q|^2|\Sigma_\omega|$ transitions, and up to $|Q|^2$ annotations. Thus, its size is bounded by $O(|Q|^2|\Sigma|)$. The time complexity to construct $\mathscr{A}(G)$ is dominated by the computation of the transitive closure of the silent transitions, i.e., $O(|Q|^3)$ [14].

Annotation removes information, and it may well happen that two different automata have equal annotated forms. The following proposition shows that this can only happen if the two original automata are conflict equivalent, so the annotation procedure does indeed yield a standardised form with respect to conflict equivalence.

Proposition 1 Let G and H be two automata such that $\mathscr{A}(G) = \mathscr{A}(H)$. Then $G \simeq_{\mathrm{conf}} H$.

Conversely, it is not true that two conflict equivalent automata have the same annotated forms. Annotations cannot be used to characterise conflict equivalence. This is due to the fact that failures equivalence [10] does not imply conflict equivalence, and the same counterexample as given in [12] applies.

3.2 Unannotation

The annotation procedure can be reversed to obtain an ordinary automaton from a given annotated automaton. The reverse operation is called *unannotation* and is characterised by the following definition.

Definition 7 Let $A = \langle \Sigma, Q, \to, Q^\circ, Ann \rangle$ be an annotated automaton. An *unannotated form* of A is any automaton $U = \langle \Sigma, Q_U, \to_U, Q^\circ \rangle$ such that the following properties hold.

(i) $Q_U = Q \cup Ann$;
(ii) $x \xrightarrow{\tau}_U (x,a)$ for all $(x,a) \in Ann$, and these are the only τ-transitions in U;
(iii) If $x, y \in Q$, then $x \xrightarrow{\sigma}_U y$ if and only if $x \xrightarrow{\sigma} y$.
(iv) If $(x,a) \in Ann$ and $\sigma \in a$, then $(x,a) \xrightarrow{\sigma}_U$;
(v) If $(x,a) \xrightarrow{\sigma}_U y$, then $\sigma \in a$ and $x \xrightarrow{\sigma} y$.

The state space of an unannotated form consists of all the *original states* of the annotated automaton plus an additional so-called *annotation state* for each annotation (i), which is linked to its original state by a silent transition (ii). Furthermore, the unannotated form contains all the transitions of the annotated automaton (iii). In addition, the annotation states must have outgoing transitions for each event in their respective annotation (iv), and these transitions must lead to some successor state reached by the same event from the corresponding original state (v).

Given an annotated automaton A, an unannotated form can be constructed by including the states and transitions according to (i), (ii), and (iii), and by arbitrarily choosing for each annotation state (x,a) and each event $\sigma \in a$ a transition $x \xrightarrow{\sigma} y$, and then including the transition $(x,a) \xrightarrow{\sigma} y$ in the unannotated form. There are several possibilities to choose transitions satisfying points (iv) and (v), but the ambiguity does not cause problems with conflict-preserving abstraction.

Proposition 2 Let A be an annotated automaton, and let U_1 and U_2 be unannotated forms of A. Then $U_1 \simeq_{\text{conf}} U_2$.

This result confirms that unannotated forms are well-defined up to conflict equivalence, so the ambiguity in Def. 7 does not affect the nonblocking property and can be exploited to minimise unannotated forms.

Example 2 In Fig. 1, automaton U is an unannotated form of the annotated automaton A''. The three annotations in A'' have been replaced by annotation states $(q_7, \{\omega\})$ $(q_8, \{\alpha\})$, and $(q_{0125}, \{\alpha\})$. Note that the transition $(q_{0125}, \{\alpha\}) \xrightarrow{\alpha} q_{0125}$ is not included in U, although it could be inherited from q_{0125}.

Complexity Given $G = \langle \Sigma, Q, \to, Q^\circ \rangle$, an unannotated form of $\mathscr{A}(G)$ has up to $|Q| + |Ann| \leq |Q| + |Q|^2$ states and up to $|\to| + |Ann| + |Ann||\Sigma_\omega| \leq |Q|^2|\Sigma_\omega|$ transitions. Its space complexity is $O(|Q|^2|\Sigma|)$, and this is also the time complexity to construct it from an annotated automaton. This worst-case is unusual in practice—in the experiments in Section 5, the number of states after unannotation is almost always less than it was before annotation.

The following result confirms that unannotation is a reverse operation of the annotation procedure, up to conflict equivalence. Conflict equivalence is preserved by annotation and subsequent unannotation.

Proposition 3 Let G be an automaton, and let U be an unannotated form of $\mathscr{A}(G)$. Then $U \simeq_{\text{conf}} G$.

In the following sections, different methods are presented to simplify annotated automata. The simplification needs to be carried out in a conflict-preserving way, and this requires an appropriate notion of conflict equivalence of annotated automata. The following definition is justified by Prop. 2 and 3, and by the fact that every annotated automaton has an unannotated form.

Definition 8 Two annotated automata A_1 and A_2 are conflict equivalent, written $A_1 \simeq_{\text{conf}} A_2$, if for every unannotated form U_1 of A_1 and for every unannotated form U_2 of A_2 it holds that $U_1 \simeq_{\text{conf}} U_2$.

3.3 Subsumption

Annotations are sets of events that must be enabled to avert blocking. More precisely, when a state is entered, at least one of the events in each of its annotations needs to be enabled in order to avert blocking. This leads to the observation that certain annotations are redundant. For example, if a state has both the annotations $\{\alpha\}$ and $\{\alpha,\beta\}$, then the latter is implied by the former. The state already requires event α to be enabled, so the fact that α or β needs to be enabled adds no additional information. The annotation $\{\alpha,\beta\}$, being a superset of $\{\alpha\}$, is said to be covered or *subsumed* by $\{\alpha\}$, and subsumed annotations can be removed without affecting conflict equivalence.

This gives rise to the following *subsumption rule*: if an annotated automaton contains annotations (x,a) and (x,b) such that $a \subsetneqq b$, then the annotation (x,b) can be removed. The removal of subsumed annotations from an annotated automaton preserves conditions (i) and (ii) in Def. 5, because no annotations are added and annotations can only be removed from states that have more than one annotation.

Example 3 In state q_0 of automaton $\mathscr{A}(G)$ in Fig. 1, the annotation $\{\alpha\}$ subsumes $\{\alpha,\beta\}$ and $\{\alpha,\beta,\gamma\}$, and the annotation $\{\alpha\}$ in state q_1 subsumes $\{\alpha,\beta,\gamma\}$. The subsumed annotations are struck out in the figure.

Proposition 4 Let $A = \langle \Sigma, Q, \rightarrow, Q^\circ, Ann\rangle$ and $A_{\text{sub}} = \langle \Sigma, Q, \rightarrow, Q^\circ, Ann_{\text{sub}}\rangle$ be two annotated automata such that $Ann_{\text{sub}} \subseteq Ann$ and for all $(x,a) \in Ann$ there exists $a_{\text{sub}} \subseteq a$ such that $(x,a_{\text{sub}}) \in Ann_{\text{sub}}$. Then $A \simeq_{\text{conf}} A_{\text{sub}}$.

Complexity The annotated form $\mathscr{A}(G)$ of $G = \langle \Sigma, Q, \rightarrow, Q^\circ\rangle$ has up to $|Q|$ annotations per state, which gives $O(|Q|^2)$ subsumption tests per state, and the cost of each test is $O(|\Sigma|)$. So the worst-case time complexity of the subsumption test for $\mathscr{A}(G)$ is $O(|Q|^3|\Sigma|)$. This makes subsumption one of the most expensive of the abstractions presented here, but experimental results show that it is worthwhile. The subsumption test is best done immediately while constructing annotated automata or introducing annotations, considerably reducing memory requirements.

3.4 Incoming Equivalence

Incoming equivalence [9] identifies two states as equivalent if they have exactly the same incoming transitions. The concept is extended to annotated automata as follows.

Definition 9 Let $A = \langle \Sigma, Q, \to, Q^\circ, Ann \rangle$ be an annotated automaton. The incoming equivalence relation $\sim_{\text{inc}} \subseteq Q \times Q$ is defined such that $x \sim_{\text{inc}} y$ if and only if the following conditions hold.

- $x \in Q^\circ$ if and only if $y \in Q^\circ$;
- For all states $z \in Q$ and all events $\sigma \in \Sigma_\omega$, it holds that $z \xrightarrow{\sigma} x$ if and only if $z \xrightarrow{\sigma} y$.

In [9], incoming equivalence is used as a restriction to make certain simplification rules applicable. Due to the improved regularity achieved by annotations, all incoming equivalent states in an annotated automaton can be merged. This merging is done using the standard automaton quotient, with the addition that, when merging several states into one, the resultant state receives the annotations of all original states.

Definition 10 Let $A = \langle \Sigma, Q, \to, Q^\circ, Ann \rangle$ be an annotated automaton, and let $\sim \subseteq Q \times Q$ be an equivalence relation. The *quotient automaton* of A modulo $\sim$ is $A/\!\sim = \langle \Sigma, Q/\!\sim, \to/\!\sim, \tilde{Q}^\circ, \tilde{A}nn \rangle$, where

$$\to/\!\sim = \{\, ([x], \sigma, [y]) \mid x \xrightarrow{\sigma} y \,\}\,; \tag{7}$$

$$\tilde{Q}^\circ = \{\, [x] \mid x \in Q^\circ \,\}\,; \tag{8}$$

$$\tilde{A}nn = \{\, ([x], a) \mid x \in Q \text{ and there exists } x' \sim x \text{ such that } (x', a) \in Ann \,\}\,. \tag{9}$$

Here, $[x] = \{\, x' \in Q \mid x' \sim x \,\}$ denotes the *equivalence class* of $x \in Q$ with respect to $\sim$, and $Q/\!\sim = \{\, [x] \mid x \in Q \,\}$ is the set of equivalence classes modulo $\sim$.

It is easily confirmed that the quotient $A/\!\sim$ of an annotated automaton A satisfies conditions (i) and (ii) in Def. 5, because every merged state receives annotations from all its original states, and the eligible events sets are increased when merging.

Proposition 5 Let $A = \langle \Sigma, Q, \to, Q^\circ, Ann \rangle$ be an annotated automaton. Then $A \simeq_{\text{conf}} A/\!\sim_{\text{inc}}$.

The merging of incoming equivalent states can be considered as a generalisation of the silent continuation rule [9]. An annotation symbolises a silent transition to an implicit state. When incoming equivalent states are merged, the nondeterministic decisions of the predecessor states are deferred by one step, expressed by the merged annotations.

Example 4 The annotated automaton A' in Fig. 1 is the result of using incoming equivalence to simplify $\mathscr{A}(G)$. States q_2 and q_5 are incoming equivalent and have been merged. The resultant state q_{25} receives the annotations $\{\alpha\}$ and $\{\alpha, \beta, \gamma\}$, but only $\{\alpha\}$ remains because of subsumption.

Complexity The complexity of partitioning an automaton based on incoming equivalence is $O(|Q|^2|\Sigma|)$. Two states are equivalent if they have equal sets of incoming transitions, which can be determined efficiently using hash codes. Hash codes can be set up in a single pass over all transitions of the automaton, of which there are up to $|Q|^2|\Sigma_\omega|$, and the construction of the simplified automaton is achieved by another loop over all transitions, in the same complexity [9]. However, the merging of some states may make other states incoming equivalent, so the abstraction should be repeated to ensure a minimal result. The maximum number of iterations is $|Q|$, as each merge except the last reduces the number of states, so the complexity to obtain a minimal abstraction by incoming equivalence is $O(|Q|^3|\Sigma|)$.

3.5 Bisimulation

Bisimulation and *observation equivalence* [13] are general tools that have been used with considerable success to simplify automata during nonblocking verification [9, 19]. Bisimulation can also be applied to annotated automata, with the added restriction that bisimilar states must have the same annotations. Nevertheless, the removal of silent transitions can transform several conflict equivalent transition structures into the same annotated states, even if they are not originally observation equivalent. So bisimulation on the annotated automaton can be more effective, particularly after the removal of subsumed annotations.

Definition 11 Let $A_1 = \langle \Sigma, Q_1, \rightarrow_1, Q_1^\circ, Ann_1 \rangle$ and $A_2 = \langle \Sigma, Q_2, \rightarrow_2, Q_2^\circ, Ann_2 \rangle$ be two annotated automata. A relation $\approx \subseteq Q_1 \times Q_2$ is called a *bisimulation* between A_1 and A_2, if the following conditions hold for all states $x_1 \in Q_1$ and $x_2 \in Q_2$ such that $x_1 \approx x_2$.

- For all $\sigma \in \Sigma_\omega$, if $x_1 \xrightarrow{\sigma} y_1$ then there exists $y_2 \in Q_2$ such that $y_1 \approx y_2$ and $x_2 \xrightarrow{\sigma} y_2$.
- For all $\sigma \in \Sigma_\omega$, if $x_2 \xrightarrow{\sigma} y_2$ then there exists $y_1 \in Q_1$ such that $y_1 \approx y_2$ and $x_1 \xrightarrow{\sigma} y_1$.
- For all $a \subseteq \Sigma_\omega$, it holds that $(x_1, a) \in Ann_1$ if and only if $(x_2, a) \in Ann_2$.

A_1 and A_2 are *bisimulation equivalent* or *bisimilar*, written $A_1 \approx A_2$, if there exists a bisimulation $\approx$ between A_1 and A_2 such that, for every initial state $x_1^\circ \in Q_1^\circ$ there exists an initial state $x_2^\circ \in Q_2^\circ$ such that $x_1^\circ \approx x_2^\circ$, and vice versa.

It is easily confirmed that conditions (i) and (ii) in Def. 5 are preserved under bisimilarity of annotated automata. This is because bisimilar states always have the same sets of annotations and eligible events.

Example 5 Automaton A'' in Fig. 1 is bisimilar to A'. States q_0, q_1, and q_{25} have been merged due to the fact that they have the same annotations and equivalent outgoing transitions. Note that this only becomes possible after annotation, subsumption, and incoming equivalence.

Proposition 6 Let A_1 and A_2 be annotated automata such that $A_1 \approx A_2$. Then $A_1 \simeq_{\text{conf}} A_2$.

Complexity Given an annotated automaton, a coarsest bisimulation relation can be found in time complexity $O(|\rightarrow| \log |Q|)$ using the algorithm in [7]. The annotated form of $G = \langle \Sigma, Q, \rightarrow, Q^\circ \rangle$ has $O(|Q|^2|\Sigma|)$ transitions, giving $O(|Q|^2|\Sigma| \log |Q|)$ time complexity for its simplification. An initial partition based on annotations can be established with lower time complexity.

3.6 Abstraction Procedure

This section explains how the above results can be used to minimise a given automaton with respect to conflict equivalence. Given an automaton G, the task is to compute a hopefully smaller abstraction G' conflict equivalent to G.

Given the complexity of the annotation procedure, it is advisable to reduce the size of the input automaton G using some standard means before constructing an annotated form. While not necessarily optimal for conflict equivalence, bisimulation or observation equivalence [13] can be computed efficiently and are known to achieve significant reduction, as is the removal of blocking states [9].

After simplification of the input automaton, the next step is to compute its annotated form $\mathscr{A}(G)$, which then is simplified in several steps. While constructing the annotated form, annotations can be checked for subsumption on the fly, suppressing the generation of any redundant annotations. The resulting annotated form is next simplified by merging incoming equivalent states, again checking for subsumption and removing annotations that become redundant. Then the result is minimised according to bisimulation equivalence.

After simplifying the annotated automaton, it is unannotated to obtain an ordinary automaton that is conflict equivalent to the input. There are different ways to construct an unannotated form that satisfies the conditions of Def. 7, as there is considerable leeway in how outgoing transitions from annotation states can be chosen, and by making clever choices, the new annotation states can become bisimilar to original states or other annotation states, making it possible to further simplify the result.

An example of the abstraction procedure is shown in Fig. 1. Automaton G is first annotated to obtain $\mathscr{A}(G)$, with subsumption being tested on the fly to suppress some annotations struck out in the figure. Next incoming equivalence leads to the abstraction A', with another annotation being suppressed due to subsumption as discussed in example 4, and the result is further simplified using bisimulation, giving A''.

Since the annotated automaton cannot be simplified further, it is replaced by its unannotated form U. As explained in example 2, the transition $(q_{0125},\{\alpha\}) \xrightarrow{\alpha} q_{0125}$ is not included in U. This choice makes the states q_8, $(q_8,\{\alpha\})$, and $(q_{0125},\{\alpha\})$ observation equivalent [13], so they can be merged in addition to states q_7 and $(q_7,\{\omega\})$. This results in the observation equivalent abstraction U'. Furthermore, the transition $q_{0125} \xrightarrow{\alpha} q_8$ is redundant according to observation equivalence [5] and can be removed, giving the final result U''.

The abstraction steps in Fig. 1 can be justified by the propositions given in the previous sections. Note that, for every annotated automaton, there exists an unannotated form although it does not always have to be constructed explicitly. Let V and V' be unannotated forms of $\mathscr{A}(G)$ and A', respectively. Then $G \simeq_{\text{conf}} V$ by Prop. 3 and $V \simeq_{\text{conf}} V' \simeq_{\text{conf}} U$ by Prop. 4–6. Furthermore, U is observation equivalent to U' and U'', which implies $U \simeq_{\text{conf}} U''$ according to [12]. Thus,

$$G \simeq_{\text{conf}} V \simeq_{\text{conf}} V' \simeq_{\text{conf}} U \simeq_{\text{conf}} U' \simeq_{\text{conf}} U'' \,. \tag{10}$$

Overall, the automaton G with nine states and 25 transitions is simplified to the conflict-equivalent automaton U'' with three states and seven transitions.

4 Formal Proofs

This section contains formal proofs of the propositions stated in the previous section. The properties of annotated automata and unannotated automata are established in Section 4.1 and 4.2, and these results are used in Section 4.3–4.5 to confirm the correctness of the abstraction rules.

4.1 Annotation

The main result about annotated forms is Prop. 1 in Section 3.1, which states that automata with equal annotated forms are conflict equivalent. Its proof depends on two lemmas that describe the relationship between paths in an automaton and its annotated form.

Lemma 7 Let $G = \langle \Sigma, Q, \rightarrow, Q^\circ \rangle$ be an automaton, and let $\mathscr{A}(G) = \langle \Sigma, Q, \rightarrow_A, Q^\circ, Ann \rangle$ be its annotated form. For all traces $s \in \Sigma^*$ and all events $\sigma \in \Sigma$, the annotated form has a path $x \xrightarrow{s\sigma}_A z$ if and only if there exists a path $x \stackrel{s}{\Rightarrow} y \xrightarrow{\sigma} z$ in G, for some $y \in Q$.

Proof The claim is proved by induction on $|s|$.

In the base case, $s = \varepsilon$, the claim follows directly from the definition (5).

For the inductive step, let $s = t\sigma'$. Then note,

$$x \xrightarrow{s\sigma}_A z \quad \Longleftrightarrow \quad x \xrightarrow{t\sigma'\sigma}_A z \quad \Longleftrightarrow \quad x \xrightarrow{t\sigma'}_A y \xrightarrow{\sigma}_A z \quad \text{for some } y \in Q\ . \tag{11}$$

By inductive assumption, $x \xrightarrow{t\sigma'}_A y$ holds if and only if $x \stackrel{s}{\Rightarrow} y' \xrightarrow{\sigma'} y$ for some $y' \in Q$, and by (5) $y \xrightarrow{\sigma}_A z$ holds if and only if $y \stackrel{\varepsilon}{\Rightarrow} z' \xrightarrow{\sigma} z$ for some $z' \in Q$. Thus, (11) becomes equivalent to,

$$x \stackrel{s}{\Rightarrow} y' \xrightarrow{\sigma'} y \stackrel{\varepsilon}{\Rightarrow} z' \xrightarrow{\sigma} z \quad \text{for some } y', z' \in Q \quad \Longleftrightarrow \quad x \stackrel{s\sigma'}{\Rightarrow} z' \xrightarrow{\sigma} z \quad \text{for some } z' \in Q\ . \quad \square$$

Lemma 8 Let $G = \langle \Sigma, Q, \rightarrow, Q^\circ \rangle$ be an automaton, and let $\mathscr{A}(G) = \langle \Sigma, Q, \rightarrow_A, Q^\circ, Ann \rangle$ be its annotated form. Also let $x, z \in Q$ and $s \in \Sigma^*$.

(i) If $x \stackrel{s}{\Rightarrow} z$, then there exists $z' \in Q$ such that $x \stackrel{s}{\Rightarrow}_A z'$ and $(z', \mathrm{Elig}_G(z)) \in Ann$.

(ii) If $x \xrightarrow{s}_A z$ and $(z, a) \in Ann$, then there exists $z' \in Q$ such that $x \stackrel{s}{\Rightarrow} z'$ and $\mathrm{Elig}_G(z') = a$.

Proof (i) Let $x \stackrel{s}{\Rightarrow} z$. If $s = \varepsilon$ then $x \stackrel{\varepsilon}{\Rightarrow} z$, so $x \xrightarrow{\varepsilon}_A x$ with $(x, \mathrm{Elig}_G(z)) \in Ann$ by Def. 6 (6). Otherwise, $s = t\sigma$ and thus $x \stackrel{t}{\Rightarrow} y \xrightarrow{\sigma} z' \stackrel{\varepsilon}{\Rightarrow} z$ for some $y, z' \in Q$. By Lemma 7, it follows that $x \stackrel{t\sigma}{\Rightarrow}_A z'$, and $(z', \mathrm{Elig}_G(z)) \in Ann$ since $z' \stackrel{\varepsilon}{\Rightarrow} z$.

(ii) Let $x \xrightarrow{s}_A z$ and $(z, a) \in Ann$. By Def. 6 (6), there exists $z' \in Q$ such that $z \stackrel{\varepsilon}{\Rightarrow} z'$ and $\mathrm{Elig}_G(z') = a$. If $s = \varepsilon$ then $x = z \stackrel{\varepsilon}{\Rightarrow} z'$ with $\mathrm{Elig}_G(z') = a$. Otherwise, $s = t\sigma$ and by Lemma 7, there exists $y \in Q$ such that $x \stackrel{t}{\Rightarrow} y \xrightarrow{\sigma} z$. Then $x \stackrel{s}{\Rightarrow} z \stackrel{\varepsilon}{\Rightarrow} z'$ with $\mathrm{Elig}_G(z') = a$. $\square$

Given these results, it is now possible to prove Prop. 1, the main result about annotated forms introduced in Section 3.1.

Proposition 1 Let G and H be two automata such that $\mathscr{A}(G) = \mathscr{A}(H)$. Then $G \simeq_{\mathrm{conf}} H$.

Proof Let $G = \langle \Sigma, Q_G, \rightarrow_G, Q_G^\circ \rangle$ and $H = \langle \Sigma, Q_H, \rightarrow_H, Q_H^\circ \rangle$, and let $T = \langle \Sigma_T, Q_T, \rightarrow_T, Q_T^\circ \rangle$ be an arbitrary automaton.

Assume that $G \parallel T$ is nonblocking. It is enough to show that this implies that $H \parallel T$ is nonblocking. Therefore, let $s \in (\Sigma \cup \Sigma_T)^*$ such that $H \parallel T \stackrel{s}{\Rightarrow} (x_H, x_T)$. Then $H \stackrel{P(s)}{\Longrightarrow} x_H$ according to (2), where $P\colon \Sigma \cup \Sigma_T \to \Sigma$ denotes the natural projection, and by Lemma 8 (i), there exists a state $x_A \in Q_H$ such that $\mathscr{A}(G) = \mathscr{A}(H) \xrightarrow{P(s)} x_A$ and $(x_A, \mathrm{Elig}_H(x_H)) \in Ann_H = Ann_G$. By Lemma 8 (ii), there also exists a state $x_G \in Q_G$ such that $G \stackrel{P(s)}{\Longrightarrow} x_G$ and $\mathrm{Elig}_G(x_G) = \mathrm{Elig}_H(x_H)$. Thus, $G \parallel T \stackrel{s}{\Rightarrow} (x_G, x_T)$.

As $G \parallel T$ is nonblocking, there exists a trace $t \in (\Sigma \cup \Sigma_T)^*$ such that $(x_G, x_T) \stackrel{t\omega}{\Rightarrow}$. Clearly, $t\omega = u\sigma v$ for some $u \in (\Sigma_T \setminus \Sigma)^*$, $\sigma \in \Sigma_\omega$, and $v \in (\Sigma_\omega \cup \Sigma_T)^*$. Then $x_G \stackrel{u}{\Rightarrow}_G x_G \xrightarrow{\sigma}_G$, i.e., $\sigma \in \mathrm{Elig}_G(x_G) = \mathrm{Elig}_H(x_H)$. If $\sigma = \omega$, then clearly $H \parallel T \stackrel{s}{\Rightarrow} (x_H, x_T) \stackrel{u\omega}{\Rightarrow}$, which is enough to show that $H \parallel T$ is nonblocking. Otherwise, if $\sigma \in \Sigma$, let $y_H \in Q_H$ such that $H \stackrel{P(s)}{\Longrightarrow} x_H \xrightarrow{\sigma} y_H$. By Lemma 7, this implies $\mathscr{A}(G) = \mathscr{A}(H) \xrightarrow{P(s)\sigma} y_H$ and $G \stackrel{P(s)\sigma}{\Longrightarrow} y_H$. Since $u \in (\Sigma_T \setminus \Sigma)^*$, it

also follows that $G \parallel T \stackrel{su\sigma}{\Longrightarrow} (y_H, y_T)$ for some state y_T of T. Since $G \parallel T$ is nonblocking, there exists a trace $w \in (\Sigma \cup \Sigma_T)^*$ such that $(y_H, y_T) \stackrel{w\omega}{\Rightarrow}$. Therefore,

$$H \parallel T \stackrel{s}{\Rightarrow} (x_H, x_T) \stackrel{u\sigma}{\Rightarrow} (y_H, y_T) \stackrel{w\omega}{\Rightarrow} \ . \tag{12}$$

Since (x_H, x_T) was chosen arbitrarily, it follows that $H \parallel T$ is nonblocking. □

4.2 Unannotation

This section proves two key results about unannotation. Unannotated forms are equal with respect to conflict equivalence (Prop. 2), and conflict equivalence is preserved when annotating and unannotating again (Prop. 3).

These results depend on the relationship between traces in an annotated automaton and its unannotated forms, which are first established. Lemma 9 shows that every nonempty path of an annotated automaton corresponds to an equivalent path of its unannotated form. Lemma 10 lifts this result to all paths of an unannotated form, considering separately the cases of original and annotation end states.

Lemma 9 Let $A = \langle \Sigma, Q, \rightarrow, Q^\circ, Ann \rangle$ be an annotated automaton, and let $U = \langle \Sigma, Q_U, \rightarrow_U, Q^\circ \rangle$ be an unannotated form of A. For all traces $s \in \Sigma^*$, all events $\sigma \in \Sigma$, and all states $x \in Q$, it holds that $x \stackrel{s\sigma}{\rightarrow} z$ if and only if $x \stackrel{s}{\Rightarrow}_U y \stackrel{\sigma}{\rightarrow}_U z$ for some $y \in Q_U$.

Proof The claim is proved by induction on $|s|$.

First consider the base case $s = \varepsilon$. If $x \stackrel{\sigma}{\rightarrow} z$, it follows directly from Def. 7 (iii) that $x \stackrel{\sigma}{\rightarrow}_U z$. Conversely, if $x \stackrel{\varepsilon}{\Rightarrow}_U y \stackrel{\sigma}{\rightarrow}_U z$, then by Def. 7 (ii) either $x = y$ or $x \stackrel{\tau}{\rightarrow}_U y$. If $x = y \stackrel{\sigma}{\rightarrow}_U z$, then $x \stackrel{\sigma}{\rightarrow} z$ by Def. 7 (iii). If $x \stackrel{\tau}{\rightarrow}_U y$, then $y = (x, a) \in Ann$ by Def. 7 (ii), and $(x, a) = y \stackrel{\sigma}{\rightarrow}_U z$ implies $x \stackrel{\sigma}{\rightarrow} z$ by Def. 7 (v).

For the inductive step, let $s = t\sigma'$, and first assume $x \stackrel{t\sigma'}{\rightarrow} y \stackrel{\sigma}{\rightarrow} z$. By inductive assumption, it follows that $x \stackrel{t\sigma'}{\Rightarrow}_U y$, and by Def. 7 (iii) it holds that $y \stackrel{\sigma}{\rightarrow}_U z$. This implies $x \stackrel{t\sigma'}{\Rightarrow}_U y \stackrel{\sigma}{\rightarrow}_U z$. Conversely, assume that $x \stackrel{t\sigma'}{\Rightarrow}_U y \stackrel{\sigma}{\rightarrow}_U z$, i.e.,

$$x \stackrel{t}{\Rightarrow}_U x' \stackrel{\sigma'}{\rightarrow}_U y' \stackrel{\varepsilon}{\Rightarrow}_U y \stackrel{\sigma}{\rightarrow}_U z \ . \tag{13}$$

Then $x \stackrel{t\sigma'}{\rightarrow} y'$ by inductive assumption, and by Def. 7 (ii), it either holds that $y' = y$, and thus $y' \stackrel{\sigma}{\rightarrow}_U z$, which implies $y' \stackrel{\sigma}{\rightarrow} z$ by Def. 7 (iii); or there is an annotation $(y', a) \in Ann$ such that $y = (y', a)$, i.e., $(y', a) \stackrel{\sigma}{\rightarrow}_U z$ and thus $y' \stackrel{\sigma}{\rightarrow} z$ by Def. 7 (v). In both cases, $x \stackrel{t\sigma'}{\rightarrow} y' \stackrel{\sigma}{\rightarrow} z$, i.e., $x \stackrel{s\sigma}{\rightarrow} z$. □

Lemma 10 Let $A = \langle \Sigma, Q, \rightarrow, Q^\circ, Ann \rangle$ be an annotated automaton, and let $U = \langle \Sigma, Q_U, \rightarrow_U, Q^\circ \rangle$ be an unannotated form of A.

(i) For all traces $s \in \Sigma^*$ and all states $x \in Q$, it holds that $A \stackrel{s}{\rightarrow} x$ if and only if $U \stackrel{s}{\Rightarrow} x$.
(ii) For all traces $s \in \Sigma^*$ and all annotations $(x, a) \in Ann$, it holds that $A \stackrel{s}{\rightarrow} x$ if and only if $U \stackrel{s}{\Rightarrow} (x, a)$.

Proof (i) Firstly, if $s = \varepsilon$, then $A \xrightarrow{\varepsilon} x$ implies $x \in Q^\circ$ and thus $U \overset{\varepsilon}{\Rightarrow} x$, and conversely $U \overset{\varepsilon}{\Rightarrow} x$ with $x \in Q$ implies $x \in Q^\circ$ by Def. 7 (ii) and thus $A \xrightarrow{\varepsilon} x$. Secondly, if $s = t\sigma$, the claim follows immediately from Lemma 9.

(ii) Let $(x,a) \in Ann$. Then $x \xrightarrow{\tau}_U (x,a)$ by Def. 7 (ii), and this is the only way how (x,a) can be reached in U. Then the claim follows from (i), because $x \in Q$ and thus $A \xrightarrow{s} x$ if and only if $U \overset{s}{\Rightarrow} x \xrightarrow{\tau} (x,a)$. □

The result that two unannotated forms of the same annotated automaton are conflict equivalent now becomes a consequence of Lemmas 9 and 10.

Proposition 2 Let A be an annotated automaton, and let U_1 and U_2 be unannotated forms of A. Then $U_1 \simeq_{\text{conf}} U_2$.

Proof Let $A = \langle \Sigma, Q, \to, Q^\circ, Ann \rangle$, and let $U_i = \langle \Sigma, Q \cup Ann, \to_i, Q^\circ \rangle$ for $i = 1,2$ be unannotated forms of A. Furthermore, let $T = \langle \Sigma_T, Q_T, \to_T, Q_T^\circ \rangle$ be an arbitrary automaton such that $U_1 \parallel T$ is nonblocking. It is enough to show that this implies that $U_2 \parallel T$ is nonblocking. Therefore, let $s \in (\Sigma \cup \Sigma_T)^*$ such that $U_2 \parallel T \overset{s}{\Rightarrow} (x, x_T)$, and consider two cases.

Case 1: $x = (x_a, a) \in Ann$. Then $U_2 \overset{P(s)}{\Longrightarrow} (x_a, a)$, which implies $A \xrightarrow{P(s)} x_a$ and $U_1 \overset{P(s)}{\Longrightarrow} (x_a, a)$ by Lemma 10 (ii). Thus $U_1 \parallel T \overset{s}{\Rightarrow} ((x_a,a), x_T)$, and since $U_1 \parallel T$ is nonblocking, there exists $t \in \Sigma^*$ such that $U_1 \parallel T \overset{s}{\Rightarrow} ((x_a,a), x_T) \overset{t\omega}{\Rightarrow}$. Write $t\omega = u\sigma v$ with $u \in (\Sigma_T \setminus \Sigma)^*$, $\sigma \in \Sigma_\omega$, and $v \in (\Sigma_\omega \cup \Sigma_T)^*$. Then $U_1 \parallel T \overset{s}{\Rightarrow} ((x_a,a), x_T) \overset{u}{\Rightarrow} ((x_a,a), x_T') \xrightarrow{\sigma} (y_1, y_T)$, so $\sigma \in \text{Elig}_{U_1}((x_a,a)) = a = \text{Elig}_{U_2}((x_a,a))$ by Def. 7 (iv) and (v), and thus $(x_a,a) \xrightarrow{\sigma}_2 y_2$ for some $y_2 \in Q$. Thus $U_2 \parallel T \overset{s}{\Rightarrow} ((x_a,a), x_T) \overset{u}{\Rightarrow} ((x_a,a), x_T') \xrightarrow{\sigma} (y_2, y_T)$. If $\sigma = \omega$, then clearly $U_2 \parallel T \overset{s}{\Rightarrow} ((x_a,a), x_T) \overset{u\omega}{\Rightarrow}$, which is enough to show that $U_2 \parallel T$ is nonblocking. Otherwise, $U_2 \parallel T \overset{su\sigma}{\Longrightarrow} (y_2, y_T)$ with $su\sigma \in (\Sigma \cup \Sigma_T)^*$ and $y_2 \in Q$, and the proof continues as in *Case 2*.

Case 2: $x \in Q$. Then $U_2 \overset{P(s)}{\Longrightarrow} x$ implies $A \xrightarrow{P(s)} x$ and $U_1 \overset{P(s)}{\Longrightarrow} x$ by Lemma 10 (i). Thus $U_1 \parallel T \overset{s}{\Rightarrow} (x, x_T)$, and since $U_1 \parallel T$ is nonblocking, there exists $w \in \Sigma^*$ such that $U_1 \parallel T \overset{s}{\Rightarrow} (x, x_T) \overset{w\omega}{\Rightarrow} (y, y_T)$ where $y \in Q$. Therefore $x \overset{P(w)\omega}{\Longrightarrow}_1 y$, which implies $x \xrightarrow{P(w)\omega} y$ and $x \overset{P(w)\omega}{\Longrightarrow}_2 y$ by Lemma 9. Then $U_2 \parallel T \overset{s}{\Rightarrow} (x, x_T) \overset{w\omega}{\Rightarrow}$, and since (x, x_T) was chosen arbitrarily, it follows that $U_2 \parallel T$ is nonblocking. □

The second main result about unannotation is that conflict equivalence is preserved when annotation is followed by unannotation. To prove this, it is helpful to first establish a lemma about annotations, namely that the annotated form of an automaton is equal to the annotated form of its unannotation. Due to the way how annotated forms are defined in this paper, Lemma 11 only applies to annotated forms of an ordinary automaton G, not to arbitrary annotated automata.

Lemma 11 Let G be an automaton, and let U be an unannotated form of $\mathscr{A}(G)$. Then $\mathscr{A}(U) = \mathscr{A}(G)$.

Proof Let $\mathscr{A}(G) = \langle \Sigma, Q, \to, Q^\circ, Ann \rangle$, let $U = \langle \Sigma, Q_U, \to_U, Q^\circ \rangle$ be an unannotated form of $\mathscr{A}(G)$, and let $\mathscr{A}(U) = \langle \Sigma, Q_U, \to_{\mathscr{A}(U)}, Q^\circ, Ann_{\mathscr{A}(U)} \rangle$. It will be shown that the reachable parts of $\mathscr{A}(G)$ and $\mathscr{A}(U)$ are equal, i.e., that $\to = \to_{\mathscr{A}(U)|Q}$ and $Ann = Ann_{\mathscr{A}(U)|Q}$, where $\to_{\mathscr{A}(U)|Q} = \to_{\mathscr{A}(U)} \cap (Q \times \Sigma_\omega \times Q_U)$ and $Ann_{\mathscr{A}(U)|Q} = Ann_{\mathscr{A}(U)} \cap (Q \times 2^{\Sigma_\omega})$.

First, let $x \xrightarrow{\sigma} y$. Then $x \in Q$ and $x \xrightarrow{\sigma}_U y$ by Def. 7 (iii), and $x \xrightarrow{\sigma}_{\mathscr{A}(U)} y$ by Def. 6 (5), and $x \xrightarrow{\sigma}_{\mathscr{A}(U)|Q} y$ as $x \in Q$.

Conversely, let $x \xrightarrow{\sigma}_{\mathscr{A}(U)|Q} y$. Then $x \in Q$ and $x \overset{\varepsilon}{\Rightarrow}_U z \xrightarrow{\sigma}_U y$ for some $z \in Q_U$ by Def. 6 (5). By Def. 7 (ii), this means that either $x = z$, which implies $x \xrightarrow{\sigma}_U y$ and $x \xrightarrow{\sigma} y$ by Def. 7 (iii), or $z = (x,a) \xrightarrow{\sigma}_U y$, which implies $x \xrightarrow{\sigma} y$ by Def. 7 (v).

Second, let $(x,a) \in \mathit{Ann}$. Then $x \in Q$ and $x \xrightarrow{\tau}_U (x,a)$ by Def. 7 (ii) and $\mathrm{Elig}_U((x,a)) = a$ by Def. 7 (iv) and (v). By Def. 6 (6), it follows that $(x,a) = (x, \mathrm{Elig}_U((x,a))) \in \mathit{Ann}_{\mathscr{A}(U)|Q}$.

Conversely, let $(x,a) \in \mathit{Ann}_{\mathscr{A}(U)|Q}$. Then $x \in Q$, and by Def. 6 (6), there exists $y \in Q_U$ such that $x \overset{\varepsilon}{\Rightarrow}_U y$ and $\mathrm{Elig}_U(y) = a$. Here, $x \overset{\varepsilon}{\Rightarrow}_U y$ means that either $x = y$ or $x \xrightarrow{\tau}_U y$.

In the case $x = y$, note that $y = x \in Q$, and $\mathrm{Elig}_U(y) = \mathrm{Elig}_A(y) \cup \bigcup_{(z,a) \in \mathit{Ann}} a = \mathrm{Elig}_A(y)$ by Def. 5 (ii), and $\mathrm{Elig}_A(y) = \mathrm{Elig}_G(y)$ by Def. 6 (5). Therefore, $(x,a) = (y, \mathrm{Elig}_U(y)) = (y, \mathrm{Elig}_A(y)) = (y, \mathrm{Elig}_G(y)) \in \mathit{Ann}$.

In the case $x \xrightarrow{\tau}_U y$, note that $y \in \mathit{Ann}$ by Def. 7 (ii). Then it follows from $\mathrm{Elig}_U(y) = a$ by Def. 7 (iv) and (v) that $(x,a) = y \in \mathit{Ann}$. □

Proposition 3 Let G be an automaton, and let U be an unannotated form of $\mathscr{A}(G)$. Then $U \simeq_{\mathrm{conf}} G$.

Proof By Lemma 11, it holds that $\mathscr{A}(U) = \mathscr{A}(G)$, which implies $U \simeq_{\mathrm{conf}} G$ by Prop. 1. □

4.3 Subsumption

This section contains the proof of Prop. 4 introduced in 3.3, which says that conflict equivalence of annotated automata is preserved under subsumption of annotations. Although lengthy, the proof can be done using the properties of the paths of unannotated forms established in 4.2.

Proposition 4 Let $A = \langle \Sigma, Q, \rightarrow, Q^\circ, \mathit{Ann} \rangle$ and $A_{\mathrm{sub}} = \langle \Sigma, Q, \rightarrow, Q^\circ, \mathit{Ann}_{\mathrm{sub}} \rangle$ be two annotated automata such that $\mathit{Ann}_{\mathrm{sub}} \subseteq \mathit{Ann}$ and for all $(x,a) \in \mathit{Ann}$ there exists $a_{\mathrm{sub}} \subseteq a$ such that $(x, a_{\mathrm{sub}}) \in \mathit{Ann}_{\mathrm{sub}}$. Then $A \simeq_{\mathrm{conf}} A_{\mathrm{sub}}$.

Proof Let $U = \langle \Sigma, Q \cup \mathit{Ann}, \rightarrow_U, Q^\circ \rangle$ and $U_{\mathrm{sub}} = \langle \Sigma, Q \cup \mathit{Ann}_{\mathrm{sub}}, \rightarrow_{U,\mathrm{sub}}, Q^\circ \rangle$ be unannotated forms of A and A_{sub}, respectively. It is to be shown that $U \simeq_{\mathrm{conf}} U_{\mathrm{sub}}$. Therefore, let $T = \langle \Sigma_T, Q_T, \rightarrow_T, Q_T^\circ \rangle$ be an arbitrary automaton.

First, assume that $U \parallel T$ is nonblocking, and let $s \in (\Sigma \cup \Sigma_T)^*$ such that $U_{\mathrm{sub}} \parallel T \overset{s}{\Rightarrow} (x, x_T)$. Then $U_{\mathrm{sub}} \overset{P(s)}{\Longrightarrow} x \in Q \cup \mathit{Ann}_{\mathrm{sub}}$. Consider two cases.

Case 1: $x = (x_a, a) \in \mathit{Ann}_{\mathrm{sub}}$. From $U_{\mathrm{sub}} \overset{P(s)}{\Longrightarrow} x = (x_a, a)$, it follows that $A_{\mathrm{sub}} \xrightarrow{P(s)} x_a$ by Lemma 10 (ii), which implies $A \xrightarrow{P(s)} x_a$ because A and A_{sub} have the same transition relations. Furthermore, since $(x_a, a) \in \mathit{Ann}_{\mathrm{sub}} \subseteq \mathit{Ann}$, it follows by Lemma 10 (ii) that $U \overset{P(s)}{\Longrightarrow} (x_a, a)$. This implies $U \parallel T \overset{s}{\Rightarrow} ((x_a, a), x_T)$, and since $U \parallel T$ is nonblocking, there exists $t \in \Sigma^*$ such that $U \parallel T \overset{s}{\Rightarrow} ((x_a, a), x_T) \overset{t\omega}{\Rightarrow}$. Write $t\omega = u\sigma v$ with $u \in (\Sigma_T \setminus \Sigma)^*$, $\sigma \in \Sigma_\omega$, and $v \in (\Sigma_\omega \cup \Sigma_T)^*$. Then $U \parallel T \overset{s}{\Rightarrow} ((x_a, a), x_T) \overset{u}{\Rightarrow} ((x_a, a), x_T') \xrightarrow{\sigma} (y, y_T)$, so $\sigma \in \mathrm{Elig}_U((x_a, a)) = a = \mathrm{Elig}_{U_{\mathrm{sub}}}((x_a, a))$ by Def. 7 (iv) and (v), and $(x_a, a) \xrightarrow{\sigma}_{U,\mathrm{sub}} y_{\mathrm{sub}}$ for some $y_{\mathrm{sub}} \in Q$. If $\sigma = \omega$, then clearly $U_{\mathrm{sub}} \parallel T \overset{s}{\Rightarrow} ((x_a, a), x_T) \overset{u\omega}{\Rightarrow}$, which is enough to show that $U_{\mathrm{sub}} \parallel T$ is nonblocking. Otherwise, $U_{\mathrm{sub}} \parallel T \overset{su\sigma}{\Longrightarrow} (y_{\mathrm{sub}}, y_T)$ with $su\sigma \in (\Sigma \cup \Sigma_T)^*$ and $y_{\mathrm{sub}} \in Q$, and the proof continues as in *Case 2*.

Case 2: $x \in Q$. From $U_{\text{sub}} \overset{P(s)}{\Longrightarrow} x$, it follows that $A_{\text{sub}} \overset{P(s)}{\longrightarrow} x$ by Lemma 10 (i), which implies $A \overset{P(s)}{\longrightarrow} x$ because A and A_{sub} have the same transition relations, which implies $U \overset{P(s)}{\Longrightarrow} x$ again by Lemma 10 (i). Then $U \parallel T \overset{s}{\Rightarrow} (x, x_T)$, and since $U \parallel T$ is nonblocking, there exists $w \in \Sigma^*$ such that $U \parallel T \overset{s}{\Rightarrow} (x, x_T) \overset{w\omega}{\Rightarrow} (z, z_T)$. This means $x \overset{P(w)\omega}{\Longrightarrow}_U z$, which implies $x \overset{P(w)\omega}{\longrightarrow} z$ by Lemma 9, which implies $x \overset{P(w)\omega}{\longrightarrow}_{\text{sub}} z$ because A and A_{sub} have the same transition relations, which implies $x \overset{P(w)\omega}{\longrightarrow}_{U,\text{sub}} z$ again by Lemma 9. Thus, $U_{\text{sub}} \parallel T \overset{s}{\Rightarrow} (x, x_T) \overset{w\omega}{\Rightarrow}$, and since (x, x_T) was chosen arbitrarily, it follows that $U_{\text{sub}} \parallel T$ is nonblocking.

Conversely, assume that $U_{\text{sub}} \parallel T$ is nonblocking, and let $s \in (\Sigma \cup \Sigma_T)^*$ such that $U \parallel T \overset{s}{\Rightarrow} (x, x_T)$. Then $U \overset{P(s)}{\Longrightarrow} x \in Q \cup Ann$. Consider two cases.

Case 1: $x = (x_a, a) \in Ann$. By assumption there exists $a_{\text{sub}} \subseteq a$ such that $(x_a, a_{\text{sub}}) \in Ann_{\text{sub}}$. From $U \overset{P(s)}{\Longrightarrow} x = (x_a, a)$, it follows that $A \overset{P(s)}{\longrightarrow} x_a$ by Lemma 10 (ii), which implies $A_{\text{sub}} \overset{P(s)}{\longrightarrow} x_a$ because A and A_{sub} have the same transition relations. Therefore, $U_{\text{sub}} \overset{P(s)}{\Longrightarrow} x_a \overset{\tau}{\longrightarrow} (x_a, a_{\text{sub}})$ by Lemma 10 (i) and by Def. 7 (ii). Thus, $U_{\text{sub}} \parallel T \overset{s}{\Rightarrow} ((x_a, a_{\text{sub}}), x_T)$, and since $U_{\text{sub}} \parallel T$ is nonblocking, there exists $t \in \Sigma^*$ such that $U_{\text{sub}} \parallel T \overset{s}{\Rightarrow} ((x_a, a_{\text{sub}}), x_T) \overset{t\omega}{\Rightarrow}$. Write $t\omega = u\sigma v$ with $u \in (\Sigma_T \setminus \Sigma)^*$, $\sigma \in \Sigma_\omega$, and $v \in (\Sigma_\omega \cup \Sigma_T)^*$. Then $U_{\text{sub}} \parallel T \overset{s}{\Rightarrow} ((x_a, a_{\text{sub}}), x_T) \overset{u}{\Rightarrow} ((x_a, a_{\text{sub}}), x'_T) \overset{\sigma}{\rightarrow} (y_{\text{sub}}, y_T)$, i.e., $\sigma \in \text{Elig}_{U_{\text{sub}}}((x_a, a_{\text{sub}})) = a_{\text{sub}} \subseteq a = \text{Elig}_U((x_a, a))$ by Def. 7 (iv) and (v), and $(x_a, a) \overset{\sigma}{\rightarrow}_U y$ for some $y \in Q$. If $\sigma = \omega$, then clearly $U \parallel T \overset{s}{\Rightarrow} ((x_a, a), x_T) \overset{u\omega}{\Rightarrow}$, which is enough to show that $U \parallel T$ is nonblocking. Otherwise, $U \parallel T \overset{su\sigma}{\Longrightarrow} (y, y_T)$ with $su\sigma \in (\Sigma \cup \Sigma_T)^*$ and $y \in Q$, and the proof continues as in *Case 2*.

Case 2: $x \in Q$. From $U \overset{P(s)}{\Longrightarrow} x$, it follows that $A \overset{P(s)}{\longrightarrow} x$ by Lemma 10 (i), which implies $A_{\text{sub}} \overset{P(s)}{\longrightarrow} x$ because A and A_{sub} have the same transition relations, which implies $U_{\text{sub}} \overset{P(s)}{\Longrightarrow} x$ again by Lemma 10 (i). Then $U_{\text{sub}} \parallel T \overset{s}{\Rightarrow} (x, x_T)$, and since $U_{\text{sub}} \parallel T$ is nonblocking, there exists $w \in \Sigma^*$ such that $U_{\text{sub}} \parallel T \overset{s}{\Rightarrow} (x, x_T) \overset{w\omega}{\Rightarrow} (z, z_T)$. This means $x \overset{P(w)\omega}{\Longrightarrow}_{U,\text{sub}} z$, which by Lemma 9 implies $x \overset{P(w)\omega}{\longrightarrow} z$, both in A and A_{sub}, and $x \overset{P(w)\omega}{\longrightarrow}_U z$. Thus, $U \parallel T \overset{s}{\Rightarrow} (x, x_T) \overset{w\omega}{\Rightarrow}$, and since (x, x_T) was chosen arbitrarily, it follows that $U \parallel T$ is nonblocking. □

4.4 Incoming Equivalence

To prove the correctness of abstractions based on automaton quotients, such as the incoming equivalence abstraction, the relationship between the traces in an automaton A and its quotient $A/\!\sim$ needs to be established. It is well-known that every trace in A also has a corresponding trace in $A/\!\sim$. The following Lemma 12 is quoted from [9] and holds for every equivalence relation. Conversely, not every path in a quotient automaton exists in the original automaton, but Lemma 13 shows how such a path can be obtained if the quotient is constructed using incoming equivalence.

Lemma 12 Let $A = \langle \Sigma, Q, \rightarrow, Q^\circ, Ann \rangle$ be an annotated automaton, and let $\sim \subseteq Q \times Q$ be an equivalence relation. Then, for all states $x, y \in Q$ and all traces $s \in \Sigma^*$ such that $x \overset{s}{\rightarrow} y$ in A, it holds that $[x] \overset{s}{\rightarrow} [y]$ in $A/\!\sim$.

Proof Let $x \overset{s}{\rightarrow} y$ in A with $s = \sigma_1 \ldots \sigma_n$. Then there exist states $x_0, \ldots, x_n \in Q$ such that

$$x = x_0 \overset{\sigma_1}{\rightarrow} x_1 \overset{\sigma_2}{\rightarrow} \cdots \overset{\sigma_n}{\rightarrow} x_n = y \,. \tag{14}$$

By Def. 10, it holds that $[x_{k-1}] \xrightarrow{\sigma_k} [x_k]$ for each $k = 1, \ldots, n$, which implies $[x] \xrightarrow{s} [y]$ in $A/\!\sim$. □

Lemma 13 Let $A = \langle \Sigma, Q, \rightarrow, Q^\circ, Ann \rangle$ be an annotated automaton, and let $\tilde{x}, \tilde{z} \in Q/\!\sim_{\mathrm{inc}}$ be two states of $A/\!\sim_{\mathrm{inc}}$.

(i) For all $s \in \Sigma^*$ and all $\sigma \in \Sigma$ such that $\tilde{x} \xrightarrow{s\sigma} \tilde{z}$, there exists $x \in \tilde{x}$ such that for all $z' \in \tilde{z}$ it holds that $x \xrightarrow{s\sigma} z'$.
(ii) For all $s \in \Sigma^*$ such that $A/\!\sim_{\mathrm{inc}} \xrightarrow{s} \tilde{z}$ and for all $z' \in \tilde{z}$, it holds that $A \xrightarrow{s} z'$.

Proof (i) The claim is proved by induction on $|s|$.

Base case: $s = \varepsilon$. As $\tilde{x} \xrightarrow{\sigma} \tilde{z}$, there must exist $x \in \tilde{x}$ and $z \in \tilde{z}$ such that $x \xrightarrow{\sigma} z$. Let $z' \in \tilde{z}$. Then $z \sim_{\mathrm{inc}} z'$, and it follows from Def. 9 that $x \xrightarrow{\sigma} z'$.

Inductive step: $s = t\sigma$. Assume that $\tilde{x} \xrightarrow{t} \tilde{y} \xrightarrow{\sigma} \tilde{z}$. Then there are states $y \in \tilde{y}$ and $z \in \tilde{z}$ such that $y \xrightarrow{\sigma} z$. By inductive assumption, there exists a state $x \in \tilde{x}$ such that $x \xrightarrow{t} y$. Let $z' \in \tilde{z}$. Then $z \sim_{\mathrm{inc}} z'$, and it follows from Def. 9 that $x \xrightarrow{t} y \xrightarrow{\sigma} z'$.

(ii) Let $\tilde{Q}^\circ = \{\, [x^\circ] \mid x^\circ \in Q^\circ \,\}$ be the set of initial states of $A/\!\sim_{\mathrm{inc}}$.

If $s = \varepsilon$, then $\tilde{z} \in \tilde{Q}^\circ$ and thus $\tilde{z} = [x^\circ]$ for some $x^\circ \in Q^\circ$, which implies $x^\circ \in \tilde{z}$. Let $z' \in \tilde{z}$. Then $x^\circ \sim_{\mathrm{inc}} z'$, which implies $z' \in Q^\circ$ by Def. 9 and thus $A \xrightarrow{\varepsilon} z'$.

Otherwise $s = t\sigma$ for some $t \in \Sigma^*$ and $\sigma \in \Sigma$, and there exists $\tilde{x} \in \tilde{Q}^\circ$ such that $\tilde{x} \xrightarrow{t\sigma} \tilde{z}$. Let $z' \in \tilde{z}$. It follows from (i) that there exists $x \in \tilde{x}$ such that $x \xrightarrow{t\sigma} z'$. Since $\tilde{x} \in \tilde{Q}^\circ$, there exists $x^\circ \in \tilde{x}$ such that $x^\circ \in Q^\circ$. Then $x^\circ \sim_{\mathrm{inc}} x$ implies $x \in Q^\circ$ and thus $A \xrightarrow{t\sigma} z'$. □

Using the above two lemmas and the properties of the paths of unannotated forms established in Section 4.2, the proof of Prop. 5 proceeds using similar ideas to that of the *Active Events Rule* [9].

Proposition 5 Let $A = \langle \Sigma, Q, \rightarrow, Q^\circ, Ann \rangle$ be an annotated automaton. Then $A \simeq_{\mathrm{conf}} A/\!\sim_{\mathrm{inc}}$.

Proof Let $U = \langle \Sigma, Q \cup Ann, \rightarrow_U, Q^\circ \rangle$ and $\tilde{U} = \langle \Sigma, Q/\!\sim_{\mathrm{inc}} \cup \tilde{A}nn, \rightarrow_{\tilde{U}}, \tilde{Q}^\circ \rangle$ be unannotated forms of A and $\tilde{A} = A/\!\sim_{\mathrm{inc}}$, respectively. It is to be shown that $U \simeq_{\mathrm{conf}} \tilde{U}$. Therefore, let $T = \langle \Sigma_T, Q_T, \rightarrow_T, Q_T^\circ \rangle$ be an arbitrary automaton.

First, assume that $U \parallel T$ is nonblocking, and let $s \in (\Sigma \cup \Sigma_T)^*$ such that $\tilde{U} \parallel T \overset{s}{\Rightarrow} (\tilde{x}, x_T)$. Then $\tilde{U} \overset{P(s)}{\Longrightarrow} \tilde{x} \in Q/\!\sim_{\mathrm{inc}} \cup \tilde{A}nn$. Consider two cases.

Case 1: $\tilde{x} = (\tilde{x}_a, a) \in \tilde{A}nn$. Then there exists $x_a \in \tilde{x}_a$ such that $(x_a, a) \in Ann$. From $\tilde{U} \overset{P(s)}{\Longrightarrow} \tilde{x} = (\tilde{x}_a, a)$, it follows that $\tilde{A} \xrightarrow{P(s)} \tilde{x}_a$ by Lemma 10 (ii), which implies $A \xrightarrow{P(s)} x_a$ by Lemma 13 (ii), and $U \overset{P(s)}{\Longrightarrow} (x_a, a)$ again by Lemma 10 (ii). Thus, $U \parallel T \overset{s}{\Rightarrow} ((x_a, a), x_T)$, and since $U \parallel T$ is nonblocking, there exists $t \in \Sigma^*$ such that $U \parallel T \overset{s}{\Rightarrow} ((x_a, a), x_T) \overset{t\omega}{\Rightarrow}$. Write $t\omega = u\sigma v$ with $u \in (\Sigma_T \setminus \Sigma)^*$, $\sigma \in \Sigma_\omega$, and $v \in (\Sigma_\omega \cup \Sigma_T)^*$. Then $U \parallel T \overset{s}{\Rightarrow} ((x_a, a), x_T) \overset{u}{\Rightarrow} ((x_a, a), x'_T) \xrightarrow{\sigma} (y, y_T)$, i.e., $\sigma \in \mathrm{Elig}_U((x_a, a)) = a = \mathrm{Elig}_{\tilde{U}}((\tilde{x}_a, a))$ by Def. 7 (iv) and (v), and $(\tilde{x}_a, a) \xrightarrow{\sigma}_{\tilde{U}} \tilde{y}$ for some $\tilde{y} \in Q/\!\sim_{\mathrm{inc}}$. If $\sigma = \omega$, then clearly $\tilde{U} \parallel T \overset{s}{\Rightarrow} ((\tilde{x}_a, a), x_T) \overset{u}{\Rightarrow} ((\tilde{x}_a, a), x'_T) \overset{\omega}{\Rightarrow}$, which is enough to show that $\tilde{U} \parallel T$ is nonblocking. Otherwise, $\tilde{U} \parallel T \overset{s}{\Rightarrow} ((\tilde{x}_a, a), x_T) \overset{u}{\Rightarrow} ((\tilde{x}_a, a), x'_T) \xrightarrow{\sigma} (\tilde{y}, y_T)$ with $su\sigma \in (\Sigma \cup \Sigma_T)^*$ and $\tilde{y} \in Q/\!\sim_{\mathrm{inc}}$, and the proof continues as in *Case 2*.

Case 2: $\tilde{x} \in Q/\!\sim_{\mathrm{inc}}$. Then $\tilde{A} \xrightarrow{P(s)} \tilde{x}$ by Lemma 10 (i). Then let $x \in \tilde{x}$, and it follows from Lemma 13 (ii) that $A \xrightarrow{P(s)} x$, which implies $U \overset{P(s)}{\Longrightarrow} x$ again by Lemma 10 (i). Thus,

$U \parallel T \overset{P(s)}{\Longrightarrow} (x,x_T)$, and since $U \parallel T$ is nonblocking, there exists $w \in \Sigma^*$ such that $U \parallel T \overset{P(s)}{\Longrightarrow} (x,x_T) \overset{w\omega}{\Rightarrow} (z,z_T)$. Then $x \overset{P(w)\omega}{\Longrightarrow}_U z$, with $z \in Q$ by Def. 7. This implies $x \overset{P(w)\omega}{\longrightarrow} z$ by Lemma 9, and thus $[x] \overset{P(w)\omega}{\longrightarrow} [z]$ in $A/\sim_{\text{inc}}$ by Lemma 12, which implies $\tilde{x} = [x] \overset{P(w)\omega}{\Longrightarrow}_{\tilde{U}} [z]$ again by Lemma 9. Thus, $\tilde{U} \parallel T \overset{s}{\Rightarrow} (\tilde{x},x_T) \overset{w\omega}{\Rightarrow}$, and since $(\tilde{x},x_T)$ was chosen arbitrarily, it follows that $\tilde{U} \parallel T$ is nonblocking.

Conversely, assume that $\tilde{U} \parallel T$ is nonblocking, and let $s \in (\Sigma \cup \Sigma_T)^*$ such that $U \parallel T \overset{s}{\Rightarrow} (x,x_T)$. Then $U \overset{P(s)}{\Longrightarrow} x \in Q \cup Ann$. Consider two cases.

Case 1: $x = (x_a,a) \in Ann$. From $U \overset{P(s)}{\Longrightarrow} (x_a,a)$, it follows that $A \overset{P(s)}{\longrightarrow} x_a$ by Lemma 10 (ii), which implies $\tilde{A} \overset{P(s)}{\longrightarrow} [x_a]$ by Lemma 12. Note that $([x_a],a) \in \tilde{A}nn$ and thus $\tilde{U} \overset{P(s)}{\Longrightarrow} ([x_a],a)$ again by Lemma 10 (ii). Thus, $\tilde{U} \parallel T \overset{s}{\Rightarrow} (([x_a],a),x_T)$, and since $\tilde{U} \parallel T$ is nonblocking, there exists $t \in \Sigma^*$ such that $\tilde{U} \parallel T \overset{s}{\Rightarrow} (([x_a],a),x_T) \overset{t\omega}{\Rightarrow}$. Write $t\omega = u\sigma v$ with $u \in (\Sigma_T \setminus \Sigma)^*$, $\sigma \in \Sigma_\omega$, and $v \in (\Sigma_\omega \cup \Sigma_T)^*$. Then $\tilde{U} \parallel T \overset{s}{\Rightarrow} (([x_a],a),x_T) \overset{u}{\Rightarrow} (([x_a],a),x'_T) \overset{\sigma}{\rightarrow} (\tilde{y},y_T)$, i.e., $\sigma \in \text{Elig}_{\tilde{U}}(([x_a],a)) = a = \text{Elig}_U((x_a,a))$ by Def. 7 (iv) and (v), and $(x_a,a) \overset{\sigma}{\rightarrow}_U y$ for some $y \in Q$. Thus $U \parallel T \overset{s}{\Rightarrow} ((x_a,a),x_T) \overset{u}{\Rightarrow} ((x_a,a),x'_T) \overset{\sigma}{\rightarrow} (y,y_T)$ with $y \in Q$. If $\sigma = \omega$, then clearly $U \parallel T \overset{s}{\Rightarrow} ((x_a,a),x_T) \overset{u\omega}{\Rightarrow}$, which is enough to show that $U \parallel T$ is nonblocking. Otherwise, $U \parallel T \overset{su\sigma}{\Longrightarrow} (y,y_T)$ with $su\sigma \in (\Sigma \cup \Sigma_T)^*$ and $y \in Q$, and the proof continues as in *Case 2*.

Case 2: $x \in Q$. Then $A \overset{P(s)}{\longrightarrow} x$ by Lemma 10 (i), which implies $\tilde{A} \overset{P(s)}{\longrightarrow} [x]$ by Lemma 12. By Def. 5, there exists $a \subseteq \text{Elig}_A(x)$ such that $(x,a) \in Ann$. Then $([x],a) \in \tilde{A}nn$, and $\tilde{U} \overset{P(s)}{\Longrightarrow} ([x],a)$ by Lemma 10 (ii). Thus, $\tilde{U} \parallel T \overset{s}{\Rightarrow} (([x],a),x_T)$, and since $\tilde{U} \parallel T$ is nonblocking, there exists $t \in \Sigma^*$ such that $\tilde{U} \parallel T \overset{s}{\Rightarrow} (([x],a),x_T) \overset{t\omega}{\Rightarrow}$. Write $t\omega = u\sigma v$ with $u \in (\Sigma_T \setminus \Sigma)^*$, $\sigma \in \Sigma_\omega$, and $v \in (\Sigma_\omega \cup \Sigma_T)^*$. Then $\tilde{U} \parallel T \overset{s}{\Rightarrow} (([x],a),x_T) \overset{u}{\Rightarrow} (([x],a),x'_T) \overset{\sigma}{\rightarrow} (\tilde{y},y_T)$. Clearly, $\sigma \in \text{Elig}_{\tilde{U}}(([x],a)) = a \subseteq \text{Elig}_A(x) = \text{Elig}_U(x)$ by Def. 7 (iii) and (v). If $\sigma = \omega$, it already follows that $U \parallel T \overset{s}{\Rightarrow} (x,x_T) \overset{u\omega}{\Rightarrow}$, i.e., $U \parallel T$ is nonblocking. Otherwise $\sigma \in \text{Elig}_A(x)$ means that $x \overset{\sigma}{\rightarrow} y$ for some $y \in Q$. Then $\tilde{A} \overset{P(s)}{\longrightarrow} [x] \overset{\sigma}{\rightarrow} [y]$ by Def. 10 and $\tilde{U} \overset{P(s)\sigma}{\Longrightarrow} [y]$ by Lemma 10 (i). Therefore $\tilde{U} \parallel T \overset{su\sigma}{\Longrightarrow} ([y],y_T)$, and since $\tilde{U} \parallel T$ is nonblocking, there exists $w \in \Sigma^*$ such that $\tilde{U} \parallel T \overset{su\sigma}{\Longrightarrow} ([y],y_T) \overset{w\omega}{\Rightarrow}$. Then $[y] \overset{P(w)\omega}{\Longrightarrow}_{\tilde{U}}$, and by Lemma 13 (i) there exists $y' \in [y]$ such that $y' \overset{P(w)\omega}{\Longrightarrow}_U$. Thus $x \overset{\sigma}{\rightarrow} y \sim_{\text{inc}} y'$, which implies $x \overset{\sigma}{\rightarrow} y'$ by Def. 9, and $x \overset{\sigma}{\rightarrow}_U y'$ by Def. 7 (iii). Thus, $U \parallel T \overset{s}{\Rightarrow} (x,x_T) \overset{u\sigma}{\Rightarrow} (y',y_T) \overset{w\omega}{\Rightarrow}$, and since (x,x_T) was chosen arbitrarily, it follows that $U \parallel T$ is nonblocking. □

4.5 Bisimulation

This section contains the proof of Prop. 6 introduced in 3.5, which states that conflict equivalence is preserved under bisimulation of annotated automata. This is best proved by showing that the unannotated forms of bisimilar annotated automata are bisimilar. For this purpose, the following standard definition of bisimulation for ordinary automata is used [13].

Definition 12 Let $G_1 = \langle \Sigma, Q_1, \rightarrow_1, Q_1^\circ \rangle$ and $G_2 = \langle \Sigma, Q_2, \rightarrow_2, Q_2^\circ \rangle$ be two automata. A relation $\approx \subseteq Q_1 \times Q_2$ is called a *bisimulation* between G_1 and G_2, if the following conditions hold for all states $x_1 \in Q_1$ and $x_2 \in Q_2$ such that $x_1 \approx x_2$.

(i) For all $\sigma \in \Sigma_{\tau,\omega}$, if $x_1 \overset{\sigma}{\rightarrow} y_1$ then there exists $y_2 \in Q_2$ such that $y_1 \approx y_2$ and $x_2 \overset{\sigma}{\rightarrow} y_2$.

(ii) For all $\sigma \in \Sigma_{\tau,\omega}$, if $x_2 \xrightarrow{\sigma} y_2$ then there exists $y_1 \in Q_1$ such that $y_1 \approx y_2$ and $x_1 \xrightarrow{\sigma} y_1$.

G_1 and G_2 are *bisimulation equivalent* or *bisimilar*, written $G_1 \approx G_2$, if there exists a bisimulation $\approx$ between G_1 and G_2 such that, for every initial state $x_1^\circ \in Q_1^\circ$ there exists an initial state $x_2^\circ \in Q_2^\circ$ such that $x_1^\circ \approx x_2^\circ$, and vice versa.

Although unannotated forms have been shown to be unique up to conflict equivalence in Prop. 2, two unannotated forms of the same annotated automaton are not necessarily bisimilar. To prove the result about bisimulation, a unique unannotated form is needed.

Definition 13 Let $A = \langle \Sigma, Q, \rightarrow, Q^\circ, Ann \rangle$ be an annotated automaton. The *standard unannotation* of A is $\mathscr{U}(A) = \langle \Sigma, Q_U, \rightarrow_U, Q^\circ \rangle$ where $Q_U = Q \cup Ann$ and

$$\rightarrow_U = \rightarrow \cup \{ (x, \tau, (x,a)) \in Q \times \{\tau\} \times Ann \} \cup$$
$$\{ ((x,a), \sigma, y) \in Ann \times \Sigma_\omega \times Q \mid \sigma \in a \text{ and } x \xrightarrow{\sigma} y \} \qquad (15)$$

The standard unannotation resolves the ambiguity in points (iv) and (v) of Def. 7 by simply including all possible transitions for every annotation state. This ensures uniqueness at the expense of minimality. It is easy to confirm that, for every annotated automaton A, the standard unannotation $\mathscr{U}(A)$ is indeed an unannotated form of A.

The standard unannotations of bisimilar automata can be shown to be bisimilar, and this is enough to complete the proof of Prop. 6.

Lemma 14 Let $A_1 = \langle \Sigma, Q_1, \rightarrow_1, Q_1^\circ, Ann_1 \rangle$ and $A_2 = \langle \Sigma, Q_2, \rightarrow_2, Q_2^\circ, Ann_2 \rangle$ be two annotated automata such that $A_1 \approx A_2$. Then $\mathscr{U}(A_1) \approx \mathscr{U}(A_2)$.

Proof Let $\mathscr{U}(A_i) = \langle \Sigma, Q_{U,i}, \rightarrow_{U,i}, Q_i^\circ \rangle$ where $Q_{U,i} = Q_i \cup Ann_i$ for $i = 1,2$, and let $\approx$ be a bisimulation between A_1 and A_2. Consider the relation $\approx_U \subseteq Q_{U,1} \times Q_{U,2}$ such that $x_1 \approx_U x_2$ if and only if one of the following two conditions holds:

$$x_1 \in Q_1,\ x_2 \in Q_2, \text{ and } x_1 \approx x_2 \qquad \text{or} \qquad (16)$$
$$\text{there exists } a \subseteq \Sigma_\omega \text{ such that } x_1 = (x_1', a) \in Ann_1,\ x_2 = (x_2', a) \in Ann_2, \text{ and } x_1' \approx x_2'\ . \qquad (17)$$

It is to be shown that $\approx_U$ is a bisimulation between $\mathscr{U}(A_1)$ and $\mathscr{U}(A_2)$. To see (i) in Def. 12, let $x_1 \approx_U x_2$ and $x_1 \xrightarrow{\sigma}_{U,1} y_1$ for some $\sigma \in \Sigma_{\tau,\omega}$. Then either (16) or (17) holds.

If (16) holds, then $x_1 \approx x_2$ with $x_1 \in Q_1$ and $x_2 \in Q_2$. Then either $y_1 \in Q_1$ or $y_1 \in Ann_1$. If $y_1 \in Q_1$, then it follows from $x_1 \xrightarrow{\sigma}_{U,1} y_1$ that $x_1 \xrightarrow{\sigma}_1 y_1$ by Def. 13. Since $x_1 \approx x_2$, by Def. 11 there exists $y_2 \in Q_2$ such that $x_2 \xrightarrow{\sigma}_2 y_2$ and $y_1 \approx y_2$. Again by Def. 13, this implies $x_2 \xrightarrow{\sigma}_{U,2} y_2$, and $y_1 \approx_U y_2$ according to (16). If on the other hand $y_1 \in Ann_1$, then $\sigma = \tau$ and $y_1 = (x_1, a)$ for some $a \subseteq \Sigma_\omega$ by Def. 7. Since $x_1 \approx x_2$ and $(x_1, a) = y_1 \in Ann_1$, it follows from Def. 11 that $(x_2, a) \in Ann_2$. Then $x_2 \xrightarrow{\tau}_{U,2} (x_2, a)$ by Def. 13 and $y_1 = (x_1, a) \approx_U (x_2, a)$ by (17).

If (17) holds, then $x_1 = (x_1', a) \in Ann_1$ and $x_2 = (x_2', a) \in Ann_2$ for some $a \subseteq \Sigma_\omega$, and $x_1' \approx x_2'$. Then it follows from $(x_1', a) \xrightarrow{\sigma}_{U,1} y_1$ by Def. 13 that $\sigma \in a$, $y_1 \in Q_1$, and $x_1' \xrightarrow{\sigma}_1 y_1$. Since $x_1' \approx x_2'$, there exists $y_2 \in Q_2$ such that $x_2' \xrightarrow{\sigma}_2 y_2$ and $y_1 \approx y_2$. Then $(x_2', a) \xrightarrow{\sigma}_{U,2} y_2$ by Def. 13 since $\sigma \in a$, and $y_1 \approx_U y_2$ by (16) since $y_1 \approx y_2$.

This shows (i) in Def. 12. The proof of (ii) is symmetric, and the condition on the initial states follows since $A_1 \approx A_2$ and A_i and $\mathscr{U}(A_i)$ have the same initial states. □

Proposition 6 Let A_1 and A_2 be annotated automata such that $A_1 \approx A_2$. Then $A_1 \simeq_{\text{conf}} A_2$.

Proof Let U_1 be an unannotated form of A_1, and let U_2 be an unannotated form of A_2. Then $U_1 \simeq_{\text{conf}} \mathscr{U}(A_1) \approx \mathscr{U}(A_2) \simeq_{\text{conf}} U_2$ by Prop. 2 and Lemma 14. The claim follows from results in [12], according to which bisimilar automata are conflict equivalent. □

5 Experimental Results

A conflict checker using annotated automata has been implemented in the DES software tool Supremica [1] and tested on the same set of industrial-scale and parametrised models as used previously in [9]. All these problems have been solved successfully, and the results are shown in Table 1.

After simplifying each individual component in a composed system such as (3), the algorithm selects a *candidate* set of automata for composition using strategies described in [9]. After synchronous composition and hiding of local events, the result is first simplified using observation equivalence and by removing obvious certain conflicts [9]. Then the annotated form is constructed and simplified using incoming equivalence and bisimulation. Subsumption is used during each of these steps. Finally, an unannotated form is obtained and further simplified by removing states with only silent outgoing transitions.

The *Annotating Method* described above has been compared to the *Heuristic Method* described in [9]. The heuristic compositional conflict checker of [9] selects and composes candidate sets of automata in the same way as the annotating method, but it uses a more straightforward set of abstraction rules to simplify automata. In addition to the *Certain Conflicts Rule* and observation equivalence, which are part of the preprocessing steps in the Annotating Method, the Heuristic Method also uses the *Active Events Rule*, the *Silent Continuation Rule*, the *Only Silent Incoming Rule*, and the *Only Silent Outgoing Rule* [9]. All these rules are directly applied to the transitions of an automaton, without computing an annotated form. This makes the rules simpler to apply, but they also have somewhat weaker abstraction potential, as it can be shown that all abstractions obtained using the above mentioned rules and more can in principle be achieved by simplifying an annotated automaton.

To make the Annotating and Heuristic Method comparable, they have been modified to ensure that both implementations select and compose the same automata in the same order, regardless of possible differences in the intermediate results. This is done to compare the effects of the different simplification methods, as opposed to comparing different choices of automata for composition (which often lead to dramatic changes). However, the chosen order of composition is no longer optimal, which explains the difference between the results in Table 1 and [9].

Table 1 shows the experimental results for nonblocking verification of 14 large models of industrial-scale applications and 9 very large parametrised models. Please refer to [9] for a more detailed description of the models. The table shows the number of reachable states of the synchronous product of each model (Size), and the number of states of the largest automaton encountered during compositional verification (Peak States), the cumulative number of states constructed during verification (Total States), and the total verification time in seconds, for both the Annotating Method and the Heuristic Method,

All experiments were run on a standard laptop computer with a 2 GHz microprocessor and 4 GB of RAM, and controlled by state limits. If during abstraction some synchronous product has more than 10,000 states, its construction is aborted and another set of automata is composed instead. If no suitable set of automata for composition can be identified, a final attempt is made to construct and check the full synchronous product of all remaining automata whether it is nonblocking. If this attempt runs out of memory, the run is aborted and the corresponding table entries are left blank.

The annotating conflict checker performs much better than the heuristic method for the parametrised dining philosophers and tree arbiter problems, which cannot be solved by the heuristic method using the given state limits and candidate selection strategy. For the industrial applications, the two methods yield similar results, with the Annotating Method

Table 1 Experimental results

		Annotating			Heuristic		
	Size	Peak States	Total States	Time [s]	Peak States	Total States	Time [s]
AGV	$2.6{\cdot}10^{7}$	10552	18054	28.1	1368	4097	4.1
AGVb	$2.3{\cdot}10^{7}$	975	1719	0.2	781	1524	0.1
verriegel3	$9.7{\cdot}10^{8}$	2346	12767	4.7	2856	14639	6.8
verriegel3b	$1.3{\cdot}10^{9}$	2346	11028	4.8	2537	11976	6.3
verriegel4	$4.5{\cdot}10^{10}$	3703	15286	5.4	2671	15106	6.1
verriegel4b	$6.3{\cdot}10^{10}$	2346	11827	4.6	2537	12968	6.3
big_bmw	$3.1{\cdot}10^{7}$	63	342	0.1	63	347	0.1
FMS	812544	86	206	0.0	125	279	0.1
SMS	312	18	119	0.0	18	120	0.0
PMS	$5.7{\cdot}10^{8}$	75	487	0.1	75	492	0.2
IPC	20592	107	195	0.0	107	195	0.1
ftechnik	$1.2{\cdot}10^{8}$	5631	21218	5.9	2450	15524	4.8
rhone_tough	$1.0{\cdot}10^{10}$	1584	5025	4.1	1584	5026	4.5
AIP	$1.0{\cdot}10^{9}$	6864	82542	30.3	6868	77512	24.7
256philo	$5.4{\cdot}10^{168}$	628	77419	21.8			
512philo	$2.9{\cdot}10^{337}$	628	156395	48.1			
1024philo	$8.5{\cdot}10^{674}$	628	314347	96.1			
128transfer	$1.6{\cdot}10^{231}$	43	11115	3.9	42	10966	10.7
256transfer	$2.4{\cdot}10^{462}$	43	22251	10.7	42	21974	9.3
512transfer	$5.8{\cdot}10^{924}$	43	44523	42.6	42	43990	34.7
128arbiter	$2.8{\cdot}10^{112}$	55	14669	10.4			
256arbiter	$5.4{\cdot}10^{224}$	55	29517	31.5			
512arbiter	$2.1{\cdot}10^{449}$	55	59213	58.1			

producing a smaller peak number of states in 5 cases, and the Heuristic Method producing a smaller peak number of states in 4 cases. The difference is particularly notable for the *AGV* and *ftechnik* models, where the annotating method results in larger automata. This seems to be caused by the annotating and unannotating steps, which may change the structure of an automaton in such a way that certain states are no longer observation equivalent. The more regular parametrised examples do not suffer from this issue, and the Annotating Method works better here.

Table 2 shows some information on the effectiveness of the individual steps taken by the annotating method. First, it shows for each model the total number of annotations created and removed by subsumption. Next, it shows the total number of states removed as unreachable after annotation (Ann.), the number of states removed by merging incoming equivalent ($\sim_{\text{inc}}$) and bisimilar ($\approx$) states, and the number of states added back in when constructing unannotated forms (Unann.). Note that $\approx$ refers to simplification of annotated automata and is in addition to observation equivalence simplification, which is performed on all automata before annotating.

In most cases, annotating helps to remove substantially more states than need to be added back during unannotation. The data clearly shows the importance of the subsumption step, which is performed directly while constructing the annotated form. While merging incoming equivalent and bisimilar states seems to have a limited effect for most industrial models, it has a marked effect for some of the more regular models in the dining philosophers and arbiter series.

Table 2 Rule Usage

	Annotations		**States**			
	Create	Subsume	Ann.	$\sim_{\text{inc}}$	$\approx$	Unann.
AGV	+63435	−58073	−1777	−34	−513	+5
AGVb	+328	−226	−0	−0	−0	+0
verriegel3	+3442	−759	−93	−7	−16	+37
verriegel3b	+3478	−777	−70	−1	−16	+19
verriegel4	+3875	−927	−93	−13	−32	+29
verriegel4b	+4578	−1540	−122	−1	−67	+42
big_bmw	+53	−27	−1	−0	−0	+1
FMS	+77	−26	−24	−0	−8	+11
SMS	+8	−8	−0	−0	−0	+0
PMS	+161	−103	−17	−9	−9	+7
IPC	+133	−58	−9	−0	−2	+4
ftechnik	+4785	−856	−26	−0	−0	+1
rhone_tough	+899	−491	−15	−0	−6	+13
AIP	+17303	−6644	−1600	−597	−216	+1054
256philo	+86128	−33106	−1756	−874	−9635	+0
512philo	+174192	−67133	−3548	−1770	−19491	+0
1024philo	+350320	−133683	−7132	−3562	−39203	+0
128transfer	+3721	−1289	−129	−0	−0	+1
256transfer	+7433	−2569	−257	−0	−0	+1
512transfer	+14857	−5129	−513	−0	−0	+1
128arbiter	+5475	−2769	−1002	−436	−61	+61
256arbiter	+11043	−5585	−2026	−884	−125	+125
512arbiter	+22179	−11217	−4074	−1780	−253	+253

6 Conclusions

This paper shows how *annotations* can be used for compositional nonblocking verification. Methods to construct annotated automata and to compute abstractions are presented, and their correctness is proved formally. Experimental results show that the performance of nonblocking verification using annotations is comparable to existing methods of simplifying automata with respect to conflict equivalence.

In addition, annotations lead to an improved structure and more regular nondeterministic automata, and help to better understand the nature and possibilities of conflict-preserving abstractions. So far, three simplification rules for annotated automata have been implemented, and it is already known that the framework allows for other more powerful ways of conflict-preserving abstraction.

In the future, the authors would like to the investigate identification of implicit transitions with respect to conflict equivalence and their selective introduction to aid bisimulation reduction of annotated automata. Another topic of future work is the investigation of alternatives to the unannotation procedure, to avoid the construction of additional states by verifying nonblocking using only annotated automata.

References

1. Åkesson, K., Fabian, M., Flordal, H., Malik, R.: Supremica—an integrated environment for verification, synthesis and simulation of discrete event systems. In: Proc. 8th Int. Workshop on Discrete Event

Systems, WODES '06, pp. 384–385. Ann Arbor, MI, USA (2006)
2. Cassandras, C.G., Lafortune, S.: Introduction to Discrete Event Systems. Kluwer (1999)
3. Clarke Jr., E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press (1999)
4. De Nicola, R., Hennessy, M.C.B.: Testing equivalences for processes. Theoretical Comput. Sci. **34**(1–2), 83–133 (1984). DOI 10.1016/0304-3975(84)90113-0
5. Eloranta, J.: Minimizing the number of transitions with respect to observation equivalence. BIT **31**(4), 397–419 (1991)
6. Feng, L., Wonham, W.M.: Supervisory control architecture for discrete-event systems. IEEE Trans. Autom. Control **53**(6), 1449–1461 (2008)
7. Fernandez, J.C.: An implementation of an efficient algorithm for bisimulation equivalence. Science of Computer Programming **13**, 219–236 (1990)
8. Flordal, H., Malik, R.: Modular nonblocking verification using conflict equivalence. In: Proc. 8th Int. Workshop on Discrete Event Systems, WODES '06, pp. 100–106. Ann Arbor, MI, USA (2006)
9. Flordal, H., Malik, R.: Compositional verification in supervisory control. SIAM J. Control and Optimization **48**(3), 1914–1938 (2009). DOI 10.1137/070695526
10. Hoare, C.A.R.: Communicating Sequential Processes. Prentice-Hall (1985)
11. Kumar, R., Shayman, M.A.: Non-blocking supervisory control of nondeterministic discrete event systems. In: Proc. American Control Conf., pp. 1089–1093. Baltimore, MD, USA (1994)
12. Malik, R., Streader, D., Reeves, S.: Conflicts and fair testing. Int. J. Found. Comput. Sci. **17**(4), 797–813 (2006)
13. Milner, R.: Communication and concurrency. Series in Computer Science. Prentice-Hall (1989)
14. Nuutila, E.: Efficient transitive closure compuation in large digraphs. Ph.D. thesis, Laboratory of Information Processing Science, Helsinki University of Technology, Finland (1995)
15. Olderog, E.R., Hoare, C.A.R.: Specification-oriented semantics for communicating processes. Acta Inf. **23**(1), 9–66 (1986)
16. Pena, P.N., Cury, J.E.R., Lafortune, S.: Verification of nonconflict of supervisors using abstractions. IEEE Trans. Autom. Control **54**(12), 2803–2815 (2009)
17. Ramadge, P.J.G., Wonham, W.M.: The control of discrete event systems. Proc. IEEE **77**(1), 81–98 (1989)
18. Rensink, A., Vogler, W.: Fair testing. Information and Computation **205**(2), 125–198 (2007). DOI 10.1016/j.ic.2006.06.002
19. Su, R., van Schuppen, J.H., Rooda, J.E., Hofkamp, A.T.: Nonconflict check by using sequential automaton abstractions based on weak observation equivalence. Automatica **46**(6), 968–978 (2010). DOI 10.1016/j.automatica.2010.02.025
20. Ware, S., Malik, R.: Compositional nonblocking verification using annotated automata. In: Proc. 10th Int. Workshop on Discrete Event Systems, WODES '10, pp. 374–379. Berlin, Germany (2010)
21. Ware, S., Malik, R.: A state-based characterisation of the conflict preorder. In: Proc. 10th Int. Workshop on the Foundations of Coordination Languages and Software Architectures, FOCLASA 2011, pp. 34–48. Aachen, Germany (2011). DOI 10.4204/EPTCS.58.3