Nonconflict check by using sequential automaton abstractions

Systems Engineering Group
Department of Mechanical Engineering
Eindhoven University of Technology
PO Box 513
5600 MB Eindhoven
The Netherlands
http://se.wtb.tue.nl/

SE Report: Nr. 2008-10

Nonconflict Check by Using Sequential Automaton Abstractions

Rong Su, Jan H. van Schuppen, Jacobus E. Rooda
Albert T. Hofkamp

ISSN: 1872-1567

SE Report: Nr. 2008-10
Eindhoven, October 2008
SE Reports are available via http://se.wtb.tue.nl/sereports

Abstract

In Ramadge-Wonham supervisory control theory we often need to check nonconflict of plants and corresponding synthesized supervisors. For a large system such a check imposes a great computational challenge because of the complexity incurred by composition of plants and supervisors. In this paper we present a novel procedure based on automaton abstractions, which removes internal transitions of relevant automata at each step, allowing the nonconflict check to be performed over relatively small automata, even though the original system can be fairly large.

1 Introduction

Since Ramadge-Wonham supervisory control theory was proposed [1, 2] in 80's, significant improvement and extensions have been made. The Ramadge-Wonham paradigm relies on synchronous product to compose local components and specifications together, upon which a standard supervisor synthesis procedure (i.e. achieving controllability, observability and nonblockingness) is performed. Unfortunately, the computational complexity of synchronous product is exponentially high with respect to the number of components and their individual sizes in terms of numbers of their states. To overcome this complexity issue, many new synthesis approaches have been developed. For example, in [3] the authors propose the concept of *modularity*, which is then extended to the concept of *local modularity* in [4]. When local supervisors are (locally) modular, a globally nonblocking supervisor becomes a product of local supervisors achievable through local synthesis. In [6] [20] [9] new modular synthesis approaches are proposed, which perform model abstractions first, upon which the standard synthesis approach is applied. In [5] the authors present a hierarchical interface-based approach, which, by imposing a specific interface invariance, decouples a large system into several independent local modules, and supervisor synthesis can be performed on each local module whose size is usually much smaller than the overall system.

When applying those new techniques, particularly modular approaches, we often need to check whether a collection of finite-state automata are nonconflicting with each other. Such a check is necessary for at least three reasons. First, a synthesis approach may require it, e.g. in [3] [4] (local) modularity needs to be tested. Second, during synthesis if we can locate conflicting modules directly, then it is much more efficient to compute appropriate coordinators, as typically seen in, e.g. [6] [20] [9]. Finally, even though theoretically a synthesis approach can guarantee a supervisor which is nonconflicting with a plant, practically we still need to test it because a synthesis program may contain coding errors causing incorrect results. There has been some work on nonconflict test, e.g. in [7] [8] [17] the authors propose to utilize model abstractions to avoid high complexity caused by synchronous product. To compute model abstraction [7] [8] use natural projections, which are required to be observers [10]. The main disadvantage of using observers is that, the alphabet of the codomain of a projection may need to be fairly large for the sake of achieving the observer property, potentially causing the size of the projected image too large for effective nonconflict test. To overcome this difficulty, we propose to use an automaton abstraction procedure, which appeared in [12] [13] for the purpose of supervisor synthesis. Our first contribution is to extend the concept of standardized automata in [13] by selflooping a special event called *marking event* at each marker state so that abstraction preserves blocking behavior, which is crucial for the success of nonconflict check. Although [17] and several other abstraction techniques are based on nondeterministic automata, e.g. in [11] [18] [19] [20], they are different from our approach. More explicitly, [11] aims to achieve weak bisimilarity between an automaton and its abstraction, and [17] [18] [19] [20] first use silence events to replace internal events, then apply rewriting rules to ensure that appropriate equivalence relations hold between automata before and after rewriting, e.g. conflict equivalence in [17] [20], supervision equivalence in [18] and synthesis equivalence in [19]. The primary goal of our abstraction technique is to create an abstraction for an automaton G, which is not necessarily weak bisimilar to G, such that any automaton S, whose alphabet is the same as that of the abstraction, is nonconflicting with the abstraction if and only if it is nonconflicting with G. The primary goal is close to achieving conflict equivalence, but with a procedure much simpler than those rewriting rules and no silence events are needed. Based on the automaton abstraction technique, our second contribution is to present an efficient sequential abstraction procedure (SAP), which bears similarity to an algorithm called Computational Procedure for

Global Consistency (CPGC) provided in [22], except that CPGC is based on natural projections, which, as we have pointed out before, is not suitable for nonconflict check unless all relevant projections are observers. The aggregative nature of SAP and CPGC is also reflected in [17]. But as mentioned above, the abstraction technique of [17] is different from ours. Our third contribution is to present a procedure for nonconflict check (PNC) using SAP, which can check nonconflict of a large number of automata. The efficiency of PNC has been illustrated by numerical experiments.

This paper is organized as follows. In Section II we introduce abstraction over nondeterministic automata and provide relevant properties. Then a procedure called PNC for nonconflict check based on sequential abstractions is presented in Section III. Conclusions are stated in Section IV. Long proofs are presented in the Appendix.

2 Automaton Abstraction and Relevant Properties

2.1 Concepts of Languages, Automaton Product and Abstraction

In the following sections we follow the notations used in [14]. Let Σ be a finite alphabet, and Σ^* the Kleene closure of Σ, i.e. the collection of all finite sequences of events taken from Σ. Given two strings $s, t \in \Sigma^*$, s is called a *prefix substring* of t, written as $s \leq t$, if there exists $s' \in \Sigma^*$ such that $ss' = t$, where ss' denotes the concatenation of s and s'. We use ϵ to denote the empty string of Σ^* such that for any string $s \in \Sigma^*$, $\epsilon s = s\epsilon = s$. For $\sigma \in \Sigma$ and $s \in \Sigma^*$, we use $\sigma \in s$ to denote that s contains σ. A subset $L \subseteq \Sigma^*$ is called a *language*. $\overline{L} = \{s \in \Sigma^* | (\exists t \in L)\, s \leq t\} \subseteq \Sigma^*$ is called the *prefix closure* of L. Given two languages $L, L' \subseteq \Sigma^*$, $LL' := \{ss' \in \Sigma^* | s \in L \wedge s' \in L'\}$.

Let $\Sigma' \subseteq \Sigma$. A map $P : \Sigma^* \to \Sigma'^*$ is called the *natural projection* with respect to (Σ, Σ'), if

1. $P(\epsilon) = \epsilon$
2. $(\forall \sigma \in \Sigma)\, P(\sigma) := \begin{cases} \sigma & \text{if } \sigma \in \Sigma' \\ \epsilon & \text{otherwise} \end{cases}$
3. $(\forall s\sigma \in \Sigma^*)\, P(s\sigma) = P(s)P(\sigma)$

Given a language $L \subseteq \Sigma^*$, $P(L) := \{P(s) \subseteq \Sigma'^* | s \in L\}$. For any two languages L, $L' \subseteq \Sigma^*$, we can show that $P(LL') = P(L)P(L')$. The inverse image mapping of P is

$$P^{-1} : 2^{\Sigma'^*} \to 2^{\Sigma^*} : L \mapsto P^{-1}(L) := \{s \in \Sigma^* | P(s) \in L\}$$

Given $L_1 \subseteq \Sigma_1^*$ and $L_2 \subseteq \Sigma_2^*$, the *synchronous product* of L_1 and L_2 is defined as:

$$L_1 || L_2 := P_1^{-1}(L_1) \cap P_2^{-1}(L_2) = \{s \in (\Sigma_1 \cup \Sigma_2)^* | P_1(s) \in L_1 \wedge P_2(s) \in L_2\}$$

where $P_1 : (\Sigma_1 \cup \Sigma_2)^* \to \Sigma_1^*$ and $P_2 : (\Sigma_1 \cup \Sigma_2)^* \to \Sigma_2^*$ are natural projections. It is clear that $||$ is commutative and associative.

Suppose Σ contains two special events: the *marking* event μ, and the *initial* event τ. Given an automaton $G = (X, \Sigma, \xi, x_0, X_m)$, where X stands for the state set, Σ for the

alphabet, $\xi : X \times \Sigma \to 2^X$ for the nondeterministic transition function, x_0 for the initial state, and X_m for the marker state set. As usual, ξ is extended to $X \times \Sigma^*$.

Definition 2.1. We say G is *standardized* if the following conditions hold,

1. $(\forall x \in X)\,[\xi(x,\tau) \neq \varnothing \iff x = x_0] \wedge (\forall \sigma \in \Sigma - \{\tau\})\,\xi(x_0,\sigma) = \varnothing$
2. $(\forall x \in X - \{x_0\})(\forall \sigma \in \Sigma)\, x_0 \notin \xi(x,\sigma)$
3. $(\forall x \in X)\, x \in X_m \Rightarrow x \in \xi(x,\mu)$ □

A standardized automaton is an automaton, in which x_0 is not marked (by conditions 1,3), τ is only defined at x_0, which only has τ outgoing transitions (by condition 1) without any incoming transition (by condition 2); and each marker state has a selflooping transition μ (by condition 3). Let $\phi(\Sigma)$ be the collection of all standardized finite-state automata over Σ.

We now introduce product and abstraction on finite-state automata that will be extensively used later. Given $G_i = (X_i, \Sigma_i, \xi_i, x_{0,i}, X_{m,i}) \in \phi(\Sigma_i)$ $(i = 1, 2)$, the *product* of G_1 and G_2, written as $G_1 \times G_2$, is an automaton in $\phi(\Sigma_1 \cup \Sigma_2)$ such that

$$G_1 \times G_2 = (X_1 \times X_2, \Sigma_1 \cup \Sigma_2, \xi_1 \times \xi_2, (x_{0,1}, x_{0,2}), X_{m,1} \times X_{m,2})$$

where $\xi_1 \times \xi_2 : X_1 \times X_2 \times (\Sigma_1 \cup \Sigma_2) \to 2^{X_1 \times X_2}$ is defined as follows,

$$(\xi_1 \times \xi_2)((x_1,x_2),\sigma) := \begin{cases} \xi_1(x_1,\sigma) \times \{x_2\} & \text{if } \sigma \in \Sigma_1 - \Sigma_2 \\ \{x_1\} \times \xi_2(x_2,\sigma) & \text{if } \sigma \in \Sigma_2 - \Sigma_1 \\ \xi_1(x_1,\sigma) \times \xi_2(x_2,\sigma) & \text{if } \sigma \in \Sigma_1 \cap \Sigma_2 \end{cases}$$

Clearly, $\times$ is commutative and associative. By a slight abuse of notations, from now on we use $G_1 \times G_2$ to denote its reachability part. $\xi_1 \times \xi_2$ is extended to $X_1 \times X_2 \times (\Sigma_1 \cup \Sigma_2)^* \to 2^{X_1 \times X_2}$.

Lemma 2.2. Let $G_i \in \phi(\Sigma_i)$ $(i = 1, 2)$ be standardized. Then $G_1 \times G_2$ is also standardized. □

By Lemma 2.2 we get that standardization is preserved under automaton product. Next, we discuss how to create an abstraction of an automaton.

Definition 2.3. Given $G = (X, \Sigma, \xi, x_0, X_m)$, let $\Sigma' \subseteq \Sigma$ and $P : \Sigma^* \to \Sigma'^*$ be the natural projection. A *marking weak bisimulation* relation on X with respect to Σ' is an equivalence relation $R \subseteq \{(x,x') \in X \times X | x \in X_m \iff x' \in X_m\}$ such that,

$$(\forall (x,x') \in R)(\forall s \in \Sigma^*)(\forall y \in \xi(x,s))(\exists s' \in \Sigma^*)\, P(s) = P(s') \wedge (\exists y' \in \xi(x',s'))\,(y,y') \in R$$

The largest marking weak bisimulation relation on X with respect to Σ' is called *marking weak bisimilarity* on X with respect to Σ', written as $\approx_{\Sigma',G}$. □

Marking weak bisimulation relation is the same as weak bisimulation relation described in [21], except for the special treatment on marker states. We now introduce abstraction.

Definition 2.4. Given $G = (X, \Sigma, \xi, x_0, X_m)$, let $\Sigma' \subseteq \Sigma$ with $\tau, \mu \in \Sigma'$. The *automaton abstraction* of G with respect to $\approx_{\Sigma',G}$ is an automaton $G/\approx_{\Sigma',G} := (Y, \Sigma', \eta, y_0, Y_m)$ where

1. $Y := X/\approx_{\Sigma',G} := \{< x >:= \{x' \in X | (x, x') \in \approx_{\Sigma',G}\} | x \in X\}$
2. $y_0 :=< x_0 > \in Y$
3. $Y_m := \{y \in Y | y \cap X_m \neq \varnothing\}$
4. $\eta : Y \times \Sigma' \to 2^Y$, where for any $(y, \sigma) \in Y \times \Sigma'$,
$$\eta(y, \sigma) := \{y' \in Y | (\exists x \in y)(\exists u, u' \in (\Sigma - \Sigma')^*)\, \xi(x, u\sigma u') \cap y' \neq \varnothing\}$$

□

An automaton abstraction always contains events τ and μ. In the rest of this paper we will discuss properties of automaton abstraction and its application in nonconflict check.

The time complexity of computing $G/\approx_{\Sigma',G}$ is mainly resulted from computing $X/\approx_{\Sigma',G}$, which can be estimated as follows. We first define a new automaton $G'' = (X, \Sigma', \xi'', x_0, X_m)$, where for any $x, x' \in X$ and $\sigma \in \Sigma$, $x' \in \xi''(x, \sigma)$ if there exist $u, u' \in (\Sigma - \Sigma')^*$ such that $x' \in \xi(x, u\sigma u')$. Then we compute $X/\approx_{\Sigma',G''}$, and we can show that the result is equal to $X/\approx_{\Sigma',G}$. The total number of transitions in G'' is no more than mn^2, where $n = |X|$ and m is the number of transitions in G. Based on a result shown in [16], the time complexity of computing $X/\approx_{\Sigma',G''}$ is $O(mn^2 \log n)$ if we ignore the complexity caused by checking the condition "$x \in X_m \iff x' \in X_m$" in Def. 2.3. If we consider this extra condition, then the overall complexity is $O(n(n-1) + mn^2 \log n)$, because we need to check at most $n(n-1)$ pairs of states.

From now on, when G is clear from the context, we simply use $\approx_{\Sigma'}$ to denote $\approx_{\Sigma',G}$, and use $< x >_{\Sigma'}$ for an element of $X/\approx_{\Sigma',G}$. If Σ' is also clear from the context, then we simply use $< x >$ for $< x >_{\Sigma'}$. We have the following result, indicating that standardization is preserved under automaton abstraction.

Lemma 2.5. Let $G \in \phi(\Sigma)$ be a standardized automaton and $\Sigma' \subseteq \Sigma$ with $\tau, \mu \in \Sigma'$. Then $G/\approx_{\Sigma'}$ is also a standardized automaton. □

To illustrate automaton abstraction, suppose a standardized automaton $G \in \phi(\Sigma)$ is depicted in Figure 1, where $\Sigma = \{\tau, a, b, \mu\}$. We take $\Sigma' = \{\tau, b, \mu\}$. Then we have

$$X/\approx_{\Sigma'} = \{< 0 >= \{0\}, < 1 >= \{1, 2\}, < 3 >= \{3\}, < 4 >= \{4\}\}$$

By Def. 2.4, the abstraction $G/\approx_{\Sigma'}$ is depicted in Figure 1. We now present some properties of automaton product and automaton abstraction.

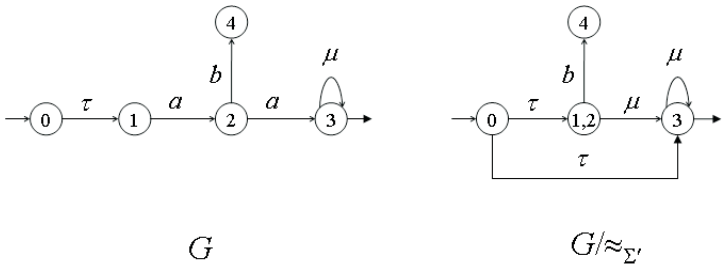

Figure 1: Example 1: A Standardized Automaton G and its Abstraction $G/\approx_{\Sigma'}$

2.2 Properties of Abstraction

We define a map $B : \phi(\Sigma) \to 2^{\Sigma^*}$, where for each $G \in \phi(\Sigma)$,

$$B(G) := \{s \in \Sigma^* | (\exists x \in \xi(x_0, s))(\forall s' \in \Sigma^*)\, \xi(x', s') \cap X_m = \varnothing\}$$

Any string $s \in B(G)$ can lead to a state x, from which no marker state is reachable, i.e. for any $s \in \Sigma^*$, $\xi(x, s) \cap X_m = \varnothing$. Such a state x is called a *blocking state* of G, and we call $B(G)$ the *blocking set* of G. A state that is not a blocking state is called a *nonblocking state*. We say G is *nonblocking* if $B(G) = \varnothing$. Similarly, we define another map $N : \phi(\Sigma) \to 2^{\Sigma^*}$ with

$$(\forall G \in \phi(\Sigma))\, N(G) := \{s \in \Sigma^* | \xi(x_0, s) \cap X_m \neq \varnothing\}$$

We call $N(G)$ the *nonblocking set* of G, which is simply the set of all strings recognized by G. It is possible that $B(G) \cap \overline{N(G)} \neq \varnothing$, due to nondeterminism. We have the following result.

Proposition 2.6. Given $G = (X, \Sigma, \xi, x_0) \in \phi(\Sigma)$, let $\Sigma' \subseteq \Sigma$, and $P : \Sigma^* \to \Sigma'^*$ be the natural projection. Then $P(B(G)) = B(G/\approx_{\Sigma'})$ and $P(N(G)) = N(G/\approx_{\Sigma'})$. □

The content of Prop. 2.6 is illustrated by the commutative diagram in Figure 2, from

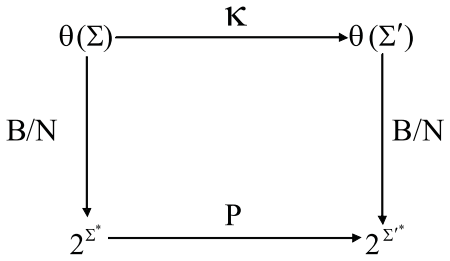

Figure 2: The Commutative Diagram for Proposition 2.6

which we can derive that, an automaton G is nonblocking if and only if $G/\approx_{\Sigma'}$ is nonblocking.

Given an automaton $G = (X, \Sigma, \xi, x_0, X_m)$, for each $x \in X$, let

$$N_G(x) := \{s \in \Sigma^* | \xi(x', s) \cap X_m \neq \varnothing \wedge \mu \in s\}$$

We can easily show that, if G is standardized, then $x \in X$ is a blocking state if and only if $N_G(x) = \varnothing$. We now introduce the following concept, which is extensively used in this paper.

Definition 2.7. Given automata $G_i = (X_i, \Sigma_i, \xi_i, x_{i,0}, X_{i.m})$ $(i = 1, 2)$, we say G_1 is *nonblocking preserving* with respect to G_2, denoted as $G_1 \sqsubseteq G_2$, if $B(G_1) \subseteq B(G_2)$, $N(G_1) = N(G_2)$ and

$$(\forall s \in \overline{N(G_1)})(\forall x_1 \in \xi_1(x_{1,0}, s))(\exists x_2 \in \xi_2(x_{2,0}, s))\, N_{G_2}(x_2) \subseteq N_{G_1}(x_1) \wedge [x_1 \in X_{1,m} \iff x_2 \in X_{2,m}]$$

G_1 is *nonblocking equivalent* to G_2, denoted as $G_1 \cong G_2$, if $G_1 \sqsubseteq G_2$ and $G_2 \sqsubseteq G_1$. □

Def. 2.7 says that, if G_1 is nonblocking preserving with respect to G_2 then their individual nonblocking parts are equal, but G_2's blocking behavior may be larger. If blocking behaviors are also equal, then G_1 and G_2 are nonblocking equivalent. We now present a few results.

Proposition 2.8. $(\forall G_1, G_2 \in \phi(\Sigma))(\forall G_3 \in \phi(\Sigma'))\, G_1 \sqsubseteq G_2 \Rightarrow G_1 \times G_3 \sqsubseteq G_2 \times G_3$. □

Corollary 2.9. $(\forall G_1, G_2 \in \phi(\Sigma))(\forall G_3 \in \phi(\Sigma'))\, G_1 \cong G_2 \Rightarrow G_1 \times G_3 \cong G_2 \times G_3$. □

Proof: Since $G_1 \cong G_2$, by Def. 2.7 we have $G_1 \sqsubseteq G_2$ and $G_2 \sqsubseteq G_1$. Then by Prop. 2.8 we get $G_1 \times G_3 \sqsubseteq G_2 \times G_3$ and $G_2 \times G_3 \sqsubseteq G_1 \times G_3$, namely $G_1 \times G_3 \cong G_2 \times G_3$. ■

Prop. 2.8 and Cor. 2.9 say nonblocking preserving and equivalence are invariant under product.

Proposition 2.10. $(\forall \Sigma' \subseteq \Sigma)(\forall G_1, G_2 \in \phi(\Sigma))\, G_1 \sqsubseteq G_2 \Rightarrow G_1 / \approx_{\Sigma'} \sqsubseteq G_2 / \approx_{\Sigma'}$. □

Corollary 2.11. $(\forall \Sigma' \subseteq \Sigma)(\forall G_1, G_2 \in \phi(\Sigma))\, G_1 \cong G_2 \Rightarrow G_1 / \approx_{\Sigma'} \cong G_2 / \approx_{\Sigma'}$. □

Proof: Use Prop. 2.10 and Def. 2.7, the corollary follows. ■

Prop. 2.10 and Cor. 2.11 say nonblocking preserving and equivalence is invariant under abstraction.

Proposition 2.12. $(\forall \Sigma'' \subseteq \Sigma' \subseteq \Sigma)(\forall G \in \phi(\Sigma))\, G / \approx_{\Sigma''} \cong (G / \approx_{\Sigma'}) / \approx_{\Sigma''}$. □

Prop. 2.12 is about the chain rule of automaton abstraction, which says an automaton abstraction can be replaced by a sequence of automaton abstractions, and the results are nonblocking equivalent to each other.

Proposition 2.13. Given $G_i \in \phi(\Sigma_i)$ with $i = 1, 2$, let $\Sigma' \subseteq \Sigma_1 \cup \Sigma_2$. If $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma'$, then we have that $(G_1 \times G_2)/\approx_{\Sigma'} \cong (G_1/\approx_{\Sigma_1 \cap \Sigma'}) \times (G_2/\approx_{\Sigma_2 \cap \Sigma'})$. □

Proposition 2.13 is about the distribution of automaton abstraction over automaton product. As an illustration we present a simple example. Suppose we have $\Sigma_1 = \{\tau, a, \mu\}$ and $\Sigma_2 = \{\tau, b, c, \mu\}$. Let $G_1 \in \phi(\Sigma_1)$ and $G_2 \in \phi(\Sigma_2)$ be shown in Figure 3. Suppose we pick

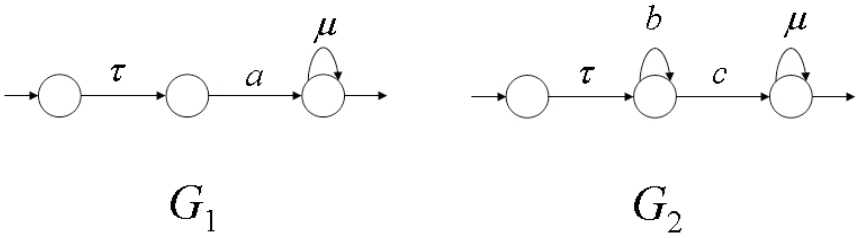

Figure 3: Example 2: G_1 and G_2

$\Sigma' = \{\tau, a, b, \mu\} \supseteq \Sigma_1 \cap \Sigma_2$. The results of $G_1 \times G_2$ and $(G_1 \times G_2)/\approx_{\Sigma'}$ are depicted in Figure 4. The results of $G_1/\approx_{\Sigma_1 \cap \Sigma'}$, $G_2/\approx_{\Sigma_2 \cap \Sigma'}$ and $(G_1/\approx_{\Sigma_1 \cap \Sigma'}) \times (G_2/\approx_{\Sigma_2 \cap \Sigma'})$ are

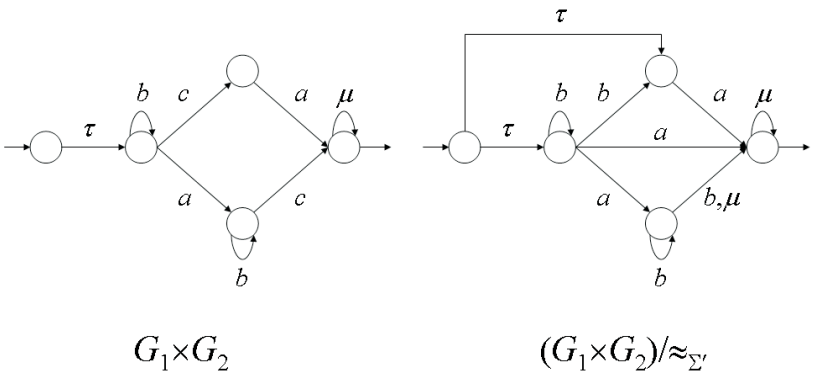

Figure 4: Example 2: $G_1 \times G_2$ and $(G_1 \times G_2)/\approx_{\Sigma'}$

depicted in Figure 5. We can check that $(G_1 \times G_2)/\approx_{\Sigma'} \cong (G_1/\approx_{\Sigma_1 \cap \Sigma'}) \times (G_2/\approx_{\Sigma_2 \cap \Sigma'})$.

If G is very large, e.g. $G = G_1 \times \cdots \times G_n$ for some very large number $n \in \mathbb{N}$, where $G_i \in \phi(\Sigma_i)$ for $i = 1, 2, \cdots, n$, how to compute $G/\approx_{\Sigma'}$? To overcome this difficulty, we propose the following algorithm.

Suppose $I = \{1, \cdots, n\}$ for some $n \in \mathbb{N}$. For $J \subseteq I$, let $\Sigma_J := \cup_{j \in J} \Sigma_j$. Let $\Sigma' \subseteq \cup_{i \in I} \Sigma_i$.

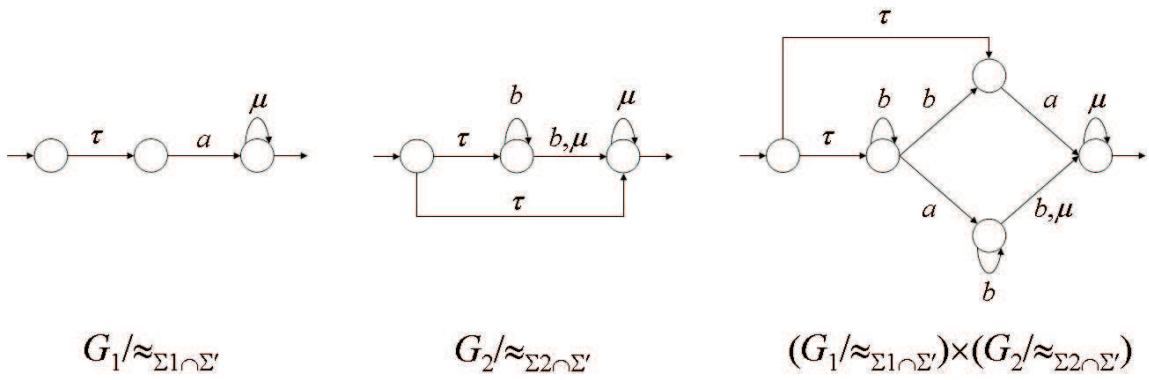

Figure 5: Example 2: $G_1/\approx_{\Sigma_1\cap\Sigma'}$, $G_2/\approx_{\Sigma_2\cap\Sigma'}$ and $(G_1/\approx_{\Sigma_1\cap\Sigma'})\times(G_2/\approx_{\Sigma_2\cap\Sigma'})$

Sequential Abstraction over Product: (SAP)
(1) Input of SAP: a collection of automata $\{G_i|i\in I\}$.
(2) For $k=1,2,\cdots,n$, we perform the following computation.

- Set $J_k:=\{1,2,\cdots,k\}$, $T_k:=\Sigma_{J_k}\cap(\Sigma_{I-J_k}\cup\Sigma')$.
- If $k=1$ then $W_1:=G_1/\approx_{T_1}$
- If $k>1$ then $W_k:=(W_{k-1}\times G_k)/\approx_{T_k}$

(3) Output of SAP: W_n. □

Theorem 2.14. Suppose W_n is computed by SAP. Then $(\times_{i\in I}G_i)/\approx_{\Sigma'}\cong W_n$. □

Proof: We use induction to show that

$$(\forall k:1\le k\le n)\,(\times_{j\in J_k}G_j)/\approx_{T_k}\cong W_k \tag{1}$$

It is clear that $G_1/\approx_{T_1}\cong W_1$. Suppose Equation (1) is true for $k\le l\in\mathbb{N}$. Then we need to show that it also holds for $k=l+1$. By the procedure,

$$\begin{aligned}(\times_{j\in J_{l+1}}G_j)/\approx_{T_{l+1}} &\cong ((\times_{j\in J_{l+1}}G_j)/\approx_{T_l\cup\Sigma_{l+1}})/\approx_{T_{l+1}} \text{ by Prop. 2.12}\\ &\cong (((\times_{j\in J_l}G_j)/\approx_{T_l})\times G_{l+1})/\approx_{T_{l+1}}\\ &\quad \text{because } \Sigma_{l+1}\cap\Sigma_{J_l}\subseteq T_l\cup\Sigma_{l+1} \text{ and Prop. 2.13 and Prop. 2.10}\\ &\cong (W_l\times G_{l+1})/\approx_{T_{l+1}}\\ &\quad \text{by the induction hypothesis and Prop. 2.8 and Prop. 2.10}\\ &= W_{l+1}\end{aligned}$$

Therefore Equation (1) holds for all k, particularly $k=n$. The proposition follows. ■

Theorem 2.14 confirms that SAP allows us to obtain an abstraction of the entire system $G=\times_{i\in I}G_i$ in a sequential way. Thus, we can avoid computing G explicitly, which may be prohibitively large for many industrial systems. The results of our numerical experiments indicate that the ordering of those automata affects the computational complexity, which is defined as the corresponding maximum number of states and transitions appearing in SAP at each step, i.e. $\max_k||W_{k-1}\times G_k||$, where $||W_{k-1}\times G_k||$ denotes the

number of states and transitions of $W_{k-1} \times G_k$. Unfortunately, finding an ordering that results in the minimum complexity requires enumeration of all possible orderings. So we come up with a heuristic rule: given $J_{k-1} = \{r_1, r_2, \cdots, r_{k-1}\}$, the choice of Σ_{r_k} at the step k in SAP maximize the ratio of the size of $\Sigma_{J_k} = \Sigma_{J_{k-1}} \cup \Sigma_{r_k}$, denoted as $|\Sigma_{J_k}|$, over the size of T_k, namely

$$r_k := \arg \max_{i \in I - J_{k-1}} \frac{|\Sigma_{J_{k-1}} \cup \Sigma_i|}{|(\Sigma_{J_{k-1}} \cup \Sigma_i) \cap (\Sigma_{I-(J_{k-1} \cup \{i\})} \cup \Sigma')|}$$

The rationality of this rule is that a large ratio of alphabets implies a large ratio of automaton sizes $|| \times_{j \in J_k} G_j||/||(\times_{j \in J_k} G_j)/\approx_{T_k}||$, which may indirectly put $||W_{k-1} \times G_{r_k}||$ under control. Although the rationality can not be formally proved, it does provide a good heuristics that usually results in small complexity in SAP, as illustrated in our numerical experiments listed in Table 1 of the next section.

Next, we discuss how to use SAP to check nonconflict of a large number of ordinary finite-state automata and provide experiment results.

3 Check Nonconflict of Finite-State Automata

Given a deterministic finite state automaton $A = (Y, \Delta, \eta, y_0, Y_m)$, we define the *closed behavior* of A as $L(A) := \{s \in \Sigma^* | \eta(y_0, s) \text{ is defined}\}$, and the *marked behavior* of A as $L_m(A) := \{s \in L(A) | \eta(y_0, s) \cap Y_m \neq \varnothing\}$. Let I be a finite index set. Given a collection of deterministic finite-state automata

$$\mathcal{A} := \{A_i = (Y_i, \Delta_i, \eta_i, y_{i,0}, Y_{i,m}) | i \in I\}$$

we say $\mathcal{A}$ is *nonconflicting* if $\overline{||_{i \in I} L_m(A_i)} = ||_{i \in I} L(A_i)$.

To check whether $\mathcal{A}$ is nonconflicting, we propose the procedure of nonconflict check (PNC):

1. Input: $\mathcal{A} = \{A_i | i \in I\}$.
2. For each $i \in I$, we create a standardized automaton $G_i = (X_i, \Sigma_i, \xi_i, x_{i,0}, X_{i,m})$ as follows:
   (a) $X_i := Y_i \cup \{\hat{x}_0\}$
   (b) $X_{i,m} := Y_{i,m}$
   (c) $x_{i,o} := \hat{x}_0$
   (d) $\Sigma_i := \Delta_i \cup \{\tau, \mu\}$
   (e) $\xi_i : X_i \times \Sigma_i \to 2^{X_i}$ is defined as follows:
   - For any $x \in Y_i$ and $\sigma \in \Delta$, $\xi(x, \sigma) := \{\eta(x, \sigma)\}$
   - $\xi(x_{i,0}, \tau) := \{y_0\}$
   - For any $x \in Y_{i,m}$, $\xi(x, \mu) := \{x\}$
3. Let $\Sigma' := \{\tau, \mu\}$ and $\Sigma = \cup_{i \in I} \Sigma_i$. Use SAP to compute W_n, where $n = |I|$.
4. Output: if $B(W_n) = \varnothing$ then claim that $\mathcal{A}$ is nonconflicting. Otherwise, claim that $\mathcal{A}$ is conflicting. □

What PNC does is simply to first convert each A_i into a standardize automaton G_i by adding an extra state $x_{i,0}$, which is connected with the initial state $y_{i,0}$ of A_i by τ, and treated as the initial state of G_i; then selflooping μ at each marker state of A_i. After that, we run SAP on those standardized automata $\{G_i|i \in I\}$, and make a conclusion on whether $\mathcal{A}$ is nonconflicting by checking the emptiness of $B(W_n)$. We will show that the claim made by PNC is correct.

Lemma 3.1. $\mathcal{A}$ is nonconflicting if and only if $B(\times_{i\in I}G_i) = \varnothing$. □

Proof: By the properties of automaton product and synchronous product, we have that $\mathcal{A}$ is nonconflicting if and only if $\overline{L_m(\times_{i\in I}A_i)} = L(\times_{i\in I}A_i)$. Since each A_i is deterministic, we have that $\overline{L_m(\times_{i\in I}A_i)} = L(\times_{i\in I}A_i)$ if and only if $B(\times_{i\in I}A_i) = \varnothing$. By the construction of $\{G_i|i \in I\}$, we have that $B(\times_{i\in I}A_i) = \varnothing$ if and only if $B(\times_{i\in I}G_i) = \varnothing$. ■

The correctness of PNC is shown in the following main result.

Theorem 3.2. $\mathcal{A}$ is nonconflicting if and only if in PNC we have $B(W_n) = \varnothing$. □

Proof: By Lemma 3.1 we get that $\mathcal{A}$ is nonconflicting if and only if $B(\times_{i\in I}G_i) = \varnothing$. Let $P : \Sigma^* \to \Sigma'^*$ be the natural projection. We have

$$B(\times_{i\in I}G_i) = \varnothing \iff P(B(\times_{i\in I}G_i)) = \varnothing$$

By Prop. 2.6 we get that

$$P(B(\times_{i\in I}G_i)) = \varnothing \iff B((\times_{i\in I}G_i)/\approx_{\Sigma'}) = \varnothing$$

By Theorem. 2.14 we have $(\times_{i\in I}G_i)/\approx_{\Sigma'} \cong W_n$. Thus, we have

$$B((\times_{i\in I}G_i)/\approx_{\Sigma'}) = \varnothing \iff B(W_n) = \varnothing$$

which means $B(\times_{i\in I}G_i) = \varnothing$ if and only if $B(W_n) = \varnothing$, and the theorem follows. ■

Theorem 3.2 confirms that we can use PNC to determine whether $\mathcal{A}$ is nonconflicting. As an illustration we apply the proposed nonconflict check approach to the following simple transfer line (STL) example, which is depicted in Figure 6. In this example we

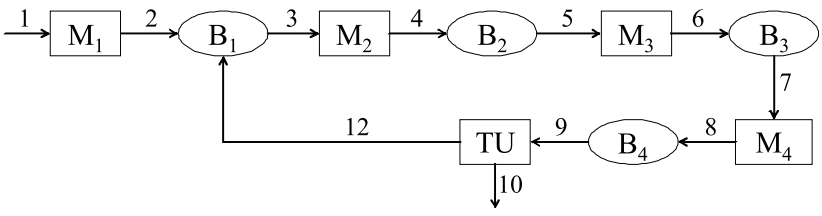

Figure 6: Example 3: A Simple Transfer Line (STL)

have 4 buffers B_1, B_2, B_3 and B_4; 4 machines M_1, M_2, M_3 and M_4, whose function is to put work pieces to a buffer or remove work pieces from a buffer; and 1 transfer unit (TU), whose function is to remove work pieces from B_4 or return (imperfect) work pieces back to B_1. The models for machines and TU are depicted in Figure 7, and models of buffers are depicted in Figure 8. We require that no buffer will be overflow or underflow. After using the modular design approach introduced in [6], we obtain 4 local supervisors

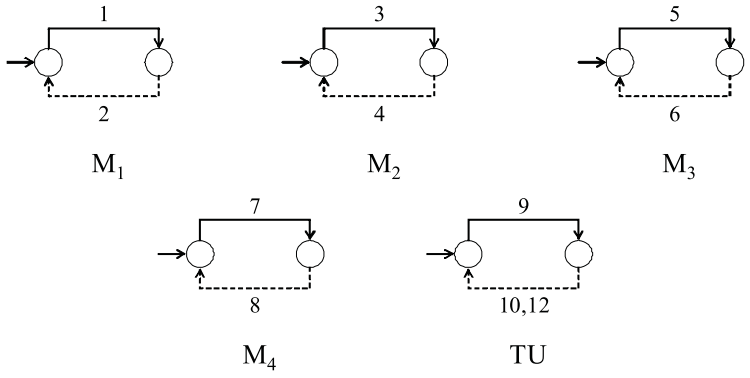

Figure 7: Example 3: Models for Machines and TU

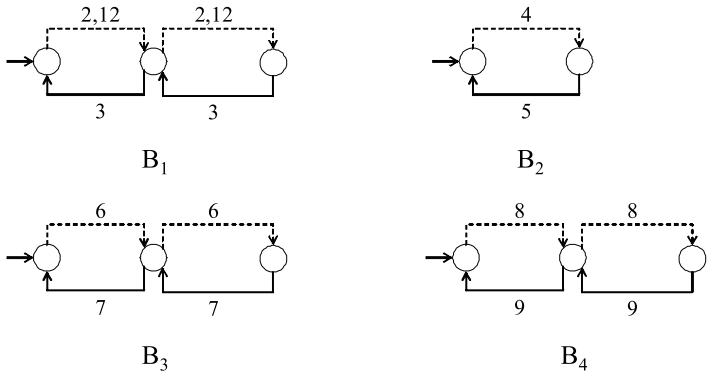

Figure 8: Example 3: Buffer Models

$SUPER_i$ $(i = 1, 2, 3, 4)$, where each $SUPER_i$ is associated with B_i; and one coordinator C, whose function is to prevent conflict among local supervisors. Their sizes are listed as follows:

$SUPER_1$ (16, 42); $SUPER_2$ (6, 8); $SUPER_3$ (10, 16); $SUPER_4$ (10, 21); C (59, 158)

where in each tuple (x, y), x denotes the number of states and y for the number of transitions.

Suppose we want to check whether local supervisors and the coordinator are nonconflicting. We apply our proposed approach. First, we convert every ordinary finite-state automaton into a standard automaton, then apply PNC on them. To show that ordering may have impact on the computational complexity of PNC, we choose two different orders and list results below.
(1) The ordering is $SUPER_1$, $SUPER_2$, $SUPER_3$, $SUPER_4$, C. The results of SAP are:

$W_1\,(7, 10)$; $W_1 \times SUPER_2\,(31, 92)$; $W_2\,(15, 42)$; $W_2 \times SUPER_3\,(71, 246)$
$W_3\,(71, 246)$; $W_3 \times SUPER_4\,(176, 554)$; $W_4\,(64, 170)$; $W_4 \times SUPER_5\,(60, 160)$; $W_5\,(3, 6)$

In W_5 we have $B(W_5) = \varnothing$. Thus, we conclude that they are nonconflicting.

(2) The ordering is $SUPER_1$, $SUPER_3$, $SUPER_2$, $SUPER_4$, C. The results of SAP are:

$W_1\,(11,18)\,;\, W_1 \times SUPER_3\,(101,422)\,;\, W_2\,(101,422)\,;\, W_2 \times SUPER_2\,(251,912)$

$W_3\,(91,308)\,;\, W_3 \times SUPER_4\,(136,385)\,;\, W_4\,(64,170)\,;\, W_4 \times SUPER_5\,(60,160)\,;\, W_5\,(3,6)$

In W_5 we have $B(W_5) = \varnothing$. Thus, we conclude that they are nonconflicting.
By using the monolithic approach, whose maximum size is (568, 1927) after synchronizing all local supervisors and the coordinator together, we confirm that, indeed they are nonconflicting. We can see that the first ordering is better than the second one. After we take out the coordinator C and redo the same check with the ordering of $SUPER_1$, $SUPER_2$, $SUPER_3$ and $SUPER_4$, we have the following results:

$$W_1\ (7,\ 10);\ W_2\ (19,\ 128);\ W_3\ (29,\ 282);\ W_4\ (4,\ 9)$$

where $B(W_4) \neq \varnothing$. Thus, we conclude that those local supervisors without the coordinator are conflicting with each other. The conclusion is consistent with the result from using the monolithic approach, whose maximum size of intermediate results is (600, 2039). Next, we apply PNC to some relatively large examples to check its efficiency.

Table 1

CN	LC	SSAC	SRPNC	MSPNC/CTPNC (s)	CMC	CPNC
1	G_1 (15, 219); G_2 (408, 3664); G_3 (39, 414); G_4 (43, 571); G_5 (33, 117)	(1159920, 23686344)	W_1 (11, 54); W_2 (35, 350); W_3 (21, 196); W_4 (31, 420); W_5 (3, 6)	(161, 3295) / 10	NC	NC
2	G_1 (24, 80); G_2 (162, 1620); G_3 (12, 80); G_4 (66, 547); G_5 (9, 78)	(1135296, 25014096)	W_1 (3, 11); W_2 (11, 140); W_3 (81, 1312); W_4 (25, 621); W_5 (4, 9)	(177, 2840) / 3	C	C
3	G_1 (16, 69); G_2 (112, 672); G_3 (108, 1332); G_4 (39, 414); G_5 (64, 404)	(1005120, 19683696)	W_1 (39, 745); W_2 (305, 6396); W_3 (8, 32); W_4 (37, 899); W_5 (3, 6)	(305, 6296) / 25	NC	NC
4	G_1 (162, 1620); G_2 (112, 672); G_3 (108, 1332); G_4 (33, 117);	(5272128, 150273792)	W_1 (9, 39); W_2 (5, 20); W_3 (11, 34); W_4 (3, 6)	(17, 206) / 0.1	NC	NC

Table 1 summarizes our test results, where 'CN' denotes *Case Number*, 'LC' for *Local Component*, 'SSAC' for *Size of Synchronization of All Components* (i.e. the numbers of states and transitions of $\times_{i\in I} G_i$), 'SRPNC' for *Sizes of Results of PNC*, 'MSPNC' for *Maximum Size* in PNC, 'CTPNC' for *Computation Time* of PNC which is realized in Python running on Intel(R) Core(TM)2 2.4GHz CPU with 3.5 GB RAM, 'CMC' for *Conclusion of Monolithic Check*, 'CPNC' for *Conclusion of PNC*, 'NC' for *nonconflicting* and 'C' for *conflicting*. We do not count the computation time for the monolithic approach because the check is done manually. But the computation time for synchronization takes 10 s for Case 1, 33s for Case 2, 23s for Case 3 and 381s for Case 4. The ordering of local components in PNC is determined by the heuristic rule described in Section II.

In the monolithic approach, by using the 'trim' operation on $\times_{i\in I}G_i$ in TCT [15], we can determine whether $\times_{i\in I}G_i$ is reachable and coreachable, i.e. we can decide whether local components are nonconflicting. Based on this approach we obtain all results in the column of CMC, which agree with the conclusions made by PNC. Clearly PNC has a substantial computational advantage over the monolithic approach.

4 Conclusions

In this paper we first introduce an automaton-based abstraction technique and provide relevant properties. Then we propose a sequential abstraction procedure (SAP), based on which we propose the procedure PNC to check nonconflict of a large number of deterministic finite-state automata, which is commonly encountered in Ramadge-Wonham supervisory control theory. Numerical experiments have shown that, sequential abstraction can help us avoid high complexity in checking nonconflict, which is usually resulted from product of a large number of automata.

Appendix

1. Proof of Lemma 2.2: Suppose $G_i = (X_i, \Sigma_i, \xi_i, x_{i,0}, X_{i,m})$ $(i = 1, 2)$. First, for any $(x_1, x_2) \in X_1 \times X_2$ we have

$$\begin{aligned}
\xi_1 \times \xi_2((x_1,x_2),\tau) \neq \varnothing \quad &\Longleftrightarrow \quad \xi_1(x_1,\tau) \neq \varnothing \wedge \xi_2(x_2,\tau) \neq \varnothing \\
&\Longleftrightarrow \quad x_1 = x_{1,0} \wedge x_2 = x_{2,0} \text{ because } G_1 \text{ and } G_2 \text{ are standardized} \\
&\Longleftrightarrow \quad (x_1,x_2) = (x_{1,0}, x_{2,0})
\end{aligned}$$

For any $\sigma \in (\Sigma_1 \cup \Sigma_2) - \{\tau\}$, if $\sigma \in \Sigma_i$ for i=1 or 2, we have $\xi_i(x_{i,0}, \sigma) = \varnothing$. Thus,

$$\xi_1 \times \xi_2((x_{1,0}, x_{2,0}), \sigma) = \varnothing$$

For any $(x_1, x_2) \in X_1 \times X_2 - \{(x_{1,0}, x_{2,0})\}$ and $\sigma \in (\Sigma_1 \cup \Sigma_2)$, since G_1 and G_2 are standardized, we have

$$\xi_1 \times \xi_2((x_1,x_2),\sigma) \neq \varnothing \Rightarrow (x_1,x_2) \neq (x_{1,0}, x_{2,0})$$

Finally, for any $(x_1, x_2) \in X_1 \times X_2$, we have

$$\begin{aligned}
(x_1,x_2) \in X_{1,m} \times X_{2,m} \quad &\Rightarrow \quad x_1 \in X_{1,m} \wedge x_2 \in X_{2,m} \\
&\Rightarrow \quad x_1 \in \xi_1(x_1,\mu) \wedge x_2 \in \xi_2(x_2,\mu) \text{ because } G_1 \text{ and } G_2 \text{ are standardized} \\
&\Rightarrow \quad (x_1,x_2) \in \xi_1 \times \xi_2((x_1,x_2),\mu)
\end{aligned}$$

Thus, $G_1 \times G_2$ is standardized. ■

2. Proof of Lemma 2.5: As in Def. 2.4, let $G = (X, \Sigma, \xi, x_0, X_m)$ and $G/\approx_{\Sigma'} = (Y, \Sigma', \eta, y_0, Y_m)$. Then for any $y \in Y = X/\approx_{\Sigma'}$ we have

$$\begin{aligned}
\eta(y,\tau) \neq \varnothing \quad &\Longleftrightarrow \quad (\exists x \in y)(\exists u, u' \in (\Sigma - \Sigma')^*)\, \xi(x, u\tau u') \neq \varnothing \\
&\Longleftrightarrow \quad (\exists x' \in \xi(x,u))\, \xi(x',\tau) \neq \varnothing \\
&\Longleftrightarrow \quad (\exists x' \in \xi(x,u))\, x' = x_0 \text{ because } G \text{ is standardized} \\
&\Longleftrightarrow \quad x = x_0 \text{ because } (\forall \hat{x} \in X - \{x_0\})(\forall \sigma \in \Sigma)\, x_0 \notin \xi(\hat{x}, \sigma) \\
&\Longleftrightarrow \quad y =< x >=< x_0 >= y_0
\end{aligned}$$

Since $\tau \in \Sigma'$, for any $\sigma \in \Sigma' - \{\tau\}$ and $u \in (\Sigma - \Sigma')^*$, we have $\xi(x_0, u\sigma) = \varnothing$. Thus

$$\eta(y_0,\sigma) = \{y =< x >\in Y | (\exists u, u' \in (\Sigma - \Sigma')^*)\, x \in \xi(x_0, u\sigma u')\} = \varnothing$$

For any $y \in Y - \{y_0\}$ and $\sigma \in \Sigma'$, since G, we have

$$\begin{aligned} \eta(y,\sigma) \neq \varnothing &\Rightarrow (\exists x \in y)(\exists u, u' \in (\Sigma - \Sigma')^*)\, \xi(x, u\sigma u') \neq \varnothing \\ &\Rightarrow x_0 \notin \xi(x, u\sigma u') \text{ because } (\forall \hat{x} \in X - \{x_0\})(\forall \sigma \in \Sigma)\, x_0 \notin \xi(\hat{x}, \sigma) \\ &\Rightarrow < x_0 >= y_0 \notin \eta(y, \sigma) \end{aligned}$$

Finally, for any $y \in Y$, we have

$$\begin{aligned} y \in Y_m &\Rightarrow (\forall x \in y)\, x \in X_m \\ &\Rightarrow x \in \xi(x, \mu) \text{ because } G \text{ is standardized} \\ &\Rightarrow < x >= y \in \eta(< x >, \mu) = \eta(y, \mu) \text{ by the definition of automaton abstraction} \end{aligned}$$

Thus, $G/\approx_{\Sigma'}$ is standardized. ■

3. Proof of Prop. 2.6: Let ξ' be the transition map of $G/\approx_{\Sigma'}$. First we show that $P(B(G)) \subseteq B(G/\approx_{\Sigma'})$. For each string $s \in P(B(G))$, there exists $t \in B(G)$ with $P(t) = s$ such that

$$(\exists x \in \xi(x_0, t))(\forall t' \in \Sigma^*)\, \xi(x, t') \cap X_m = \varnothing$$

Since G is standardized, we get that $< x >\in \xi'(< x_0 >, P(t))$. Furthermore, from the condition

$$(\forall t' \in \Sigma^*)\, \xi(x, t') \cap X_m = \varnothing$$

we can derive that

$$(\forall s' \in \Sigma'^*)\, \xi'(< x >, s') \cap (X_m/\approx_{\Sigma'}) = \varnothing$$

Thus, $s = P(t) \in B(G/\approx_{\Sigma'}$.
Next we show that $B(G/\approx_{\Sigma'}) \subseteq P(B(G))$. For each string $s \in B(G/\approx_{\Sigma'})$, we have

$$(\exists < x >\in \xi'(< x_0 >, s))(\forall s' \in \Sigma'^*)\, \xi'(< x' >, s') \cap (X_m/\approx_{\Sigma'}) = \varnothing$$

Clearly, $x \notin X_m$. By the definition of automaton abstraction,

$$(\exists t \in \Sigma^*)\, P(t) = s \wedge x \in \xi(x_0, t)) \wedge (\forall t' \in \Sigma^*)\, \xi(x', t') \cap X_m \neq \varnothing \Rightarrow t' \in (\Sigma - \Sigma')^*$$

We claim that x is a blocking state of G because, otherwise, there exists $t' \in \Sigma^*$ such that $\xi(x, t') \cap X_m \neq \varnothing$. Since G is standardized, we get that $\xi(x, t'\mu) \cap X_m \neq \varnothing$. But $s'\mu \notin (\Sigma - \Sigma')^*$. Since x is a blocking state, we get $t \in B(G)$, thus $P(t) = s \in P(B(G))$. To show that $P(N(G)) \subseteq N(G/\approx_{\Sigma'})$, let $s \in P(N(G))$. Then

$$(\exists t \in N(G))\, P(t) = s \wedge \xi(x_0, t) \cap X_m \neq \varnothing$$

Since G is standardized, $\xi'(< x_0 >, P(t)) \cap (X_m/\approx_{\Sigma'}) \neq \varnothing$. Thus, $s \in N(G/\approx_{\Sigma'})$.
Finally we show that $N(G/\approx_{\Sigma'}) \subseteq P(N(G))$. Let $s \in N(G/\approx_{\Sigma'})$. Then

$$\xi'(< x_0 >, s) \cap (X_m/\approx_{\Sigma'}) \neq \varnothing$$

Thus, there exists $t \in \Sigma^*$ with $P(t) = s$ such that $\xi(x_0, t) \cap X_m \neq \varnothing$, which means $t \in N(G)$. Thus, $P(t) = s \in P(N(G))$. ■

4. Proof of Proposition 2.8: Let $G_i = (X_i, \Sigma_i, \xi_i, x_{i,0}, X_{i,m})$ with $i = 1, 2, 3$, where $\Sigma_1 = \Sigma_2 = \Sigma$ and $\Sigma_3 = \Sigma'$. Let $P : (\Sigma \cup \Sigma')^* \to \Sigma^*$ and $P' : (\Sigma \cup \Sigma')^* \to \Sigma'^*$ be natural projections. We first show that $N(G_1 \times G_3) = N(G_2 \times G_3)$. Clearly, We have $N(G_1 \times G_3) = N(G_1)||N(G_3)$. Since $G_1 \sqsubseteq G_2$, we have $N(G_1) = N(G_2)$. Thus, we have

$$N(G_1 \times G_3) = N(G_1)||N(G_3) = N(G_2)||N(G_3) = N(G_2 \times G_3)$$

To show that $B(G_1 \times G_3) \subseteq B(G_2 \times G_3)$, let $s \in B(G_1 \times G_3)$. By the definition of automaton product, there exists $x_1 \in X_1$ such tat $x_1 \in \xi_1(x_{1,0}, P(s))$. There are two cases to consider. Case 1: x_1 is a blocking state. Then $P(s) \in B(G_1) \subseteq B(G_2)$. Thus, $s \in B(G_2 \times G_3)$. Case 2: x_1 is a nonblocking state. Since $G_1 \sqsubseteq G_2$, there exists

$x_2 \in \xi_2(x_{2,0}, P(s))$ such that $N_{G_1}(x_1) \supseteq N_{G_2}(x_2)$. Since $s \in B(G_1 \times G_3)$, there exists $x_3 \in X_3$ such that $(x_1, x_3) \in \xi_1 \times \xi_3((x_{1,0}, x_{3,0}), s)$ and $N_{G_1 \times G_3}(x_1, x_3) = \varnothing$. We have

$$N_{G_2 \times G_3}(x_2, x_3) = N_{G_2}(x_2) || N_{G_3}(x_3) \subseteq N_{G_1}(x_1) || N_{G_3}(x_3) = N_{G_1 \times G_3}(x_1, x_3) = \varnothing$$

Thus, (x_2, x_3) is a blocking state of $G_2 \times G_3$, which means $s \in B(G_2 \times G_3)$. Therefore, in either case we have $B(G_1 \times G_3) \subseteq B(G_2 \times G_3)$.
Finally, follow the argument in Case 2, for any string $s \in (\Sigma \cup \Sigma')^*$ and any state $(x_1, x_3) \in \xi_1 \times \xi_3((x_{1,0}, x_{3,0}), s)$, we have $(x_2, x_3) \in \xi_2 \times \xi_3((x_{2,0}, x_{3,0}), s)$ such that $N_{G_2 \times G_3}(x_2, x_3) \subseteq N_{G_1 \times G_3}(x_1, x_3)$. ■

5. Proof of Prop. 2.10: Let $G_i = (X_i, \Sigma, \xi_i, x_{i,0}, X_{i,m})$, where $i = 1, 2$, and $P : \Sigma^* \to \Sigma'^*$ be the natural projection. Since $G_1 \sqsubseteq G_2$, by Prop. 2.6 we have

$$N(G_1/\approx_{\Sigma'}) = P(N(G_1)) = P(N(G_2)) = N(G_2/\approx_{\Sigma'})$$

To show $B(G_1/\approx_{\Sigma'}) \subseteq B(G_2/\approx_{\Sigma'})$, let ξ_i' $(i = 1, 2)$ be the transition map of $G_i/\approx_{\Sigma'}$. For any $s \in B(G_1/\approx_{\Sigma'})$, there exists $x_1 \in X_1$ such that

$$< x_1 > \in \xi_1'(< x_{1,0} >, s) \wedge (\forall s' \in \Sigma'^*)\, \xi_1'(< x_1 >, s') \cap (X_{1,m}/\approx_{\Sigma'}) = \varnothing$$

which means there exists $t \in \Sigma^*$ such that

$$P(t) = s \wedge x_1 \in \xi_1(x_{1,0}, t) \wedge (\forall t' \in \Sigma^*)\, \xi_1(x_1, t') \cap X_{1,m} \neq \varnothing \Rightarrow t' \in (\Sigma - \Sigma')^*$$

Thus, $N_{G_1}(x_1) \subseteq (\Sigma - \Sigma')^*$. There are two cases. Case 1: x_1 is a blocking state of G_1. Then $t \in B(G_1)$, which means $s = P(t) \in P(B(G_1))$. Since $G_1 \sqsubseteq G_2$, we have $B(G_1) \subseteq B(G_2)$. Thus, by Prop. 2.6, we have $s \in P(B(G_2)) \subseteq B(G_2/\approx_{\Sigma'})$. Case 2: x_1 is a nonblocking state. Thus $t \in N(G_1)$. Clearly $x_1 \notin X_{1,m}$. Since $G_1 \sqsubseteq G_2$, there exists $x_2 \in X_2$ such that

$$x_2 \in \xi_2(x_{2,0}, t) \wedge N_{G_1}(x_1) = N_{G_2}(x_2) \wedge [x_1 \in X_{1,m} \iff x_2 \in X_{2,m}]$$

Since $N_{G_1}(x_1) \subseteq (\Sigma - \Sigma')^*$, we have

$$(\forall t' \in \Sigma^*)\, \xi_2(x_2, t') \cap X_{1,m} \neq \varnothing \Rightarrow t' \in (\Sigma - \Sigma')^*$$

Since $x_1 \notin X_{1,m}$, we have $x_2 \notin X_{2,m}$. Thus, by the definition of automaton abstraction,

$$< x_2 > \in \xi_2'(< x_{2,0} >, P(t)) \wedge (\forall s' \in \Sigma'^*)\, \xi_2'(< x_2 >, s') \cap (X_{2,m}/\approx_{\Sigma'}) = \varnothing$$

which means $s = P(t) \in B(G_2/\approx_{\Sigma'})$. Thus, in either case $B(G_1/\approx_{\Sigma'}) \subseteq B(G_2/\approx_{\Sigma'})$. Finally, for each $s \in \overline{N(G_1/\approx_{\Sigma'})}$, there exists $x_1 \in X_1$ such that

$$< x_1 > \in \xi_1'(< x_{1,0} >, s) \wedge (\exists s' \in \Sigma'^*)\, \xi_1'(< x_1 >, s') \cap (X_{1,m}/\approx_{\Sigma'}) \neq \varnothing$$

which means there exists $t \in \Sigma^*$ with $P(t) = s$ such that

$$x_1 \in \xi_1'(x_{1,0}, t) \wedge (\exists t' \in \Sigma'^*)\, P(t') = s' \wedge \xi_1(x_1, t') \cap X_{1,m} \neq \varnothing$$

Clearly, $t \in \overline{N(G_1)}$. Thus, by $G_1 \sqsubseteq G_2$, we have

$$(\exists x_2 \in \xi_2(x_{2,0}, t))\, N_{G_1}(x_1) \supseteq N_{G_2}(x_2) \wedge [x_1 \in X_{1,m} \iff x_2 \in X_{2,m}]$$

Since G_2 is standardized, we get $< x_2 > \in \xi_2'(< x_{2,0} >, s)$. For any $s' \in N_{G_2/\approx_{\Sigma'}}(< x_2 >)$, we have $\xi_2'(< x_2 >, s') \cap X_{2,m}/\approx_{\Sigma'} \neq \varnothing$. Thus, there exists $t' \in \Sigma^*$ with $P(t') = s'$ such that $\xi_2(x_2, t') \cap X_{2,m} \neq \varnothing$. Since $\mu \in s'$, we have $\mu \in t'$. Thus, $t' \in N_{G_2}(x_2) \subseteq N_{G_1}(x_1)$. So

$$\xi_1'(< x_1 >, s') \cap (X_{1,m}/\approx_{\Sigma'}) \neq \varnothing$$

namely $s' \in N_{G_1/\approx_{\Sigma'}}(< x_1 >)$. Thus, $N_{G_2/\approx_{\Sigma'}}(x_2) \subseteq N_{G_1/\approx_{\Sigma'}}(x_1)$. ■

6. Proof of Prop. 2.12: Let ξ'' be the transition map of $G/\approx_{\Sigma''}$, and ξ''' be the transition map of $(G/\approx_{\Sigma'})/\approx_{\Sigma''}$. Let $P_{12} : \Sigma^* \to \Sigma'^*$, $P_{13} : \Sigma^* \to \Sigma''^*$ and $P_{23} : \Sigma'^* \to \Sigma''^*$ be

natural projections. We first show that $G/\approx_{\Sigma''} \sqsubseteq (G/\approx_{\Sigma'})/\approx_{\Sigma''}$.
By Prop. 2.6 we have

$$N(G/\approx_{\Sigma''}) = P_{13}(N(G)) = P_{23}(P_{12}(N(G))) = P_{23}(N(G/\approx_{\Sigma'})) = N((G/\approx_{\Sigma'})/\approx_{\Sigma''})$$

We now show $B(G/\approx_{\Sigma''}) \subseteq B((G/\approx_{\Sigma'})/\approx_{\Sigma''})$. Let $s \in B(G/\approx_{\Sigma''})$. There exists $x \in X$ such that

$$< x >_{\Sigma''} \in \xi''(< x_0 >_{\Sigma''}, s) \wedge (\forall s' \in \Sigma''^*)\, \xi''(< x >_{\Sigma''}, s') \cap X_m/\approx_{\Sigma''} = \varnothing$$

Thus, there exists $t \in \Sigma^*$ with $P_{13}(t) = s$ such that

$$x \in \xi(x_0, t) \wedge (\forall t' \in \Sigma^*)\, \xi(x, t') \cap X_m \neq \varnothing \Rightarrow t' \in (\Sigma - \Sigma'')^* \tag{2}$$

We have two cases to consider. Case 1: x is a blocking state. Then clearly $t \in B(G)$. By Prop. 2.6, $P_{13}(t) = s = P_{23}(P_{12}(t)) \in P_{23}(P_{12}(B(G))) \subseteq B((G/\approx_{\Sigma'})/\approx_{\Sigma''})$. Case 2: x is a nonblocking state. Clearly $x \notin X_m$, which means $<< x >_{\Sigma'}>_{\Sigma''} \notin (X_m/\approx_{\Sigma'})/\approx_{\Sigma''}$. Thus, from Expression (2) and the definition of automaton abstraction, we get that

$$<< x >_{\Sigma'}>_{\Sigma''} \in \xi'''(<< x_0 >_{\Sigma'}>_{\Sigma''}, s) \wedge (\forall s' \in \Sigma''^*)\, \xi'''(<< x >_{\Sigma'}>_{\Sigma''}, s') \cap ((X_m/\approx_{\Sigma'})/\approx_{\Sigma''}) = \varnothing$$

Thus, $s \in B((G/\approx_{\Sigma'})/\approx_{\Sigma''})$. In either case we have $B(G/\approx_{\Sigma''}) \subseteq B((G/\approx_{\Sigma'})/\approx_{\Sigma''})$.
Let $s \in \overline{N(G/\approx_{\Sigma''})}$. For any $x \in X$ with $< x >_{\Sigma''} \in \xi''(< x_0 >_{\Sigma''}, s)$, we have that

$$(\exists t \in \Sigma^*)\, P_{13}(t) = s \wedge x \in \xi(x_0, t)$$

Since G is standardize, if $s = \epsilon$, then $t = \epsilon$, which means $x = x_0$. Clearly, we have the following expression: $<< x_0 >_{\Sigma'}>_{\Sigma''} \in \xi'''(<< x_0 >_{\Sigma'}>_{\Sigma''}, \epsilon)$. If $s \neq \epsilon$, then by the definition of automaton abstraction and the assumption that $\Sigma'' \subseteq \Sigma' \subseteq \Sigma$, we get that $<< x >_{\Sigma'}>_{\Sigma''} \in \xi'''(<< x_0 >_{\Sigma'}>_{\Sigma''}, s)$. Thus, in either case we have $<< x >_{\Sigma'}>_{\Sigma''} \in \xi'''(<< x_0 >_{\Sigma'}>_{\Sigma''}, s)$. We now show that

$$N_{(G/\approx_{\Sigma'})/\approx_{\Sigma''}}(<< x >_{\Sigma'}>_{\Sigma''}) \subseteq N_{G/\approx_{\Sigma''}}(< x >_{\Sigma''})$$

Let $s' \in N_{(G/\approx_{\Sigma'})/\approx_{\Sigma''}}(<< x >_{\Sigma'}>_{\Sigma''})$. Then there exists $t' \in \Sigma^*$ with $P_{13}(t') = s'$ such that $\xi(x, t') \cap X_m \neq \varnothing$. Since $\mu \in s'$, we have $\mu \in t'$. Thus, $P_{13}(t') \neq \epsilon$. By the definition of automaton abstraction, we get that $\xi''(< x >_{\Sigma''}, P_{13}(t')) \cap X_m/\approx_{\Sigma''} \neq \varnothing$. Thus, $s' \in N_{G/\approx_{\Sigma''}}(< x >_{\Sigma''})$, namely $N_{(G/\approx_{\Sigma'})/\approx_{\Sigma''}}(<< x >_{\Sigma'}>_{\Sigma''}) \subseteq N_{G/\approx_{\Sigma''}}(< x >_{\Sigma''})$.
Next, to show $(G/\approx_{\Sigma'})/\approx_{\Sigma''} \sqsubseteq G/\approx_{\Sigma''}$, we first show $B((G/\approx_{\Sigma'})/\approx_{\Sigma''}) \subseteq B(G/\approx_{\Sigma''})$.
Let $s \in B((G/\approx_{\Sigma'})/\approx_{\Sigma''})$. Then there exists $x \in X$ such that

$$<< x >_{\Sigma'}>_{\Sigma''} \in \xi'''(<< x_0 >_{\Sigma'}>_{\Sigma''}, s) \wedge (\forall s' \in \Sigma''^*)\, \xi'''(<< x >_{\Sigma'}>_{\Sigma''}, s') \cap ((X_m/\approx_{\Sigma'})/\approx_{\Sigma''}) = \varnothing$$

Thus, there exists $t \in \Sigma^*$ with $P_{13}(t) = s$ such that

$$x \in \xi(x_0, t) \wedge (\forall t' \in \Sigma^*)\, \xi(x, t') \cap X_m \neq \varnothing \Rightarrow t' \in (\Sigma - \Sigma'')^* \tag{3}$$

We have two cases to consider. Case 1: x is a blocking state. Then clearly $t \in B(G)$. By Prop. 2.6, $P_{13}(t) = s \in P_{13}(B(G)) \subseteq B(G/\approx_{\Sigma''})$. Case 2: x is a nonblocking state. Clearly $x \notin X_m$, which means $< x >_{\Sigma''} \notin X_m/\approx_{\Sigma''}$. Thus, from Expression (3) we get that

$$< x >_{\Sigma''} \in \xi''(< x_0 >_{\Sigma''}, s) \wedge (\forall s' \in \Sigma''^*)\, \xi''(< x >_{\Sigma''}, s') \cap (X_m/\approx_{\Sigma''}) = \varnothing$$

Thus, $s \in B(G/\approx_{\Sigma''})$. In either case we have $B((G/\approx_{\Sigma'})/\approx_{\Sigma''}) \subseteq B(G/\approx_{\Sigma''})$.
Let $s \in \overline{N((G/\approx_{\Sigma'})/\approx_{\Sigma''})}$. For any $x \in X$ with $<< x >_{\Sigma'}>_{\Sigma''} \in \xi'''(<< x_0 >_{\Sigma'}>_{\Sigma''}, s)$, we have

$$(\exists t \in \Sigma^*)\, P_{13}(t) = s \wedge x \in \xi(x_0, t)$$

Since G is standardize, if $s = \epsilon$, then $t = \epsilon$, which means $x = x_0$. Clearly, we have the following expression: $< x_0 >_{\Sigma''} \in \xi''(< x_0 >_{\Sigma''}, \epsilon)$. If $s \neq \epsilon$, then by the assumption that $\Sigma'' \subseteq \Sigma' \subseteq \Sigma$, we get $< x >_{\Sigma''} \in \xi''(< x_0 >_{\Sigma''}, s)$. Thus, in either case we have $< x >_{\Sigma''} \in \xi''(< x_0 >_{\Sigma''}, s)$. We now show that

$$N_{G/\approx_{\Sigma''}}(< x >_{\Sigma''}) \subseteq N_{(G/\approx_{\Sigma'})/\approx_{\Sigma''}}(<< x >_{\Sigma'}>_{\Sigma''})$$

Let $s' \in N_{G/\approx_{\Sigma''}}(< x >_{\Sigma''})$. Then

$$(\exists t' \in \Sigma^*)\, P_{13}(t') = s' \wedge \xi(x, t') \cap X_m \neq \varnothing$$

Since $\mu \in s'$, we have $\mu \in t'$. Thus, $P_{13}(t') \neq \epsilon$. By the definition of abstraction, we have

$$\xi'''(<< x >_{\Sigma'}>_{\Sigma''}, P_{13}(t')) \cap ((X_m/\approx_{\Sigma'})/\approx_{\Sigma''}) \neq \varnothing$$

Thus, $s' \in N_{(G/\approx_{\Sigma'})/\approx_{\Sigma''}}(<< x >_{\Sigma'}>_{\Sigma''})$, namely

$$N_{G/\approx_{\Sigma''}}(< x >_{\Sigma''}) \subseteq N_{(G/\approx_{\Sigma'})/\approx_{\Sigma''}}(<< x >_{\Sigma'}>_{\Sigma''})$$

The proposition follows. ■

7. Proof of Prop. 2.13: Let $G_i = (X_i, \Sigma_i, \xi_i, x_{i,0}, X_{i,m}) \in \phi(\Sigma_i)$ with $i = 1, 2$. For notation simplicity let $\hat{\Sigma}_i = \Sigma_i \cap \Sigma'$, and $P : (\Sigma_1 \cup \Sigma_2)^* \to \Sigma'^*$, $P_i : \Sigma_i^* \to \hat{\Sigma}_i^*$, $\hat{P}_i : \Sigma'^* \to \hat{\Sigma}_i^*$ and $Q_i : (\Sigma_1 \cup \Sigma_2)^* \to \Sigma_i^*$ be natural projections, ξ' the transition map of $(G_1 \times G_2)/\approx_{\Sigma'}$ and ξ_i' be the transition map of $G_i/\approx_{\hat{\Sigma}_i}$ $(i = 1, 2)$.
First, we have the following,

$$\begin{aligned} N((G_1 \times G_2)/\approx_{\Sigma'}) &= P(N(G_1 \times G_2)) \text{ by Prop. 2.6} \\ &= P(N(G_1)||N(G_2)) \\ &= P_1(N(G_1))||P_2(N(G_2)) \text{ because } \Sigma_1 \cap \Sigma_2 \subseteq \Sigma' \\ &= N(G_1/\approx_{\hat{\Sigma}_1})||N(G_2/\approx_{\hat{\Sigma}_2}) \text{ by Prop. 2.6} \\ &= N((G_1/\approx_{\hat{\Sigma}_1}) \times (G_2/\approx_{\hat{\Sigma}_2})) \end{aligned}$$

Next, we show

$$B((G_1 \times G_2)/\approx_{\Sigma'}) \subseteq B((G_1/\approx_{\hat{\Sigma}_1}) \times (G_2/\approx_{\hat{\Sigma}_2}))$$

Let $s \in B((G_1 \times G_2)/\approx_{\Sigma'})$. Then there exists $(x_1, x_2) \in X_1 \times X_2$ such that

$$< (x_1, x_2) >_{\Sigma'} \in \xi'(< (x_{1,0}, x_{2,0}) >_{\Sigma'}, s) \wedge (\forall s' \in \Sigma'^*)\, \xi'(< (x_1, x_2) >_{\Sigma'}, s') \cap ((X_{1,m} \times X_{2,m})/\approx_{\Sigma'}) = \varnothing$$

which means $(x_1, x_2) \notin X_{1,m} \times X_{2,m}$ and there exists $t \in (\Sigma_1 \cup \Sigma_2)^*$ with $P(t) = s$ such that

$$(x_1, x_2) \in \xi_1 \times \xi_2((x_{1,0}, x_{2,0}), t) \wedge (\forall t' \in \Sigma^*)\, \xi_1 \times \xi_2((x_1, x_2), t') \cap (X_{1,m} \times X_{2,m}) \neq \varnothing \Rightarrow t' \in ((\Sigma_1 \cup \Sigma_2) - \Sigma')^*$$

Since G_1 and G_2 are standardized, from $(x_1, x_2) \in \xi_1 \times \xi_2((x_{1,0}, x_{2,0}), t)$ and the fact that $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma'$ we can derive that

$$(< x_1 >_{\hat{\Sigma}_1}, < x_2 >_{\hat{\Sigma}_2}) \in \xi_1' \times \xi_2'((< x_{1,0} >_{\hat{\Sigma}_1}, < x_{2,0} >_{\hat{\Sigma}_1}), s)$$

We claim that $(< x_1 >_{\hat{\Sigma}_1}, < x_2 >_{\hat{\Sigma}_2})$ is a blocking state of $(G_1/\approx_{\hat{\Sigma}_1}) \times (G_2/\approx_{\hat{\Sigma}_2})$. Otherwise, there exists $s' \in \Sigma'^*$ such that

$$\xi_1' \times \xi_2'((< x_1 >_{\hat{\Sigma}_1}, < x_2 >_{\hat{\Sigma}_2}), s') \cap ((X_{1,m}/\approx_{\hat{\Sigma}_1}) \times (X_{2,m}/\approx_{\hat{\Sigma}_2})) \neq \varnothing$$

Since $(x_1, x_2) \notin X_{1,m} \times X_{2,m}$, we get $(< x_1 >_{\hat{\Sigma}_1}, < x_2 >_{\hat{\Sigma}_2}) \notin (X_{1,m}/\approx_{\hat{\Sigma}_1}) \times (X_{2,m}/\approx_{\hat{\Sigma}_2})$. Thus, $s' \neq \epsilon$, which means there exists $t' \in \Sigma^*$ with $P(t') = s' \notin ((\Sigma_1 \cup \Sigma_2) - \Sigma')^*$ such that $\xi_1 \times \xi_2((x_1, x_2), t') \cap (X_{1,m} \times X_{2,m}) \neq \varnothing$ - contradict the fact that

$$(\forall t' \in \Sigma^*)\, \xi_1 \times \xi_2((x_1, x_2), t') \cap (X_{1,m} \times X_{2,m}) \neq \varnothing \Rightarrow t' \in ((\Sigma_1 \cup \Sigma_2) - \Sigma')^*$$

From the claim we get that $s \in B((G_1/\approx_{\hat{\Sigma}_1}) \times (G_2/\approx_{\hat{\Sigma}_2}))$.
Let $s \in \overline{N((G_1 \times G_2)/\approx_{\Sigma'})}$. For any $(x_1, x_2) \in X_1 \times X_2$ with

$$< (x_1, x_2) >_{\Sigma'} \in \xi'(< (x_{1,0}, x_{2,0}) >_{\Sigma'}, s)$$

we have

$$(\exists t \in \Sigma^*)\, P_(t) = s \wedge (x_1, x_2) \in \xi((x_{1,0}, x_{2,0}), t)$$

Since G_1 and G_2 are standardize, if $s = \epsilon$, then $t = \epsilon$, which means $(x_1, x_2) = (x_{1,0}, x_{2,0})$. Clearly, we have the following expression:

$$(< x_{1,0} >_{\hat{\Sigma}_1}, < x_{2,0} >_{\hat{\Sigma}_2}) \in \xi_1' \times \xi_2'((< x_{1,0} >_{\hat{\Sigma}_1}, < x_{2,0} >_{\hat{\Sigma}_2}), \epsilon)$$

If $s \neq \epsilon$, then by the assumption that $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma'$, we get

$$(< x_1 >_{\hat{\Sigma}_1}, < x_2 >_{\hat{\Sigma}_2}) \in \xi_1' \times \xi_2'((< x_{1,0} >_{\hat{\Sigma}_1}, < x_{2,0} >_{\hat{\Sigma}_2}), s)$$

Thus, in either case we have

$$(< x_1 >_{\hat{\Sigma}_1}, < x_2 >_{\hat{\Sigma}_2}) \in \xi_1' \times \xi_2'((< x_{1,0} >_{\hat{\Sigma}_1}, < x_{2,0} >_{\hat{\Sigma}_2}), s)$$

We now show that

$$N_{(G_1/\approx_{\hat{\Sigma}_1})\times(G_2/\approx_{\hat{\Sigma}_2})}(< x_1 >_{\hat{\Sigma}_1}, < x_2 >_{\hat{\Sigma}_2}) \subseteq N_{(G_1\times G_2)/\approx_{\Sigma'}}(< (x_1, x_2) >_{\Sigma'})$$

Let $s' \in N_{(G_1/\approx_{\hat{\Sigma}_1})\times(G_2/\approx_{\hat{\Sigma}_2})}(< x_1 >_{\hat{\Sigma}_1}, < x_2 >_{\hat{\Sigma}_2})$. Then there exists $t' \in \Sigma^*$ with $P(t') = s'$ such that $\xi_1 \times \xi_2((x_1, x_2), t') \cap (X_{1,m} \times X_{2,m}) \neq \varnothing$. Since $\mu \in s'$, we have $\mu \in t'$. Thus, $P(t') \neq \epsilon$. By the definition of automaton abstraction, we have

$$\xi'(< (x_1, x_2) >_{\Sigma'}, P(t')) \cap (X_{1,m} \times X_{2,m})/\approx_{\Sigma'} \neq \varnothing$$

Thus, $s' \in N_{(G_1\times G_2)/\approx_{\Sigma'}}(< (x_1, x_2) >_{\Sigma'})$, namely

$$N_{(G_1/\approx_{\hat{\Sigma}_1})\times(G_2/\approx_{\hat{\Sigma}_2})}(< x_1 >_{\hat{\Sigma}_1}, < x_2 >_{\hat{\Sigma}_2}) \subseteq N_{(G_1\times G_2)/\approx_{\Sigma'}}(< (x_1, x_2) >_{\Sigma'})$$

Thus, $(G_1 \times G_2)/\approx_{\Sigma'} \sqsubseteq (G_1/\approx_{\hat{\Sigma}_1}) \times (G_2/\approx_{\hat{\Sigma}_2})$.
To show $(G_1/\approx_{\hat{\Sigma}_1}) \times (G_2/\approx_{\hat{\Sigma}_2}) \sqsubseteq (G_1 \times G_2)/\approx_{\Sigma'}$, we first show that

$$B((G_1/\approx_{\hat{\Sigma}_1}) \times (G_2/\approx_{\hat{\Sigma}_2})) \subseteq B((G_1 \times G_2)/\approx_{\Sigma'})$$

Let $s \in B((G_1/\approx_{\hat{\Sigma}_1}) \times (G_2/\approx_{\hat{\Sigma}_2}))$. Then there exists $(x_1, x_2) \in X_1 \times X_2$ such that

$$(< x_1 >_{\hat{\Sigma}_1}, < x_2 >_{\hat{\Sigma}_2}) \in \xi_1' \times \xi_2'(< (< x_{1,0} >_{\hat{\Sigma}_1}, < x_{2,0} >_{\hat{\Sigma}_2}), s) \tag{4}$$

and

$$(\forall s' \in \Sigma'^*)\, \xi_1' \times \xi_2'((< x_1 >_{\hat{\Sigma}_1}, < x_2 >_{\hat{\Sigma}_2}), s') \cap ((X_{1,m}/\approx_{\hat{\Sigma}_1}) \times (X_{2,m}/\approx_{\hat{\Sigma}_2})) = \varnothing \tag{5}$$

From Expression (4) we get that

$$(\exists t \in (\Sigma_1 \cup \Sigma_2)^*)\, P(t) = s \wedge (x_1, x_2) \in \xi_1 \times \xi_2((x_{1,0}, x_{2,0}), t) \tag{6}$$

From Expression (5) we get that $(x_1, x_2) \notin X_{1,m} \times X_{2,m}$. Since G_1 and G_2 are standardized, from Expression (6) and the fact that $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma'$ we have

$$< (x_1, x_2) >_{\Sigma'} \in \xi'(< (x_{1,0}, x_{2,0}) >_{\Sigma'}, s)$$

We claim that $< (x_1, x_2) >_{\Sigma'}$ is a blocking state of $(G_1 \times G_2)/\approx_{\Sigma'}$. Otherwise, there exists $s' \in \Sigma'^*$ such that

$$\xi'(< (x_1, x_2) >_{\Sigma'}, s') \cap ((X_{1,m} \times X_{2,m})/\approx_{\Sigma'}) \neq \varnothing$$

Since G_1 and G_2 are standardized, we get that

$$\xi'(< (x_1, x_2) >_{\Sigma'}, s'\mu) \cap ((X_{1,m} \times X_{2,m})/\approx_{\Sigma'}) \neq \varnothing$$

Clearly, $\hat{P}_i(s'\mu) \neq \epsilon$. Thus, there exists $t' \in \Sigma^*$ with $P(t') = s'\mu$ such that

$$\xi_1 \times \xi_2((x_1, x_2), t'\mu) \cap (X_{1,m} \times X_{2,m}) \neq \varnothing$$

Since $\hat{P}_i(s'\mu) \neq \epsilon$ for $i = 1, 2$ and $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma'$, we have

$$\xi_1' \times \xi_2'((< x_1 >_{\hat{\Sigma}_1}, < x_2 >_{\hat{\Sigma}_2}), s'\mu) \cap ((X_{1,m}/\approx_{\hat{\Sigma}_1}) \times (X_{2,m}/\approx_{\hat{\Sigma}_2})) \neq \varnothing$$

which contradicts Expression (5). Thus, the claim is true, namely $s \in B((G_1 \times G_2)/\approx_{\Sigma'})$.
Let $s \in \overline{N((G_1/\approx_{\hat{\Sigma}_1}) \times (G_2/\approx_{\hat{\Sigma}_2}))}$. For any $(x_1, x_2) \in X_1 \times X_2$ with

$$(< x_1 >_{\hat{\Sigma}_1}, < x_2 >_{\hat{\Sigma}_2}) \in \xi_1' \times \xi_2'((< x_{1,0} >_{\hat{\Sigma}_1}, < x_{2,0} >_{\hat{\Sigma}_2}), s)$$

we have

$$(\exists t \in \Sigma^*)\, P_(t) = s \wedge (x_1, x_2) \in \xi((x_{1,0}, x_{2,0}), t)$$

Since G_1 and G_2 are standardize, if $s = \epsilon$, then $t = \epsilon$, which means $(x_1, x_2) = (x_{1,0}, x_{2,0})$. Clearly, we have the following expression:

$$< (x_{1,0}, x_{2,0}) >_{\Sigma'} \in \xi'(< (x_{1,0}, x_{2,0}) >_{\Sigma'}, \epsilon)$$

If $s \neq \epsilon$, then by the assumption that $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma'$, we get

$$< (x_1, x_2) >_{\Sigma'} \in \xi'(< (x_{1,0}, x_{2,0}) >_{\Sigma'}, s)$$

Thus, in either case we have

$$< (x_1, x_2) >_{\Sigma'} \in \xi'(< (x_{1,0}, x_{2,0}) >_{\Sigma'}, s)$$

We now show that

$$N_{(G_1 \times G_2)/\approx_{\Sigma'}}(< (x_1, x_2) >_{\Sigma'}) \subseteq N_{(G_1/\approx_{\hat{\Sigma}_1}) \times (G_2/\approx_{\hat{\Sigma}_2})}(< x_1 >_{\hat{\Sigma}_1}, < x_2 >_{\hat{\Sigma}_2})$$

Let $s' \in N_{(G_1 \times G_2)/\approx_{\Sigma'}}(< (x_1, x_2) >_{\Sigma'})$. Then there exists $t' \in \Sigma^*$ with $P(t') = s'$ such that $\xi_1 \times \xi_2((x_1, x_2), t') \cap (X_{1,m} \times X_{2,m}) \neq \varnothing$. Since $\mu \in s'$, we have $\mu \in t'$. Thus, $\hat{P}_i(P(t')) \neq \epsilon$ $(i = 1, 2)$. By the definition of automaton abstraction and $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma'$, we have

$$\xi_1' \times \xi_2'((< x_1 >_{\hat{\Sigma}_1}, < x_2 >_{\hat{\Sigma}_2}), P(t')) \cap ((X_{1,m}/\approx_{\hat{\Sigma}_1}) \times (X_{2,m}/\approx_{\hat{\Sigma}_2})) \neq \varnothing$$

Thus, $s' \in N_{(G_1/\approx_{\hat{\Sigma}_1}) \times (G_2/\approx_{\hat{\Sigma}_2})}(< x_1 >_{\hat{\Sigma}_1}, < x_2 >_{\hat{\Sigma}_2})$, namely

$$N_{(G_1 \times G_2)/\approx_{\Sigma'}}(< (x_1, x_2) >_{\Sigma'}) \subseteq N_{(G_1/\approx_{\hat{\Sigma}_1}) \times (G_2/\approx_{\hat{\Sigma}_2})}(< x_1 >_{\hat{\Sigma}_1}, < x_2 >_{\hat{\Sigma}_2})$$

Thus, $(G_1/\approx_{\hat{\Sigma}_1}) \times (G_2/\approx_{\hat{\Sigma}_2}) \sqsubseteq (G_1 \times G_2)/\approx_{\Sigma'}$. ■

Bibliography

[1] P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete event systems. *SIAM J. Control and Optimization*, 25(1):206–230, 1987.

[2] W.M. Wonham and P.J. Ramadge. On the supremal controllable sublanguage of a given language. *SIAM J. Control and Optimization*, 25(3):637–659, 1987.

[3] W.M. Wonham and P.J. Ramadge. Modular supervisory control of discrete event systems. *Maths. of Control, Signals & Systems*, 1(1):13–30, 1988.

[4] M.H. de Queiroz and J.E.R. Cury. Modular supervisory control of composed systems. In *Proc. of American Control Conference*, pages 4051–4055, Chicago, USA, June 2000.

[5] R.J. Leduc, M. Lawford and W.M. Wonham. Hierarchical interface-based supervisory control-part II: parallel case. *IEEE Trans. Automatic Control*, 50(9):1336–1348, 2005.

[6] L. Feng and W.M. Wonham. Computationally efficient supervisor design: abstraction and modularity. In *Proc. 8th International Workshop on Discrete Event Systems (WODES06)*, pages 3–8, 2006.

[7] P.N. Pena, J.E.R. Cury and S. Lafortune. Testing modularity of local supervisors: an approach based on abstractions. In *Proc. 8th International Workshop on Discrete Event Systems (WODES06)*, pages 107–112, 2006.

[8] P.N. Pena, A.E. Carrilho da Cunha, J.E.R. Cury and S. Lafortune. New results on the nonconflict test of modular supervisors. In *Proc. 9th International Workshop on Discrete Event Systems (WODES08)*, pages 468–473, 2008.

[9] K. Schmidt and C. Breindl. On maximal permissiveness of hierarchical and modular supervisory control approaches for discrete event systems. In *Proc. 9th International Workshop on Discrete Event Systems (WODES08)*, pages 462–467, 2008.

[10] K.C. Wong and W.M. Wonham. On the computation of observers in discrete-event systems. *Discrete Event Dynamic Systems*, 14(1):55-107, 2004.

[11] R. Su and J.G. Thistle. A distributed supervisor synthesis approach based on weak bisimulation. In *Proc. 8th International Workshop on Discrete Event Systems (WODES06)*, pages 64–69, 2006.

[12] R. Su, J.H. van Schuppen and J.E. Rooda. Synthesizing nonblocking distributed supervisors based on automaton abstraction. In *Proc. 47th IEEE Conference on Decision and Control (CDC09)*, 2008.

[13] R. Su, J.H. van Schuppen and J.E. Rooda. *Model abstraction of nondeterministic finite state automata in supervisor synthesis*. submitted to *IEEE Trans. Automatic Control*, 2008.

[14] W. M. Wonham. *Supervisory Control of Discrete-Event Systems*. Systems Control Group, Dept. of ECE, University of Toronto. URL: www.control.utoronto.ca/DES, July 1, 2007.

[15] *Design Software: XPTCT (Version 121 for Windows 98/XP)*. Systems Control Group, Dept. of ECE, University of Toronto. URL: www.control.utoronto.ca/DES, July 1, 2008.

[16] J.C. Fernandez. An implementation of an efficient algorithm for bisimulation equivalence. *Science of Computer Programming*, 13(2-3): 219-236, 1990

[17] H. Flordal and R. Malik. Modular nonblocking verification using conflict equivalence. In *Proc. 8th International Workshop on Discrete Event Systems (WODES06)*, pages 100–106, 2006.

[18] H. Flordal, R. Malik, M. Fabian and K. Akesson. Compositional synthesis of maximally permissive supervisors using supervisor equivalence. In *Discrete Event Dynamic Systems*, 17(4):475-504, 2007.

[19] R. Malik and H. Flordal. Yet another approach to compositional synthesis of discrete event systems. In *Proc. 9th International Workshop on Discrete Event Systems (WODES08)*, pages 16–21, 2008.

[20] R.C. Hill, D.M. Tilbury and S. Lafortune. Modular supervisory control with equivalence-based conflict resolution. In *Proc. 2008 American Control Conference (ACC08)*, pages 491–498, 2008.

[21] R. Milner. Operational and algebraic semantics of concurrent processes. *Handbook of theoretical computer science (vol. B): formal models and semantics*, pp. 1201-1242, MIT Press, 1990

[22] R. Su and W.M. Wonham. Global and local consistencies in distributed fault diagnosis for discrete-event systems. *IEEE Trans. Automatic Control*, 50(12):1923-1935, 2005.