State-Vector Transition Model Applied to Supervisory Control

State-Vector Transition Model Applied to Supervisory Control

Bengt Lennartson, Sajed Miremadi, Zhennan Fei,
Mona Noori Hosseini, Martin Fabian, and Knut Åkesson
Automation Research Group, Signals and Systems, Chalmers University of Technology
SE-412 96 Gothenburg, Sweden
{bengt.lennartson, miremads, zhennan, noori, fabian, knut.akesson}@chalmers.se

Abstract

In supervisory control theory, a supervisor restricts the plant in order to fulfill given specifications. A problem for larger industrial applications is that the resulting supervisor is not easily implemented and comprehensible for the users. To tackle this problem, an efficient method has recently been introduced to characterize a supervisor by tractable logic conditions, referred to as guards. This approach has been developed for a specific type of automata with variables called extended finite automata (EFAs). An extension of this approach to a more general class of models is presented in this paper. It means that classical supervisory control problems for automata and Petri nets are easily and efficiently solved, but also generalized based on the suggested approach. The synthesis procedure is naturally modeled and efficiently computed based on binary decision diagrams.

1 Introduction

Supervisory control theory (SCT) [13] means that a control function, called *supervisor*, is automatically synthesized based on a plant model (the system to be controlled) and a specification. The supervisor restricts the behavior of the plant to ensure that the system never violates the given specification. A standard approach to determine a synthesized supervisor is to explicitly represent all states that are allowed to be reached in the closed-loop system. However, the resulting supervisor may then require more memory than available, and the final supervisor is a black box, where it is not clear from a user perspective why some events become disabled after the synthesis. One way to obtain compact and comprehensible models for the supervisor as well as the plant is to combine discrete states/locations with variables. The variables may then appear in *guards* and *actions*. Guard expressions at the transitions restrict the behavior of the system, while actions update the variables.

There exist a number of frameworks that are based on automata extended with variables such as [17, 2]. In [17], the states of a given supervisor are encoded using Boolean variables, but the variables are used in guards and actions attached to the events (not transitions) of the model. In [2], to ensure a least restrictive supervisor it is assumed that all variables are local, i.e., not shared between automata.

In [14, 11], an extended framework called *extended finite automata* (EFAs) is presented that overcomes the above-mentioned restrictions, making the framework suitable for SCT. For instance, it is possible to update the variables, which are global, in different automata, and to use EFAs to model both plant and supervisor. In [12] a supervisor is synthesized based on an EFA plant and a set of forbidden locations. For large problems the approach can suffer from an early state-space explosion while generating the plant with the forbidden locations. In addition, it is not allowed to assign new values to some variables in the plant and other variables in the specification. In this paper we show that this can be a serious restriction. Indeed, the assignment of variables in different local models, and its formal treatment, is a key contribution of this paper.

In [8] the supervisor is represented as a set of control functions, which is relatively close to the approach presented in this paper. The major focus is however to design a nonblocking supervisor for huge systems, while the synthesis result is represented as BDDs, which normally is not easy to interpret by users.

Supervisory synthesis based on Petri nets is presented in [7, 5, 3, 16]. In these methods, the specifications are added to the plants in the form of linear predicates. The resulting controller can also be formulated as guards on the marking vector, which for special cases correspond to control places [6]. However, each approach has some restrictions. The non-blocking problem is not considered in [7]. In [5] the liveness problem is considered but only for controlled marked graphs. The approach proposed in [3] is applicable if the supervisory net has a convex reachability set, and in [16] the request for a minimally restrictive supervisor is abandoned.

In this paper the supervisory guard generation, recently presented for automata [10] and EFAs [11], is generalized to a more general model class called *state-vector transition* (SVT) models. This means that well established supervisory control problems for automata, EFAs and Petri nets are easily and efficiently solved, but also generalized based on the suggested approach. The SVT model, including its full synchronous composition, can be directly

formulated as binary decision diagrams (BDDs) [1]. It implies that large systems can be synthesized symbolically, significantly reducing the traditional state space explosion in terms of memory and computation time.

By the suggested framework, SCT can be solved for large and complex systems, and in the same way for both automata, EFAs and Petri nets, resulting in comprehensible control guards. Therefore, we argue that the presented framework is a unified, flexible and attractive approach. The SVT model has no specific graphical model; both automata and Petri nets with added guards and actions are interesting user choices. The focus of the SVT model is not on user interaction, more on the mathematical model behind, and related algorithmic aspects.

2 Generic Discrete Event Model

A generic discrete event model is presented in this section based on a tuple x, including an ordered set of discrete variables. The domain of the individual variables X_j can be symbolic states as in automata, or integer values as in Petri nets. Each value of x represents a state, and since the tuple x can be seen as a vector, x is considered as the *state-vector* of a state space system with a discrete state space X. A transition from one value of x to an updated next value $\acute{x}$ is enabled when a related predicate on the current and next value $\mathcal{C}(x, \acute{x})$ is satisfied. The transition takes place when it is enabled and a related event σ occurs. At the same time the state vector x is updated to $\acute{x}$.

Communication between different discrete event models is often obtained by common events and full synchronous composition [4], as in automata and Petri nets. In EFAs, communication and synchronization can also be determined by shared variables that are updated in more than one EFA. Therefore, we assume for simplicity that any variable in the state vector x can be assigned to new values in more than one model. This means that x is only partially updated in a local transition, and the rest of the variables implicitly keep their current values. Some complications then occur when local models are synchronized for a specific event, especially for variables that are not updated in any local model. This problem, which has also been treated in [14] [11], is further developed and generalized in this paper.

2.1 State-Vector Transition Model

Consider a tuple of discrete state variables $(x_1, \ldots, x_n)$, where a subset of them are included in a *state vector* x. Let Ω_x be an index set, where $j \in \Omega_x$ for all state variables x_j that are included in x. The domain of definition of each x_j is a finite set X_j. An SVT model will now be defined based on this tuple of state variables. The reason to introduce the index set Ω_x is that variables will later be arbitrarily shared between different local models.

Definition 1 (State-Vector Transition Model)
A state-vector transition model G is a 5-tuple

$$G = \langle X, \Sigma, T, \mathcal{X}_i, \mathcal{X}_m \rangle \tag{1}$$

where:

(i) $X = \times_{j \in \Omega_x} X_j$ is the finite domain of definition of a vector x of variables $x_j \in X_j$, where $j \in \Omega_x$ for all x_j that are included in x.

(ii) Σ is a finite set of events.

(iii) T is a finite set of transitions. Each transition is a 3-tuple $t = (\mathcal{C}, \Omega_\mathcal{C}, \sigma)$, where:

- $\mathcal{C} : X \times X \to \mathbb{B}$ is a predicate on the current value x and the next value $\acute{x}$, defining the enabling condition for the transition,
- $\Omega_\mathcal{C}$ is an index set, and $j \in \Omega_\mathcal{C}$ for all $\acute{x}_j$ where there is a condition on $\acute{x}_j$ in $\mathcal{C}(x, \acute{x})$,
- $\sigma \in \Sigma$.

(iv) $\mathcal{X}_i : X \to \mathbb{B}$ is a predicate, defining possible initial values of x.

(v) $\mathcal{X}_m : X \to \mathbb{B}$ is a predicate, defining desired marked values of x. □

Transition Relation and Keep Current Value A transition $t = (\mathcal{C}, \Omega_\mathcal{C}, \sigma)$ is *enabled* when the predicate $\mathcal{C}(x, \acute{x})$ is evaluated to true. An enabled transition is then executed when the event σ occurs. If there is no condition on $\acute{x}_j$ in $\mathcal{C}(x, \acute{x})$, the index $j \in \Omega_x \backslash \Omega_\mathcal{C}$ and the state variable x_j should keep its current value, i.e. $\acute{x}_j = x_j$. The *complete transition predicate* $\Phi(x, \acute{x})$ for transition t can therefore be expressed as

$$\Phi(x, \acute{x}) \triangleq \mathcal{C}(x, \acute{x}) \bigwedge_{j=\Omega_x \backslash \Omega_\mathcal{C}} \acute{x}_j = x_j \tag{2}$$

Consider e.g. the predicate $\mathcal{C} \triangleq x_1 = 1 \wedge \acute{x}_2 = 3$ and the index set $\Omega_x = \{1, 2\}$. Then $\Omega_\mathcal{C} = \{2\}$, and the complete transition predicate $\Phi \triangleq x_1 = 1 \wedge \acute{x}_2 = 3 \wedge \acute{x}_1 = x_1$. Observe that $\Omega_\mathcal{C}$ can be automatically generated by parsing the predicate $\mathcal{C}(x, \acute{x})$, and including the indices for those variables that have conditions on $\acute{x}_j$ in $\mathcal{C}(x, \acute{x})$.

Also note the condition on the next value $\acute{x} \in X$. When for instance the domain $X = \{0, 1, 2\}$, it means that the conditions $\acute{x} = x + 1$ and $\acute{x} = x - 1$ implicitly include the additional guards $x < 2$ and $x > 0$, respectively. These conditions on the current value of x do not need to be explicitly introduced, since they are achieved by the domain of definition for $\acute{x}$.

The reason why the keep-current-value predicate $\bigwedge_{j \in \Omega_x \backslash \Omega_\mathcal{C}} \acute{x}_j = x_j$ is separated from the predicate condition $\mathcal{C}(x, \acute{x})$ is that shared variables can be updated in different SVT models. This implies that the two predicates are required to be handled differently in the synchronization defined in Section 3. Another benefit of the keep-current-value predicate is that conditions on the next value only need to be introduced by the user for those variables where the updated value is different from the current one.

To be able to separate the keep-current-value predicate $\bigwedge_{j\in\Omega_x\setminus\Omega_\mathcal{C}} \acute{x}_j = x_j$ from the predicate condition $\mathcal{C}(x,\acute{x})$ in the complete transition predicate (2), we have to assume that there are no OR conditions in $\mathcal{C}$ between update of *different shared variables*. This has the implication that $\mathcal{C} \triangleq \acute{x}_1 = 1 \vee \acute{x}_1 = 3$ is acceptable, while $\mathcal{C} \triangleq \acute{x}_1 = 1 \vee \acute{x}_2 = 3$ is not acceptable. The latter case has simply to be separated into two conditions and a choice between two alternative transitions.

The following example illustrates the comments above, but also shows how a Petri net model can be simplified by introducing shared variables.

Example 1 Consider the classical Petri net (PN) model in Fig. 1, where two common resources R_1 and R_2 are required by both sequences but in opposite order. The places and arcs between the two straight sequences model the mutual exclusion conditions for the two resources. In Fig. 2 an alternative PN model is presented, where the two shared resource places are replaced by the shared variables R_1 and R_2. With the domain $\{0,1\}$ for both R_1 and R_2, the resources are booked (+) and unbooked (-) by the short-cut command

$$R_k^{\pm} \triangleq \acute{R}_k = R_k \pm 1 \qquad k = 1,2 \tag{3}$$

This alternative PN model has the benefit of showing the sequential part flow graphically, while the resource booking is more easily specified by logical conditions, especially for larger systems.

The two modular PNs in Fig. 2 are now formally defined as two synchronized SVT models G^k, $k = 1,2$. The state variables are the number of tokens m_j^k in each place p_j^k, i.e. $x_{4(k-1)+j} = m_j^k$, and the resource variables, $x_{k+8} = R_k$. This gives the state index sets $\Omega_x^1 = \{1,2,3,4,9,10\}$ and $\Omega_x^2 = \{5,6,7,8,9,10\}$. Assuming there are initially m_0^k tokens in place p_1^k, and all tokens in p_4^k is the marking condition, the initial and marking predicates are

$$\mathcal{X}_i^k \triangleq x^k = (m_0^k, 0, 0, 0, 0, 0)$$
$$\mathcal{X}_m^k \triangleq x^k = (0, 0, 0, m_0^k, 0, 0)$$

In Fig. 2 $m_0^1 = 3$ and $m_0^2 = 2$. With the domain of definition for $m_j^k \in \{0, 1, \ldots, m_0^k\}$, the predicate condition for the first transition in G^1 is

$$\mathcal{C}_1^1 \triangleq \acute{m}_1^1 = m_1^1 - 1 \wedge \acute{m}_2^1 = m_2^1 + 1 \wedge \acute{R}_1 = R_1 + 1$$

One token is moved from the first to the second place, when there is at least one token in the first place and R_1 is available. Remind that the domain of definitions for m_1^1 and R_1 imply that the predicate $\mathcal{C}_1^1$ is only satisfied when the current value of $m_1^1 > 0$ and $R_1 = 0$. Since only m_1^1, m_2^1, and R_1 are updated, the index set $\Omega_{\mathcal{C},1}^1 = \{1,2,9\}$, and the complete transition predicate becomes, cf. (2)

$$\Phi_1^1 \triangleq \mathcal{C}_1^1 \wedge \acute{m}_3^1 = m_3^1 \wedge \acute{m}_4^1 = m_4^1 \wedge \acute{R}_2 = R_2$$

Generating corresponding predicates for all transitions, it only remains to handle the synchronization between G^1 and G^2. This is described in Example 3. □

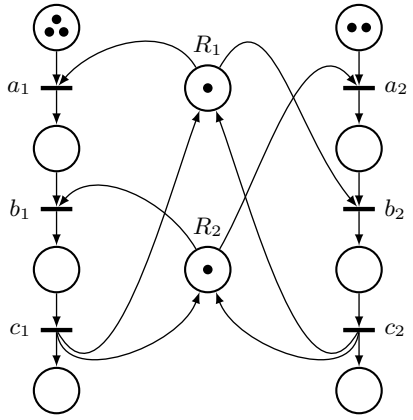

Figure 1 Petri net including two shared resources R_1 and R_2.

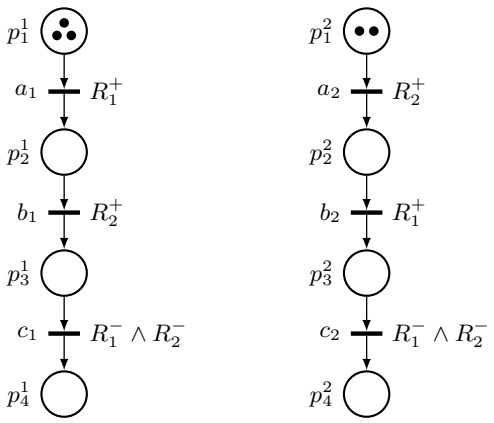

Figure 2 Two modular Petri net sequences, where resources are booked (+) / unbooked (-) by the short-cut condition $R_k^{\pm}$ (3).

The presented SVT model is similar to the EFA model introduced in [14], where the locations here are generalized to a tuple of variables. This means that Petri nets can also be naturally represented as SVT models, illustrated above. The formulation is simplified, since the distinction between locations and variables is avoided by considering one tuple of variables, the state vector x. Guards and actions in an EFA transition are unified in the common condition predicate $\mathcal{C}(x,\acute{x})$. In EFAs guards and actions are defined for variables but not for locations. Hence, guards involving locations require an explicit introduction of corresponding location variables in EFAs.

2.2 Explicit State Transition Model

The state-vector transition model in (1) is now formulated as an explicit state transition model. This can be

considered as an *evaluated* SVT model, where the complete predicate $\Phi(x, \acute{x})$ in (2) is determined for all possible combinations of state variables.

Definition 2 (Explicit State Transition Model) An explicit state transition model of an SVT model $G = \langle X, \Sigma, T, \mathcal{X}_i, M\rangle$ is a 5-tuple

$$\widehat{G} = \langle X, \Sigma, \rightarrow, X_i, X_m\rangle \tag{4}$$

where X and Σ are defined in Definition 1, and

- $\rightarrow \triangleq \{(x, \sigma, \acute{x}) \in X \times \Sigma \times X \mid \exists t = (\mathcal{C}, \Omega_{\mathcal{C}}, \sigma) \in T : \mathcal{C}(x, \acute{x}) \bigwedge_{j=\Omega_x \setminus \Omega_{\mathcal{C}}} \acute{x}_j = x_j\}$
- $X_i \triangleq \{x \in X \mid \mathcal{X}_i(x)\}$
- $X_m \triangleq \{x \in X \mid \mathcal{X}_m(x)\}$ □

Observe that $\widehat{G}$ is an automaton, where preferably different values of the state vector x are expressed symbolically. The explicit state transition relation is written $x \xrightarrow{\sigma} \acute{x}$, which can recursively be extended to strings in Σ^*. A path from a state x to a state y, which generates a string s, is denoted $x \xrightarrow{s} y$. A corresponding path from an initial state in $\widehat{G}$ to a state x is written $\widehat{G} \xrightarrow{s} x$, while a path from a state x to a marked state in X_m is denoted $x \xrightarrow{s} X_m$. If a path from x to y, generating a string s, is not defined, it is written $x \overset{s}{\nrightarrow} y$.

The state-vector transition model G in (1), and its corresponding explicit state transition model $\widehat{G}$ in (4), naturally include nondeterministic behavior. A *deterministic* SVT model, however, has a single initial state, and the transitions $x \xrightarrow{\sigma} \acute{x}$ and $x \xrightarrow{\sigma} \grave{x}$ always imply $\acute{x} = \grave{x}$.

3 Synchronous Composition

The synchronous composition of SVT models is now defined and analyzed. The results are used in the following sections on supervisor synthesis.

3.1 Extended Full Synchronous Composition

The definition is based on Hoar's full synchronous composition [4], but extended to include shared variables. This is similar to the formulation in [14], but here adapted to the SVT model.

Definition 3 (Extended Full Synchronous Composition) Let $G^k = \langle X^k, \Sigma^k, T^k, \mathcal{X}_i^k, \mathcal{X}_m^k\rangle$, $k = 1, 2$, be two SVT models, where $X^k = \times_{j\in\Omega_x^k} X_j$. The extended full synchronous composition (EFSC) of G^1 and G^2 is then defined as

$$G^1 \| G^2 = \langle X, \Sigma^1 \cup \Sigma^2, T, \mathcal{X}_i^1 \wedge \mathcal{X}_i^2, \mathcal{X}_m^1 \wedge \mathcal{X}_m^2\rangle$$

where $X = \times_{j\in\Omega_x^1\cup\Omega_x^2} X_j$, and $T = T^1 \times T^2$. For each combination of $t^k = (\mathcal{C}^k, \Omega_{\mathcal{C}}^k, \sigma) \in T^k$, $k = 1, 2$, the predicate $\mathcal{C}(x, \acute{x})$ and the index set $\Omega_{\mathcal{C}}$ of the synchronized transition $t = (t^1, t^2) = (\mathcal{C}, \Omega_{\mathcal{C}}, \sigma) \in T$ are defined as

$$\mathcal{C}(x, \acute{x}) \triangleq \begin{cases} \mathcal{C}^1(x^1, \acute{x}^1) \wedge \mathcal{C}^2(x^2, \acute{x}^2) & \sigma \in \Sigma^1 \cap \Sigma^2 \\ \mathcal{C}^1(x^1, \acute{x}^1) & \sigma \in \Sigma^1 \setminus \Sigma^2 \\ \mathcal{C}^2(x^2, \acute{x}^2) & \sigma \in \Sigma^2 \setminus \Sigma^1 \end{cases} \tag{5}$$

and

$$\Omega_{\mathcal{C}} \triangleq \begin{cases} \Omega_{\mathcal{C}}^1 \cup \Omega_{\mathcal{C}}^2 & \sigma \in \Sigma^1 \cap \Sigma^2 \\ \Omega_{\mathcal{C}}^1 & \sigma \in \Sigma^1 \setminus \Sigma^2 \\ \Omega_{\mathcal{C}}^2 & \sigma \in \Sigma^2 \setminus \Sigma^1 \end{cases} \tag{6}$$

□

Example 2 To illustrate this definition, consider a synchronized system $G^1 \| G^2$ with only shared variables in $x = (x_1, x_2, x_3)$, i.e. $\Omega_x^1 = \Omega_x^2 = \{1, 2, 3\}$. Given the local transitions $(\acute{x}_1 = 1 \wedge x_3 \geq 0, \{1\}, a) \in T^1$ and $(x_3 = 1 \wedge \acute{x}_3 = 0, \{3\}, a) \in T^2$, the corresponding synchronized transition, according to (5) and (6), becomes

$$(\acute{x}_1 = 1 \wedge x_3 \geq 0 \wedge x_3 = 1 \wedge \acute{x}_3 = 0, \{1, 3\}, a) \in T$$

This results in the following complete transition predicate

$$\Phi(x, \acute{x}) = \acute{x}_1 = 1 \wedge x_3 = 1 \wedge \acute{x}_3 = 0 \wedge \acute{x}_2 = x_2$$ □

Based on Definition 3 we find that the index set for the state vector x of $G^1 \| G^2$ is $\Omega_x^1 \cup \Omega_x^2$. This means that the shared variables defined by the intersection $\Omega_x^1 \cap \Omega_x^2$ are not repeated. Furthermore, the set of variables is not fixed; it is extended when new SVT models are added by synchronization.

When the next value conditions in G^1 and G^2 are in conflict (due to update of one or more shared variables to different values), no transition will occur. The reason is that the synchronized predicate $\mathcal{C}^1 \wedge \mathcal{C}^2$ will not be satisfied due to the conflict between $\mathcal{C}^1$ and $\mathcal{C}^2$.

Example 3 In Example 1 the individual SVT models G_1 and G_2 were defined for the PN model in Fig. 2. In this example the synchronous composition $G^1 \| G^2$ is presented. First we observe that the synchronized state vector has the index set $\Omega_x = \{1, 2, \ldots, 10\}$. Since all transitions have unique events, the only thing to add to the local models G^1 and G^2 is the keep-current-value conditions on the marking variables for the PN that is not involved in the actual transition. The complete transition predicate $\Phi_1(x, \acute{x})$ for the first transition in the left PN sequence in Fig. 2 now also needs keep-current-value conditions on $m_1^2, \ldots, m_4^2$. This results in the complete transition predicate

$$\Phi_1 \triangleq \mathcal{C}_1^1 \wedge \acute{m}_3^1 = m_3^1 \wedge \acute{m}_4^1 = m_4^1 \wedge \acute{R}_2 = R_2 \bigwedge_{j=1,\ldots,4} \acute{m}_j^2 = m_j^2$$

for the synchronized transition. Adding such keep-current-value conditions to all transitions and taking the OR condition between them, a complete logical transition model for $G^1 \| G^2$ is finally achieved. This transition model can be directly transformed to a binary decision diagram, which means that efficient symbolic algorithms for supervisory synthesis can be applied. □

3.2 Full Synchronous Composition

In the following analysis we will further investigate the relation between synchronization of state-vector transition models and explicit state transition models. Therefore also the full synchronous composition (FSC) of explicit state transition models is introduced.

Definition 4 (Full Synchronous Composition)
Let $\widehat{G}^k = \langle X^k, \Sigma^k, \rightarrow_k, X_i^k, X_m^k \rangle$, $k=1,2$, be two SVT models on explicit state transition form. The full synchronous composition (FSC) of $\widehat{G}^1$ and $\widehat{G}^2$ is then defined as

$$\widehat{G}^1 \| \widehat{G}^2 = \langle X^1 \times X^2, \Sigma^1 \cup \Sigma^2, \rightarrow, X_i^1 \times X_i^2, X_m^1 \times X_m^2 \rangle$$

where $\Phi^k(x^k, \acute{x}^k) = \mathcal{C}^k(x^k, \acute{x}^k) \bigwedge_{j=\Omega_x^k \setminus \Omega_{\mathcal{C}}^k} \acute{x}_j = x_j$ and

$$\begin{aligned} \rightarrow \triangleq \{ ((x^1,x^2),\sigma,(\acute{x}^1,\acute{x}^2)) \in X^1 \times X^2 \times \Sigma \times X^1 \times X^2 \mid \\ \big(\Phi^1(x^1,\acute{x}^1) \wedge \Phi^2(x^2,\acute{x}^2) \wedge \sigma \in \Sigma^1 \cap \Sigma^2\big) \bigvee \\ \big(\Phi^1(x^1,\acute{x}^1) \wedge \acute{x}^2 = x^2 \wedge \sigma \in \Sigma^1 \setminus \Sigma^2\big) \bigvee \\ \big(\Phi^2(x^2,\acute{x}^2) \wedge \acute{x}^1 = x^1 \wedge \sigma \in \Sigma^2 \setminus \Sigma^1\big) \} \end{aligned}$$ □

Compared to the extended full synchronous composition (EFSC) for SVT models in Definition 3, one main difference is that no shared variables are included in this definition. The FSC is only based on shared events, while the synchronization of SVT models includes both shared events and shared variables.

In Definition 4 the state vectors x^1 and x^2 for the two models $\widehat{G}^1$ and $\widehat{G}^2$ are not joined, which implies that the total vector for the FSC becomes (x^1, x^2) compared to the EFSC, where the shared variables among x_1 and x_2 are only included ones. Furthermore, the index sets $\Omega_{\mathcal{C}}^1$ and $\Omega_{\mathcal{C}}^2$ are not joined in the FSC, which they are in the EFSC. These differences have major implications on the resulting behavior of the FSC for explicit state transitions models and the EFSC for SVT models, as will be illustrated in the following analysis.

3.3 Analysis

The unique features of SVT models are the introduction of shared variables that can be updated by different local SVT models, the keep-current-value mechanism, and the flexible specification of the involved variables, including the shared ones. The consequences of this nontrivial but powerful sharing mechanism will now be further analyzed, first by an example and then by a proposition.

Example 4 Consider two SVT models G^1 and G^2 with two shared variable (x_1, x_2) and two shared events $\{a, b\}$. The local transition sets for G^1 and G^2 are $T^1 = \{(\acute{x}_1 = 1, \{1\}, a), (x_2 > 0, \varnothing, b)\}$ and $T^2 = \{(\acute{x}_2 = 1, \{2\}, a), (x_1 > 0, \varnothing, b)\}$. By generating the complete transition models for G_1, G_2, and $G_1 \| G_2$, based on Definition 1, (2), and Definition 3, and assuming the initial state $(x_1, x_2) = (0, 0)$, the following languages are obtained

$$\mathcal{L}(G^1) = \mathcal{L}(G^2) = a^* \quad \mathcal{L}(G^1 \| G^2) = a(a+b)^*$$

The reason for the more limited behavior of the local models G_1 and G_2 is that there is a condition (a guard) on the variable that is not updated by the local model itself but by its "sister" model. G^1 has a guard on x_2 that is updated by G^2 and vice versa. In $G^1 \| G^2$, where both variables are updated simultaneously, the guards are satisfied after the first a-transition has happened, which implies that the b-transition can also be executed. □

This example shows that a richer behavior can be achieved after an EFSC of SVT models, compared to the individual models. It never happens in an ordinary FSC, where two synchronized automata limit each others behaviors or run independently.

Proposition 1
According to Definition 3 and 4 and the assumptions below, the following relations are valid for SVT models G^1, G^2 and G^3.

(a) The synchronization operator is commutative and associative, i.e. $G^1 \| G^2 = G^2 \| G^1$ and $(G^1 \| G^2) \| G^3 = G^1 \| (G^2 \| G^3)$.

(b) Assume that $\Omega_x^1 \cap \Omega_x^2 \neq \varnothing$. Then generally $\widehat{G^1 \| G^2} \neq \widehat{G}^1 \| \widehat{G}^2$.

(c) Assume that $\Omega_x^1 \cap \Omega_x^2 = \varnothing$ (no shared variables). Then $\widehat{G^1 \| G^2} = \widehat{G}^1 \| \widehat{G}^2$.

(d) Assume that $\Omega_x^1 \cap \Omega_{\mathcal{C}}^2 = \varnothing$ and $\Sigma^2 \subseteq \Sigma^1$. Then $\widehat{G}^1 \| \widehat{G^1 \| G^2} = \widehat{G^1 \| G^2}$.

Proof: (a) Follows by Definition 3 and the fact that the conjunction and the set union operators are commutative and associative.

(b) Results from Example 4, since $\Sigma^1 = \Sigma^2$ implies that $\mathcal{L}(\widehat{G}^1 \| \widehat{G}^2) = \mathcal{L}(\widehat{G}^1) \cap \mathcal{L}(\widehat{G}^2) = a^* \neq \mathcal{L}(\widehat{G^1 \| G^2}) = a(a+b)^*$

(c) Assume without loss of generality that the state vector x for $G^1 \| G^2$ is reordered such that $x = (x^1, x^2)$, where x^1 and x^2 are the local state vectors for G^1 and G^2. Remind that there are no shared variables between x^1 and x^2. Then the state space of $\widehat{G^1 \| G^2}$, $X = X^1 \times X^2$, i.e. the state space of $\widehat{G}^1 || \widehat{G}^2$. Furthermore, generating the complete transition predicates (2) for $G^1 || G^2$, following Definition 3, then gives the same explicit state space model $\widehat{G^1 \| G^2}$, based on Definition 2, as $\widehat{G}^1 || \widehat{G}^2$, according to Definition 4.

(d) For $G^1 \| G^2$ and $\sigma \in \Sigma^1 \cap \Sigma^2$, the set $\Omega_x \setminus \Omega_{\mathcal{C}}$ in (2) can, according to Definition 3, be rewritten as $(\Omega_x^1 \cup \Omega_x^2) \setminus (\Omega_{\mathcal{C}}^1 \cup \Omega_{\mathcal{C}}^2) = (\Omega_x^1 \cup \Omega_x^2 \setminus \Omega_x^1) \setminus (\Omega_{\mathcal{C}}^1 \cup \Omega_{\mathcal{C}}^2) = \Omega_x^1 \setminus (\Omega_{\mathcal{C}}^1 \cup \Omega_{\mathcal{C}}^2) \cup \Omega_x^2 \setminus \Omega_x^1 \setminus \Omega_{\mathcal{C}}^2$. Based on the assumption $\Omega_x^1 \cap \Omega_{\mathcal{C}}^2 = \varnothing$, we then get

$$\Omega_x \setminus \Omega_{\mathcal{C}} = \Omega_x^1 \setminus \Omega_{\mathcal{C}}^1 \cup \Omega_x^2 \setminus (\Omega_x^1 \cup \Omega_{\mathcal{C}}^2)$$

for $\sigma \in \Sigma^1 \cap \Sigma^2$. Now introduce the functions

$$\Phi^1(x^1, \acute{x}^1) \;=\; \mathcal{C}^1(x^1, \acute{x}^1) \bigwedge_{j=\Omega_x^1 \setminus \Omega_{\mathcal{C}}^1} \acute{x}_j = x_j \qquad (7)$$

$$\Phi^{21}(x^2, \sigma, \acute{x}^2) \;=\; \big(\mathcal{C}^2(x^2, \acute{x}^2) \bigwedge_{j=\Omega_x^2 \setminus (\Omega_x^1 \cup \Omega_{\mathcal{C}}^2)} \acute{x}_j = x_j \wedge \sigma \in \Sigma^2\big)$$
$$\vee\big(\bigwedge_{j=\Omega_x^2 \setminus \Omega_x^1} \acute{x}_j = x_j \wedge \sigma \notin \Sigma^2\big) \qquad (8)$$

as well as the assumption $\Sigma^2 \subseteq \Sigma^1$ and Definition 3. Then the explicit state transition relation for $G^1 \| G^2$ can be expressed as

$$\rightarrow_{\widehat{G^1 \| G^2}} = \{(x, \sigma, \acute{x}) \mid \Phi^1(x^1, \acute{x}^1) \wedge \Phi^{21}(x^2, \sigma, \acute{x}^2)\} \qquad (9)$$

To emphasize separate state variables in the FSC $\widehat{G^1 \| \widehat{G^1 \| G^2}}$, the state vector $\bar{x}^1 \in X^1$ is used in G^1. According to Definition 4, including the observation that the event set for $G^1 \| G^2$ is Σ^1, we then obtain

$$\rightarrow_{\widehat{G}^1 \| \widehat{G^1 \| G^2}} ((\bar{x}^1, x), \sigma, (\acute{\bar{x}}^1, \acute{x})) \mid \Phi^1(\bar{x}^1, \acute{\bar{x}}^1) \wedge$$
$$\Phi^1(x^1, \acute{x}^1) \wedge \Phi^{21}(x^2, \sigma, \acute{x}^2)\} \qquad (10)$$

Since x^1 and $\bar{x}^1$ have the same domain of definition X^1, and $\Phi^1(\bar{x}^1, \acute{\bar{x}}^1)$ is true for all $((\bar{x}^1, x), (\acute{\bar{x}}^1, \acute{x}))$ such that $\Phi^1(x^1, \acute{x}^1) \wedge \Phi^{21}(x^2, \sigma, \acute{x}^2)$ is true, it is enough to consider those $(x, \acute{x})$ where $\Phi^1(x^1, \acute{x}^1) \wedge \Phi^{21}(x^2, \sigma, \acute{x}^2)$ is true in (10). In other words, (9) generates the same transitions as (10), which shows that $\widehat{G}^1 \| \widehat{G^1 \| G^2} = \widehat{G^1 \| G^2}$ when $\Omega_x^1 \cap \Omega_{\mathcal{C}}^2 = \varnothing$ and $\Sigma^2 \subseteq \Sigma^1$. □

Part (c) of this proposition is an important special case, since both automata and Petri nets have no shared variables. Part (d) will be used in the following supervisor synthesis.

4 Supervisory Control

Supervisory control theory (SCT) [13] is a general theory to automatically synthesize supervisors based on a given plant and specification. A plant G is normally given as a number of synchronized sub-plants, i.e. $G = G^1 \| \cdots \| G^{N_G}$. Local specifications K^j, $j = 1, \ldots, N_K$ are also synchronized to a common specification $K = K^1 \| \cdots \| K^{N_K}$. Typical examples of specifications are a) additional guards and actions on existing or new variables, such as the resource booking $R_k^{\pm}$ (3) in Example 1, b) marked and explicitly forbidden states, and c) dynamic specifications that restrict possible alternatives in the plant. The total specification of the controlled system is obtained by synchronizing the specification K with the plant G. The result $G \| K$ is also a *candidate of a supervisor* to control the plant such that the specification K is fulfilled.

All synchronizations in $G \| K$ are based on the EFSC, which means that shared variables can be updated by any local SVT model. Before the supervisor synthesis is introduced, an important assumption need to be included. No variables in the plant G are allowed to be updated to new values by the specification K. It means that the set $\Omega_{\mathcal{C}}^K$ has no common elements with Ω_x^G. This is a natural assumption, which still means that guards on current values of plant variables can be included in the specification. This assumption has an important implication, which is shown in the following proposition.

In this proposition, $\widehat{G'}$ being a subautomaton of $\widehat{G}$, denoted $\widehat{G'} \subseteq \widehat{G}$, means that all elements in the tuple of $\widehat{G'}$ are subsets of corresponding elements in $\widehat{G}$, except for the event sets that are equal, i.e. $\Sigma^{G'} = \Sigma^G$.

Proposition 2 (Refinement)
Let $G = \langle X^G, \Sigma^G, T^G, \mathcal{X}_i^G, \mathcal{X}_m^G \rangle$ *and* $K = \langle X^K, \Sigma^K, T^K, \mathcal{X}_i^K, \mathcal{X}_m^K \rangle$. *Assume that* $\Omega_x^G \cap \Omega_{\mathcal{C}}^K = \varnothing$, $\Sigma^K \subseteq \Sigma^G$, *and consider an arbitrary sub-automaton* $\widehat{S} \subseteq \widehat{G \| K}$. *Then* $\widehat{G} \| \widehat{G \| K} = \widehat{G \| K}$ *and* $\widehat{G} \| \widehat{S} = \widehat{S}$.

Proof: From Proposition 1(d), with $G^1 = G$, $G^2 = K$, it follows that $\widehat{G} \| \widehat{G \| K} = \widehat{G \| K}$. Since $\widehat{S} \subseteq \widehat{G \| K}$, we also know that $\widehat{G \| K} \| \widehat{S} = \widehat{S}$. Synchronizing $\widehat{G} \| \widehat{G \| K} = \widehat{G \| K}$ with $\widehat{S}$ then gives $\widehat{G} \| \widehat{S} = \widehat{S}$. □

Assuming that $\widehat{S}$ is a supervisor, a standard assumption in SCT is that the closed loop system is $\widehat{G} \| \widehat{S}$. The result of this proposition then shows that the supervisor in itself is a model of the closed loop system.

4.1 Controllability and Nonblocking

In SCT, the events are divided into two disjoint subsets: *controllable events*, denoted by Σ_c, that can be prevented from executing by the supervisor; and *uncontrollable events*, denoted by Σ_u, which cannot be influenced by the supervisor [13]. Two important properties are considered in SCT namely controllability and nonblocking.

Definition 5 (Controllability)
Let $\widehat{G}^k = \langle X^k, \Sigma^k, \rightarrow_k, X_i^k, X_m^k \rangle$, $k = 1, 2$, be two SVT models on explicit state transition form, where $\Sigma^2 \subseteq \Sigma^1$. Then $\widehat{G}^2$ is *controllable* with respect to $\widehat{G}^1$ and a set of uncontrollable events $\Sigma_u \subseteq \Sigma^1$ if, for every string $s \in {\Sigma^1}^*$ and every uncontrollable event $\sigma_u \in \Sigma_u \cap \Sigma^2$ such that $\widehat{G}^1 \| \widehat{G}^2 \xrightarrow{s} (x^1, x^2)$ and $x^1 \xrightarrow{\sigma_u} \acute{x}^1$ in $\widehat{G}^1$, it also holds that $x^2 \xrightarrow{\sigma_u} \acute{x}^2$ in $\widehat{G}^2$. If this transition in G^2 does not exist, x^2 is an *uncontrollable state* that needs to be forbidden. □

Definition 6 (Nonblocking)
Let $\widehat{G} = \langle X, \Sigma, \rightarrow, X_i, X_m \rangle$. A state $x \in X$ in $\widehat{G}$ is *reachable* if $\widehat{G} \xrightarrow{s} x$, and it is *coreachable* if $x \xrightarrow{s} X_m$. An SVT model, on explicit state transition form $\widehat{G}$, is *nonblocking* if every reachable state is also coreachable. □

By generating a subautomaton $\widehat{S} \subseteq \widehat{G \| K}$ that is nonblocking, Proposition 2 shows that this property also holds for the closed loop system $\widehat{G} \| \widehat{S} = \widehat{S}$.

Concerning controllability, Proposition 2 can also be used to generate a controllable supervisor $\widehat{S}$, with respect

to the plant $\widehat{G}$, that is a subautomaton $\widehat{S} \subseteq \widehat{G\|K}$. The key is to first identify all *uncontrollable states* in $\widehat{G\|K}$ with respect to $\widehat{G}$. The identification of these forbidden states is shown in the following proposition.

Proposition 3 (Uncontrollable states)
Let $G = \langle X^G, \Sigma^G, T^G, \mathcal{X}_i^G, \mathcal{X}_m^G\rangle$ and $K = \langle X^K, \Sigma^K, T^K, \mathcal{X}_i^K, \mathcal{X}_m^K\rangle$. Assume that $\Omega_x^G \cap \Omega_{\mathcal{C}}^K = \varnothing$, $\Sigma^K \subseteq \Sigma^G$. The set of uncontrollable states in $\widehat{G\|K}$ with respect to $\widehat{G}$ is then given by the set

$$\begin{aligned} X_u^{G\|K} = \{&x \in X^{G\|K} \,|\, \exists (t^G, t^K) \in T^{G\|K}, \\ &t^G = (\mathcal{C}^G, \Omega_{\mathcal{C}}^G, \sigma), t^K = (\mathcal{C}^K, \Omega_{\mathcal{C}}^K, \sigma) \wedge \\ &\sigma \in \Sigma_u \wedge \forall \acute{x}^G \in X^G \wedge \forall \acute{x}^K \in X^K : \\ &\mathcal{C}^G(x^G, \acute{x}^G) \wedge \neg \mathcal{C}^K(x^K, \acute{x}^K)\} \end{aligned}$$

Proof: Follows from Proposition 1(d) for $G^1 = G$ and $G^2 = K$, the transition relation (10) and the related conflict between Φ^1 in (7) and Φ^{21} in (8), which reduces to $\neg(\mathcal{C}^1 \rightarrow \mathcal{C}^2) = \mathcal{C}^1 \wedge \neg\mathcal{C}^2 = \mathcal{C}^G \wedge \neg\mathcal{C}^K$. □

4.2 Supervisor state set generation

Based on the supervisor candidate $\widehat{G\|K}$ a supervisor $\widehat{S}$ will now be constructed. It is done by generating a set of forbidden states X_f that are removed from $\widehat{G\|K}$, such that desired properties are achieved. This is achieved using the following two definitions.

Definition 7 (Forbidden state model)
Let $\widehat{G} = \langle X, \Sigma, \rightarrow, X_i, X_m\rangle$. A corresponding model including a set of *forbidden states* $X_f \subseteq X$ is then defined as $\widehat{G}_{\setminus X_f} = \langle X \setminus X_f, \Sigma, \rightarrow_{\setminus X_f}, X_i \setminus X_f, X_m \setminus X_f\rangle$ where $\rightarrow_{\setminus X_f} = \{(x, \sigma, \acute{x}) \,|\, x, \acute{x} \notin X_f\}$ □

Definition 8 (Forbidden state operator)
Let $\widehat{G} = \langle X, \Sigma, \rightarrow, X_i, X_m\rangle$ where $\Sigma_u \subseteq \Sigma$. The *forbidden state operator* $\Theta^{\widehat{G}}(X_f) : 2^X \rightarrow 2^X$ for $\widehat{G}$ is defined as

$$\begin{aligned} \Theta^{\widehat{G}}(X_f) \;=\; &X_f \cup \{x \in X \,|\, \forall s_u \in \Sigma_u^* \wedge \forall y \in X \\ &\text{such that } x \xrightarrow{s_u} y, \text{ it holds that } y \overset{s}{\nrightarrow}_{\setminus X_f} X_m\} \end{aligned}$$ □

First note that $y \overset{s}{\nrightarrow}_{\setminus X_f} X_m$ means that state y is not coreachable, i.e. it is a blocking state. Given a set of forbidden states X_f this operator adds additional forbidden states, including all those states x from which a string s_u of uncontrollable events leads to blocking or forbidden states y, based on the current set of forbidden states X_f. This generation includes the empty string $s_u = \varepsilon$. Hence, $\Theta^{\widehat{G}}(X_f)$ adds all blocking states to X_f, but also the uncontrollable states leading to blocking or forbidden states.

This forbidden state operator is applied to the supervisor candidate $\widehat{G\|K}$, where the initially forbidden states are the uncontrollable states $X_u^{G\|K}$, defined in Proposition 3, and any explicitly forbidden states $X_f^{G\|K}$, specified by the user. Hence, the recursive equation

$$X_f^{j+1} := \Theta^{\widehat{G\|K}}(X_f^j) \qquad X_f^0 = X_u^{G\|K} \cup X_f^{G\|K} \quad (11)$$

is iterated until a fixed point $X_f = \Theta^{\widehat{G\|K}}(X_f)$ is reached. Remind that $\Sigma^{G\|K} = \Sigma^G$ and let $\widehat{S} = \langle X^S, \Sigma^G, \rightarrow_S, X_i^S, X_m^S\rangle$ be the automaton for the supervisor. Based on this fix point calculation $\widehat{S}$ can be expressed as

$$\widehat{S} = \widehat{G\|K}_{\setminus X_f} \quad (12)$$

and the supervisor states, the *safe states*, are $X^S = X^{G\|K} \setminus X_f$.

4.3 Supervisor Guards

To obtain supervisor guards that can be added to the original SVT model G, which guarantee that the closed loop system will stay within the safe state set X^S, two specific state sets are introduced.

$$\begin{aligned} X_\sigma^S &= \{x \in X^S \,|\, x \xrightarrow{\sigma}_S\} \\ X_{\neg\sigma}^S &= \{x \in X^S \,|\, x \xrightarrow{\sigma}_{G\|K} \wedge\, x \overset{\sigma}{\nrightarrow}_S\} \end{aligned}$$

The set X_σ^S includes all states in X^S where the execution of σ is defined for S, while $X_{\neg\sigma}^S$ includes all corresponding states where the execution of σ is defined for $G\|K$, but not for S. Thus, for all states in X_σ^S the event σ must be allowed by S, while for all states in $X_{\neg\sigma}^S$ the event σ must be forbidden. For all other states it does not matter if the supervisor allows or forbids σ. For every event $\sigma \in \Sigma_c \subseteq \Sigma_G$ a supervisor predicate is therefore generated such that:

$$\mathcal{C}_\sigma^S(x) = \begin{cases} \mathbf{T} & x \in X_\sigma^S \\ \mathbf{F} & x \in X_{\neg\sigma}^S \\ \text{don't care} & x \in X \setminus (X_\sigma^S \cup X_{\neg\sigma}^S) \end{cases} \quad (13)$$

More details on how this can be done based on BDDs and heuristic rules are shown in [10].

These predicates are added to the plant SVT models G^k, $k = 1 \ldots, N_G$, by replacing the transition relation sets T_k with $T_S^k = \{(\mathcal{C}^k \wedge \mathcal{C}_\sigma^S, \Omega_{\mathcal{C}}^k, \sigma) \,|\, (\mathcal{C}^k, \Omega_{\mathcal{C}}^k, \sigma) \in T^k\}$. The resulting SVT models, called G_S^k, are composed to $G_S = G_S^1 \| \cdots \| G_S^{N_G}$ and synchronized with the specification K. This results in the final SVT supervisor

$$\mathcal{S} = G_S \| K$$

Theorem 4
Let $G = \langle X^G, \Sigma^G, T^G, \mathcal{X}_i^G, \mathcal{X}_m^G\rangle$ and $K = \langle X^K, \Sigma^K, T^K, \mathcal{X}_i^K, \mathcal{X}_m^K\rangle$. Assume that $\Omega_x^G \cap \Omega_{\mathcal{C}}^K = \varnothing$, $\Sigma^K \subseteq \Sigma^G$. By iterating (11) until a fix point X_f is achieved, a controllable, nonblocking, and maximally permissive supervisor $\widehat{S} = \widehat{G\|K}_{\setminus X_f}$ is obtained. The resulting supervisor is a model of the closed loop system, i.e. $\widehat{S} = \widehat{G}\|\widehat{S}$, and a

corresponding SVT supervisor $\mathcal{S} = G_S \| K$, where G_S includes guards added to G, generates the same closed loop system, i.e. $\widehat{G_S \| K} = \widehat{G \| K}_{\setminus X_f}$.

Proof: Since $\Omega_x^G \cap \Omega_c^K = \varnothing$, $\Sigma^K \subseteq \Sigma^G$ and $\widehat{S} = \widehat{G \| K} \setminus X_f \subseteq \widehat{G \| K}$, Proposition 2 shows that $\widehat{S} = \widehat{G} \| \widehat{S}$. The set of uncontrollable states given by Proposition 3 are avoided, including explicitly forbidden states, by initialize the forbidden state iteration (11) by $X_f^0 = X_u^{G\|K} \cup X_f^{G\|K}$.

In every iteration of (11), based on the current set of forbidden states X_j^f, any additional blocking states B_j are added to X_f by $s_u = \varepsilon$. When $\widehat{G \| K} \xrightarrow{s} x$ and $x \xrightarrow{s_u} y$, where $y \in X_j^f \cup B_j$ and $s_u \in \Sigma_u^*$, the string ss_u can be executed in both $\widehat{G}$ and $\widehat{G \| K}$. This is the fact, since according to Proposition 2, $\mathcal{L}(\widehat{G} \| \widehat{G \| K}) = \mathcal{L}(\widehat{G}) \cap \mathcal{L}(\widehat{G \| K}) \subseteq \mathcal{L}(\widehat{G})$. All such *extended uncontrollable states* x must therefore be forbidden, since via a string ss_u they will lead to a forbidden or blocking state y in $\widehat{G \| K}$, while the corresponding state in the plant $\widehat{G}$ is reachable.

Exactly those extended uncontrollable states, including the blocking states, are added to the set of forbidden states in (11). When the fix point $X_f = \Theta^{G\|K}(X_f)$ is reached, no more blocking and extended uncontrollable states are identified. Therefore, the resulting supervisor $\widehat{S} = \widehat{G \| K}_{\setminus X_f}$ is controllable and nonblocking. Since $\widehat{S} = \widehat{G} \| \widehat{S}$, the closed loop system is also nonblocking.

In each iteration of (11) only those states that necessarily need to be forbidden are added to the set of forbidden states X_f. Together with the fact that $\Theta^{\widehat{G\|K}}$ (11) is a monotonic function, and a classical result from lattice theory by Tarski [15], it can be shown, see [9], that the resulting $\widehat{S}$ in (12) is also maximally permissive.

Finally, since the generated guards in (13) guarantee that the supervisor $\mathcal{S}$ accepts all transitions that are accepted by the supervisor $\widehat{S}$ and forbid all those that are not allowed by $\widehat{S}$, the same closed loop system are achieved for $\widehat{\mathcal{S}} = \widehat{G_S \| K}$ and $\widehat{S} = \widehat{G \| K}_{\setminus X_f}$. □

5 Conclusions

A state-vector transition model has been presented from which a nonblocking, controllable and maximal permissive supervisor can be implemented in terms of control guards added to the original model. The suggested algorithm is naturally formulated by BDDs, which means that large and complex systems can be handled efficiently. Furthermore, since the SVT model include automata, EFAs and Petri nets as special cases, the presented framework is a unified, flexible and attractive approach for supervisor synthesis.

Acknowledgement

This work was carried out within the Wingquist Laboratory VINN Excellence Centre within the Area of Advance - Production at Chalmers, and supported by VINNOVA an the Swedish Science Foundation. The support is gratefully acknowledged.

References

[1] S. B. Akers. Binary Decision Diagrams. *IEEE Transactions on Computers*, 27:509–516, June 1978.

[2] B. Gaudin and P. H. Deussen. Supervisory Control on Concurrent Discrete Event Systems with Variables. In *American Control Conference, 2007. ACC '07*, pages 4274 – 4279, New York, NY, USA, 2007. IEEE.

[3] A. Giua and F. DiCesare. Blocking and controllability of Petri nets in supervisory control. *IEEE Transactions on Automatic Control*, 39(4):818–823, Apr. 1994.

[4] C. A. R. Hoare. *Communicating sequential processes*, volume 21 of *Series in Computer Science*. ACM, Aug. 1978.

[5] L. E. Holloway and B. H. Krogh. On closed-loop liveness of discrete event systems under maximally permissive control. *IEEE Transactions on Automatic Control*, 37(5):692–697, 1992.

[6] L. E. Holloway, B. H. Krogh, and A. Giua. A survey of Petri net methods for controlled discrete event systems. *Discrete Event Dynamic Systems*, 7(2):151–190, 1997.

[7] Y. Li and W. Wonham. Control of vector discrete-event systems. II. Controller synthesis. *IEEE Transactions on Automatic Control*, 39(3):512–531, Mar. 1994.

[8] C. Ma and W. Wonham. Nonblocking supervisory control of state tree structures. *IEEE Transactions on Automatic Control*, 51(5):782–793, May 2006.

[9] R. Malik and H. Flordal. Compositional synthesis of discrete event systems via synthesis equivalence. Technical Report 05/2008, University of Waikato, 2008.

[10] S. Miremadi, K. Akesson, and B. Lennartson. Symbolic Computation of Reduced Guards in Supervisory Control. *IEEE Transactions on Automation Science and Engineering*, 8(4):754–765, 2011.

[11] S. Miremadi, B. Lennartson, and K. Akesson. A BDD-Based Approach for Modeling Plant and Supervisor by Extended Finite Automata. *IEEE Transactions on Control Systems Technology*, PP(99):1–15, 2011.

[12] L. Ouedraogo, R. Kumar, R. Malik, and K. Åkesson. Symbolic approach to nonblocking and safe control of Extended Finite Automata. In *2010 IEEE International Conference on Automation Science and Engineering*, pages 471–476. IEEE, Aug. 2010.

[13] P. Ramadge and W. Wonham. The control of discrete event systems. *Proceedings of the IEEE, Special Issue on Discrete Event Dynamic Systems*, 77(1):81–98, 1989.

[14] M. Sköldstam, K. Åkesson, and M. Fabian. Modeling of discrete event systems using finite automata with variables. *Decision and Control, 2007 46th IEEE Conference on*, pages 3387–3392, 2007.

[15] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309, 1955.

[16] K. Yamalidou, J. O. Moody, M. D. Lemmon, and P. J. Antsaklis. Feedback control of Petri nets based on place invariats. *Automatica*, 32(1):15–28, 1996.

[17] Y. Yang and R. Gohari. Embedded supervisory control of discrete-event systems. *IEEE International Conference on Automation Science and Engineering, 2005.*, pages 410–415, 2005.