Variable Abstraction and Approximations in Supervisory Control Synthesis

Marcelo Teixeira[1] Robi Malik[2] José E. R. Cury[1] Max. H. de Queiroz[1]

***Abstract*— This paper proposes a method to simplify Extended Finite-state Automata (EFA) in such a way the least restrictive controllable supervisor is preserved. The method is based on variable abstraction, which involves the identification and removal of irrelevant variables from a model. Variable abstraction preserves controllability, and the paper shows how approximations can be used to ascertain least restrictiveness of the synthesis result. The approach has the modelling benefits of Extended Finite-state Automata, leads to optimal control solutions, and reduces the synthesis cost. An example of a manufacturing system illustrates the contributions.**

I. Introduction

Supervisory Control Theory [1] formally describes the *synthesis* of controllers for Discrete Event Systems, mathematically grounded on the *Finite-state Automata (FA)* formalism [2]. By nature, FA are limited in expressive power, particularly when modelling systems with *data dependency*. Furthermore, processing large FA is computationally expensive and leads to *state-space explosion*.

While FA are a good graphical way to capture control states, data dependency is more naturally modelled using *variables*. Several formalisms combine automata with variables. *Synchronous programming languages* [3] describe concurrent behaviours in textual form and translate them to FA. *Statecharts* [4] and *Abstract State Machines* [5] are formalisms of automata with variables, used for verification and refinement. In the context of supervisory control theory, *Extended Finite-state Automata (EFA)* are a simple formalism of automata with variables, which can be used to define and synthesise supervisors [6]–[8].

Variables greatly simplify modelling tasks and produce more concise and more readable models, yet the state-space explosion problem remains. When analysing a system, all possible values of the variables need to be taken into account, and this can give very large state spaces. This problem can be mitigated by *symbolic representations* of the state space [8]–[10], and by *variable abstraction* to simplify models by removing variables that are irrelevant for particular properties [9].

This paper proposes a way to exploit variable abstraction in synthesis. This is more difficult in synthesis than in verification, because a synthesised supervisor typically is required to satisfy several properties at the same time. Variable abstraction has been used for safety properties in synthesis [11], but this method does not preserve least restrictiveness of supervisors. The solution proposed in this paper is inspired by the idea of *approximations* [12], [13], which make it possible to determine that the supervisor synthesised for an abstracted model is least restrictive.

This paper is structured as follows. Section II introduces Extended Finite-state Automata in the context of supervisory control theory. Section III presents variable abstraction and approximations, and their use in supervisor synthesis. The proposed method is applied to a small manufacturing system to demonstrate its benefits. Finally, conclusions and perspectives of future work are discussed in Section IV.

This work was supported by the Brazilian National Council for Scientific and Technological Development (CNPq).

[1]M. Teixeira, J. E, R. Cury, and M. H. de Queiroz are with the Departamento de Automação e Sistemas, Universidade Federal de Santa Catarina, Florianópolis, Brazil ({mt,cury,max}@das.ufsc.br).

[2]R. Malik is with the Department of Computer Science, The University of Waikato, Hamilton, New Zealand (robi@waikato.ac.nz).

II. Preliminaries

A. Events, Traces and Languages

Traces and languages are a simple means to describe discrete system behaviours [2]. Their basic building blocks are *events*, which are taken from a finite *alphabet* Σ. Then Σ^* denotes the set of all finite *traces* of the form $\sigma_1\sigma_2\ldots\sigma_n$ of events from Σ, including the *empty trace* ε. A subset $L \subseteq \Sigma^*$ is called a *language*. The *concatenation* of two traces $s, t \in \Sigma^*$ is written as st. Traces and languages can also be concatenated, for example, $sL = \{\, st \in \Sigma^* \mid t \in L \,\}$. The *prefix-closure* of a language L is $\overline{L} = \{\, s \in \Sigma^* \mid st \in L \text{ for some } t \in \Sigma^* \,\}$, and L is *prefix-closed* if $\overline{L} = L$.

B. Extended Finite-State Automata

Extended Finite-State Automata (EFA) are structures of states, similar to conventional *Finite-State Automata (FA)*, but augmented with *updates* associated to the transitions [6]–[8]. Updates are formulas containing variables.

A *variable* v is an entity associated with a finite domain $\mathrm{dom}(v)$ and an initial value $v^\circ \in \mathrm{dom}(v)$. The domain of a variable set $V = \{v_0, \ldots, v_n\}$ is $\mathrm{dom}(V) = \mathrm{dom}(v_0) \times \cdots \times \mathrm{dom}(v_n)$, and its elements are written as $\bar{v} = (\bar{v}_0, \ldots, \bar{v}_n) \in \mathrm{dom}(V)$ with $\bar{v}_i \in \mathrm{dom}(v_i)$. A second set of variables, called *next-state variables* and denoted by $V' = \{\, v' \mid v \in V \,\}$ with $\mathrm{dom}(V') = \mathrm{dom}(V)$, is used to describe how variables are updated by transitions.

For example, let x be a variable with domain $\mathrm{dom}(x) = \{0, \ldots, 5\}$ and initial value $x^\circ = 0$. A transition with update $x' = x + 1$ changes the variable x by adding 1 to its current value, if it currently is less than 5. Otherwise (if $x = 5$) the transition is disabled and no updates are performed. Another possibility is to write the formula $x' = \min(x + 1, 5)$, in which case the transition remains enabled when $x = 5$. The update $x = 3$ disables a transition unless $x = 3$ in the current state, and allows all possible next-state values of x.

Differently, the update $x' = 3$ always enables its transition, and the value of x in the next state is forced to be 3.

Formally, an EFA is described by a 6-tuple $A_V = \langle \Sigma, V, Q, Q^\circ, Q^\omega, \rightarrow \rangle$, where:

- Σ is the alphabet of events;
- $V = \{v_1, \ldots, v_n\}$ is the set of variables;
- Q is the finite set of states;
- $Q^\circ \subseteq Q$ is the set of initial states;
- $Q^\omega \subseteq Q$ is the set of marked states;
- $\rightarrow \subseteq Q \times \Sigma \times \Pi_V \times Q$ is the state transition relation, where Π_V is the set of Boolean formulas over $V \cup V'$.

The term $x \xrightarrow{\sigma:p} y$ denotes the presence of a transition in A_V, from state x to state y with event $\sigma \in \Sigma$ and update $p \in \Pi_V$.

An EFA A_V can also be interpreted from another perspective (*unfolded* interpretation) as an ordinary FA $A = \langle \Sigma, Q_A, Q_A^\circ, Q_A^\omega, \rightarrow \rangle$ where:

- $Q_A = Q \times \mathrm{dom}(V)$;
- $Q_A^\circ = Q^\circ \times \{(v_1^\circ, \ldots, v_n^\circ)\}$;
- $Q_A^\omega = Q^\omega \times \mathrm{dom}(V)$;
- $\rightarrow$ is such that $(x, \bar{v}) \xrightarrow{\sigma} (y, \bar{v}')$ for $\bar{v}, \bar{v}' \in \mathrm{dom}(V)$, if there exists $x \xrightarrow{\sigma:p} y$ such that $p(\bar{v}, \bar{v}') = true$.

The unfolded state set Q_A includes the values of the variables as part of each state. The unfolded transition relation is defined based on the transition relation of A_V, by taking into account the conditions imposed by the updates on the variable values. The unfolded transition relation is extended to strings in Σ^* by $(x, \bar{v}) \xrightarrow{\varepsilon} (x, \bar{v})$ for all $(x, \bar{v}) \in Q_A$ and $(x, \bar{v}) \xrightarrow{s\sigma} (x'', \bar{v}'')$ if $(x, \bar{v}) \xrightarrow{s} (x', \bar{v}') \xrightarrow{\sigma} (x'', \bar{v}'')$ for some $(x', \bar{v}') \in Q_A$.

Further, $A_V \xrightarrow{s} (x, \bar{v})$ means that there exists $(x^\circ, \bar{v}^\circ) \in Q_A^\circ$ such that $(x^\circ, \bar{v}^\circ) \xrightarrow{s} (x, \bar{v})$. The *open-loop behaviour* and the *marked behaviour* of A_V are the languages

$$L(A_V) = \{\, s \in \Sigma^* \mid A_V \xrightarrow{s} (x, \bar{v}) \in Q_A \,\} ;$$
$$L^\omega(A_V) = \{\, s \in \Sigma^* \mid A_V \xrightarrow{s} (x, \bar{v}) \in Q_A^\omega \,\} .$$

C. EFA Properties and Operations

This section summarises how some common FA properties and operations are defined for EFA. First, two kinds of determinism are of interest for EFA.

Definition 1: An EFA $A_V = \langle \Sigma, V, Q, Q^\circ, Q^\omega, \rightarrow \rangle$ is:

- *state-deterministic* if $|Q^\circ| \leq 1$, and $x \xrightarrow{\sigma:p_1} y_1$ and $x \xrightarrow{\sigma:p_2} y_2$ implies $y_1 = y_2$, for all $x, y_1, y_2 \in Q$, $\sigma \in \Sigma$, and $p_1, p_2 \in \Pi_V$;
- *V-deterministic* if $(x, \bar{v}) \xrightarrow{\sigma} (y, \bar{w})$ and $(x, \bar{v}) \xrightarrow{\sigma} (y, \bar{w}')$ always implies $\bar{w} = \bar{w}'$.

Definition 2: An update $p \in \Pi_V$ is *total* if, for all $\bar{v} \in \mathrm{dom}(V)$, there exists $\bar{v}' \in \mathrm{dom}(V)$ such that $p(\bar{v}, \bar{v}') = true$. An EFA is total if all its updates are total.

An EFA is total if none of its transitions is inhibited by an update. Adding total updates to an FA introduces behaviour like a *distinguisher* [12], [13], merely recording information in variables without introducing synchronisation constraints.

Definition 3: Given two EFA $A = \langle \Sigma_A, V_A, Q_A, Q_A^\circ, Q_A^\omega, \rightarrow_A \rangle$ and $B = \langle \Sigma_B, V_B, Q_B, Q_B^\circ, Q_B^\omega, \rightarrow_B \rangle$, the *synchronous composition* of A and B is $A \parallel B = \langle \Sigma_A \cup \Sigma_B, V_A \cup V_B, Q_A \times Q_B, Q_A^\circ \times Q_B^\circ, Q_A^\omega \times Q_B^\omega, \rightarrow \rangle$, where:

- $(x_A, x_B) \xrightarrow{\sigma:p_A \wedge p_B} (y_A, y_B)$ if: $\sigma \in \Sigma_A \cap \Sigma_B$, $x_A \xrightarrow{\sigma:p_A}_A y_A$, and $x_B \xrightarrow{\sigma:p_B}_B y_B$;
- $(x_A, x_B) \xrightarrow{\sigma:p_A} (y_A, x_B)$ if: $\sigma \in \Sigma_A \setminus \Sigma_B$ and $x_A \xrightarrow{\sigma:p_A}_A y_A$;
- $(x_A, x_B) \xrightarrow{\sigma:p_B} (x_A, y_B)$ if: $\sigma \in \Sigma_B \setminus \Sigma_A$ and $x_B \xrightarrow{\sigma:p_B}_B y_B$.

Shared events between two EFA are synchronised in lockstep synchronisation [14], while other events are interleaved. In addition, the updates are combined by conjunction.

Definition 4: EFA $A_V = \langle \Sigma_A, V_A, Q_A, Q_A^\circ, Q_A^\omega, \rightarrow_A \rangle$ is a *subautomaton* of EFA $B_V = \langle \Sigma_B, V_B, Q_B, Q_B^\circ, Q_B^\omega, \rightarrow_B \rangle$, written $A \subseteq B$, if

- $\Sigma_A = \Sigma_B$ and $V_A = V_B$;
- $Q_A \subseteq Q_B$, $Q_A^\circ \subseteq Q_B^\circ$, and $Q_A^\omega \subseteq Q_B^\omega$;
- If $x \xrightarrow{\sigma:p_A}_A y$, then there exists a transition $x \xrightarrow{\sigma:p_B}_B y$ such that p_A logically implies p_B.

In words, a subautomaton $A \subseteq B$ results from the removal of some states or transitions, or from strengthening of updates in B. Clearly, $A \subseteq B$ implies $L(A) \subseteq L(B)$ and $L^\omega(A) \subseteq L^\omega(B)$. A particular class of subautomata is obtained by the operation of restriction.

Definition 5: Let $A_V = \langle \Sigma, V, Q, Q^\circ, Q^\omega, \rightarrow \rangle$ be an EFA, and let $X \subseteq Q \times \mathrm{dom}(V)$. The *restriction* of A_V to X is the EFA $A_{V\,|X} = \langle \Sigma, V, Q_{|X}, Q_{|X}^\circ, Q_{|X}^\omega, \rightarrow_{|X} \rangle$, where

- $Q_{|X} = \{\, x \in Q \mid (x, \bar{v}) \in X \text{ for some } \bar{v} \in \mathrm{dom}(V) \,\}$;
- $Q_{|X}^\circ = \{\, (x^\circ, \bar{v}^\circ) \mid x^\circ \in Q^\circ \,\}$;
- $Q_{|X}^\omega = Q_{|X} \cap Q^\omega$;

and $x \xrightarrow{\sigma:p \wedge q_y}_{|X} y$, if $x \xrightarrow{\sigma:p} y$ and q_y is a formula over V' such that $q_y(\bar{v}') = true$ if and only if $(y, \bar{v}') \in X$.

Lemma 1: Let $A_V = \langle \Sigma, V, Q, Q^\circ, Q^\omega, \rightarrow \rangle$ be an EFA and $X \subseteq Q \times \mathrm{dom}(V)$. Then $A_{V\,|X} \subseteq A_V$.

D. Supervisory Control with EFA

A key question in supervisory control theory is whether a given *plant* behaviour can be restricted through control in such a way that a given *specification* is satisfied [2]. For this purpose, the event alphabet Σ is partitioned into the set Σ_c of *controllable* events, whose occurrence can be inhibited through control, and the set Σ_u of *uncontrollable* events that cannot be directly disabled.

More precisely, given a prefix-closed plant behaviour $L \subseteq \Sigma^*$ and a specification behaviour $K \subseteq \Sigma^*$, it is desired to construct a so-called *supervisor* S, which restricts L to K by disabling only controllable events. A necessary and sufficient condition for the existence of S is controllability: a language $K \subseteq \Sigma^*$ is *controllable* [2] with respect to (wrt) a prefix-closed language $L \subseteq \Sigma^*$ if $\overline{K}\Sigma_\mathrm{u} \cap L \subseteq \overline{K}$. If the specification language K is controllable, then a supervisor achieving this behaviour can be implemented by an automaton representing K, which disables any controllable events not eligible in K. If the specification is *not* controllable, then it can be reduced to the *supremal controllable sublanguage*

$$\sup\mathcal{C}(K, L) = \bigcup \{\, K' \subseteq K \mid K' \text{ is controllable wrt } L \,\} .$$

$\sup\mathcal{C}(K, L)$ represents the largest sub-behaviour of K that can be achieved by controlling the plant behaviour L, and the process of computing it is known as *supervisor synthesis* [2].

If the specification and plant are given as EFA E_V and G_V, respectively, the controllability condition is extended as follows to consider the variables.

Definition 6: Let $E_V = \langle \Sigma, V, Q_E, Q_E^\circ, Q_E^\omega, \rightarrow_E \rangle$ and $G_V = \langle \Sigma, V, Q_G, Q_G^\circ, Q_G^\omega, \rightarrow_G \rangle$ be two EFA. E_V is *V-controllable* with respect to G_V if the following holds for all $s \in \Sigma^*$, all $\mu \in \Sigma_u$, and all $\bar{v}, \bar{v}' \in \mathrm{dom}(V)$: if $E_V \xrightarrow{s} (x_E, \bar{v})$ and $G_V \xrightarrow{s} (x_G, \bar{v}) \xrightarrow{\mu} (x'_G, \bar{v}')$ then there exists $x'_E \in Q_E$ such that $E_V \xrightarrow{s} (x_E, \bar{v}) \xrightarrow{\mu} (x'_E, \bar{v}')$.

V-controllability differs from standard controllability in that the specification must not only be able to process all uncontrollable events that are possible in the plant, on the occurrence of an uncontrollable event it must also update the variables in the same way as the plant. Differently, for controllable events, the specification can disable some or all of the associated variable updates.

Synthesis of EFA is defined using subautomata instead of languages. For this purpose, the specification E_V is composed with the plant G_V, and synthesis is performed over $E_V \parallel G_V$. Similarly to the classical case, the set

$$\mathcal{C}_V = \{\, K_V \subseteq E_V \| G_V \mid K_V \text{ is } V\text{-controllable wrt } G_V \,\}$$

contains a supremal EFA, denoted $\sup\mathcal{C}_V(E_V, G_V)$, representing the most permissive behaviour that can be implemented in G_V while satisfying the specification E_V. Prop. 2 shows that *V*-controllability is a generalisation of standard controllability in the deterministic case.

Proposition 2: Let G_V and E_V be state-deterministic EFA, such that G_V is also *V*-deterministic. Then

$$\sup\mathcal{C}(L(E_V \parallel G_V), L(G_V)) = L(\sup\mathcal{C}_V(E_V, G_V)) \ .$$

In addition to controllability, the controlled behaviour is typically required to be nonblocking.

Definition 7: An EFA $A_V = \langle \Sigma, V, Q, Q^\circ, Q^\omega, \rightarrow \rangle$ is *nonblocking* if $A_V \xrightarrow{s} (x, \bar{v})$ implies $(x, \bar{v}) \xrightarrow{t} (y, \bar{w})$ for some $(y, \bar{w}) \in Q^\omega$.

Given the above definitions, a *Supervisory Control Problem (SCP)* for EFA can be formulated as follows.

Problem 1 (SCP-V): Given state-deterministic EFA E_V and G_V for the specification and plant, such that G_V is total, find a nonblocking subautomaton $K_V \subseteq E_V \parallel G_V$ that is controllable with respect to G_V.

E_V and G_V are assumed to be modelled by state-deterministic EFA. G_V is also assumed to be total, that is, it records state changes in variables without imposing constraints. If the EFA $\sup\mathcal{C}_V(E_V, G_V)$ is nonblocking, then it is the supremal solution for the SCP-V, and can implement a supervision system by disabling all controllable events eligible in G_V that are not eligible in $\sup\mathcal{C}_V(E_V, G_V)$.

E. An Example of a Manufacturing System

This section demonstrates the use of EFA for modelling a simple manufacturing system shown in Fig. 1. The system consists of a robot (R) and two machines (M_1 and M_2) linked by buffers B_1 and B_2 with capacities of 10 and 5

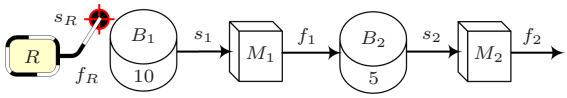

Fig. 1. Manufacturing System with intermediate buffering.

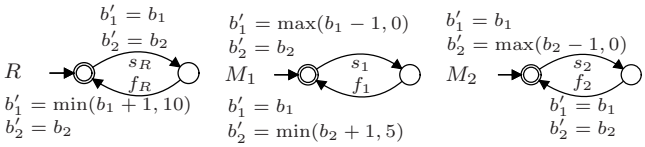

Fig. 2. EFA model of plant.

workpieces, respectively. The robot R takes workpieces from storage (event s_R) and stacks them on buffer B_1 (event f_R). Machine M_1 removes workpieces from B_1 (event s_1), manufactures and stacks them on B_2 (event f_1), and M_2 picks up workpieces from B_2 (event s_2), manufactures and releases them from the system (event f_2). Events f_1 and f_2 are uncontrollable, the others are controllable.

The plant is modelled by the EFA R, M_1, and M_2 shown in Fig. 2. This model uses two variables b_1 and b_2, representing the number of workpieces in each buffer, with domains $\mathrm{dom}(b_1) = \{0, \ldots, 10\}$ and $\mathrm{dom}(b_2) = \{0, \ldots, 5\}$ and initial values $b_1^\circ = b_2^\circ = 0$. In R, when a workpiece is unloaded to B_1 by event f_R, the number b_1 of workpieces in B_1 increases by 1 (update $b_1' = \min(b_1 + 1, 10)$). Likewise, the occurrence of event s_1 in M_1 decreases b_1 by 1, event f_1 increases the number b_2 of workpieces in B_2 by 1, and event s_2 decreases b_2 by 1.

The control objective is to avoid overflow and underflow of the buffers B_1 and B_2, which is modelled by EFA O_1, U_1, O_2, and U_2 in Fig. 3. In O_1, for example, the formula $b_1 < 10$ prohibits the robot to stack a workpiece on B_1 when the buffer is full (10 workpieces). The composed plant EFA $G_V = R \parallel M_1 \parallel M_2$ has 8 states, and the specification $E_V = O_1 \parallel U_1 \parallel O_2 \parallel U_2$ has just one state. Thanks to the variables, the requirement to avoid overflow and underflow is expressed concisely and independently of the buffer capacities.

The composition $E_V \| G_V$ and synthesis result $\sup\mathcal{C}_V(E_V, G_V)$ of these EFA unfold to 528 and 484 states, respectively, which is the same as with a standard FA model. The use of EFA does not reduce the state space, because the additional states generated by the variables must be considered in synthesis. It is shown in the remainder of this paper how variable abstraction can be used to simplify an EFA model, and reduce to effort to synthesise supervisors.

III. Variable Abstraction in Synthesis

This section shows how variable abstraction [9] can simplify EFA and avoid unfolding of variables in synthesis. This makes it possible to retain the modelling benefits of EFA,

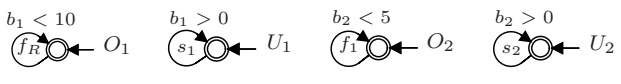

Fig. 3. EFA model for overflow and underflow avoidance specifications.

while at the same time making synthesis computationally more efficient. Conditions for optimality are also provided.

A. Variable Abstraction

Variable abstraction is a standard means of automaton simplification in model checking [9], which involves the removal of variables from an EFA. This is done through existential quantification.

Definition 8: Let A_V be an EFA and $W \subseteq V$. The *existential abstraction* of A_V is the EFA $\exists W A_V$ obtained from A_V by replacing each transition $x \xrightarrow{\sigma:p} y$ in A_V by $x \xrightarrow{\sigma:\exists W \exists W' p} y$ in $\exists W A_V$.

The existentially quantified formula $\exists W \exists W' p$ is defined over variables $U = V \setminus W$. It is true for $\bar{u}, \bar{u}' \in \mathrm{dom}(U)$ if there exist value combinations $\bar{w}, \bar{w}' \in \mathrm{dom}(W)$ such that $p(\bar{u}, \bar{w}, \bar{u}'\bar{w}')$ is true. For example, if $\mathrm{dom}(x) = \{0, \ldots, 9\}$ and $\mathrm{dom}(y) = \{0, 1\}$, then $\exists y \exists y' (x' = y)$ is equivalent to $x' = 0 \vee x' = 1$, and $\exists x \exists y \exists x' \exists y' (x' = y)$ is *true*.

By making $W = V$, one produces the coarsest abstraction $\exists V A_V$ of A_V, which is equivalent to the FA obtained by erasing all updates from A_V. Existential abstraction increases the behaviour of an EFA by removing constraints, and it is easy to show that $A_V \subseteq \exists W A_V$. It is known that existential abstraction preserves safety properties [9], but it does not necessarily preserve nonblocking or synthesis results.

In order to use variable abstraction in synthesis, the following alternative kind of controllability is defined.

Definition 9: Let E_V and G_V be two EFA. E_V is *$\exists$-controllable* with respect to G_V if the following holds for all $s \in \Sigma^*$, $\mu \in \Sigma_\mathrm{u}$ and $\bar{v} \in \mathrm{dom}(V)$: if $E_V \xrightarrow{s} (x_E, \bar{v})$ and $G_V \xrightarrow{s} (x_G, \bar{v}) \xrightarrow{\mu}$ then $E_V \xrightarrow{s} (x_E, \bar{v}) \xrightarrow{\mu}$, where $(x, \bar{v}) \xrightarrow{\mu}$ denotes the existence of x' and $\bar{v}'$ such that $(x, \bar{v}) \xrightarrow{\mu} (x', \bar{v}')$.

In words, E_V is $\exists$-controllable with respect to G_V if every uncontrollable event eligible in the plant G_V is also eligible in the specification E_V. Yet, unlike with *V*-controllability, the variables in the successor states of the specification may be different from the plant. With $\exists$-controllability, the specification has the power to choose the values of the variables in the successor states on the occurrence of uncontrollable events. $\exists$-controllability extends to EFA the Σ_u-preserving property [12], [13] that relates distinguished languages.

Similarly to *V*-controllability, the set

$$\mathcal{C}_\exists = \{\, K_V \subseteq E_V \parallel G_V \mid K_V \text{ is } \exists\text{-controllable wrt } G_V \,\}$$

contains a supremal EFA, denoted $\mathrm{sup}\mathcal{C}_\exists(E_V, G_V)$. Due to the unusual properties of $\exists$-controllability, this is unlikely to be a useful supervisor: it is only used to evaluate abstractions.

B. Synthesis of Control using Abstractions

The following results show that a solution to the SCP-V can be synthesised using an abstraction $\exists W G_V$ of the plant G_V. While still controllable, the synthesis result $\mathrm{sup}\mathcal{C}_V(E_V, \exists W G_V)$ may be more restrictive than $\mathrm{sup}\mathcal{C}_V(E_V, G_V)$. The following Theorem 3 provides a way to measure how suboptimal such an abstracted synthesis result is, by extending to EFA a result about the inclusion of distinguished languages [12], [13].

Theorem 3: Let G_V and E_V be state-deterministic EFA, such that G_V is total, and let $W \subseteq V$. Then

$$\begin{aligned} &L^\omega(\mathrm{sup}\mathcal{C}_V(E_V, \exists W G_V) \parallel G_V) \\ &\subseteq L^\omega(\mathrm{sup}\mathcal{C}_V(E_V, G_V)) \\ &\subseteq L^\omega(\mathrm{sup}\mathcal{C}_\exists(E_V, \exists W G_V) \parallel G_V)\ . \end{aligned}$$

By the first set inclusion in Theorem 3, a supervisor synthesised for an abstraction $\exists W G_V$, when composed with the original plant G_V, forms a controllable supervisor for this plant. If it also is nonblocking, then it solves the SCP-V.

Corollary 4: Let G_V and E_V be state-deterministic EFA, such that G_V is total, and let $W \subseteq V$. If $\mathrm{sup}\mathcal{C}_V(E_V, \exists W G_V) \parallel G_V$ is nonblocking, then it solves the SCP-V.

Moreover, Theorem 3 states upper and lower approximations for the optimal solution. $\mathrm{sup}\mathcal{C}_V(E_V, \exists W G_V)$ gives a supervisor that may be too restrictive, while $\mathrm{sup}\mathcal{C}_\exists(E_V, \exists W G_V)$ gives an upper approximation. If these approximations are equal, then the resulting supervisor is optimal.

Corollary 5: Let G_V and E_V be state-deterministic EFA, such that G_V is total, and let $W \subseteq V$. If

$$\mathrm{sup}\mathcal{C}_V(E_V, \exists W G_V) = \mathrm{sup}\mathcal{C}_\exists(E_V, \exists W G_V)\ , \qquad (1)$$

then

$$L^\omega(\mathrm{sup}\mathcal{C}_V(E_V, \exists W G_V) \parallel G_V) = L^\omega(\mathrm{sup}\mathcal{C}_V(E_V, G_V))\ .$$

Based on these results, an optimal solution for the SCP-V can be obtained using an abstraction, if this solution composed with the plant is nonblocking and (1) is satisfied. The nonblocking property can be checked without constructing a full synchronous product using compositional verification [15]. The following section shows how to compute the approximations in (1) without unfolding all the variables.

C. Computing Abstract Supervisors

Checking condition (1) requires the computation of abstract supervisors $\mathrm{sup}\mathcal{C}_V(E_V, \exists W G_V)$ and $\mathrm{sup}\mathcal{C}_\exists(E_V, \exists W G_V)$. Although the variables in W have been removed from the plant G_V, they appear in the specification E_V, so the EFA $E_V \| \exists W G_V$ still uses all the variables. It is shown in the following how the abstract supervisors can nevertheless be computed without unfolding the variables in W.

When an EFA is obtained by abstraction, it may not be necessary to unfold all its variables. This is formalised by the concept of variable restriction.

Definition 10: The *variable restriction* of EFA $A_V = \langle \Sigma, V, Q, Q^\circ, Q^\omega, \rightarrow \rangle$ to $U \subseteq V$ is the EFA $A_{V|U} = \langle \Sigma, U, Q, Q^\circ, Q^\omega, \rightarrow \rangle$.

The variable restriction is only well-defined if all updates of the EFA only contain variables in the reduced set. For example, if $V = U \dot\cup W$, then $(\exists W A_V)_{|U}$ is an EFA defined over variables U, while $\exists W A_V$ is defined over all the variables in V according to Def. 8.

The following algorithm computes $\mathrm{sup}\mathcal{C}_V(E_V \parallel \exists W G_V)$ without unfolding the abstracted variables in W. When an uncontrollable event occurs in the abstracted plant $\exists W G_V$, the variables in W can assume all possible values. Therefore, synthesis must remove the source states of any uncontrollable transition, for which the specification fails to allow all next-state values. These source states can be identified in advance.

This idea is captured by the concept of *strongly controllable states*. A state is strongly controllable, if the specification allows all possible next-state values of the variables in W for all uncontrollable events eligible in the plant. States that are not strongly controllable are unsafe and must be removed by synthesis. A state x may be strongly controllable for some variable values $\bar{w} \in \mathrm{dom}(W)$ but not for others. If $(x, \bar{w})$ is not strongly controllable, then all states with uncontrollable transitions to x are also unsafe, because the plant may take the system to $(x, \bar{w})$ on an uncontrollable event. However, controllable transitions to x may be possible if the unsafe next-state values of the variables in W are prevented by synthesis.

Algorithm to compute $\mathrm{sup}\mathcal{C}_V(E_V \parallel \exists W G_V)$*:*

1. Let $U = V \setminus W$, and construct the unfolded EFA $(\exists W E_V \parallel \exists W G_V)_{|U}$. Its states have the form $(x_E, x_G, \bar{u}) \in Q_E \times Q_G \times \mathrm{dom}(U)$.

2. Find the sets of *strongly controllable states* as follows:

$$\mathbf{SCS}_V = \{\, (x_E, x_G, \bar{u}, \bar{w}) \in Q_E \times Q_G \times \mathrm{dom}(V) \mid \text{for all } \bar{u}' \in \mathrm{dom}(U),\ \bar{w}' \in \mathrm{dom}(W), \text{ and } \mu \in \Sigma_\mathrm{u} \text{ such that } (x_G, \bar{u}) \xrightarrow{\mu} (x'_G, \bar{u}') \text{ in } (\exists W G_V)_{|U}, \text{ there exists } x'_E \in Q_E \text{ such that } (x_E, \bar{u}, \bar{w}) \xrightarrow{\mu} (x'_E, \bar{u}', \bar{w}') \text{ in } E_V \,\} \ ;$$

$$\mathbf{SCS}_U = \{\, (x_E, x_G, \bar{u}) \in Q_E \times Q_G \times \mathrm{dom}(U) \mid \text{for all } \bar{w} \in \mathrm{dom}(W), \text{ it holds that } (x_E, x_G, \bar{u}, \bar{w}) \in \mathbf{SCS}_V \,\} \ .$$

3. Find the supremal strongly controllable state set. A state set $X \subseteq Q_E \times Q_G \times \mathrm{dom}(U)$ is *strongly controllable* with respect to $\exists W G_V$ if, for all $(x_E, x_G, \bar{u}) \in X$ and all uncontrollable transitions $(x_G, \bar{u}) \xrightarrow{\mu} (x'_G, \bar{u}')$ in $(\exists W G_V)_{|U}$, there exists a transition $(x_E, \bar{u}) \xrightarrow{\mu} (x'_E, \bar{u}')$ in $(\exists W E_V)_{|U}$ such that $(x'_E, x'_G, \bar{u}') \in X \cap \mathbf{SCS}_U$. The union of strongly controllable state sets is again strongly controllable, so it is possible to compute

$$\widehat{\mathbf{SCS}}_U = \bigcup \{\, X \subseteq Q_E \times Q_G \times \mathrm{dom}(U) \mid X \text{ is strongly controllable wrt } \exists W G_V \,\} \ .$$

4. Construct the restriction

$$\hat{K}_U = (\exists W E_V \parallel \exists W G_V)_{|U \,|\widehat{\mathbf{SCS}}_U} \ .$$

5. Construct the result EFA

$$\hat{K}_V = \langle \Sigma, V, Q_E \times Q_G, Q^\circ, Q_E^\omega \times Q_G^\omega, \rightarrow \rangle \qquad (2)$$

where

$$Q^\circ = \{\, (x_E^\circ, x_G^\circ) \in Q_E^\circ \times Q_G^\circ \mid (x_E^\circ, x_G^\circ, \bar{u}^\circ) \in \widehat{\mathbf{SCS}}_U \text{ and } (x_E^\circ, x_G^\circ, \bar{u}^\circ, \bar{w}^\circ) \in \mathbf{SCS}_V \,\}$$

and

$$(x_E, x_G) \xrightarrow{\sigma : \hat{p} \wedge p_E \wedge p_{\mathbf{SCS}}\langle x'_E, x'_G \rangle} (x'_E, x'_G)$$

if $(x_E, x_G) \xrightarrow{\sigma : \hat{p}} (x'_E, x'_G)$ in $\hat{K}_U$, and $x_E \xrightarrow{\sigma : p_E} x'_E$ in E_V, and $p_{\mathbf{SCS}}\langle x'_E, x'_G \rangle$ is a formula over V' such that $p_{\mathbf{SCS}}\langle x'_E, x'_G \rangle(\bar{v}') = true$ if $(x'_E, x'_G, \bar{v}') \in \mathbf{SCS}_V$.

The set $\widehat{\mathbf{SCS}}_U$ can be computed in step 3 on the state space of $(\exists W E_V \| \exists W G_V)_{|U}$ without unfolding the variables in W. The states in $\mathbf{SCS}_V$ and $\mathbf{SCS}_U$ can be represented symbolically and computed in advance, or membership in these sets can be evaluated one transition at a time when computing $\widehat{\mathbf{SCS}}_U$. In this case, the potentially large set $\mathbf{SCS}_V$ is never actually computed.

Based on these observations, the updates in the specification E_V, which may contain variables in W, do not affect the computation of $\widehat{\mathbf{SCS}}_U$ in step 3. The updates from E_V can simply be copied to the synthesis result $\hat{K}_V$ in step 5. Theorem 6 confirms that this algorithm produces the desired result $\mathrm{sup}\mathcal{C}_V(E_V \parallel \exists W G_V)$.

Theorem 6: Let E_V and G_V be two state-deterministic EFA, and let $\hat{K}_V$ be the EFA (2) computed by the above algorithm. It holds that $\hat{K}_V = \mathrm{sup}\mathcal{C}_V(E_V, \exists W G_V)$.

It is next shown how to compute $\mathrm{sup}\mathcal{C}_\exists(E_V \parallel \exists W G_V)$ without unfolding the abstracted variables in W. With $\exists$-controllability, the specification can choose the next-state values of variables on the occurrence of uncontrollable events, making it possible to enter states that are not strongly controllable. Therefore, the following algorithm replaces strongly controllable states by *weakly controllable states*.

A state x, and uncontrollable transitions leading to it, can be retained in synthesis as long as it is weakly controllable for some value combination $\bar{w} \in \mathrm{dom}(W)$. For this approach to be feasible without considering the updates of the specification E_V during synthesis, the specification cannot impose constraints on the next-state values of the variables. This additional requirement is not a strong one, as uncontrollable specification transitions that attempt to constrain next-state values are likely to be removed by $\mathrm{sup}\mathcal{C}_V$ anyway.

Algorithm to compute $\mathrm{sup}\mathcal{C}_\exists(E_V \parallel \exists W G_V)$*:*

1. Let $U = V \setminus W$, and construct the unfolded EFA $(\exists W E_V \parallel \exists W G_V)_{|U}$.

2. Find the sets of *weakly controllable states* as follows:

$$\mathbf{WCS}_V = \{\, (x_E, x_G, \bar{v}) \in Q_E \times Q_G \times \mathrm{dom}(V) \mid \text{for all } \mu \in \Sigma_\mathrm{u}, \text{ if } (x_G, \bar{v}) \xrightarrow{\mu} \text{ in } \exists W G_V \text{ then also } (x_E, \bar{v}) \xrightarrow{\mu} \text{ in } E_V \,\} \ ;$$

$$\mathbf{WCS}_U = \{\, (x_E, x_G, \bar{u}) \in Q_E \times Q_G \times \mathrm{dom}(U) \mid \text{there exists } \bar{w} \in \mathrm{dom}(W) \text{ such that } (x_E, x_G, \bar{u}, \bar{w}) \in \mathbf{WCS}_V \,\} \ .$$

3. Synthesise the EFA

$$\hat{H}_U = \mathrm{sup}\mathcal{C}_\exists((\exists W E_V \| \exists W G_V)_{|U \,|\mathbf{WCS}_U}, (\exists W G_V)_{|U}) \ .$$

4. Construct the result EFA

$$\hat{H}_V = \langle \Sigma, V, Q_E \times Q_G, Q^\circ, Q_E^\omega \times Q_G^\omega, \rightarrow \rangle \qquad (3)$$

where

$$Q^\circ = \{\, (x_E^\circ, x_G^\circ) \in Q_E^\circ \times Q_G^\circ \mid (x_E^\circ, x_G^\circ, \bar{u}^\circ, \bar{w}^\circ) \in \mathbf{WCS}_V \,\}$$

and

$$(x_E, x_G) \xrightarrow{\hat{p} \wedge p_E \wedge p_{\mathbf{WCS}}\langle x'_E, x'_G \rangle} (x'_E, x'_G)$$

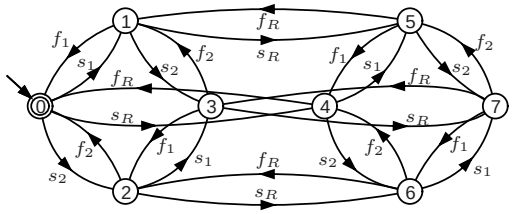

Fig. 4. The abstraction $\exists b_1 \exists b_2 E_V \parallel \exists b_1 \exists b_2 G_V$.

if $(x_E, x_G) \overset{\sigma:\hat{p}}{\to} (x'_E, x'_G)$ in $\hat{H}_U$, and $x_E \xrightarrow{\sigma:p_E} x'_E$ in E_V, and $p_{\mathbf{WCS}}\langle x'_E, x'_G\rangle$ is a formula over V' such that $p_{\mathbf{WCS}}\langle x'_E, x'_G\rangle(\bar{v}') = true$ if $(x'_E, x'_G, \bar{v}') \in \mathbf{WCS}_V$.

Theorem 7: Let E_V and G_V be two state-deterministic EFA, such that all updates on uncontrollable transitions in E_V are defined over the variables V (not using V'). Furthermore, let $\hat{H}_V$ be the EFA (3) computed by the above algorithm. It holds that $\hat{H}_V = \sup\mathcal{C}_\exists(E_V, \exists W G_V)$.

The above algorithms can be used to compute the approximations in (1) and determine whether a supervisor synthesised from an abstraction is least restrictive. One question that remains is how to choose the set W of variables for abstraction. It is computationally advantageous to choose an abstraction that is as coarse as possible, starting with $W = V$. Yet, this reduces the amount of information available to synthesis, and may give a solution that is too restrictive or no solution at all. The following example shows how the set of variables can be refined in cases where (1) is not satisfied.

D. Manufacturing System Revisited

Consider again the manufacturing system introduced in Section II-E. Given the plant G_V in Fig. 2 and the specification E_V in Fig. 3, the first step is to consider the coarsest abstraction obtained by erasing both variables b_1 and b_2.

Fig. 4 shows the synchronous composition of the abstractions $\exists b_1 \exists b_2 E_V \parallel \exists b_1 \exists b_2 G_V$. State 1 is not strongly controllable, because the uncontrollable f_1-transition has the update $b_2 < 5$ in the specification, which is not enabled when $b_2 = 5$. Then state 3 is unsafe because of the uncontrollable transition $3 \overset{f_2}{\to} 1$, which may take the abstracted plant $\exists b_1 \exists b_2 G_V$ to state 1 with $b_2 = 5$. Therefore, states 3 and, for similar reasons, 7 are removed from $S'_V = \sup\mathcal{C}_V(E_V, \exists b_1 \exists b_2 G_V)$.

On the other hand, state 1 is weakly controllable, because event f_1 is allowed by the specification when $b_2 \neq 5$. Then an $\exists$-controllable supervisor can also allow state 3, by choosing a next-state value $b'_2 \neq 5$ when executing $3 \overset{f_2}{\to} 1$. Thus $S'_V \neq S'_\exists = \sup\mathcal{C}_\exists(E_V, \exists b_1 \exists b_2 G_V)$, so the condition (1) is not satisfied, and it cannot be concluded that $S'_V \parallel G_V$ is the optimal solution. In fact, a supervisor based on this abstraction will never allow machine M_1 to start.

A better result is obtained by improving the abstraction. The fact that state 1 fails to be strongly controllable because of certain values of b_2 suggests to retain this variable in the abstraction, so the next attempt considers $\exists b_1 E_V \parallel \exists b_1 G_V$. This EFA has the same structure as Fig. 4, but this time the possible values $b_2 = 0, \ldots, 5$ are considered in each state, and the EFA unfolds to 48 states. State $(1, 5)$ is not strongly controllable, but this time it is only reached by the uncontrollable transition $(3, 5) \overset{f_2}{\to} (1, 5)$. Only states $(1, 5)$, $(3, 5)$, $(5, 5)$, $(7, 5)$ are found to be unsafe. Synthesis avoids them by adding the condition $b_2 < 5$ to the s_1-transitions, resulting in a 44-state EFA for $S''_V = \sup\mathcal{C}_V(E_V, \exists b_1 G_V)$.

It turns out that $S''_V = S''_\exists$, and these EFA are nonblocking when composed with the plant G_V. By Corollaries 4 and 5, the optimal solution has been found. This is achieved by exploring an state space of 48 states, while the standard synthesis explores the unfolded state space of 528 states.

IV. Conclusions

A method for variable abstraction in the synthesis of supervisors from Extended Finite-State Automata (EFA) models has been presented. It has been shown that, under certain circumstances, variables can be existentially quantified out from an EFA without affecting the synthesis result. While the proposed method produces least restrictive controllable supervisors, the nonblocking property is only indirectly supported through an additional check. Future research will investigate methods for identifying abstractions automatically, and the possibility of including nonblocking-preserving abstractions in the same framework.

References

[1] P. J. G. Ramadge and W. M. Wonham, "The control of discrete event systems," *Proc. IEEE*, vol. 77, no. 1, pp. 81–98, Jan. 1989.

[2] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd ed. Springer, 2008.

[3] N. Halbwachs, *Synchronous Programming of Reactive Systems*. Kluwer, 1993.

[4] D. Harel, "Statecharts: a visual formalism for complex systems," *Science of Computer Programming*, vol. 8, no. 3, pp. 231–274, 1987.

[5] E. Börger and R. Stärk, *Abstract State Machines*. Springer, 2003.

[6] Y. Chen and F. Lin, "Modeling of discrete event systems using finite state machines with parameters," in *Proc. 2010 IEEE Int. Conf. Control Applications (CCA)*, Anchorage, AK, USA, 2000, pp. 941–946.

[7] M. Sköldstam, K. Åkesson, and M. Fabian, "Modeling of discrete event systems using finite automata with variables," in *Proc. 46th IEEE Conf. Decision and Control, CDC '07*, Dec. 2007, pp. 3387–3392.

[8] L. Ouedraogo, R. Kumar, R. Malik, and K. Åkesson, "Nonblocking and safe control of discrete-event systems modeled as extended finite automata," *IEEE Trans. Autom. Sci. Eng.*, vol. 8, no. 3, pp. 560–569, July 2011.

[9] B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and P. Schnoebelen, *Systems and Software Verification*. Springer, 2001.

[10] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang, "Symbolic model checking: 10^{20} states and beyond," *Information and Computation*, vol. 98, no. 2, pp. 142–170, 1992.

[11] T. Le Gall, B. Jeannet, and H. Marchand, "Supervisory control of infinite symbolic systems using abstract interpretation," in *Proc. 46th IEEE Conf. Decision and Control, CDC '05*, Dec. 2005, pp. 30–35.

[12] G. Bouzon, M. H. de Queiroz, and J. E. R. Cury, "Exploiting distinguishing sensors in supervisory control of DES," in *Proc. 7th Int. Conf. Control and Automation, ICCA '09*, Christchurch, New Zealand, Dec. 2009, pp. 442–447.

[13] J. E. R. Cury, M. H. de Queiroz, G. Bouzon, and M. Teixeira, "Supervisory control of discrete event systems with distinguishers," submitted to journal, 2012.

[14] C. A. R. Hoare, *Communicating Sequential Processes*. Prentice-Hall, 1985.

[15] H. Flordal and R. Malik, "Compositional verification in supervisory control," *SIAM J. Control and Optimization*, vol. 48, no. 3, pp. 1914–1938, 2009.