Symbolic Synthesis of a Class of Discrete-event Controllers for Process Systems

A. Sanchez*, J. Reza, J. Douriet and R. E. Gonzalez
Depto. de Ingeniería Eléctrica y Computación
Centro de Investigacion y Estudios Avanzados (Cinvestav)
Apdo. Postal 31-438, Guadalajara 45091, Jalisco, México

Abstract
This paper presents a predicate-based symbolic synthesis framework for discrete-event controllers forcing at most one control action in each stage of the control pattern. Binary Decision Diagrams (BDD)-encoded algorithms carry out the calculations required to synthesize a controller of this nature. Numeric experiments show a better performance than other known algorithms. Controllers for systems with state spaces of up to $2.9 \times 10^6$ were synthesized in a standard PC without using decomposition or modularization. The synthesis tools are available from www.gdl.cinvestav.mx/sspc.

Keywords: Discrete-event systems, controller synthesis, forced actions

1. Introduction
A symbolic synthesis framework is presented for a class of discrete-event (DES) controllers applied to systems modelled as finite state machines (FSM). This class of controllers, termed procedural controllers, are capable of enforcing controllable transitions upon the system in order to fulfil behavioural specifications (Sanchez et al., 1999). Their application in the batch processing industries has been studied previously, quantifying operational and economic benefits (Sanchez et al., 2002). The synthesis framework is based on a predicate representation of the FSM and their structural properties (Reza, 2002) previously proposed for the synthesis of supervisory controllers (Kumar and Garg, 1995; Zhang and Wonham, 2002). Section 2 takes from Sanchez et al., (1999) the notions of procedural controller, controllability, completeness and the theorem guaranteeing the existence of a controller. Based on these concepts, the symbolic framework is introduced in section 3, followed by a fix-point operator for calculating the maximal superstructure of controllers and an algorithm for the synthesis of non-blocking procedural controllers. All algorithms were encoded using Binary Decision Diagrams (BDD) (Andersen, 1997). As an example, procedural controllers were synthesized in section 4 for a set of pressurized reactors operating in parallel with state spaces of up to $2.9(10^6)$ without using either decomposition or modularization techniques. Results of numerical experiments are shown confirming and quantifying the exponential nature (time and memory) of the algorithms.

*Corresponding author. e-mail address: arturo@gdl.cinvestav.mx
2. Fundamentals

Models are built using finite state machines (FSMs) and their associated languages \((L, L_m)\) in a standard fashion. The transition set \(\Sigma\) is composed of two disjoint sets of uncontrollable \(\Sigma_u\) and controllable \(\Sigma_c\) transitions. The plant \(P\) synchronizes with the controller \(C\) such that the closed loop behavior is given by \(L(C/P) = L(P) \cap L(C)\) and \(L_m(C/P) = L_m(P) \cap L(C)\). A procedural controller can be understood as an FSM \(C = \{X, \Sigma, \gamma, x_0, X_m\}\) with \(X\) and \(X_m\) as the sets of controller states and marked states.

Transition function \(\gamma : X \times \Sigma \rightarrow X\) is built for each \(x \in X\) and \(\sigma \in \Sigma\) such that either 1) \(\sigma \in \Sigma_u \land (\forall \sigma' \in \Sigma, \gamma (x, \sigma)\) is undefined) or 2) \(\sigma \in \Sigma_c \land (\forall \sigma' \in \Sigma, \gamma (x, \sigma)\) is undefined). Thus, \(C\) can be in either 1) a state in which any of the plant-enabled uncontrollable transitions can occur, or 2) a state in which one and only one controllable transition occurs. The controller executing synchronously with the plant pre-empts the execution of uncontrollable transitions from some of the plant states by executing controllable transitions.

A controller is said to be complete with respect to a plant \(P\) if for each \(s \in \Sigma^*\) with both \(\gamma(x_0, s)\) and \(\delta(q_0, ss\sigma_s)\) defined, imply that either \(\gamma(x_0, s\sigma_s)\) is defined or \(\exists \sigma_c \in \Sigma_c\) such that \(\gamma(x_0, s\sigma_c)\) is defined and \(\gamma(x_0, s\Sigma_c)\) is undefined. The controller objective is to maintain the closed-loop language within certain boundaries. Thus, a language \(K \subseteq L\) is said to be controllable with respect to \(P\) if \(\forall s \in K\) either \(s\Sigma_u \in L \land s\Sigma_c \in \overline{K}\), or \(\exists \sigma_c \in \Sigma_c\) such that \((s\sigma_c \in \overline{K}) \land (\forall \sigma_c \in \Sigma_c \mid s\sigma_c \notin L \cap \overline{K})\). That is, for every state in an FSM generating a controllable language \(K\) either 1) all the uncontrollable transitions end in one of the states of the FSM, regardless of the existence of controllable transitions or, 2) there is at least one controllable transition leaving the state and ending in another state of the FSM. This means that even if there are uncontrollable transitions leading outside the FSM, it is possible to keep the plant states in the desired set by proper control actions. The FSM that generates all possible controllable languages in identified as the superstructure of controllers. From this superstructure a procedural controller must be chosen using optimality criteria.

The aim is to synthesize complete procedural controllers. Sanchez et al. (1999) showed that a given controller superstructures \(M\), generating language \(K(M) \subseteq L(P)\) and \(K_m = L_m(P) \cap K (K_m \neq \emptyset)\), if \(M\) is trim and \(K_m\) is closed and controllable, then there is a non-blocking and complete controller \(C\), with \(L(C/P) \subseteq K_m\).

3. Efficient Calculation of Procedural Controllers

Predicates can be used to express structural properties of FSMs (e.g. Zhang and Wonham, 2002). They also facilitate the development of synthesis algorithms that can be encoded using efficient tools, such as Binary Decision Diagrams.
3.1 Predicate representation of FSM operations and their BDD implementation

A predicate $Pr$ about the state set of an FSM $M = \{Q, \Sigma, \delta, q_o, Q_m\}$ may be interpreted as a function of the form $Pr : Q \rightarrow \{0, 1\}$. Thus, $Pr$ identifies a subset of states

$Pr(q) = 1$.

In the same fashion, $Re : Q \times \Sigma \times Q \rightarrow \{0, 1\}$ describes a function corresponding to the set

$$Re := \{(q, \sigma, q') \in Q \times \Sigma \times Q \mid Re(q, \sigma, q') = 1\}.$$ Thus, $M$ can be represented by

$M' = \{Q, \Sigma, ||, ||, ||, ||, ||, ||, ||, ||, \}$ with

- $T : Q \times \Sigma \times Q \rightarrow \{0, 1\}$ as the transition function predicate. The triplet $(q, \sigma, q')$ satisfies $T$ if and only if $q' = \delta(q, \sigma)$.
- $I : Q \rightarrow \{0, 1\}$ as the initial state predicate. A state $q$ satisfies $I$ if and only if $q = q_o$.
- $Mr : Q \rightarrow \{0, 1\}$ as the marked states predicate s.t. $q$ satisfies $Mr$ if and only if $q \in Qm$.

3.2 Fix-point operators for supremal controllable sublanguages

Let $P$ and $E$ be the plant and specification FSMs. The starting point for calculating the supremal controllable language can be a composition of these two FSMs:

$PE = \{Q \times X, \Sigma, ||, ||, ||, ||, ||, ||, ||, ||, \}$ with

$$|| || := \{(q, x) \in Q \times X \mid q \in ||, || and x \in ||, ||\}$$

$$Mr_{PE} := \{(q, x) \in Q \times X \mid q \in ||, || and x \in ||, ||\}$$

$$PE := \{(q, x), \sigma, (q', x') \} \text{ defined according to the composition operation}\$$

Thus, the transition relation $PE$ is a function with domain $W := (Q \times X) \times \Sigma \times (Q \times X)$. Now, let $T_Z$ be a transition relation defined on the same domain. $T_Z$ establishes the relation between a plant and specification states. Using this predicate representation, a fix-point operator $\Omega : 2^W \rightarrow 2^W$ for the superstructure of procedural controllers is:

$$\Omega(T_Z) := (T_{PE} \setminus T_Z) - \{(q, x), \sigma, (q', x') \in T_Z \mid (q, x) \in K_1, \sigma, \in \Sigma\} - \{(q, x), \sigma, (q', x') \in T_Z \mid (q, x) \in K_2, \sigma, \in \Sigma\} - \{(q, x), \sigma, (q', x') \in T_Z \mid (q, x) \in K_3, \sigma, \in \Sigma\} - \{(q, x), \sigma, (q', x') \in T_Z \mid (q, x) \in K_4, \sigma, \in \Sigma\}$$

where:

- $K_1 := \{(q, x) \exists (q, \sigma, q_z) \in T_Z \mid \sigma, \in \Sigma\}$ and $(q, x), \sigma, (q_z, x_z) \not\in T_Z$.
- $K_2 := \{(q, x) \exists (q, \sigma, q_z) \in T_Z \mid \sigma, \in \Sigma\}$ and $(q, x), \sigma, (q_z, x_z) \in T_Z$.
- $K_3 := \{(q, x) \exists (q, \sigma, q_z) \in T_Z \mid \sigma, \in \Sigma\}$ and $(q, x), \sigma, (q_z, x_z) \not\in T_Z$.
- $K_4 := \{(q, x) \exists (q, \sigma, q_z) \in T_Z \mid \sigma, \in \Sigma\}$ and $(q, x), \sigma, (q_z, x_z) \not\in T_Z$.

$K_1$ is the set of state pairs $(q, x)$ from which at least one uncontrollable transition is enabled from $q$ in the process FSM and is not enabled in the equivalent state of $T_Z$ but there is at least one controllable transition capable of preempting the occurrence of any
process-enabled uncontrollable transition. $K_2$ represents the set of state pairs $(q, x)$ from which at least one uncontrollable transition is enabled from $q$ in the process FSM and is not enabled in the equivalent state of $\|T_Z\|$, such that there is no controllable transition capable of preempting the occurrence of an uncontrollable transition. $\|Co\|$ guarantees that the surviving states are coreachable. This set is obtained by successively computing the pre-image of the set of marked states $\|Co\| := \bigcup_{i=0}^{\infty} Pre^i \|Mr\|$, where $Pre^i \|P\|$ is defined inductively as $Pre^0 \|Mr\| := \|Mr\|$ and $Pre^{i+1} \|Mr\| := Pre(Pre^i \|Mr\|)$, for all $k \in \mathbb{N}$. This fix-point operation produces the desired supremal controllable language (Reza, 2002).

### 3.3 Calculation of Procedural Controllers
Based on the maximal superstructure, the algorithm shown in fig. 1 gives as a result a non-blocking complete procedural controller (Reza, 2002). The algorithm terminates obtaining the desired procedural controller. Its complexity is cubic on the number of states. As mentioned in the introduction, the encoding of these algorithms was carried out using BDDs in a standard fashion.

### 4. Example. Pressurization Cycle of Reactors
Sanchez et al. (1999) presented this example to illustrate the synthesis of procedural controllers. The objective is to design a procedural controller for the reactor unit shown in figure 2. The operational goals of the controller are: i) to pressurize-depressurize the reactor in a cyclic manner and ii) to avoid high-pressure operating states. The process and specification FSM models for one reactor are shown in figures 3 and 4. In the initial process state (state 0 in figure 3), the operator must press the on/off button. The controller responds by opening the valve and then it waits for the normal pressure switch to be activated (states 2-5-9). Once this signal is received, the controller shuts the valve off and waits for the button to be depressed by the operator in order to reach the desired (marked) state of normal operation (states 9-6-3). Once the reaction is over, the reactor is depressurized and the process returns to the initial state (states 3-0). Assuming that $L(C/P) = L(C) \cap L(P) = L(C)$, the FSM realizing the procedural controller is shown in figure 5. It can be observed that the closed-loop behavior meets the specification.

Performance experiments were carried out synthesizing the superstructure and a procedural controller for a set of reactor units operating in parallel. Table 1 shows the results of the synthesis experiments carried out on a PC Pentium 4 at 2.4 GHz with 1 GB RAM. The largest reachable state set that could be handled in reasonable times was of $2.9(10^6)$ equivalent to 6 reactors operating in parallel.

### 5. Conclusions
The experiment results show that the BDD-encoded algorithms for the synthesis of the superstructure and the controller alleviate to a certain extent the problem of state explosion. These results compares favorably against the best published results known to us whose maximum sizes were of the order of $4.5 \times 10^4$ (Baird, 1999). Sanchez et al. (2003) presented and benchmarked BDD-encoded algorithms for other operations.
required for the controller synthesis such as the synchronous and asynchronous products. The tools can be obtained from www.gdl.cinvestav.mx/sspc.

Acknowledgements
AS kindly acknowledges partial financial support from Conacyt, Mexico (project 365935U). Calculations were carried out by Eduardo Ramirez.

References

Table 1: Memory, time and nodes for calculation of maximal superstructure and procedural controller

<table>
<thead>
<tr>
<th>Controller Superstructure</th>
<th>Procedural Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>$</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>144</td>
</tr>
<tr>
<td>3</td>
<td>1,728</td>
</tr>
<tr>
<td>4</td>
<td>20,736</td>
</tr>
<tr>
<td>5</td>
<td>248,832</td>
</tr>
<tr>
<td>6</td>
<td>2.9(10$^6$)</td>
</tr>
</tbody>
</table>
Procedural Controller

**R:** fsm

**Z:** fsm;

A B X X:

 sets; temp : relation

* Max. superstructure Z as starting point *

C Q, , , , :

**C** TI Mr

= Σ =

* States with ctrble and unctrble trans */

{ 11: , : ( (( , ), ,( , ))

AC u C

X qx Q q x T σ=∈ ∃ ∈

22) ( (( , ), ,( , ))

u u C c C

qx q x T ∈Σ ∧ ∃ ∈ ∈Σ

(, )

Aqx X

σσ=− ∈ Σ

* Finding coreachable states */

If (( ( , )) tempImg q x Co ⊆

then

{ 11: ((, ), ,( ', ')) :

CC cq x q x σσ=− ∈Σ

else

{ 11: ((, ), ,( ', ')) :

CC uq x q x σσ=− ∈Σ

Next (q, x)

* States with more than one conrtble trans *

{ 11 2 2, ), ,( , )),(( , ), ,( , )) }

C cq x q x q x s q x T σ ∈

For each Error!do

{}:: (( , ), ,( ', '))

Xc c Cqx q x T σΣ= ∃ ∈

repeat

Choose σ c ∈Σ X

:{

XX c σΣ= Σ −

}:( ( , ), ,( ', ')) :

Cctemp T q x q x σσ=− ≠

* Coreachable set with current ctrbl trans  */

If (( ( , )) tempImg q x Co ⊆

then

{ 11: ((, ), ,( ', ')) :

CC q x q x σσ=− ≠

Until (( ( , )) tempImg q x Co ⊆

Next (q, x)

* Erase unreachable states *

{ 11: ((, ), ,( , )) : (, )

CC T qx q x qx A l σ=∩ ∈

* Create new FSM C from TC*/

{ 11: Q , , , ,

CC CCT I M =Σ

return C

Figure 1: Algorithm for procedural controller synthesis

Figure 2: Pressurized Reactor Unit

Figure 3: Process model FSM

Figure 4: Specification model FSM

Figure 5: Controller model FS