An Approach for Supervisor Reduction of Discrete-event Systems

Huimin Zhang
Guangxi Normal University, China

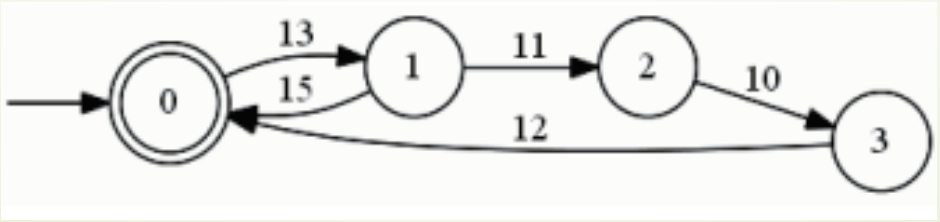
1. Basics of SCT and automaton

Automaton

A deterministic finite automaton (DFA), denoted by **G**, is a quintuple
$\mathbf{G} = (Q, \Sigma, \delta, q_0, \Gamma, Q_m)$,

where

– Q is the state set,
– Σ is the alphabet,
– $\delta: Q \times \Sigma \rightarrow Q$ is the transition function,
– $q_0 \in Q$ is the initial state,
– $\Gamma : Q \rightarrow 2^{\Sigma}$ is the active event function; $\Gamma(q)$ is the set of all events σ for which $\delta(q, \sigma)$!,
– $Q_m \subseteq Q$ is the set of marker states.



1. Basics of SCT and automaton

Automaton

Let $G_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, \Gamma_1, Q_{m1})$ and $G_2 = (Q_2, \Sigma_2, \delta_2, q_{02}, \Gamma_2, Q_{m2})$ be two automata. The synchronous product of automata G_1 and G_2 is defined as $G_1 \parallel G_2 = Ac(Q_1 \times Q_2, \Sigma, \delta, (q_{01}, q_{02}), \Gamma_{1 \parallel 2}, Q_{m1} \times Q_{m2})$ where

$$\delta((q_1, q_2), \sigma) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)) & \text{if } \sigma \in \Gamma_1(x_1) \cap \Gamma_2(x_2) \\ (\delta_1(x_1, \sigma), x_2) & \text{if } \sigma \in \Gamma_1(x_1) \setminus \Sigma_2 \\ (x_1, \delta_2(x_2, \sigma),) & \text{if } \sigma \in \Gamma_2(x_2) \setminus \Sigma_1 \\ \text{undefined} & \text{otherwise} \end{cases}$$

1. Basics of SCT and automaton

Supervisory control theory

Let $\mathbf{G} = (Q, \Sigma, \delta, q_0, \Gamma, Q_m)$
be a (nonempty) controlled DES, with $\sum = \sum_c \cup \Sigma_u$.

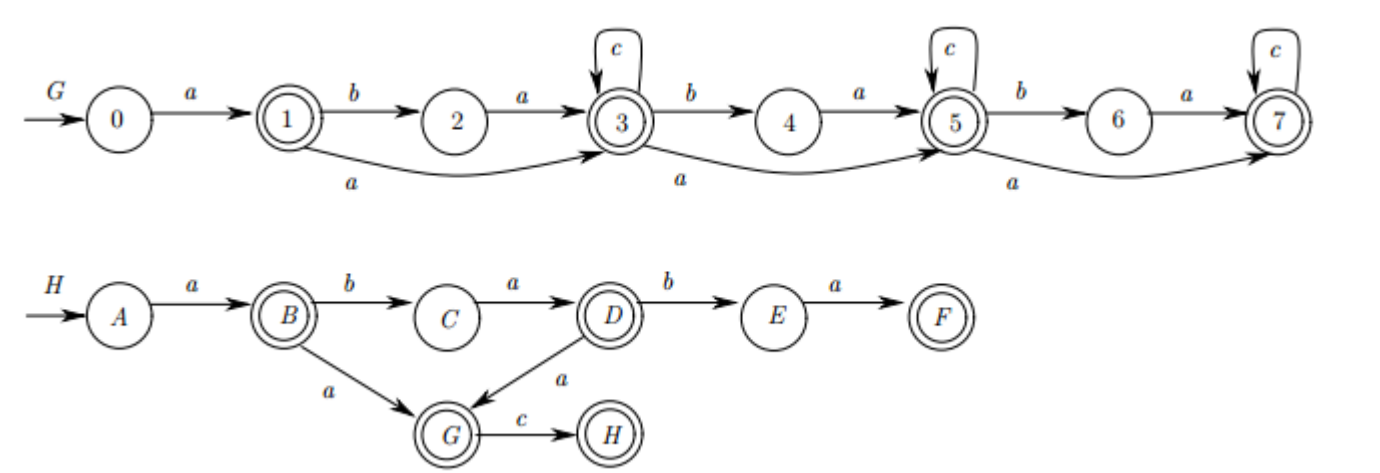
A *supervisory control for* **G** is any map $V : L(\mathbf{G}) \rightarrow \Gamma$. The pair $(\mathbf{G}; V)$ will be written $V/\mathbf{G}$, to suggest '**G** under the supervision of V'. The *closed behavior* of $V/\mathbf{G}$ is defined to be the language
$L(V/\mathbf{G}) \subseteq L(\mathbf{G})$ described as follows.
(i) $\epsilon \in L(V/\mathbf{G})$
(ii) If $s \in L(V/\mathbf{G})$, $\sigma \in V(s)$, and $s\sigma \in L(\mathbf{G})$ then $s\sigma \in L(V/\mathbf{G})$
(iii) No other strings belong to $L(V/\mathbf{G})$.

[1]Vaz, A. F., Wonham, W. M.: On supervisor reduction in discrete-event systems. International Journal of Control, 44(2), 475–491 (1986)

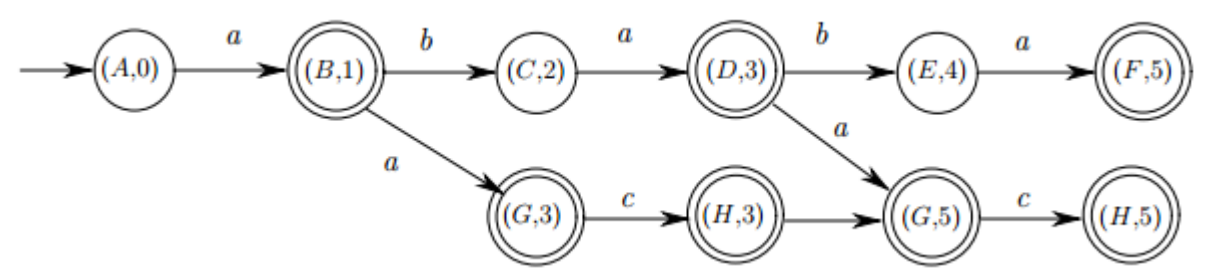
[2] Su, R., Wonham, W. M.: Supervisor reduction for discrete-event systems. Discrete Event Dynamic Systems, 14(1), 31–53 (2004)

2. Motivation

Iterative calculation of the supremal supervisor w.r.t G and H



Automata *G* and *H*



Automaton H_0

3. Problem formalization

Given a plant $G = (X, \Sigma, f, \Gamma_G, x_0, X_m)$, a specification $H = (Y, \Sigma, g, \Gamma_H, y_0, Y)$ and the supremal supervisor $\mathbf{S} = (Q, \Sigma, \delta, q_0, \Gamma_S, Q_m)$ with respect to G and H, compute a reduced supervisor $R = (Y', \Sigma, \xi, y_0, \Gamma_R, Y_{m'})$ such that

$$\boldsymbol{L_m(G \parallel R) = L_m(S),}$$
$$\boldsymbol{L(G \parallel R) = L(S),}$$
$$\textbf{and } \boldsymbol{|R| \leq |S|}$$

The supremal supervisor with respect to G and H is computed by Alg. 1, where $Q \subseteq X \times Y$, $Q_m \subseteq X_m \times Y_m$ and $q_0 = (x_0, y_0)$. Each state q of the supervisor S is denoted by a two-tuples (x, y), where $x \in X$ and $y \in Y$. For each string $s \in L(S)$, if $\delta(q_0, s) = (x, y)$, then we have $f(x_0, s) = x$ and $g(y_0, s) = y$.

3. Problem formalization

Algorithm 1: A standard algorithm for computing the supremal supervisor

Input: The plant G and the specification H

Output: The supremal supervisor with respect to G and H

Let $H_0 = H \parallel G = (Y_0, \Sigma, g_0, \Gamma_{H_0}, (x_0, y_0), Y_{0,m})$;

Set $i = 0$;

Set $Y'_i = \{(x, y) \in Y_i : \Gamma_G(x) \cap \Sigma_u \subseteq \Gamma_{H_i}(x, y)\}$;

Set $g'_i = g_i | Y'_i$ (The notation | stands for "restricted to");

Set $Y'_{i,m} = Y_{i,m} \cap Y'_i$;

$H_{i+1} = Trim(Y'_i, \Sigma, g'_i, (x_0, y_0), Y'_{i,m})$;

while $H_{i+1} \neq H_i$ *and* H_{i+1} *is not an empty automaton* **do**

    Set $H_{i+1} = (Y_{i+1}, \Sigma, g_{i+1}, (x_0, y_0), Y_{i+1,m})$;

    set $i = i + 1$;

    Set $Y'_i = \{(x, y) \in Y_i : \Gamma_G(x) \cap E_{uc} \subseteq \Gamma_{H'}(x, y)\}$;

    Set $g'_i = g_i | Y'_i$;

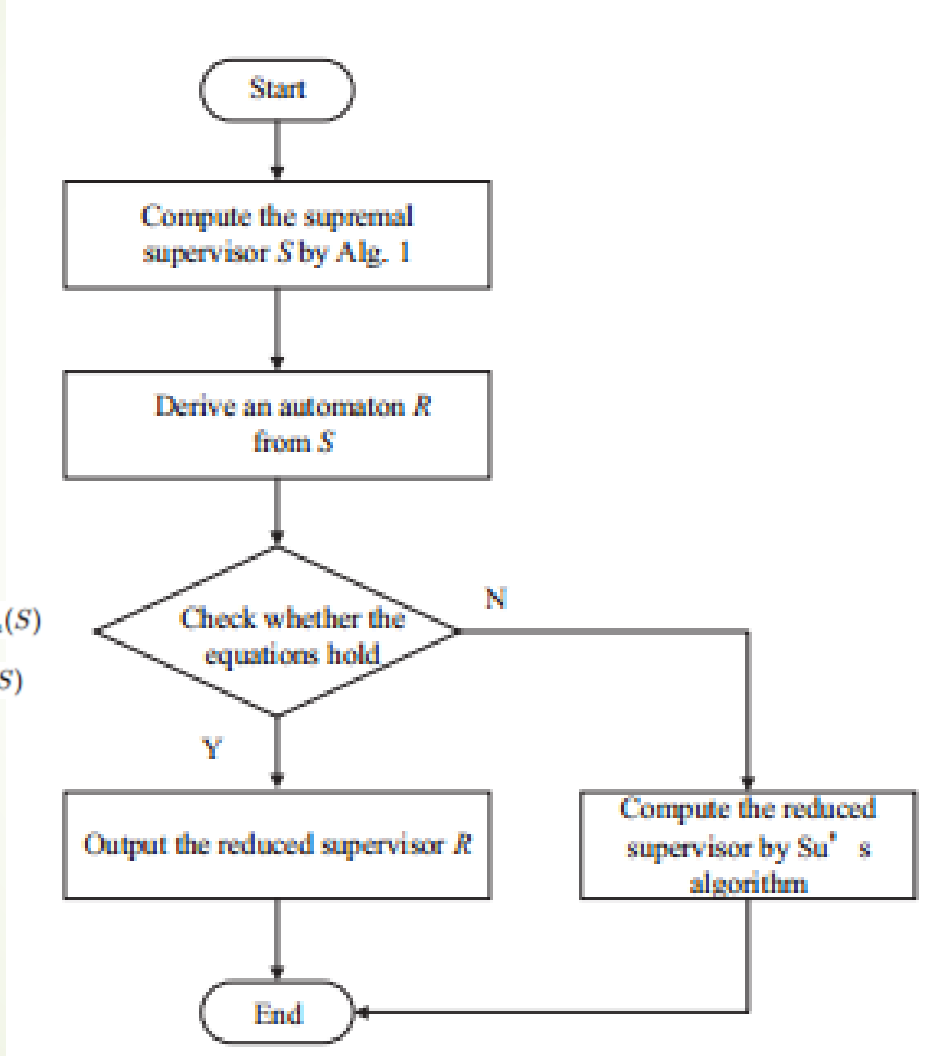
    Set $Y'_{i,m} = Y_{i,m} \cap Y'_i$;

    $H_{i+1} = Trim(Y'_i, \Sigma, g'_i, (x_0, y_0), Y'_{i,m})$;

end

Output H_{i+1};

4. Framework of the proposed approach



Framework for computing a reduced supervisor

4. Framework of the proposed approach

Derivation of *R* from the supremal supervisor *S*

Algorithm 2: Derivation of R from the supremal supervisor S

Input: The plant $G = (X, \Sigma, f, -, x_0, X_m)$ and the specification $H = (Y, \Sigma, g, -, y_0, Y)$

Output: An automaton R

1 Compute the supremal supervisor $\mathbf{S} = (Q, \Sigma, \delta, q_0, -, Q_m)$ with respect to G and H by Alg. 1 where $Q \subseteq X \times Y$, $Q_m \subseteq X_m \times Y_m$ and $q_0 = (x_0, y_0)$;
2 Let $Y' = \emptyset$ and $Y'_m = \emptyset$;
3 **for** *each state* $(x, y) \in Q$ **do**
4 Set $Y' = Y' \cup \{y | (x, y) \in Q\}$;
5 **if** $(x, y) \in Q_m$ **then**
6 Set $Y'_m = Y'_m \cup \{y | (x, y) \in Q_m\}$;
7 **end**
8 **for** *each event* $\sigma \in \Sigma$ **do**
9 **if** $\delta((x, y), \sigma) = (x', y')$ **then**
10 $\xi(y, \sigma) = y'$;
11 **end**
12 **end**
13 **end**
14 Output $R = (Y', \Sigma, \xi, y_0, -, Y'_m)$;

4. Framework of the proposed approach

Derivation of *R* from the supremal supervisor *S*

Proposition 1 *In Alg. 2, the output automaton* $R = (Y', \Sigma, \xi, y_0, -, Y_{m'})$ *is closed and deterministic.*

Proposition 2 $|R| \leq |H|$.

Proposition 3 *If T is isomorphic to S, R is a reduced supervisor with respect to the plant G and the specification H.*

4. Framework of the proposed approach

A sufficient condition for guaranteeing *T* isomorphic to *S*

Definition 2. *The supremal supervisor S is called inseparable with respect to the plant if there exist a string* $w \in \Sigma^*$, *states* $q \in Q$, $x \in X$ *and* $y \in Y'$ *such that the following conditions hold:*

(a) $\delta(q_0, w) = q$ *and* $\sigma \notin \Gamma_S(q)$ *in the supremal supervisor S;*

(b) $f(x_0, w) = x$ *and* $\sigma \notin \Gamma_G(x)$ *in the plant G;*

(c) $\xi(y_0, w) = y$ *and* $\sigma \notin \Gamma_R(y)$ *in the automaton R obtained from Alg. 2;*

Otherwise, S is called separable with respect to the plant.

4. Framework of the proposed approach

A sufficient condition for guaranteeing *T* isomorphic to *S*

Algorithm 3: Check whether the supervisor S is separable

Input: The supervisor S, the plant G and the automaton R obtained by Alg. 2
Output: The supervisor S is separable or not

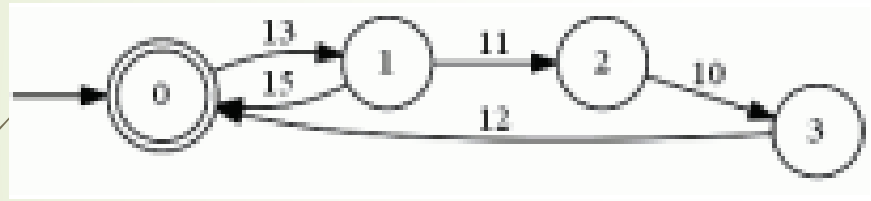
```
for each state q ∈ Q in the plant G do
    Choose a string s in Σ* such that δ(q0, s) = q in S;
    Denote f(x0, s) = x in G and ξ(y0, s) = y in R;
    for each σ ∈ ΓG(x) do
        if σ ∈ ΓR(y) and σ ∉ ΓS(q) then
            return False;
        end
    end
end
return True;
```

Proposition 4 *If S is separable with respect to the plant G, we have* $L(S) = L(G \parallel R)$ *and* $L_m(S) = L_m(G \parallel R)$.

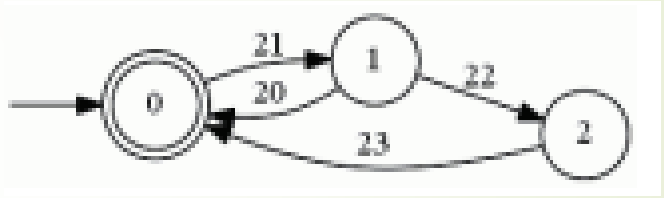
5. Illustrative examples

Example 1

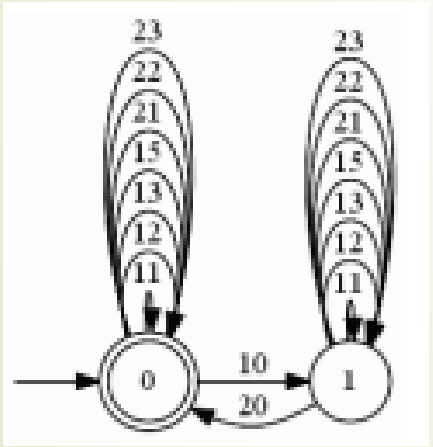
The plant G is the synchronization of M_1 with event set $\Sigma_1 = \{10, 11, 12, 13, 15\}$ and M_2 with event set $\Sigma_2 = \{20, 21, 22, 23\}$. The controllable event sets of M_1 and M_2 are $\{11, 13, 15\}$ and $\{21, 23\}$, respectively.



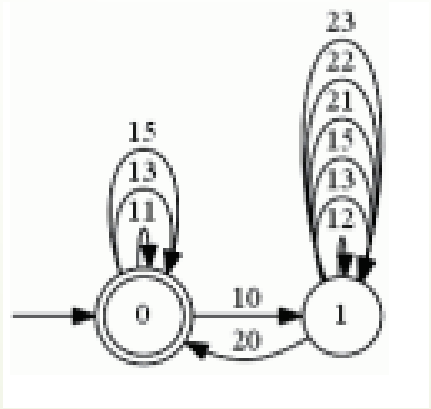
M_1



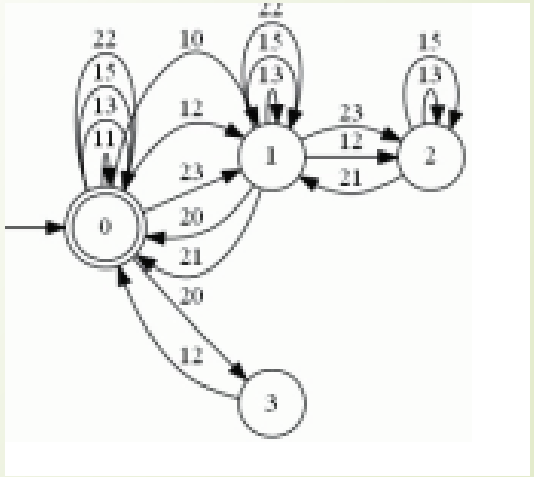
M_2



H



R

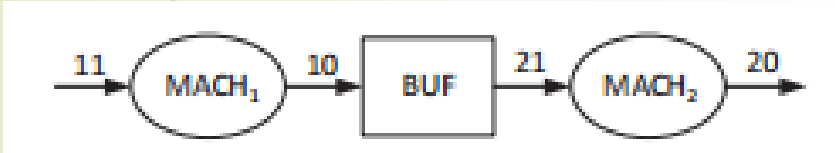


R_1

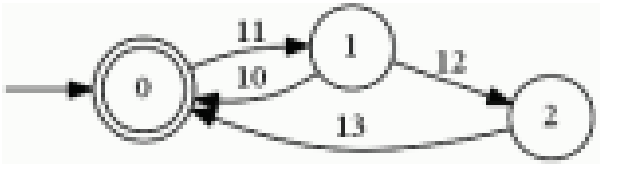
5. Illustrative examples

Example 2

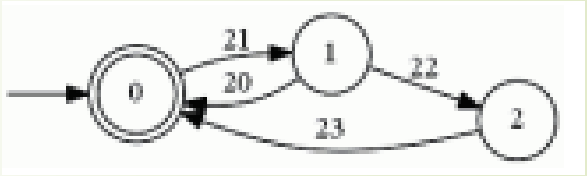
The **Small Factory** system consists of two machines **MACH**1, **MACH**2 and a buffer **BUF**.The controllable events set is *{*11, 13, 21, 23*}*



Small Factory



MACH1



MACH2

The specifications are described as follows.

– The buffer should neither underflow nor overflow, which means that **MACH**2 cannot take a part from **BUF** when the buffer is empty and **MACH**1 cannot put a part into **BUF** when the buffer is full.

– If both machines break down, **MACH**2 must be repaired ahead of **MACH**1.

5. Illustrative examples

Example 2

Size of the reduced supervisors computed by TCT and the proposed approach

buffer size	specification	supervisor	TCT	proposed approach	isomorphic
2	(6, 35)	(21, 47)	**(5, 18)**	**(6, 25)**	Y
4	(10, 61)	(39, 96)	**(9, 34)**	**(10, 47)**	Y
6	(14, 87)	(57, 139)	**(13, 50)**	**(14, 69)**	Y
8	(18 ,113)	(75, 185)	**(17, 66)**	**(18, 91)**	Y
10	(22, 139)	(93, 231)	**(21, 82)**	**(22, 113)**	Y
12	(26, 165)	(93, 231)	**(25, 98)**	**(26, 134)**	N

6. Conclusion and future work

- We propose an approach for computing a reduced supervisor by projecting state labels of the supremal supervisor on that of the specification. However, the proposed approach cannot work in all the DES. To alleviate this limitation, a sufficient condition for the algorithm is presented that makes the proposed algorithm work well. State size of the reduced supervisor is equal or less than that of the specification. Since the redued supervisor is a subset of the specification, it is explicit for the designer to understand the meaning of the control actions of the supervisor.
- In the future work, we will explore more general sufficient conditions for the proposed algorithm to compute reduced supervisors.

Thanks.