Using Automaton Abstraction in Synthesis of Distributed Supervisors

Dr Rong Su
S1-B1b-59, School of EEE
Nanyang Technological University
Tel: +65 6790-6042, Email: rsu@ntu.edu.sg
Outline

• Review of Automaton Abstraction
• Concepts of Supervisors and Relevant Properties
• Synthesis of Distributed Supervisors
• Example
• Conclusions
The Standardized Automata

Suppose $G = (X, \Sigma, \xi, x_0, X_m)$. Bring in a new event symbol $\tau$.

- $\tau$ will be treated as uncontrollable and unobservable.

An automaton $G = (X, \Sigma \cup \{\tau\}, \xi, x_0, X_m)$ is standardized if

- $x_0 \notin X_m$
- $(\forall x \in X) \xi(x, \tau) \neq \emptyset \iff x = x_0$
- $(\forall \sigma \in \Sigma) \xi(x_0, \sigma) = \emptyset$
- $(\forall x \in X)(\forall \sigma \in \Sigma \cup \{\tau\}) x_0 \notin \xi(x, \sigma)$

Let $\phi(\Sigma)$ be the collection of all standardized automata over $\Sigma$. 
Marking Awareness

- $G \in \Phi(\Sigma)$ is *marking aware* with respect to $\Sigma' \subseteq \Sigma$, if

$$
(\forall x \in X - X_m) (\forall s \in \Sigma^*) \xi(x,s) \cap X_m \neq \emptyset \Rightarrow P(s) \neq \varepsilon
$$

where $P: \Sigma^* \rightarrow \Sigma'^*$ is the natural projection.
Main Result

• **Theorem:** Given $\Sigma$ and $\Sigma' \subseteq \Sigma$, let $G \in \phi(\Sigma)$ and $S \in \phi(\Sigma')$. Then
  
  $B((G/\approx_{\Sigma'})\times S) = \emptyset \Rightarrow B(G\times S) = \emptyset$

  $G$ is marking aware w.r.t. $\Sigma'$ $\Rightarrow [B((G/\approx_{\Sigma'})\times S) = \emptyset \iff B(G\times S) = \emptyset]$
Outline

• Review of Automaton Abstraction
• Concepts of Supervisors and Relevant Properties
• Synthesis of Distributed Supervisors
• Example
• Conclusions
Basic Concepts

- Given a nondeterministic automaton $G = (X, \Sigma, \xi, x_0, X_m)$, let
  
  - $L(G) := \{s \in \Sigma^* | \xi(x_0, s) \neq \emptyset\}$ : the closed behavior
  
  - $N(G) := \{s \in \Sigma^* | \xi(x_0, s) \cap X_m \neq \emptyset\}$ : the nonblocking set
  
  - $B(G) := \{s \in \Sigma^* | (\exists x \in \xi(x_0, s))(\forall s' \in \Sigma^*) \xi(x, s') \cap X_m = \emptyset\}$ : the blocking set
  
  - $(\forall x \in X) E_G(x) := \{\sigma \in \Sigma | \xi(x, \sigma) \neq \emptyset\}$ : the enabling set
State Controllability

• Definition 1

Given \( G = (X, \Sigma, \xi, x_0, X_m) \) and \( \Sigma' \subseteq \Sigma \), let \( A = (Y, \Sigma', \eta, y_0, Y_m) \) and \( P: \Sigma^* \to \Sigma'^* \) be the natural projection. \( A \) is called \textit{state-controllable} with respect to \( G \), if

\[
(\forall s \in L(G \times A))(\forall x \in \xi(x_0, s))(\forall y \in \eta(y_0, P(s))) \ E_G(x) \cap \Sigma_{uc} \cap \Sigma' \subseteq E_A(y)
\]
State Observability

**Definition 2**

Given $G = (X, \Sigma, \xi, x_0, X_m)$ and $\Sigma' \subseteq \Sigma$, let $A = (Y, \Sigma', \eta, y_0, Y_m)$. We say $A$ is *state-observable* with respect to $(G, P_o)$ if for any $s, s' \in L(G \times A)$ with $P_o(s) = P_o(s')$,

$$(\forall (x,y) \in \xi \times \eta((x_0, y_0), s)) (\forall (x', y') \in \xi \times \eta((x_0, y_0), s')) E_{G \times A}(x, y) \cap E_G(x') \cap \Sigma' \subseteq E_A(y')$$

\[ P_o(s) = P_o(s') \]
State Normality

**Definition 3**

Given $G = (X, \Sigma, \xi, x_0, X_m)$ and $\Sigma' \subseteq \Sigma$, let $A = (Y, \Sigma', \eta, y_0, Y_m)$ and $P: \Sigma^* \rightarrow \Sigma'^*$ be the natural projection. We say $A$ is state-normal with respect to $(G, P_0)$ if for any $s \in L(G \times A)$ and $s' \in P_0^{-1}(P_0(s))$,

$$(\forall (x, y) \in \xi \times \eta((x_0, y_0), s'))(\forall s'' \in \Sigma^*) \quad P_0(s's'') = P_0(s) \land \xi(x, s'') \neq \emptyset \Rightarrow \eta(y, P(s'')) \neq \emptyset$$
Nonblocking Supervisor

• Definition 4

Given $G \in \phi(\Sigma)$ and $H \in \phi(\Delta)$ with $\Delta \subseteq \Sigma' \subseteq \Sigma$, an automaton $S \in \phi(\Sigma')$ is a nonblocking supervisor of $G$ under $H$, if $S$ is deterministic and the following conditions hold:

- $N(G \times S) \subseteq N(G \times H)$
- $B(G \times S) = \emptyset$
- $S$ is state-controllable with respect to $G$
- $S$ is state-observable with respect to $G$ and $P_o$
Supremal Nonblocking State-Normal Supervisor

• Let

\[ CN(G,H) : = \{ S \in \phi(\Sigma) \mid S \text{ is a NSN supervisor of } G \text{ w.r.t. } H \land L(S) \subseteq L(G) \} \]

where NSN denotes “Nonblocking State-Normal”

• We can show that \( CN(G,H) \) contains a unique element \( S^* \) such that

\[ (\forall S \in CN(G,H)) \ N(S) \subseteq N(S^*) \]

We call \( S^* \) the \textit{supremal} NSN supervisor of \( G \) under \( H \)

• \( S^* \) is computable with the complexity of \( O(||G|| \times ||H|| \times e^{||G|| \times ||H||}) \)
Main Results

• Let $G \in \phi(\Sigma)$ and a deterministic specification $H \in \phi(\Delta)$ with $\Delta \subseteq \Sigma' \subseteq \Sigma$.

Theorem 1

$S \in \phi(\Sigma')$ is a nonblocking supervisor of $G/\approx_{\Sigma'}$ with respect to $H$...

$\Rightarrow$

$S$ is a nonblocking supervisor of $G$ with respect to $H$
**Main Results (cont.)**

- Let $G \in \phi(\Sigma)$ and a deterministic specification $H \in \phi(\Delta)$ with $\Delta \subseteq \Sigma' \subseteq \Sigma$.
- Suppose $G$ is marking aware w.r.t. $\Sigma'$ and $\Sigma_o \subseteq \Sigma'$.

**Theorem 2**

$S \in \phi(\Sigma')$ is a nonblocking supervisor of $G/\approx_{\Sigma'}$ with respect to $H$ if and only if $S$ is a nonblocking supervisor of $G$ with respect to $H$. 
Outline

• Review of Automaton Abstraction
• Concepts of Supervisors and Relevant Properties
• Synthesis of Distributed Supervisors
• Example
• Conclusions
A distributed system with respect to given alphabets $\{\Sigma_i|i\in I\}$ is a collection of nondeterministic finite-state automata

$$G:=\{G_i=(X_i, \Sigma_i, \xi_i, x_{i,0}, X_{i,m})\in \Phi(\Sigma_i)|i\in I\}$$

where $\Sigma_i = \Sigma_{i,c} \cup \Sigma_{i,uc} = \Sigma_{i,o} \cup \Sigma_{i,uo}$. The compositional behavior of $G$ is specified by $\times_{i\in I}G_i$.

We assume that, $(\forall i,j\in I) i\neq j \Rightarrow \Sigma_{i,c} \cap \Sigma_{j,uc} = \emptyset \land \Sigma_{i,o} \cap \Sigma_{j,uo} = \emptyset$
Nonblocking Distributed Supervisor

Given a distributed system \( G = \{G_i \in \phi(\Sigma_i) | i \in I\} \) and deterministic specifications \( H = \{H_j \in \phi(\Delta_j) | j \in J\} | \Delta_j \subseteq \bigcup_{i \in I} \Sigma_i \land j \in J\} \), synthesize a set of deterministic automata \( S = \{S_k \in \phi(\Gamma_k) | \Gamma_k \subseteq \bigcup_{i \in I} \Sigma_i \land k \in K\} \) such that the following conditions hold,

- \( N((\times_{i \in I} G_i) \times (\times_{k \in K} S_k)) \subseteq N((\times_{i \in I} G_i) \times (\times_{j \in J} H_j)) \)
- \( B((\times_{i \in I} G_i) \times (\times_{k \in K} S_k)) = \emptyset \)
- \( \times_{k \in K} S_k \) is state-controllable w.r.t. \( \times_{i \in I} G_i \)
- \( \times_{k \in K} S_k \) is state-observable w.r.t. \( \times_{i \in I} G_i \) and \( P_o : (\bigcup_{i \in I} \Sigma_i)^* \rightarrow (\bigcup_{i \in I} \Sigma_{i,o})^* \)
An Aggregative Synthesis Approach (ASP)

• Inputs: standardized $G=\{G_i \in \phi(\Sigma_i) | i \in I\}$, $H=\{H_i \in \phi(\Delta_j) | j \in J\}$ $|\Delta_j \subseteq \bigcup_{i \in I} \Sigma_i \land j \in J\}$
• Initially set $W_1 := G_1$, $J_1 := \{j \in J | \Delta_j \subseteq \Sigma_1\}$, $Q_1 := J_1$ and $T_1 := \Sigma_1$
• For $k=1, \ldots, n$,
  • If $J_k \neq \emptyset$, let $V_k := \times_{j \in J_k} H_j$. Otherwise, set $V_k$ as a recognizer of $\Sigma_i^*$.
  • Synthesize the supremal NSN supervisor $S_k$ of $W_k$ under $V_k$.
  • Terminate when $S_k$ is empty or $k=n$. Otherwise, do the following.
  • Set $I_{k+1} := \{i \in I | k+1 \leq i \leq n\}$, $\Sigma_{k+1} := \bigcup_{i \in I_{k+1}} \Sigma_i$ and $\Theta_{k+1} := \bigcup_{j \in J-Q_k} \Delta_j$.
  • Choose $\Sigma_{Ak} \subseteq T_k$ with $(\Sigma_{k+1} \cup \Theta_{k+1}) \cap T_k \subseteq \Sigma_{Ak}$. Let $A_k := (W_k \times S_k)/\approx_{\Sigma_{Ak}}$.
  • $W_{k+1} := A_k \times G_{k+1}$, $Q_{k+1} := \{j \in J | \Delta_j \subseteq \bigcup_{i=1}^{k+1} \Sigma_i\}$.
  • $J_{k+1} := Q_{k+1} - Q_k$, $T_{k+1} := \Sigma_{Ak} \cup \Sigma_{k+1}$.
• When terminate upon $k$, output $S = \{S_1, S_2, \ldots, S_k\}$. 
Aggregative Synthesis

$G_1 \quad \ldots \quad G_{k-1}$

$\mathbf{S}_1 \times \ldots \times \mathbf{S}_{k-1}$

$A(k-1)$

$G_k$

$V(k)$

$S_k$

$I(k) = \{k, \ldots, n\}$

has been processed

to be processed

$G_{k+1} \quad \ldots \quad G_n$

EE6226 Discrete Event Systems
• **Theorem**

The ASP always terminates, and if every $S_k$ ($k=1,2,\ldots,n$) is nonempty, then $\{S_k \mid k=1,2,\ldots,n\}$ a nonblocking distributed supervisor of $\mathcal{G}$ under $\mathcal{H}$. 
Main Difficulty for Aggregative Synthesis

• How to order components so that it yields a solution?
Parallel Synthesis – Coordinated Distributed Control

\[ C : C/G \text{ is nonblocking} \]

\[ G = A_1 \times A_2 \]

\[ A_1 = (S_1/G_1)/\approx_{\Sigma_1\cap\Sigma'} \]

\[ A_2 = (S_2/G_2)/\approx_{\Sigma_2\cap\Sigma'} \]

abstraction

\[ S_1 \quad G_1 \]

\[ S_2 \quad G_2 \]
Multi-Level Coordinators

- $\Sigma'' \subseteq \Sigma_1 \cup \Sigma_2 \cup \Sigma_3 \cup \Sigma_4$
- $(\Sigma_1 \cup \Sigma_2) \cap (\Sigma_3 \cup \Sigma_4) \subseteq \Sigma''$

$G = A_{12} \times A_{34}$

$C$

$G_1 \times G_2 \times G_3 \times G_4$

$S_1 \wedge S_2 \wedge S_3 \wedge S_4 \wedge C_{12} \wedge C_{34} \wedge C$

- $\Sigma'' \subseteq \Sigma_1 \cup \Sigma_2 \cup \Sigma_3 \cup \Sigma_4$
- $(\Sigma_1 \cup \Sigma_2) \cap (\Sigma_3 \cup \Sigma_4) \subseteq \Sigma''$

$G_1 \times G_2$
$S_1 \wedge S_2 \wedge C_{12}$

$C_{12}$

$G_{12}$

$A_1$

$S_1 \wedge G_1$

$A_2$

$S_2 \wedge G_2$

$G_3 \times G_4$
$S_3 \wedge S_4 \wedge C_{34}$

$C_{34}$

$G_{34}$

$A_3$

$S_3 \wedge G_3$

$A_4$

$S_4 \wedge G_4$
Main Difficulty for Coordinated Control

• How to define those coordinator alphabets?
Outline

- Review of Automaton Abstraction
- Concepts of Supervisors and Relevant Properties
- Synthesis of Distributed Supervisors
- Example
- Conclusions
Cluster Tools
Component Models – Load and Exit Locks

\[ R_1\text{-pick-}L_{in} \]

Entering Load Lock \( L_{in} \)

\[ R_1\text{-drop-}L_{out} \]

Exit Load Lock \( L_{out} \)
Component Models – Chambers

\[ R_i \text{-pick-} C_{ij} \]

\[ R_i \text{-drop-} C_{ij} \]

\[ R_i \text{-pick-} C_{ij} \]

\[ R_i \text{-drop-} C_{ij} \]

\[ R_i \text{-pick-} C_{ij} \]

\[ R_i \text{-drop-} C_{ij} \]

\[ C_{ij} \]
Component Models – Buffers
Component Models – Robots

$R_1$: pick-$L_{\text{in}}$, pick-$C_{11}$, pick-$C_{12}$, pick-$B_1$

$R_1$: drop-$L_{\text{out}}$, drop-$C_{11}$, drop-$C_{12}$, drop-$B_1$

$R_2$: pick-$B_1$, pick-$C_{21}$, pick-$C_{22}$

$R_2$: drop-$B_1$, drop-$C_{21}$, drop-$C_{22}$

$R_3$: pick-$B_2$, pick-$C_{31}$, pick-$C_{32}$

$R_3$: drop-$B_2$, drop-$C_{31}$, drop-$C_{32}$

$R_4$: pick-$B_3$, pick-$C_{41}$, pick-$C_{42}$, pick-$C_{43}$

$R_4$: drop-$B_3$, drop-$C_{41}$, drop-$C_{42}$, drop-$C_{43}$
Specifications
Create Standardized Automata

- Let
  - \( G_1 := \mu(C_{41}) \times \mu(C_{42}) \times \mu(C_{43}) \times \mu(R_4) \times \mu(B_3) \)
  - \( G_2 := \mu(C_{31}) \times \mu(C_{32}) \times \mu(R_3) \times \mu(B_2) \)
  - \( G_3 := \mu(C_{21}) \times \mu(C_{22}) \times \mu(R_2) \times \mu(B_1) \)
  - \( G_4 := \mu(C_{11}) \times \mu(C_{12}) \times \mu(R_1) \times \mu(L_{in}) \times \mu(L_{out}) \)

and

- \( H_1 := \mu(H_{41}) \times \mu(H_{42}) \times \mu(H_{43}) \times \mu(H_{44}) \)
- \( H_2 := \mu(H_{31}) \times \mu(H_{32}) \times \mu(H_{33}) \times \mu(H_{34}) \)
- \( H_3 := \mu(H_{21}) \times \mu(H_{22}) \times \mu(H_{23}) \times \mu(H_{24}) \)
- \( H_4 := \mu(H_{11}) \times \mu(H_{12}) \times \mu(H_{13}) \times \mu(H_{14}) \)
Aggregative Synthesis

• Synthesize the supremal nonblocking state-normal supervisor $S_1$ of $G_1$ under $H_1$.
  – Use `make_supervisor('G1.cfg', 'H1.cfg', 'S1.cfg') :: S1 (112, 222)`

• Perform abstraction
  – Use `make_sequential_abstraction('G1.cfg, S1.cfg', 'R3-pick-B3, R3-drop-B3, R3-pick-B3, R4-drop-B3', 'A1.cfg') :: A1 (15, 24)`
Aggregative Synthesis (cont.)

• Form a new plant model
  – Use make_product(`G2.cfg, A1.cfg’, `W2.cfg’) :: W2 (985, 4053)

• Synthesize the supremal nonblocking state-normal supervisor $S_2$ of $W_2$ under $H_2$.
  – Use make_supervisor(`W2.cfg’, `H2.cfg’, `S2.cfg’) :: S1 (140, 288)

• Perform abstraction
Aggregative Synthesis (cont.)

- Form a new plant model
  - Use `make_product('G3.cfg, A2.cfg', 'W3.cfg') :: W3 (985, 4053)`

- Synthesize the supremal nonblocking state-normal supervisor $S_3$ of $W_3$ under $H_3$.
  - Use `make_supervisor('W3.cfg', 'H3.cfg', 'S3.cfg') :: S1 (140, 288)`

- Perform abstraction
  - Use `make_sequential_abstraction('W3.cfg, S3.cfg', 'R1-pick-B1, R1-drop-B1, R2-pick-B1, R2-drop-B1', 'A3.cfg') :: A3 (15, 24)`
Aggregative Synthesis (cont.)

- Form a new plant model
  - Use `make_product('G4.cfg, A3.cfg', 'W4.cfg')` :: W4 (253, 913)

- Synthesize the supremal nonblocking state-normal supervisor $S_4$ of $W_4$ under $H_4$.
  - Use `make_supervisor('W4.cfg', 'H4.cfg', 'S4.cfg')` :: S4 (68, 126)

- Perform nonconflict check
  - Use `make_nonconflicting_check('G1.cfg, G2.cfg, G3.cfg, G4.cfg, S1.cfg, S2.cfg, S3.cfg, S4.cfg')` :: ok
Homework

• Compute a coordinated distributed supervisor.
  – You can decide the number and the locations of your coordinators.
Conclusions

- **Advantages**
  - The abstraction technique is less restrictive than using observers
  - It can reduce space complexity as long as a system is loosely coupled
  - The synthesis approach has a limited degree of reusability when a system’s architecture is changed

- **Disadvantages**
  - The abstraction technique may bring in extra restriction on supervisors
  - The aggregative approach requires a “good” ordering of components
  - The coordinated control needs good choices of coordinator alphabets