Using Automaton Abstraction in Synthesis of Distributed Supervisors

Rong Su
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 \textit{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. $\Sigma'$ $\Rightarrow [B((G/\approx_{\Sigma'}) \times S) = \emptyset \Leftrightarrow B(G \times S) = \emptyset]$
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 $P: \Sigma^* \rightarrow \Sigma'^*$ be the natural projection. $A$ is called state-controllable with respect to $G$, if

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

\[
\sigma \in \Sigma_{uc} \cap \Sigma'
\]

$$G \quad \begin{array}{c} \text{s} \\ \text{x} \end{array} \quad P(s) \quad A$$

\[
\begin{array}{c} \text{s} \\ \text{x}_0 \end{array} \quad \begin{array}{c} \text{s} \\ \text{y} \end{array} \quad \begin{array}{c} \text{s} \\ \text{y}_0 \end{array}
\]
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 $P_o(s) = P_o(s')$,

$$(\forall (x,y) \in \xi \times \eta((x_0,y_0),s))(\forall (x',y') \in \xi \times \eta((x_0,y_0),s')) \; E_{G \times A}(x,y) \cap E_G(x') \cap \Sigma' \subseteq E_A(y')$$
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 $s' \in P_0^{-1}(P_0(s))$,

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

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

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 supremal NSN supervisor of \( G \) under \( H \)

\( S^* \) is computable with the complexity of \( O(||G|| \times ||H||^e||G|| \times ||H||) \)
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 \( \Sigma_0 \subseteq \Sigma' \).

Theorem 2

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

\[ \iff \]

\( S \) 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
Concept of Distributed System

- 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 $H = \{ H_i \in \phi(\Delta_j) | j \in J \} | \Delta_j \subseteq \bigcup_{i \in I} \Sigma_i \land j \in J \}$, 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 $(\Sigma_{Ik+1} \cup \Theta_{k+1}) \cap T_k \subseteq \Sigma_{Ak}$. 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

has been processed

I(k) = {k, …, n} to be processed

\[ G_1 \cdots G_{k-1} \]

\[ S_1 \times \cdots \times S_{k-1} \]

\[ V(k) \]

\[ A(k-1) \]

\[ S_k \]

\[ G_k \]

\[ G_{k+1} \cdots G_n \]
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 $G$ under $H$. 
Main Difficulty for Aggregative Synthesis

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

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

\[ G = A_1 \times A_2 \]

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

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

\[ G_1 \times G_2 \]

\[ S_1 \wedge S_2 \wedge C \]
Multi-Level Coordinators

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

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

$G_1 \times G_2 \times G_3 \times G_4$
$S_1 \wedge S_2 \wedge S_3 \wedge S_4 \wedge C_{12} \wedge C_{34} \wedge C$

$G_{12}$

$G_{1}$
$S_1$

$G_{2}$
$S_2$

$G_{3}$
$S_3$

$G_{4}$
$S_4$

$C_{12}$

$A_1$

$A_2$

$A_3$

$A_4$

$C_{34}$

$G_{34}$
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

Entering Load Lock $L_{in}$

Exit Load Lock $L_{out}$
Component Models – Chambers

Component Models – Chambers

Component Models – Chambers

Component Models – Chambers

Component Models – Chambers

Component Models – Chambers

Component Models – Chambers
Component Models – Buffers

\[ R_i \text{-pick-} B_i \quad R_{i+1} \text{-pick-} B_i \]

\[ R_i \text{-drop-} B_i \quad R_{i+1} \text{-drop-} B_i \]

\[ R_i \text{-pick-} B_i \quad R_{i+1} \text{-pick-} B_i \]

\[ B_i \]
Component Models – Robots

\[ R_1 \text{-drop-} L_{\text{out}} \]
\[ R_1 \text{-drop-} C_{11} \]
\[ R_1 \text{-drop-} C_{12} \]
\[ R_1 \text{-drop-} B_1 \]

\[ R_1 \text{-pick-} L_{\text{in}} \]
\[ R_1 \text{-pick-} C_{11} \]
\[ R_1 \text{-pick-} C_{12} \]
\[ R_1 \text{-pick-} B_1 \]

\[ R_2 \text{-drop-} B_1 \]
\[ R_2 \text{-drop-} C_{21} \]
\[ R_2 \text{-drop-} C_{22} \]

\[ R_2 \text{-pick-} B_1 \]
\[ R_2 \text{-pick-} C_{21} \]
\[ R_2 \text{-pick-} C_{22} \]

\[ R_3 \text{-drop-} B_2 \]
\[ R_3 \text{-drop-} C_{31} \]
\[ R_3 \text{-drop-} C_{32} \]
\[ R_3 \text{-drop-} B_3 \]

\[ R_3 \text{-pick-} B_2 \]
\[ R_3 \text{-pick-} C_{31} \]
\[ R_3 \text{-pick-} C_{32} \]
\[ R_3 \text{-pick-} B_3 \]

\[ R_4 \text{-drop-} B_3 \]
\[ R_4 \text{-drop-} C_{41} \]
\[ R_4 \text{-drop-} C_{42} \]
\[ R_4 \text{-drop-} C_{43} \]

\[ R_4 \text{-pick-} B_3 \]
\[ R_4 \text{-pick-} C_{41} \]
\[ R_4 \text{-pick-} C_{42} \]
\[ R_4 \text{-pick-} C_{43} \]
Specifications

\[
\begin{align*}
R_1\text{-drop-}C_{11} & \quad H_{11} \quad R_1\text{-pick-}L_{in} \\
R_1\text{-drop-}B_1 & \quad H_{12} \quad R_1\text{-pick-}C_{11} \\
R_1\text{-drop-}C_{12} & \quad H_{13} \quad R_1\text{-pick-}B_1 \\
R_1\text{-drop-}L_{out} & \quad H_{14} \quad R_1\text{-pick-}C_{12} \\
R_2\text{-drop-}C_{21} & \quad H_{21} \quad R_2\text{-pick-}B_1 \\
R_2\text{-drop-}B_2 & \quad H_{22} \quad R_2\text{-pick-}C_{21} \\
R_2\text{-drop-}B_2 & \quad H_{23} \quad R_2\text{-pick-}B_2 \\
R_2\text{-drop-}B_1 & \quad H_{24} \quad R_2\text{-pick-}C_{22} \\
R_3\text{-drop-}C_{31} & \quad H_{31} \quad R_3\text{-pick-}B_2 \\
R_3\text{-drop-}B_3 & \quad H_{32} \quad R_3\text{-pick-}C_{31} \\
R_3\text{-drop-}B_3 & \quad H_{33} \quad R_3\text{-pick-}B_3 \\
R_3\text{-drop-}C_{32} & \quad H_{34} \quad R_3\text{-pick-}C_{32} \\
R_4\text{-drop-}C_{41} & \quad H_{41} \quad R_4\text{-pick-}B_3 \\
R_4\text{-drop-}C_{42} & \quad H_{42} \quad R_4\text{-pick-}C_{41} \\
R_4\text{-drop-}C_{43} & \quad H_{43} \quad R_4\text{-pick-}C_{42} \\
R_4\text{-drop-}B_3 & \quad H_{44} \quad R_4\text{-pick-}C_{43}
\end{align*}
\]
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