Modular Supervisory Control

Rong Su
Outline

• Motivation
• Ramadge-Wonham Modular Supervisory Control
• Queiroz-Cury Extension
• Coordinated Modular Supervisory Control
• Example
• Interface-based Approach
• Conclusions
Divide & Conquer

Machine $G_1$

Specification $R_1$

Machine $G_2$

Specification $R_2$

Machine $G_3$
Construct Local Supervisors (by TCT)

- \( G = G_1 \times G_2 \times G_3 \) \( (G = \text{Sync}(\text{Sync}(G_1, G_2), G_3)) \) (8 ; 24)
- \( \text{SPEC}_1 = \text{Selfloop}(R_1, \{a_1, a_3, b_2, b_3\}) \) (2 ; 10)
- \( \text{SPEC}_2 = \text{Selfloop}(R_2, \{a_1, a_2, b_1, b_3\}) \) (2 ; 10)
- \( \text{SUPER}_1 = \text{Supcon}(G, R_1) \) (12 ; 28)
- \( \text{SUPER}_2 = \text{Supcon}(G, R_2) \) (12 ; 28)
- \( \text{Nonconflict}(\text{SUPER}_1, \text{SUPER}_2) = \text{true} \)
- \( R = R_1 \times R_2 \) \( (R = \text{Sync}(R_1, R_2)) \) (4 ; 16)
- \( \text{SUPER} = \text{Supcon}(G, R) \) (18 ; 32)
- \( \text{Isomorph}(\text{SUPER}, \text{Sync}(\text{SUPER}_1, \text{SUPER}_2)) = \text{true} \)
What to Gain?

- $\text{Minsuper} = \text{Supreduce}(G, \text{SUPER}, \text{SUPER}) \ (4; 13)$

- $\text{Minsuper}_1 = \text{Supreduce}(G, \text{SUPER}_1, \text{SUPER}_1) \ (2; 2)$

- $\text{Minsuper}_2 = \text{Supreduce}(G, \text{SUPER}_2, \text{SUPER}_2) \ (2; 2)$

- $|A| := \text{the total number of states and transitions of } A$

  $$|\text{SUPER}_1| + |\text{SUPER}_2| < |\text{SUPER}|$$
Motivation of Modular Control

Reduce complexity by allocating control tasks to local supervisors!
Outline

• Motivation
• Ramadge-Wonham Modular Supervisory Control
• Queiroz-Cury Extension
• Coordinated Modular Supervisory Control
• Example
• Interface-based Approach
• Conclusions
Architecture of Modular Supervisory Control

local specification

E₁ → S₁ → G → S₂ → E₂
Composition of Local Supervisors (1)

• Recall that $S$ is a proper supervisor of $G$ if
  \- $L_m(S) \cap L_m(G)$ is controllable with respect to $G$
  \- $L_m(S) \cap L_m(G) = L(S) \cap L(G)$
  \- $S$ is nonblocking, i.e. $L_m(S) = L(S)$

• Let $S/G$ denote the supervision of $S$ over $G$
  \- $L_m(S/G) := L_m(S) \cap L_m(G)$
  \- $L(S/G) := L(S) \cap L(G)$

• Given $S_1$ and $S_2$, let $S_1 \land S_2 := \text{reachable}(S_1 \times S_2)$
Composition of Local Supervisors (2)

- **Theorem 1 (Ramadge-Wonham)**
  - Given two proper supervisors $S_1$ and $S_2$ of $G$, we have
    \[
    L_m((S_1 \land S_2)/G) = L_m(S_1/G) \cap L_m(S_2/G) \\
    L((S_1 \land S_2)/G) = L(S_1/G) \cap L(S_2/G)
    \]
  - Furthermore, $S_1 \land S_2$ is a proper supervisor of $G$ if and only if
    - $S_1 \land S_2$ is nonblocking
    - $L_m(S_1/G)$ and $L_m(S_2/G)$ are nonconflicting, i.e.
      \[
      L_m(S_1/G) \cap L_m(S_2/G) = L_m(S_1/G) \cup L_m(S_2/G) = L(S_1/G) \cap L(S_2/G)
      \]
Composition of Local Supervisors (3)

- Let $C(G, E) := \{K \subseteq L_m(G) \cap L_m(E) | \overline{K} \Sigma_{uc} \cap L(G) \subseteq \overline{K}\}$.
- Let $\text{sup} C(G, E)$ be the greatest element of $C(G, E)$.
- Theorem 2 (Wonham-Ramadge)
  - Given a plant $G$ and two specifications $E_1, E_2$, if $\text{sup} C(G, E_1)$ and $\text{sup} C(G, E_2)$ are nonconflicting, then
    \[
    \text{sup} C(G, E_1 \times E_2) = \text{sup} C(G, E_1) \cap \text{sup} C(G, E_2)
    \]
The General Procedure for RW Modular Design

- Given $G$ and $E_1$, $E_2$
- $S_1 = \text{Supcon}(G, E_1)$
- $S_2 = \text{Supcon}(G, E_2)$
- $\text{Nonconflict}(S_1, S_2) = \text{true}$?
  - If yes, then $\{S_1$ and $S_2\}$ is a modular supervisor of $G$ w.r.t. $E_1$, $E_2$
  - Otherwise, the problem is unsolvable by RW modular control theory
    • But we can compute a coordinator to solve the conflicting part of $(S_1 \land S_2)/G$
Inadequacy of RW Modular Control Theory (MCT)

• More on implementation simplicity than synthesis simplicity
  – It is computationally expensive to verify the condition $\sup \mathcal{C}(G, E_1)$ and $\sup \mathcal{C}(G, E_2)$ are nonconflicting
  – If the condition doesn’t hold, RWMCT doesn’t tell what to do next?
Example – Resource Competition

User 1: $G_1$

User 2: $G_2$

Resource A: $R_A$

Resource B: $R_B$
Specification

- Deadlock should not happen.
A “Naive” Modular Supervisor

Local Supervisor: $S_A$  Local Supervisor: $S_B$
Facts

- $S_A$ is a proper supervisor of $G_1 \times G_2 \times R_A$

- $S_B$ is a proper supervisor of $G_1 \times G_2 \times R_B$

- Nevertheless, $L_m(S_A)$ and $L_m(S_B)$ are conflicting.

- We can check that $G_1 \times G_2 \times R_A \times R_B \times S_A \times S_B$ has deadlock!
Outline

- Motivation
- Ramadge-Wonham Modular Supervisory Control
- Queiroz-Cury Extension
- Coordinated Modular Supervisory Control
- Example
- Interface-based Approach
- Conclusions
An Extended Architecture
Main Result

• (Product) Plant: \( \{G_i \in \phi(\Sigma_i) \mid i \in I \land (\forall j \in I) \ j \neq i \Rightarrow \Sigma_i \cap \Sigma_j = \emptyset \} \)

• Specifications: \( \{E_i \in \phi(\Sigma_i) \mid i \in I \} \)

• Let \( G = \times_{i \in I} G_i \) and \( E = \times_{i \in I} E_i \)

• Let \( S_i \) be a proper supervisor of \( G_i \) with respect to \( E_i \)

• Theorem 3 (Queiroz-Cury)
  – \( \land_{i \in I} S_i \) is a proper supervisor of \( G \) with respect to \( E \) if \( \land_{i \in I} S_i \) is nonblocking and \( \{L_m(S_i/G_i) \mid i \in I \} \) is (synchronously) nonconflicting.

  – Furthermore, if \( \{\text{sup}_C(G_i,E_i) \mid i \in I \} \) is (synchronously) nonconflicting then
    \[ \text{sup}_C(G,E) = \|_{i \in I} \text{sup}_C(G_i,E_i) \]
The inadeqracy of RW modular control theory still exists!

But we can do something about it …
One Solution to The Inadequacy

\[ P: \Sigma^* \rightarrow \Sigma'^* \]

\[ \Sigma = \Sigma_1 \cup \Sigma_2 \]
\[ \Sigma' \supseteq \Sigma_1 \cap \Sigma_2 \]
Outline

- Motivation
- Ramadge-Wonham Modular Supervisory Control
- Queiroz-Cury Extension
- Coordinated Modular Supervisory Control
- Example
- Interface-based Approach
- Conclusions
Example – Resource Competition Revisit

Graphs $G_1$ and $G_2$ with resource competition $C$.

Selfloops: $\{b_1, b_2\}$ for $S_A$; $\{a_1, a_2\}$ for $S_B$.
$S_A \land S_B \land C$ is a proper supervisor of $G_1 \times G_2 \times R_A \times R_B$
The Concept of L-observer

- Given $L \subseteq \Sigma^*$ and $\Sigma' \subseteq \Sigma$, let $P: \Sigma^* \rightarrow \Sigma'^*$ be the natural projection
- $P$ is called an L-observer if

$$(\forall t \in P(L))(\forall s \in L) \ P(s) \leq t \Rightarrow (\exists u \in \Sigma^*) \ su \in L \land P(su) = t$$

\[ \begin{array}{ccc}
\text{t'} & \rightarrow & \text{t''} \\
\uparrow & & \uparrow \\
P(s) = t' & \rightarrow & P(u) = t'' \\
\text{s} & \rightarrow & \text{u} \\
\rightarrow & \rightarrow & \rightarrow \\
& \rightarrow & su \in L \\
\end{array} \]
The Main Property of L-observer (MPLO)

- $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma' \subseteq \Sigma_1 \cup \Sigma_2$
- If
  - $P_1: \Sigma_1^* \rightarrow (\Sigma_1 \cap \Sigma')^*$ is $L_1$-observer
  - $P_2: \Sigma_2^* \rightarrow (\Sigma_2 \cap \Sigma')^*$ is $L_2$-observer
- then
  - $P: (\Sigma_1 \cup \Sigma_2)^* \rightarrow \Sigma'^*$ is $L_1||L_2$-observer
Application of MPLO

- Given $L \subseteq \Sigma^*$ and $\Sigma' \subseteq \Sigma$, let $P: \Sigma^* \rightarrow \Sigma'^*$ be the $L$-observer.
- Let $\Sigma'' \subseteq \Sigma'$ and $L'' \subseteq \Sigma''^*$, then

$$P(L) \text{ and } L'' \text{ is nonconflicting } \iff L \text{ and } L'' \text{ is nonconflicting}$$

$$L \text{ nonconflicting} \quad L'' \text{ nonconflicting}$$

$$P \quad L \quad \downarrow \quad P(L) \quad L''$$

$$\overline{L \parallel L''} = \overline{L} \parallel \overline{L''}$$

$$\overline{P(L) \parallel L''} = \overline{P(L)} \parallel \overline{L''}$$
Coordinated Modular Supervisory Control

- Given $\Sigma$, let $\phi(\Sigma)$ denote the set of all FSAs over $\Sigma$.
- Given two alphabets $\Sigma_1$ and $\Sigma_2$, let $G_1 \in \phi(\Sigma_1)$ and $G_2 \in \phi(\Sigma_2)$.
- Let $S_i$ be a proper supervisor of $G_i$ ($i=1,2$).
- Let $\Sigma' \subseteq \Sigma_1 \cup \Sigma_2$ with $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma'$.
- Suppose $P_i : \Sigma_i^* \rightarrow (\Sigma_i \cap \Sigma')^*$ be an $L_m(S_i/G_i)$-observer, where $i=1,2$.
- Let $P_1(S_1/G_1)$ denote an automaton, where
  - $L(P_1(S_1/G_1)) = P_1(L(S_1/G_1))$ and $L_m(P_1(S_1/G_1)) = P_1(L_m(S_1/G_1))$
- Let $G := P_1(S_1/G_1) \times P_2(S_2/G_2)$
- Compute a coordinator $C \in \phi(\Sigma')$ such that $C/G$ is nonblocking

Theorem 4
- Given the above setup, $S_1 \wedge S_2 \wedge C$ is a proper supervisor of $G_1 \times G_2$. 
Illustration of Coordinator Synthesis

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

\[ G = P_1(S_1/G_1) \times P_2(S_2/G_2) \]

- \( P_1(S_1/G_1) \) - observer
- \( P_2(S_2/G_2) \) - observer

- \( S_1 \)
- \( G_1 \)
- \( S_2 \)
- \( G_2 \)
Multi-Level Coordinators

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

$G = P_{12}((S_1 \land S_2 \land C_{12})/(G_1 \times G_2)) \times P_{34}((S_3 \land S_4 \land C_{34})/(G_3 \times G_4))$

$P_{12}: (\Sigma_1 \cup \Sigma_2)^* \rightarrow ((\Sigma_1 \cup \Sigma_2) \cap \Sigma'')^*$

$P_{34}: (\Sigma_3 \cup \Sigma_4)^* \rightarrow ((\Sigma_3 \cup \Sigma_4) \cap \Sigma'')^*$
Outline

• Motivation
• Ramadge-Wonham Modular Supervisory Control
• Queiroz-Cury Extension
• Coordinated Modular Supervisory Control
• Example
• Interface-based Approach
• Conclusions
Simple Transfer Line (STL)
Component Models

M1

M2

M3

M4

TU
Buffer Specifications

B₁

B₂

B₃

B₄
Partition of STL

PLANT_1

PLANT_2
Local Synthesis with TCT

- \text{PLANT}_1 = M_1 \times M_2 \times M_3 \times TU \quad (\text{use Sync}) \quad (16, 72)
- \text{PLANT}_2 = M_3 \times M_4 \times TU \quad (8, 28)
- \text{SPEC}_1 = \text{Selfloop}(\text{Sync}(B_1, B_2), \{1, 6, 9, 10\}) \quad (4, 42)
- \text{SPEC}_2 = \text{Selfloop}(\text{Sync}(B_3, B_4), \{5, 10, 12\}) \quad (9, 51)
- \text{SUPER}_1 = \text{Supcon}(\text{PLANT}_1, \text{SPEC}_1) \quad (48, 146)
- \text{SUPER}_2 = \text{Supcon}(\text{PLANT}_2, \text{SPEC}_2) \quad (50, 137)
- \text{SUPER}_1 \text{ and } \text{SUPER}_2 \text{ are conflicting}
Create an Coordinator

\[ \Sigma_1 = \{1,2,3,4,5,6,9,10,12\} \]

\[ \Sigma_2 = \{5,6,7,8,9,10,12\} \]

\[ \Sigma_c \supseteq \Sigma_1 \cap \Sigma_2 = \{5,6,9,10,12\} \]
Coordinator Synthesis

• Preparation
  – Set the coordinator’s alphabet as $\Sigma_c = \{1, 5, 6, 9, 10, 12\}$
  – We can check that both $P_1$ and $P_2$ are observers.

• Create local abstractions
  – $P_{PLANT_1}$ = Project($SUPER_1, \{1, 5, 6, 9, 10, 12\}$) (14, 40)
  – $P_{PLANT_2}$ = Project($SUPER_2, \{5, 6, 9, 10, 12\}$) (18, 41)

• Create a specification SPEC, recognizing $\Sigma_c^*$.

• Synthesis
  – $P_{PLANT}$ = Sync($P_{PLANT_1}, P_{PLANT_2}$) (63, 168)
  – $C = \text{Supcon}(P_{PLANT}, \text{SPEC})$ (59, 158)
Verification

- C, SUPER₁ and SUPER₂ are nonconflicting
Monolithic Supervisor Synthesis

- PLANT = Sync(PLANT₁, PLANT₂)  \hspace{1cm} (32 , 176)
- SPEC = Selfloop(Sync(Sync(Sync(B₁,B₂),B₃),B₄), \{1,10\}) \hspace{1cm} (54 , 414)
- SUPER = Supcon(PLANT, SPEC) \hspace{1cm} (568 , 1927)

Isomorphic(Sync(C,Sync(SUPER₁, SUPER₂)),SUPER)=true
Comparison

• Monolithic Approach
  – Plant : (32, 176)
  – Supervisor : (568, 1927)
  – The largest intermediate computational result : (568, 1927)

• Coordinated Modular Approach
  – Local Plants : (16, 72), (8, 28)
  – Local Supervisors : (48, 146), (50, 137)
  – Coordinator : (59, 158)
  – The largest intermediate computational result : (63, 168)
Outline

• Motivation
• Ramadge-Wonham Modular Supervisory Control
• Queiroz-Cury Extension
• Coordinated Modular Supervisory Control
• Example
• Interface-based Approach
• Conclusions
Motivations

- Each component has a fixed interface
- Each component’s internal behaviour is unseen to outsiders
- Components communicate with each other through interfaces
Example 1 – Digital Circuit
Example 2 – Component-Based Software

Data Retrieval (DR)

Video Signal Processor (VSP)

Audio Signal Processor (ASP)

Video-Audio Synchronizer (VAS)
Our Goal

• Use interfaces to separate components, allowing local synthesis
The System Architecture

High-Level Component $G_H \in \phi(\Sigma_H)$ (where $\phi(\Sigma_H)$ contains all FSAs over $\Sigma_H$)

Interfaces

Low-Level Components

• For any $i,j \in \{H,L1,...,Ln\}$
  - $\Sigma_i = \Sigma_{i,c} \cup \Sigma_{i,uc}$
  - $i \neq j \Rightarrow \Sigma_{i,c} \cap \Sigma_{j,uc} = \emptyset$

• For any $i,j \in \{L1,...,Ln\}$
  - $\Sigma_H \cap \Sigma_L = \Sigma_{Li}$
  - $i \neq j \Rightarrow \Sigma_{Li} \cap \Sigma_{Lj} = \Sigma_{Li} \cap \Sigma_{Lj}$
Separable Requirements

- At the high level: $E_H \in \phi(\Sigma_H)$

- At the low level: $\{E_{Li} \in \phi(\Sigma_{Li}) \mid i=1,\ldots,n\}$

A requirement can “touch” different components via interface events!
A Supervisor Synthesis Problem

- Compute an interface-based modular (IBM) supervisor \( \{S_H, S_{L1}, \ldots, S_{Ln}\} \),
  - Requirements: \( L_m(S_H/G_H) \subseteq L_m(E_H) \land (\forall i \in \{1, \ldots, n\}) \ L_m(S_{Li}/G_{Li}) \subseteq L_m(E_{li}) \)
  - Nonblockingness: \( \overline{L_m(S/G)} = L(S/G) \) where
    - \( L_m(G) = L_m(G_H)\|L_m(G_{L1})\|\ldots\|L_m(G_{Ln}) \) and \( L(G) = L(G_H)\|L(G_{L1})\|\ldots\|L(G_{Ln}) \)
    - \( L_m(S) = L_m(S_H)\|L_m(S_{L1})\|\ldots\|L_m(S_{Ln}) \) and \( L(S) = L(S_H)\|L(S_{L1})\|\ldots\|L(S_{Ln}) \)
  - Controllability: \( L(S/G)\Sigma_{uc} \cap L(G) \subseteq L(S/G) \)
  - Interface Invariance:
    \[
    (\forall i \in \{1, \ldots, n\}) \ P_i(L_m(S_{Li}/G_{Li})) = L_m(G_{li})
    \]
    where \( P_i : \Sigma_{Li}^* \rightarrow \Sigma_{li}^* \) is an \( L_m(S_{Li}/G_{Li}) \)-observer
Theorem 1:

Given $\mathcal{G} = \{G_H, G_{Li}, G_{li} \mid i=1,\ldots,n\}$ and $\mathcal{E} = \{E_H, E_{Li} \mid i=1,\ldots,n\}$, the largest IBM supervisor, denoted as the supremal IBM supervisor, in terms of component-wise set inclusion exists.
Local Supervisor Synthesis (1)

- At the high level
  - Plant: $G = G_H \times G_{I_1} \times \ldots \times G_{I_n}$
  - Requirement: $E_H$
  - Synthesize $S_H \in \phi(\Sigma_H)$, where
    - $L_m(S_H/G) \subseteq L_m(E_H)$
    - $L_m(S_H/G) = L(S_H/G)$
    - $L(S_H/G)\Sigma_{H,uc} \cap L(G) \subseteq L(S_H/G)$
Local Supervisor Synthesis (2)

- At the low level, for each local component \( G_{Li} \) \((i \in \{1, \ldots, n\})\)
  
  - **Plant**: \( G_{Li} \)
  
  - **Requirement**: \( E_{Li} \)
  
  - **Synthesize** \( S_{Li} \in \phi(\Sigma_{Li}) \), where

1. \( L_m(S_{Li}/G_{Li}) \subseteq L_m(E_{Li}) \)
2. \( L_m(S_{Li}/G_{Li}) = L(S_{Li}/G_{Li}) \)
3. \( L(S_{Li}/G_{Li}) \Sigma_{Li,uc} \cap L(G_{Li}) \subseteq L(S_{Li}/G_{Li}) \)
4. \( P_i(L_m(S_{Li}/G_{Li})) = L_m(G_{Li}) \), where \( P_i : \Sigma_{Li}^* \rightarrow \Sigma_{Li}^* \) is an \( L_m(S_{Li}/G_{Li}) \)-observer
Theorem 2:

The largest language $L_m(S_{Li}/G_{Li})$ satisfying conditions 1-4 exists.

(Why?)
• The largest language $L_m(S_{Li}/G_{Li})$ in Theorem 2 is computable.
Theorem 3:

\{S_H, S_{L1}, \ldots, S_{Ln}\} is the supremal IBM supervisor w.r.t. \( \mathcal{G} \) and \( \mathcal{E} \).
Conclusions

• Advantages of Modular Supervisory Control
  – It is easy to present a system in a modular way
  – It is computationally tractable compared to the monolithic approach
  – It possesses a certain level of implementation flexibility

• Disadvantage of Modular Supervisory Control
  – Modular control is more conservative than centralized control (why?)
  – The observer property is required during model abstraction