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 τ.
  - τ will be treated as uncontrollable and unobservable.
- An automaton $G = (X,\Sigma\cup\{\tau\},\xi,x_0,X_m)$ is *standardized* if
  - $x_0 \notin X_m$
  - $(\forall x\in X)\ \xi(x,\tau) \neq \varnothing \Leftrightarrow x = x_0$
  - $(\forall \sigma\in\Sigma)\ \xi(x_0,\sigma) = \varnothing$
  - $(\forall x\in X)(\forall \sigma\in\Sigma\cup\{\tau\})\ x_0\notin\xi(x,\sigma)$
- Let $\phi(\Sigma)$ be the collection of all standardized automata over Σ.

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 $P: \Sigma^* \to \Sigma'^*$ is the natural projection.

Main Result

- **Theorem:** Given Σ and $\Sigma' \subseteq \Sigma$, let $G \in \phi(\Sigma)$ and $S \in \phi(\Sigma')$. Then
  - $B((G/\approx_{\Sigma'}) \times S) = \varnothing \Rightarrow B(G \times S) = \varnothing$
  - G is marking aware w.r.t. $\Sigma' \Rightarrow [B((G/\approx_{\Sigma'}) \times S) = \varnothing \Leftrightarrow B(G \times S) = \varnothing]$

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
  - $L(G) := \{s \in \Sigma^* | \xi(x_0,s) \neq \varnothing\}$: the closed behavior
  - $N(G) := \{s \in \Sigma^* | \xi(x_0,s) \cap X_m \neq \varnothing\}$: the nonblocking set
  - $B(G) := \{s \in \Sigma^* | (\exists x \in \xi(x_0,s))(\forall s' \in \Sigma^*)\ \xi(x,s') \cap X_m = \varnothing\}$: the blocking set
  - $(\forall x \in X)\ E_G(x) := \{\sigma \in \Sigma \mid \xi(x,\sigma) \neq \varnothing\}$: 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

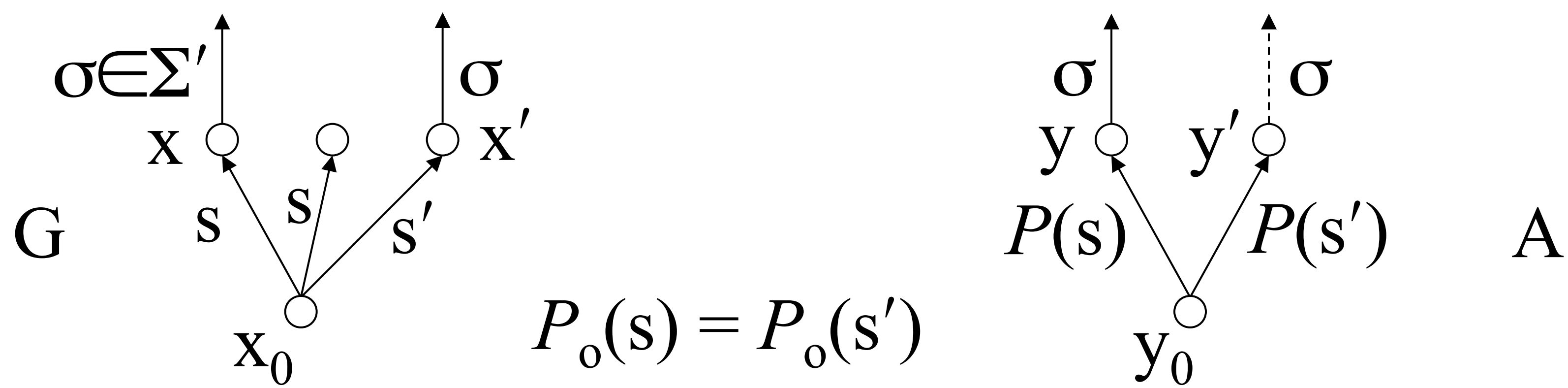
$$(\forall s \in L(G \times A))(\forall x \in \xi(x_0,s))(\forall y \in \eta(y_0,P(s)))\ E_G(x) \cap \Sigma_{uc} \cap \Sigma' \subseteq E_A(y)$$

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 $P_o(s)=P_o(s')$,

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

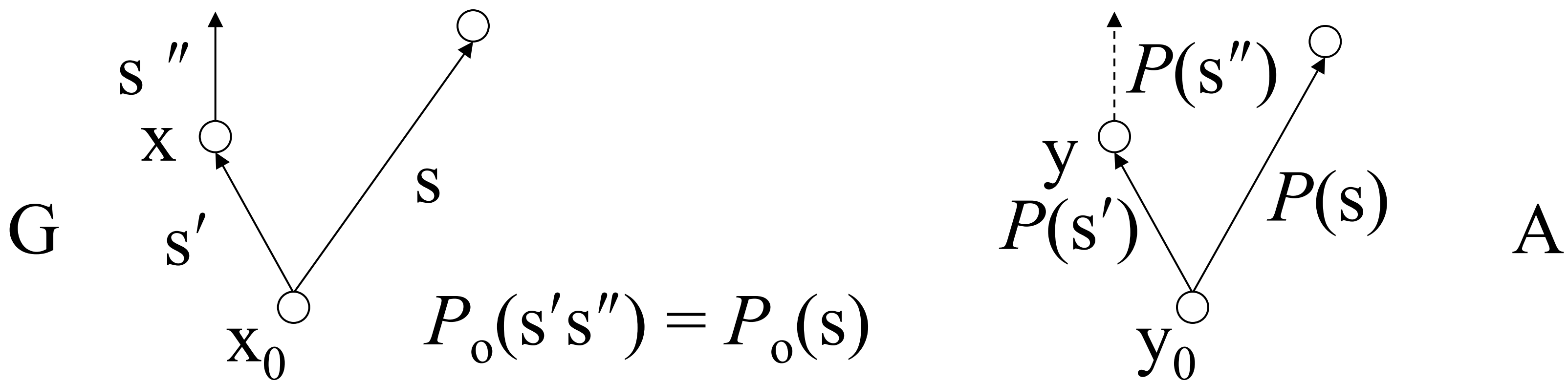


State Normality

- **Definition 3**

Given G = (X,Σ,ξ,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. We say A is *state-normal* with respect to (G, P_0) if for any $s\in L(G\times A)$ and $s'\in P_o^{-1}(P_o(s))$,

$$(\forall(x,y)\in\xi\times\eta((x_0,y_0),s'))(\forall s''\in\Sigma^*)\ P_o(s's'')=P_o(s)\wedge\xi(x,s'')\neq\varnothing\Rightarrow\eta(y,P(s''))\neq\varnothing$$



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:

  - $N(G \times S) \subseteq N(G \times H)$
  - $B(G \times S) = \varnothing$
  - 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

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

where NSN denotes "Nonblocking State-Normal"

- We can show that $\mathcal{CN}(G,H)$ contains a unique element S^* such that

$$(\forall S \in \mathcal{CN}(G,H))\ N(S) \subseteq N(S^*)$$

We call S^* the *supremal* NSN supervisor of G under H

- S^* is computable with the complexity of $\mathcal{O}(\|G\| \times \|H\| e^{\|G\| \times \|H\|})$

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

$$\Longrightarrow$$

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. Σ' and $\Sigma_o \subseteq \Sigma'$.

Theorem 2

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

$$\Leftrightarrow$$

S is a nonblocking supervisor of G with respect to H

Outline

- Review of Automaton Abstraction
- Concepts of Supervisors and Relevent 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

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

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

- We assume that, $(\forall i,j \in I)\ i \neq j \Rightarrow \Sigma_{i,c} \cap \Sigma_{j,uc} = \varnothing \wedge \Sigma_{i,o} \cap \Sigma_{j,uo} = \varnothing$

Nonblocking Distributed Supervisor

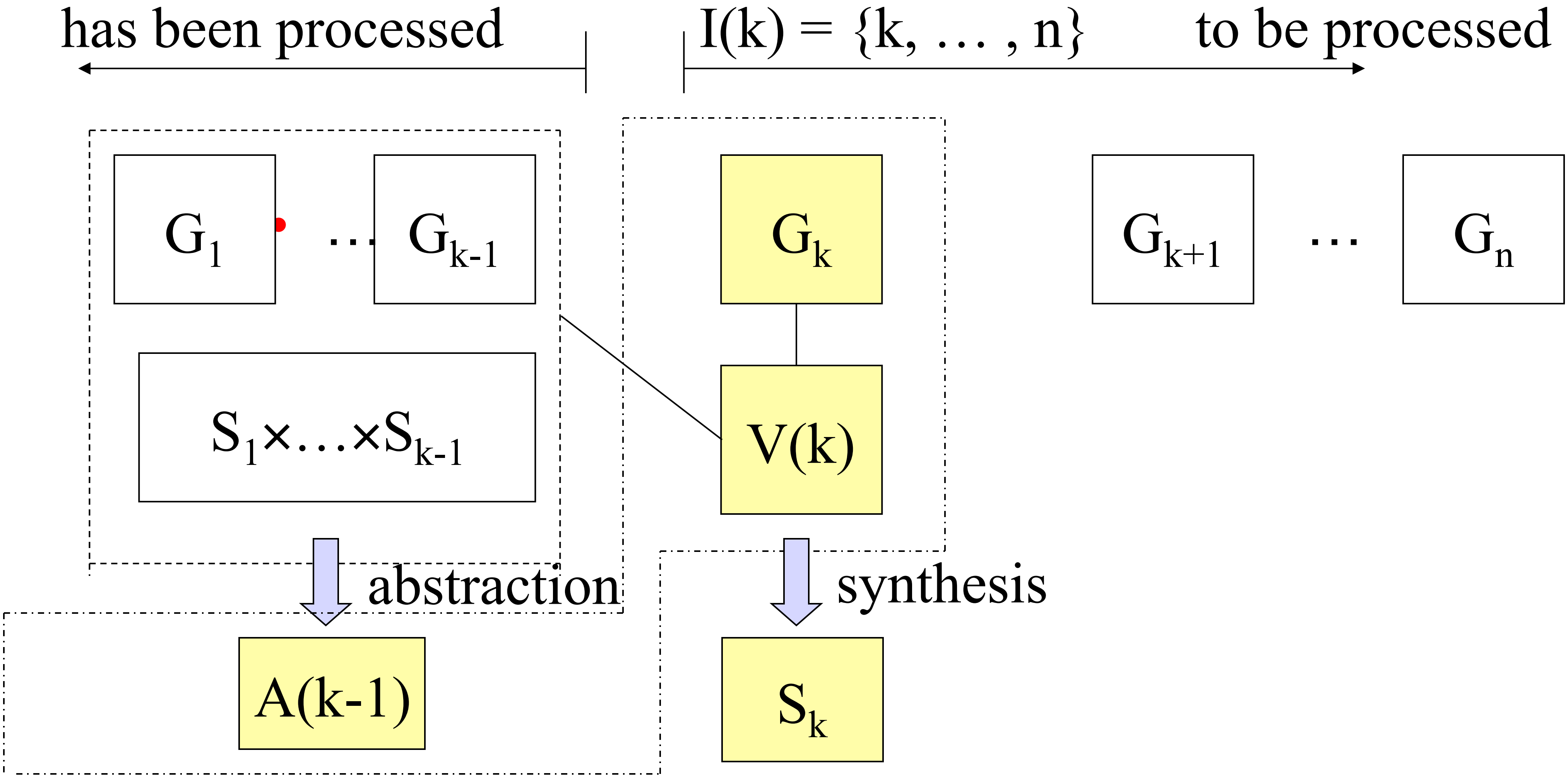
Given a distributed system $\mathcal{G} = \{G_i \in \phi(\Sigma_i) | i \in I\}$ and deterministic specifications $\mathcal{H} = \{H_i \in \phi(\Delta_j) | j \in J\} | \Delta_j \subseteq \cup_{i\in I} \Sigma_i \wedge j \in J\}$, synthesize a set of deterministic automata $\mathcal{S} = \{S_k \in \phi(\Gamma_k)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

- $N((\times_{i\in I} G_i)\times(\times_{k\in K} S_k)) \subseteq N((\times_{i\in I} G_i)\times(\times_{j\in J} H_j))$
- $B((\times_{i\in I} G_i)\times(\times_{k\in K} S_k)) = \varnothing$
- $\times_{k\in K} S_k$ 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 $P_o:(\cup_{i\in I} \Sigma_i)^* \rightarrow (\cup_{i\in I} \Sigma_{i,o})^*$

An Aggregative Synthesis Approach (ASP)

- Inputs: standardized $\mathcal{G}=\{G_i\in\phi(\Sigma_i)|i\in I\}$, $\mathcal{H}=\{H_j\in\phi(\Delta_j)|j\in J\}\,|\Delta_j\subseteq\cup_{i\in I}\Sigma_i \wedge j\in J\}$
- 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 $J_k\neq\varnothing$, let $V_k:=\times_{j\in Jk}H_j$. Otherwise, set V_k as a recognizer of Σ_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 $I_{k+1}:=\{i\in I|k+1\leq i\leq n\}$, $\Sigma_{Ik+1}:=\cup_{i\in Ik+1}\Sigma_i$ and $\Theta_{k+1}:=\cup_{j\in J\text{-}Qk}\Delta_j$.
  - Choose $\Sigma_{Ak}\subseteq T_k$ with $(\Sigma_{Ik+1}\cup\Theta_{k+1})\cap T_k\subseteq\Sigma_{Ak}$. Let $A_k:=(W_k\times S_k)/\approx_{\Sigma Ak}$.
  - $W_{k+1}:=A_k\times G_{k+1}$, $Q_{k+1}:=\{j\in J|\Delta_j\subseteq\cup_{i=1}^{k+1}\Sigma_i\}$.
  - $J_{k+1}:=Q_{k+1}-Q_k$, $T_{k+1}:=\Sigma_{Ak}\cup\Sigma_{k+1}$.
- When terminate upon k, output $\mathcal{S}=\{S_1,S_2,\ldots,S_k\}$.

Aggregative Synthesis



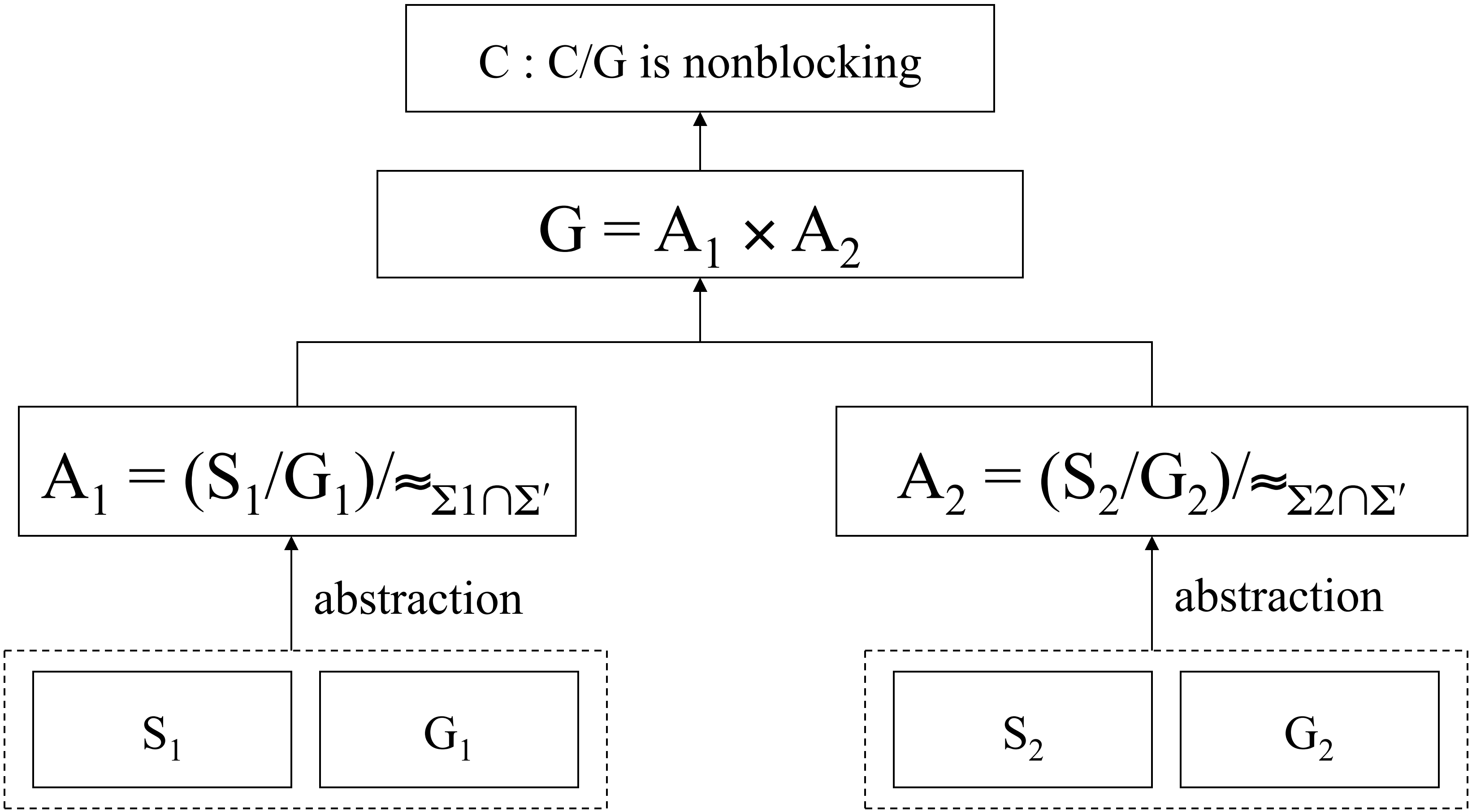
- **Theorem**

  The ASP always terminates, and if every S_k (k=1,2,…,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

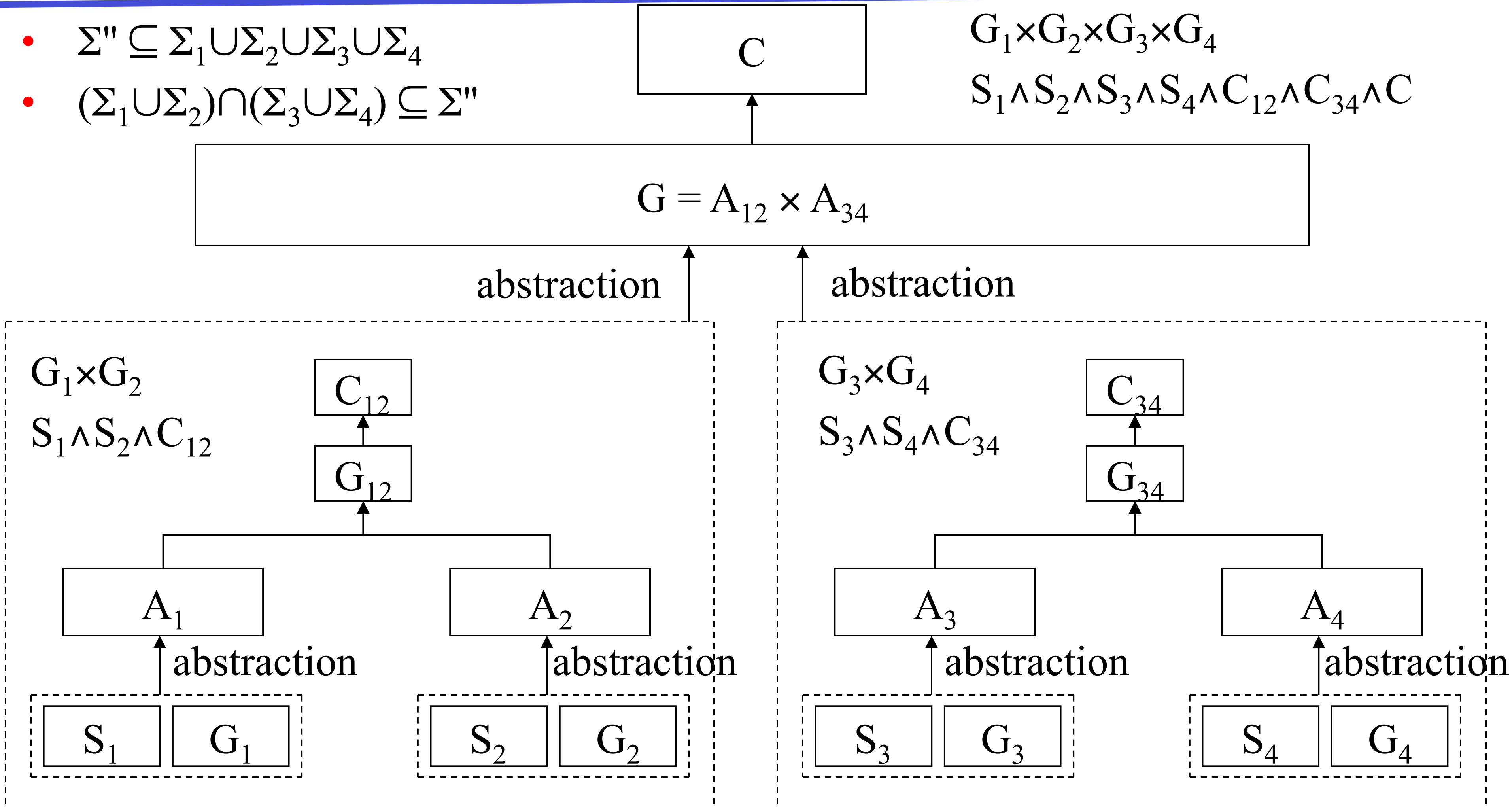
- How to order components so that it yields a solution?

Parallel Synthesis – Coordinated Distributed Control



Multi-Level Coordinators

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



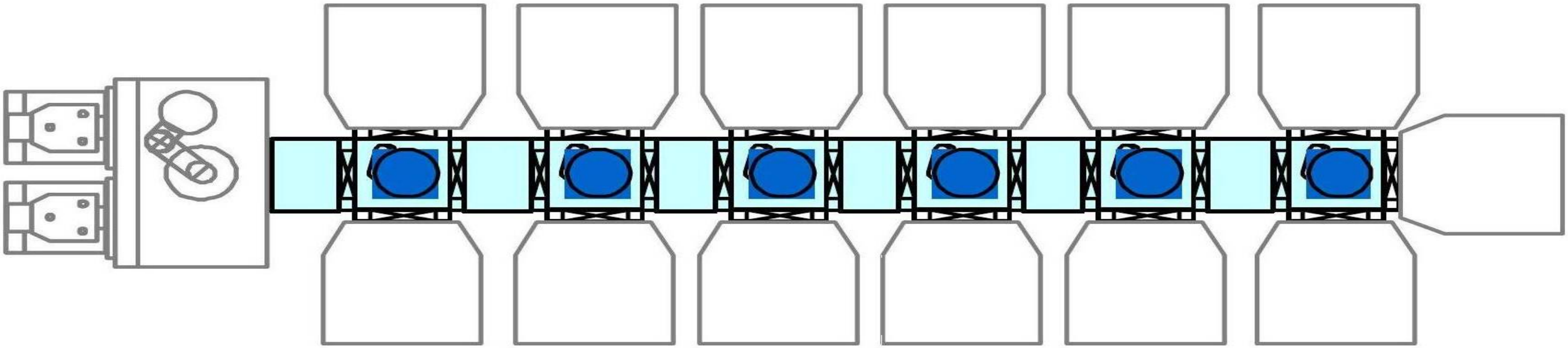
Main Difficulty for Coordinated Control

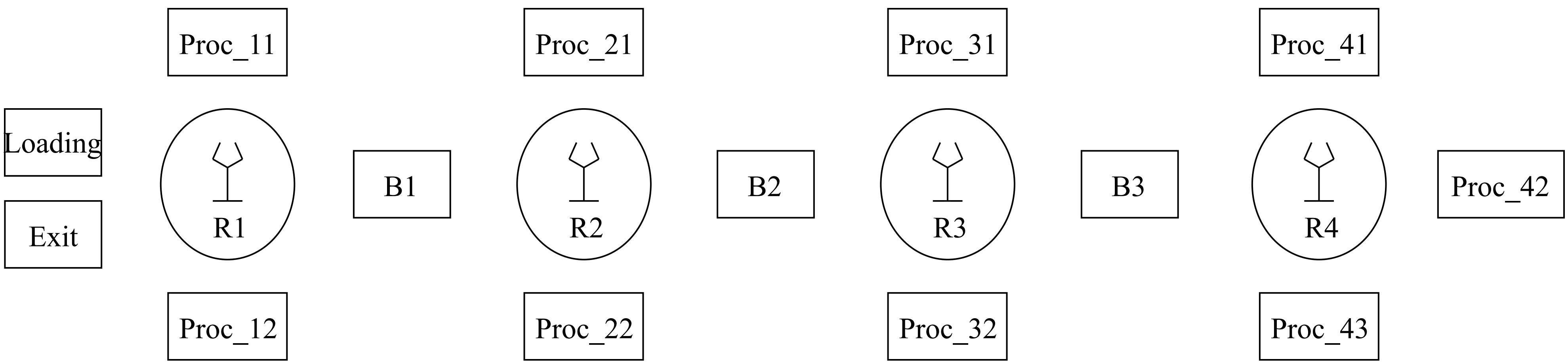
- How to define those coordinator alphabets?

Outline

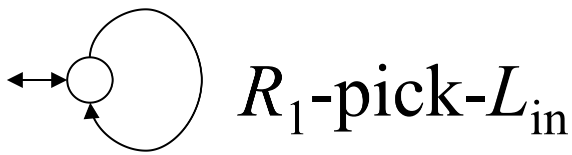
- Review of Automaton Abstraction
- Concepts of Supervisors and Relevent Properties
- Synthesis of Distributed Supervisors
- Example
- Conclusions

Cluster Tools

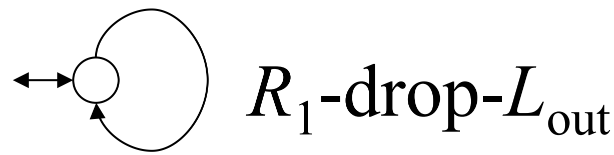




Component Models – Load and Exit Locks

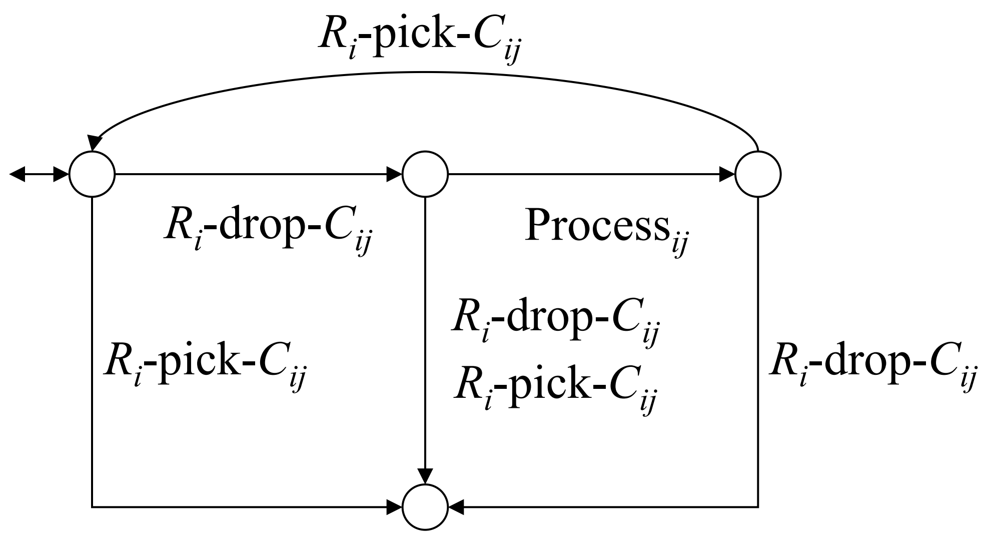


Entering Load Lock L_{in}



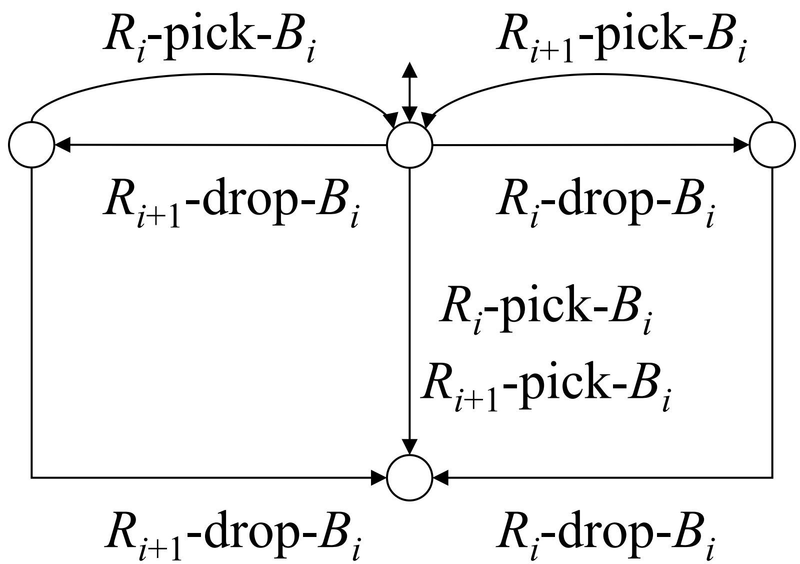
Exit Load Lock L_{out}

Component Models – Chambers



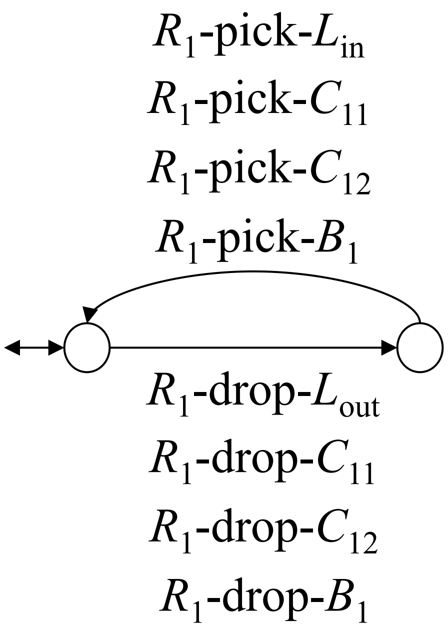
C_{ij}

Component Models – Buffers

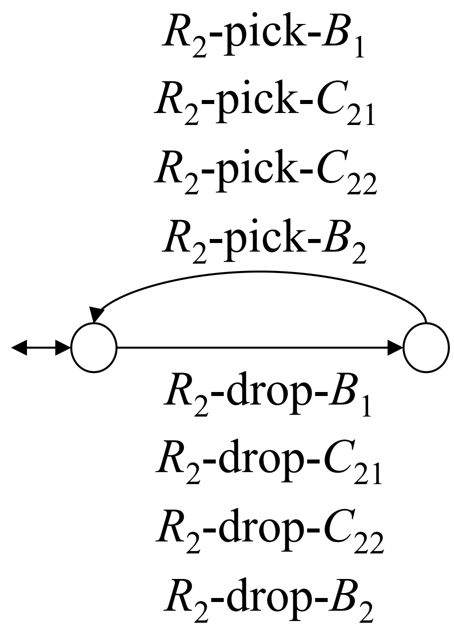


B_i

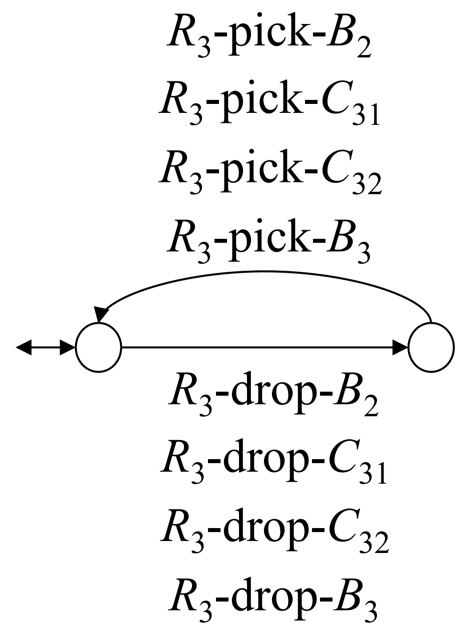
Component Models – Robots



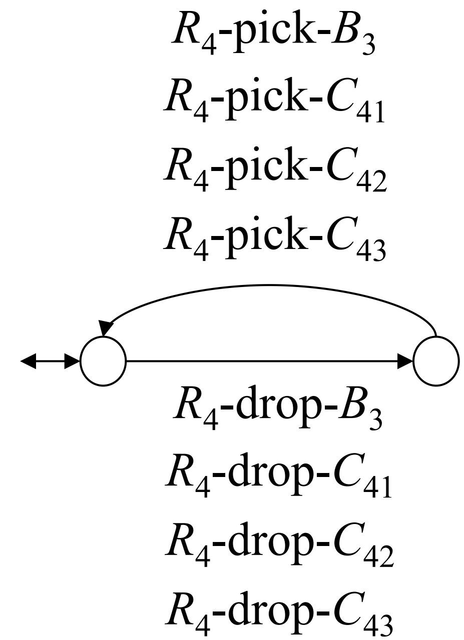
R_1



R_2

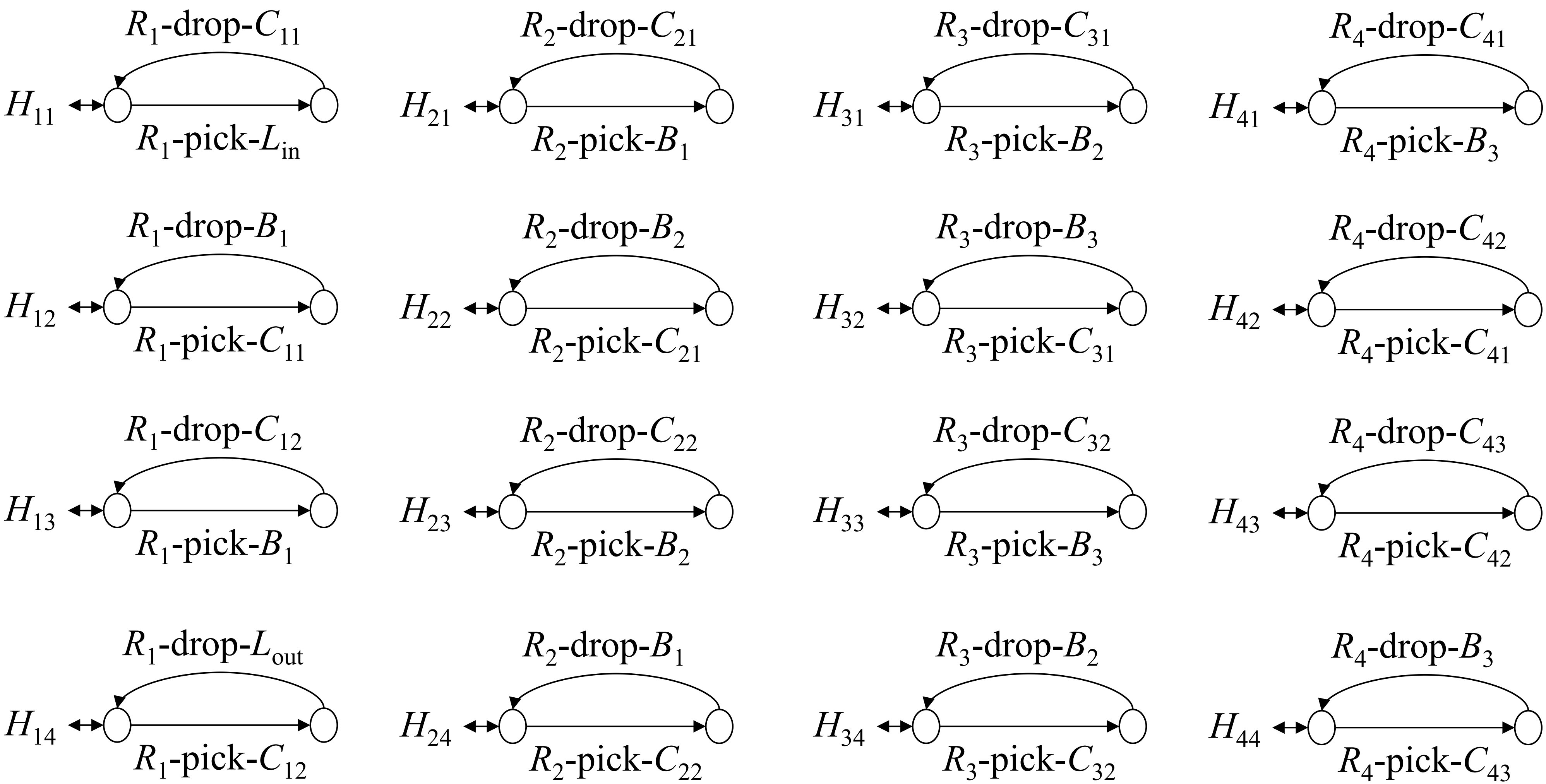


R_3



R_4

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
  - Use make_sequential_abstraction(`W2.cfg, S2.cfg', `R2-pick-B2, R2-drop-B2, R3-pick-B2, R3-drop-B2', `A2.cfg') :: A2 (15, 24)

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