CAS 745
Supervisory Control of Discrete-Event Systems

Slides 3: Supervision of Discrete-Event Systems: Basics

Dr. Ryan Leduc

Department of Computing and Software
McMaster University

Material based on W. M. Wonham, Supervisory Control of Discrete-Event Systems, Department of Electrical and Computer Engineering, University of Toronto, July 2004. Lecture notes of Professor Wonham also used.

DES Supervision Intro - §3.1

- Discrete-event systems are used to model physical systems whose associated processes can be seen as:
- *Discrete* (in time and (usually) state space), *asynchronous* (driven by events, not a clock) and *generative* (event selection method not modelled).
- Will represent the *plant* (system of interest) as a generator of a formal language.
- Will add a control mechanism that can vary the behaviour of the plant within prescribed limits.
- The desired behaviour of the controlled plant is that its generated language be contained in a specification language.
- We will consider this problem solved for a specific plant when we can show a suitable controller exists and can be constructed.

DES Representation - §3.2

- We will represent our discrete-event system (DES) as a generator:

$$\mathbf{G} = (Q, \Sigma, \delta, q_o, Q_m)$$

  where Σ is finite alphabet of *event labels*, Q is the *state set*, $\delta : Q \times \Sigma \to Q$ is the *(partial) transition function*, q_o the *initial state*, and $Q_m \subseteq Q$ the subset of *marker states*.

- A *transition* or *event* of $\mathbf{G}$ is a triple of form (q, σ, q') where $\delta(q, \sigma) = q'$

- We refer to q as the *exit state*, q' as the *entrance state*, and σ as the *event label*.

- The *event set* of $\mathbf{G}$ is the set of these triples, so defined.

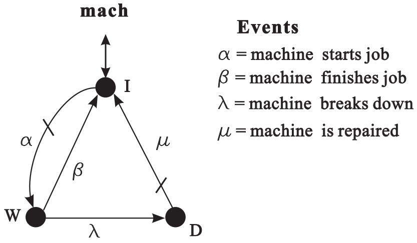
Control Technology

- We next partition are alphabet as follows:

$$\Sigma = \Sigma_c \,\dot\cup\, \Sigma_u$$

- *Controllable events* (Σ_c) are events that can be enabled or disabled by an external agent. They can only occur if they have been enabled.

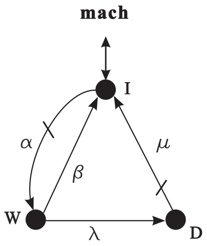
- Controllable events are often indicated on a graph by a slash through the event arrow. In DES, **mach**, $\Sigma_c = \{\alpha, \mu\}$



Control Technology - II

- *Uncontrollable events* (Σ_u) are events that can not be disabled by an external agent.

- Once the plant is in a state that the event is possible, it can no longer be prevented. In DES, **mach**, $\Sigma_u = \{\beta, \lambda\}$



Events

α = machine starts job

β = machine finishes job

λ = machine breaks down

μ = machine is repaired

TCT Procedure: create

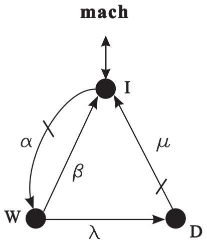
- To create and save (as a .DES file) a DES using TCT (You can download it at: http://www.control.toronto.edu/DES/), use the **create** operator.
- It will ask the user to enter the DES name, the number of states (state *size*), list of marker states, and transitions (event triples).
- The state set is integer set $\{0, 1, 2, \ldots, \mathit{size} - 1\}$.
- The initial state is always 0.
- For **mach**, state set would be $\{0, 1, 2\}$, and the marker state set $\{0\}$.
- Event labels are integers in range $\{0, 1, \ldots, 999\}$.
- Controllable events must be odd numbers and uncontrollable events even.

TCT Procedure: create - II

- Transitions are entered as tuples (q, σ, q') where $\delta(q, \sigma) = q'$.
- For **mach**, we could take: $I = 0$, $W = 1$, $D = 2$, $\alpha = 1$, $\beta = 0$, $\lambda = 2$, and $\mu = 3$.

  We would then have the following list of transitions:

```
0   1   1
1   0   0
1   2   2
2   3   0
```

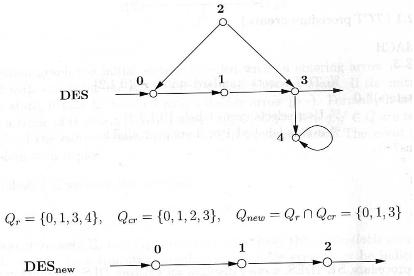


Events
α = machine starts job
β = machine finishes job
λ = machine breaks down
μ = machine is repaired

Trim: DES

- **Defn:** a DES is trim if it is both reachable and coreachable.
- Implemented in TCT by the **trim** function, which returns the trimmed version of its parameter DES.
- If a DES is trim, it is nonblocking, but a DES can be nonblocking, yet not trim.
- **Defn:** For $K \subseteq \Sigma^*$ then DES $\mathbf{G}$ represents K if $\mathbf{G}$ is nonblocking and $L_m(\mathbf{G}) = K$. This implies $L(\mathbf{G}) = \overline{K}$

$\mathbf{DES_{new} = trim(DES)}$



$Q_r = \{0,1,3,4\}, \quad Q_{cr} = \{0,1,2,3\}, \quad Q_{new} = Q_r \cap Q_{cr} = \{0,1,3\}$

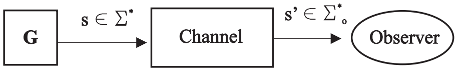
Natural Projection - §3.2

- Let $\Sigma_o \subseteq \Sigma$. We take Σ_o to be the set of observable events through some communication channel.
- **Defn:** We define the natural projection of Σ^* onto Σ_o^*, $P : \Sigma^* \to \Sigma_o^*$, as follows:

$$\begin{aligned} P(\epsilon) &= \epsilon \\ P(\sigma) &= \begin{cases} \epsilon & \text{if } \sigma \notin \Sigma_o \\ \sigma & \text{if } \sigma \in \Sigma_o \end{cases} \\ P(s\sigma) &= P(s)P(\sigma), \quad s \in \Sigma^*, \sigma \in \Sigma \end{aligned}$$

eg. For $\Sigma = \{\alpha, \beta, \lambda\}$, $\Sigma_o = \{\alpha, \lambda\}$, and $s = \alpha\beta\alpha\lambda\beta\alpha$, we get:

$$P(s) = P(\alpha)P(\beta)P(\alpha)P(\lambda)P(\beta)P(\alpha) = \alpha\alpha\lambda\alpha$$



Natural Projection - II

- Clearly, P is catenative. ie. for $s_1, s_2 \in \Sigma^*$,

$$P(s_1 s_2) = P(s_1)P(s_2)$$

- Let $L \subseteq \Sigma^*$. We get:

$$PL := \{P(s) |\, s \in L\} \subseteq \Sigma_o^*$$

- The TCT procedure **project** operates on a DES $\mathbf{G}$ and event set $\Sigma_{\text{null}} := \Sigma - \Sigma_o$.

- It returns a minimal DES $\mathbf{G}_{\text{new}} = \textbf{project}(\mathbf{G}, \Sigma_{\text{null}})$ with alphabet Σ_o such that:

$$L(\mathbf{G}_{\text{new}}) = PL(\mathbf{G}) \quad L_m(\mathbf{G}_{\text{new}}) = PL_m(\mathbf{G})$$

Inverse Image Fcn of P

- For $P : \Sigma^* \rightarrow \Sigma_o^*$, we define its inverse image function, $P^{-1} : \text{Pwr}(\Sigma_o^*) \rightarrow \text{Pwr}(\Sigma^*)$, as:

$$P^{-1}(H) := \{s \in \Sigma^* \,|\, P(s) \in H\}, \text{ for } H \subseteq \Sigma_o^*$$

- For $\Sigma = \{\alpha, \beta, \lambda, \mu\}$, $\Sigma_o = \{\alpha, \lambda\}$, and $s_o = \alpha\alpha\lambda\alpha$, we get:

$$P^{-1}(\{s_o\}) = \{\beta, \mu\}^*\alpha\{\beta, \mu\}^*\alpha\{\beta, \mu\}^*\lambda\{\beta, \mu\}^*\alpha\{\beta, \mu\}^* \subseteq \Sigma^*$$

- For a DES $\mathbf{G}$ over Σ_o, and event set $\Sigma_{\text{aux}} = \Sigma - \Sigma_o$, the TCT procedure **selfloop**$(\mathbf{G}, \Sigma_{\text{aux}})$ returns DES $\mathbf{G}_{\text{new}}$ where:

$$L(\mathbf{G}_{\text{new}}) = P^{-1}(L(\mathbf{G})) \quad L_m(\mathbf{G}_{\text{new}}) = P^{-1}(L_m(\mathbf{G})$$

Synchronous Product

- When we are modelling a system, it is often easier to model it as several smaller DES, than as one large one.
- For plants, we use the synchronous product operator to combine the component DES into a single, more complex DES.
- **Defn:** Let $L_1 \subseteq \Sigma_1^*$, $L_2 \subseteq \Sigma_2^*$, and $\Sigma = \Sigma_1 \cup \Sigma_2$. Let $\mathsf{P}_i : \Sigma^* \to \Sigma_i^*$, $i = 1, 2$, be natural projections.
- The synchronous product of L_1 and L_2 is defined to be:

$$L_1 || L_2 \quad = \quad \mathsf{P}_1^{-1}(L_1) \cap \mathsf{P}_2^{-1}(L_2)$$

where $\mathsf{P}_i^{-1}(L_i) = \{s \in \Sigma^* | \, P_i(s) \in L_i\}$

Synchronous Product - II

- For DES $\mathbf{G}_i = (Y_i, \Sigma_i, \delta_i, y_{o_i}, Y_{m_i})$ $(i = 1, 2)$, the synchronous product is defined to be a reachable DES:

  $$\mathbf{G} = \mathbf{G}_1 || \mathbf{G}_2 = (Y \subseteq Y_1 \times Y_2, \Sigma_1 \cup \Sigma_2, \delta, y_o, Y_m)$$

  with the properties:

  $$L_m(\mathbf{G}) = L_m(\mathbf{G_1}) || L_m(\mathbf{G_2}), \quad L(\mathbf{G}) = L(\mathbf{G_1}) || L(\mathbf{G_2})$$

- Can think of $\mathbf{G}_1$ and $\mathbf{G}_2$ as acting cooperatively by agreeing to synchronize events they have in common.
- As long as the event sets of the DES are listed explicitly, then $||$ is associative.
- The TCT procedure is called **sync**.
- Unfortunately, TCT defines the event set of a DES to be all event labels that appear as a transition in the DES, and thus is not always associative. See ex 3.3.5.

Intro Assembly Station 1*

- The structure of assembly station 1 (AS1) is as given in diagram on next page.
- It contains a conveyor belt for bringing pallets to the station, and removing the pallet after processing.
- When pallet reaches station, it is stopped at the pallet gate.
- When pallet is allowed through gates, it is stopped at the pallet stop.

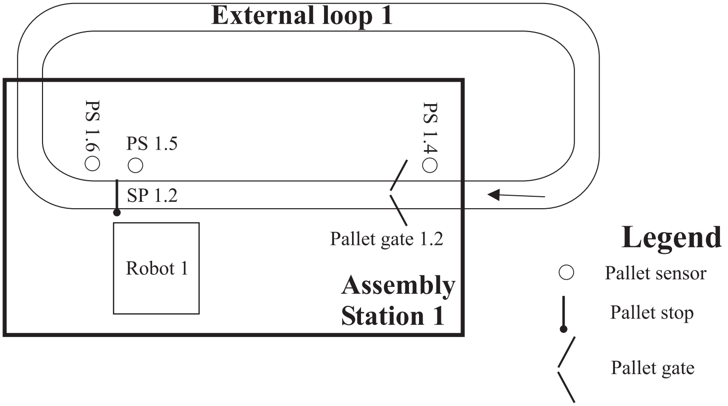
$\ast$ Material based on R.J. Leduc. *Hierarchical Interface-based Supervisory Control.* Doctoral Thesis, Dept. of Elec. & Comp. Engrg., Univ. of Toronto, 2002

Intro Assembly Station i - II

- Pallet is held here while robot performs assembly operations.
- Robot can perform two tasks, labelled task1A and task1B. Robot can also breakdown.
- Pallet stop is then opened to allow the pallet to leave AS1.
- Sensors detect the presence of a pallet.

Diagram of Assembly Station 1*

Fig. A: AS1



* This is a simplified version of assembly station 1 of the AIP example.

Modelling AS1

- We want to model AS1 as a DES plant.
- Typically we have two types of behavior we need to model:
  1. Local Behavior: behavior local to a component ie. what tasks the robot can do, how it performs these tasks.
  2. Interaction behavior: this is behavior that models how component interact with other component. ie. when the robot should perform a task, depending on the movement of a pallet.

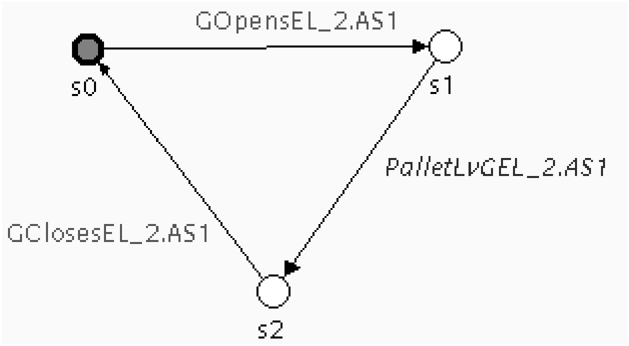
Modelling AS1 - II

- For our example, our local behavior would be that of the sensors, gate, pallet stop, and robot.
- For our example, some components are so simple (ie a sensor) we have combined the local behavior with the interaction behavior in the same DES.
- Our interaction behavior would be how these items are physically positioned with respect to the movement of a pallet.
- For example, a pallet can't leave the gate, until a pallet is detected at the gate by a sensor.
- Another example is that a pallet can't arrive at the pallet stop until it leaves the gate.

Plant Models: Local Behavior Only

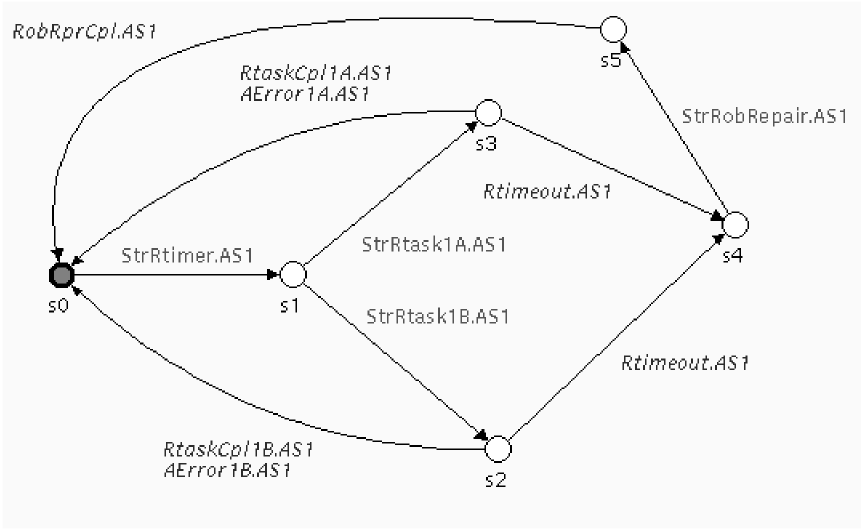
- Notation used: solid circle = marked state, circle with heavy outline = initial state. Event labels in italics are uncontrollable.

Fig B: PalletGateEL_2



Plant Models: Local Behavior Only - II

Fig C: Robot1



Plant Models: Interaction Behavior

Fig D: DepGate_2

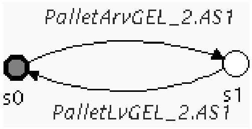
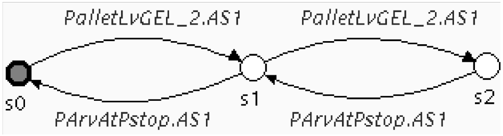


Fig E: DepGateNPstop



Plant Models: Interaction Behavior - II

- Our complete plant model would be:

$$\textbf{PLANT} = \text{sync}(B, C, D, E, F, G)$$

where B refers to the DES in Fig. B etc.

Fig F: PSenAtPstop

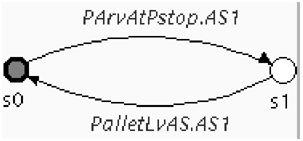
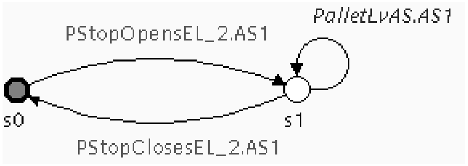


Fig G: PalletSTopEL_2



Meet Operator

- **Defn:** For the special case of $\Sigma_1 \cap \Sigma_2 = \emptyset$, the synchronous product becomes the *shuffle product.* The resulting language consists of all possible interleavings of strings from the original two languages.

- **Defn:** For DES $\mathbf{G}_i = (Y_i, \Sigma, \delta_i, y_{o_i}, Y_{m_i})$ $(i = 1, 2)$, the TCT procedure **meet** returns a reachable DES $\mathbf{G} =$ **meet**$(\mathbf{G}_1, \mathbf{G}_2)$:

  with the properties:

$$L_m(\mathbf{G}) = L_m(\mathbf{G_1}) \cap L_m(\mathbf{G_2}), \quad L(\mathbf{G}) = L(\mathbf{G_1}) \cap L(\mathbf{G_2})$$

- The **meet** operator is equivalent to the **sync** operator when $\Sigma_1 = \Sigma_2 = \Sigma$.

Meet Operator - II

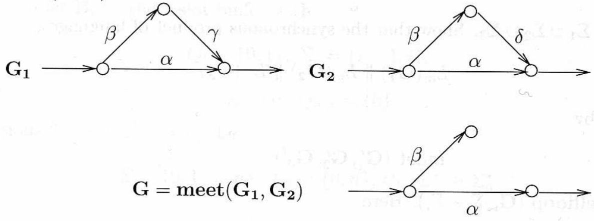
- As $L_m(\mathbf{G_i}) \subseteq L(\mathbf{G_i})$, $i = 1, 2$, we have:

$$L_m(\mathbf{G}) \subseteq L(\mathbf{G})$$

- **NOTE:** The **meet** operator will block any event label that does not occur (have a transition) in both DES.

- As below, $\mathbf{G_i}$ nonblocking $\not\Rightarrow$ **meet**$(\mathbf{G}_1, \mathbf{G}_2)$ nonblocking!

Example 3.3.2 (Meet)



Meet Algorithm

Assume we are given DES $\mathbf{G}_i = (Y_i, \Sigma, \delta_i, y_{o_i}, Y_{m_i})$ $(i = 1, 2)$, and we want to construct their **meet** DES, $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$.

Define Variables:

Let $y_o = (y_{o,1}, y_{o,2})$, $Y_m = \emptyset$

$Y_{\mathsf{rch}} = \emptyset$ // states reached by algorithm

$Y_{\mathsf{pend}} = \emptyset$ // states waiting to be examined

$\delta = \emptyset$ //initialize transition function - store as tuples

$\delta^{-1} = \emptyset$ // initialize inverse transition function
// will store as tuples of form $(y, \sigma, Y_{\mathsf{inv}})$ where
// $\delta^{-1}(\{(y', \sigma)\}) = \{y \in Y \,|\, \delta(y, \sigma) = y'\}$

Meet Algorithm - II

```
push($y_o, Y_{\mathsf{rch}}$)
push($y_o, Y_{\mathsf{pend}}$)
while ($Y_{pend} \neq \emptyset$) {
    $y = (y_1, y_2) =$ pop($Y_{\mathsf{pend}}$)
    if $((y_1 \in Y_{m_1})\,\&\,(y_2 \in Y_{m_2}))$ then
        push($(y_1, y_2), Y_m$)
    for $\sigma$ in $\Sigma$ {
        if $((\delta_1(y_1, \sigma)!)\,\&\,(\delta_2(y_2, \sigma)!))$ then {
          $y' = (y'_1, y'_2) = (\delta_1(y_1, \sigma), \delta_2(y_2, \sigma))$
          push($(y, \sigma, y'), \delta$)
          $\delta^{-1}(\{(y', \sigma)\}) = \delta^{-1}(\{(y', \sigma)\}) \cup \{y\}$
          if $(y' \notin Y_{\mathsf{rch}})$ then {
            push($y', Y_{\mathsf{rch}}$)
            push($y', Y_{\mathsf{pend}}$)
          }
        }  // end of if
} }  // end of for and while loop
```

Meet Algorithm - III

- At the conclusion of the algorithm, we have constructed a suitable δ, y_o, and Y_m.
- We then take $Y = Y_{\text{rch}}$, and we have our:

$$\mathbf{G} = \textbf{meet}(\mathbf{G}_1, \mathbf{G}_2) = (Y, \Sigma, \delta, y_o, Y_m)$$

- To construct $\textbf{meet}(\mathbf{G}_1, \mathbf{G}_2)$, we don't actually need to construct δ^{-1}.
- It was included to show how it can be constructed at the same time as the **meet**.
- We will use δ^{-1} in our nonblocking algorithm.
- To save memory when checking nonblocking, we do not need to store δ, and for δ^{-1} we do not need to store the event label.
- For controllability, we dont have to store state info, δ or δ^{-1}.

Sync Algorithm

- Assume we are given DES $\mathbf{G}_i = (Y_i, \Sigma_i, \delta_i, y_{o_i}, Y_{m_i})$ $(i = 1, 2)$, and we want to construct $\mathbf{G} = \mathbf{G}_1 || \mathbf{G}_2$.
- Let $\mathbf{G}_{\mathsf{new}_i} =$ **selfloop**$(\mathbf{G}_i, \Sigma_{\mathsf{aux}_i})$, $\Sigma_{\mathsf{aux}_i} = \Sigma - \Sigma_i$, $i = 1, 2$
- We then construct $\mathbf{G}$ by applying our **meet** algorithm to $\mathbf{G}_{\mathsf{new}_i}$, $i = 1, 2$.

Nonconflicting (see Section §3.6)

- Let $L_1, L_2 \subseteq \Sigma^*$
- In general,

$$\overline{L_1 \cap L_2} \subsetneq \overline{L_1} \cap \overline{L_2}$$

- ie. let $\Sigma = \{\alpha, \beta\}$, $L_1 = \{\alpha\alpha\}$, and $L_2 = \{\alpha\beta\}$
- We have:

$$\overline{L_1 \cap L_2} = \overline{\emptyset} = \emptyset \text{ and } \overline{L_1} \cap \overline{L_2} = \{\epsilon, \alpha, \alpha\alpha\} \cap \{\epsilon, \alpha, \alpha\beta\} = \{\epsilon, \alpha\}$$

- **Defn:** We say L_1 and L_2 are *nonconflicting* if:

$$\overline{L_1 \cap L_2} = \overline{L_1} \cap \overline{L_2}$$

- For DES $\mathbf{G}_i = (Y_i, \Sigma_i, \delta_i, y_{o_i}, Y_{m_i})$ $(i = 1, 2)$, let $L_i = L_m(\mathbf{G}_i)$.
- TCT procedure: **nonconflict**$(\mathbf{G}_1, \mathbf{G}_2) =$ TRUE

  if $\mathbf{G}_1$, $\mathbf{G}_2$ are nonblocking, and L_1, L_2 are *nonconflicting*.

Nonconflicting - II

- The TCT procedure actually checks that $\mathbf{G} = \textbf{meet}(\mathbf{G}_1, \mathbf{G}_2)$ is reachable and coreachable (ie. nonblocking).

- **Proposition:** If $\mathbf{G}_1$, $\mathbf{G}_2$ are nonblocking, then $L_m(\mathbf{G}_1)$, $L_m(\mathbf{G}_2)$ are nonconflicting iff $\mathbf{G} = \textbf{meet}(\mathbf{G}_1, \mathbf{G}_2)$ is nonblocking.

Proof: Assume $\mathbf{G}_1$, $\mathbf{G}_2$ are nonblocking. **(1)**

Must show implies $L_m(\mathbf{G}_1)$, $L_m(\mathbf{G}_2)$ are nonconflicting iff $\mathbf{G}$ is nonblocking.

A) Show $L_m(\mathbf{G}_1)$, $L_m(\mathbf{G}_2)$ are nonconflicting $\Rightarrow$ $\mathbf{G}$ is nonblocking.

Assume $L_m(\mathbf{G}_1)$, $L_m(\mathbf{G}_2)$ are nonconflicting. **(2)**

Must show implies $\mathbf{G}$ is nonblocking.

Nonconflicting - III

Sufficient to show: $\overline{L_m(\mathbf{G})} = L(\mathbf{G})$

From **(2)**, we have:

$$\overline{L_m(\mathbf{G}_1) \cap L_m(\mathbf{G}_2)} = \overline{L_m(\mathbf{G}_1)} \cap \overline{L_m(\mathbf{G}_2)}$$

From **(1)**, we can conclude:

$$\overline{L_m(\mathbf{G}_i)} = L(\mathbf{G}_i), \quad i = 1, 2$$

Substituting in, gives:

$$\overline{L_m(\mathbf{G}_1) \cap L_m(\mathbf{G}_2)} = L(\mathbf{G}_1) \cap L(\mathbf{G}_2)$$

$\Rightarrow \overline{L_m(\mathbf{G})} = L(\mathbf{G})$ by defn of **meet**

Part A complete.

Nonconflicting - IV

B) Show $\mathbf{G}$ nonblocking $\Rightarrow L_m(\mathbf{G}_1)$, $L_m(\mathbf{G}_2)$ are nonconflicting.

Follows immediately from reversing the proof of Part A.

From Parts A and B, we can now conclude $L_m(\mathbf{G}_1)$, $L_m(\mathbf{G}_2)$ are nonconflicting iff $\mathbf{G}$ is nonblocking.

QED

- To find out if DES $\mathbf{G}_1$ is nonblocking using TCT, check that:

$$\textbf{nonconflict}(\mathbf{G}_1, \mathbf{G}_1) = \text{TRUE}$$

Nonblocking Algorithm

Assume we are given DES $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$, δ^{-1} (the inverse transition function), $N_Y \geq 0$ (number of reachable states of $\mathbf{G}$), and $N_{Y_m} \geq 0$ (number of reachable marked states of $\mathbf{G}$).

We will also assume that $\mathbf{G}$ is reachable, for simplicity.

Want to check that all reachable states are coreachable.

Define Variables:

```
Let $Y_{\mathsf{fnd}} = Y_m$  // states reached by algorithm

$Y_{\mathsf{pend}} = Y_m$  // states waiting to be examined

// Will use $N_Y$ as number of states still to check
$N_Y = N_Y - N_{Y_m}$

if $(N_Y \leq 0)$ then
    return "system is nonblocking"
```

Nonblocking Algorithm - II

```
while ($Y_{pend} \neq \emptyset$) {
    $y = \text{pop}(Y_{\text{pend}})$
    for $\sigma$ in $\Sigma$ {
        $Y' = \delta^{-1}(\{(y,\sigma)\})$
        for $y'$ in $Y'$ {
            if ($y' \notin Y_{\text{fnd}}$) then {
                push($y', Y_{\text{fnd}}$)
                push($y', Y_{\text{pend}}$)
                $N_Y = N_Y - 1$
                if ($N_Y \leq 0$) then
                    return "system is nonblocking"
            } // end of if
        } // end of for $y'$ loop
    } // end of for $\sigma$ loop
} // end of while loop

// if reached here, then system blocks
return "system has" $N_Y$ "noncoreachable states"
```

Controllability and Supervision - §3.4

- Let $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$, be a nonempty controlled DES with event set:

$$\Sigma = \Sigma_c \,\dot{\cup}\, \Sigma_u$$

- As before, Σ_c is the set of events that can be disabled by an external agent (that we will call a *supervisor*).

- **Defn:** A control pattern is a subset of Σ that contains all uncontrollable events. It represents the events to be currently enabled.

- We define the set of all control patterns as follows:

$$\Gamma = \{\gamma \in \mathrm{Pwr}(\Sigma) \,|\, \gamma \supseteq \Sigma_u\}$$

- **Defn:** A supervisory control for $\mathbf{G}$ is any map $V : L(\mathbf{G}) \rightarrow \Gamma$

Closed Loop Behaviour

- We write $V/\mathbf{G}$ for the pair $(\mathbf{G}, V)$ to imply $\mathbf{G}$ under the supervision of V.

- **Defn:** The closed behaviour of $V/\mathbf{G}$ is the language $L(V/\mathbf{G}) \subseteq L(\mathbf{G})$ defined 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 $L(V/\mathbf{G})$

- Clearly $L(V/\mathbf{G})$ is prefix-closed and nonempty and in the range $\{\epsilon\} \subseteq L(V/\mathbf{G}) \subseteq L(\mathbf{G})$

Marked Behavior

- **Defn:** The marked behaviour of $V/\mathbf{G}$ is defined to be:

$$L_m(V/\mathbf{G}) := L(V/\mathbf{G}) \cap L_m(G)$$

- ie strings in $L_m(G)$ that are still possible under supervision by V.

- We have: $\emptyset \subseteq L_m(V/\mathbf{G}) \subseteq L_m(G)$

- **Defn:** We say V is *nonblocking* for $\mathbf{G}$ if

$$\overline{L_m(V/\mathbf{G})} = L(V/\mathbf{G})$$

Controllable Languages

- Want to be able to determine if a language $K \subseteq \Sigma^*$ could qualify as a marked behavior of some supervisory control V for $\mathbf{G}$.
- **Defn:** A language $K \subseteq \Sigma^*$ is controllable with respect to $\mathbf{G}$ if

$$(\forall s \in \overline{K})(\forall \sigma \in \Sigma_u)\, s\sigma \in L(\mathbf{G}) \Rightarrow s\sigma \in \overline{K}$$

- *One step interpretation:* For a string $s \in \overline{K} \cap L(\mathbf{G})$, any uncontrollable event ($\sigma \in \Sigma_u$) allowed by $\mathbf{G}$ (ie $s\sigma \in L(\mathbf{G})$) will also be accepted by $\overline{K}$ (ie $s\sigma \in \overline{K}$).
- For $S \subseteq \Sigma^*$ and $\Sigma_o \subseteq \Sigma$, then define:

$$S\Sigma_o := \{s\sigma |\, s \in S \,\&\, \sigma \in \Sigma_o\}$$

- Can now restate controllable defn more concisely: language $K \subseteq \Sigma^*$ is controllable with respect to $\mathbf{G}$ iff

$$\overline{K}\Sigma_u \cap L(\mathbf{G}) \subseteq \overline{K}$$

Controllable Languages and L-Closed

- Clearly, $\emptyset$, $L(\mathbf{G})$, and Σ^* are controllable wrt to $\mathbf{G}$.
- Constrains only $s \in \overline{K} \cap L(\mathbf{G})$.
- For Language $L \subseteq \Sigma^*$, we say L is closed if $L = \overline{L}$.
- **Defn:** Let $K \subseteq L \subseteq \Sigma^*$. The language K is L-closed if: $K = \overline{K} \cap L$
- Means that K contains all of its prefixes that belong to L.
- If we want to use $\overline{K}$ to represent $L(V/\mathbf{G})$ for some supervisory control V for $\mathbf{G}$, then $K = \overline{K} \cap L_m(\mathbf{G}) = L(V/\mathbf{G}) \cap L_m(\mathbf{G})$ means that it qualifies as the marked behaviour.

Controllable Languages Theorem

- **Theorem 3.4.1** Let $K \subseteq L_m(\mathbf{G})$, $K \neq \emptyset$. There exists a *nonblocking supervisory control* V for $\mathbf{G}$ such that $L_m(V/\mathbf{G}) = K$ if and only if:

  i) K is controllable with respect to $\mathbf{G}$ and

  ii) K is $L_m(\mathbf{G})$-closed.

  Proof: see proof in course notes.

- We refer to a nonblocking supervisory control (the "for $\mathbf{G}$" is understood) as an *NSC.*

- **Corollary:** Let $K \subseteq L(\mathbf{G})$ be nonempty and closed. There exists a supervisory control V for $\mathbf{G}$ such that $L(V/\mathbf{G}) = K$ if and only if K is controllable with respect to $\mathbf{G}$.

Marking Supervisory Controls

- Want to also allow supervisory to specify marked behavior.
- **Defn:** Let $M \subseteq L_m(\mathbf{G})$. A marking nonblocking supervisory control for pair $(M, \mathbf{G})$, or *(MNSC)*, is a map $V : L(\mathbf{G}) \to \Gamma$ with the marked behaviour of $V/\mathbf{G}$ defined to be:

$$L_m(V/\mathbf{G}) := L(V/\mathbf{G}) \cap M$$

- **Theorem 3.4.2** Let $K \subseteq L_m(\mathbf{G})$, $K \neq \emptyset$. There exists a MNSC V for $(K, \mathbf{G})$ such that $L_m(V/\mathbf{G}) = K$ if and only if K is controllable with respect to $\mathbf{G}$.

Proof:

Let $K \subseteq L_m(\mathbf{G})$, $K \neq \emptyset$. **(1)**

Marking Supervisory Controls - II

(if) Assume K is controllable with respect to $\mathbf{G}$ **(2)**

Must show implies there exists a MNSC V for $(K, \mathbf{G})$ such that $L_m(V/\mathbf{G}) = K$.

First, we must construct suitable $V : L(\mathbf{G}) \rightarrow \Gamma$.

For $s \in L(\mathbf{G})$, let:

$$V(s) := \Sigma_u \cup \{\sigma \in \Sigma_c |\, s\sigma \in \overline{K}\} \qquad (\mathbf{3})$$

Clearly, V is a supervisory control.

Step 1) Show that $L(V/\mathbf{G}) = \overline{K}$

Marking Supervisory Controls - III

A) Show $L(V/\mathbf{G}) \subseteq \overline{K}$

Let $s \in L(V/\mathbf{G})$. **(4)**

We will show this implies $s \in \overline{K}$.

We will do this by induction on length of string.

Let $m = |s|$.

We thus know:

$$(\exists s_0, \ldots, s_m \in \Sigma^*) \quad (s_0 = \epsilon)\&(s_m = s)\&(s_n \leq s_{n+1})\&(|s_n| = n), \quad n = 0, 1, \ldots, m-1 \qquad \textbf{(5)}$$

As $s \in L(V/\mathbf{G})$ and fact that $L(V/\mathbf{G})$ is closed, we have:
$s_n \in L(V/\mathbf{G})$, $n = 0, \ldots, m$ **(6)**

Base case: Show that $s_0 \in \overline{K}$

From **(5)**, we have $s_0 = \epsilon$.

Marking Supervisory Controls - IV

We have $\epsilon \in \overline{K}$ as $K \neq \emptyset$, by **(1)**.

We thus have $s_0 \in \overline{K}$, as required.

Base case complete.

Inductive Step:

Assume $s_{n-1} \in \overline{K}$. **(7)**

Must show implies $s_n \in \overline{K}$.

From **(5)**, we have:

$$(\exists \sigma \in \Sigma)\, s_{n-1}\sigma = s_n$$

Marking Supervisory Controls - V

As $s_n \in L(V/\mathbf{G})$ (by **(6)**), we have:

$$s_{n-1}\sigma \in L(V/\mathbf{G}) \tag{8}$$

We must now show that $s_{n-1}\sigma \in \overline{K}$.

We have two cases: 1) $\sigma \in \Sigma_u$ 2) $\sigma \in \Sigma_c$

Case 1) $\sigma \in \Sigma_u$

From **(8)** and defn of $L(V/\mathbf{G})$, we have:

$$s_{n-1}\sigma \in L(\mathbf{G})$$

$\Rightarrow s_{n-1}\sigma \in \overline{K}\Sigma_u \cap L(\mathbf{G})$, as $s_{n-1} \in \overline{K}$ by **(7)**.

$\Rightarrow s_{n-1}\sigma \in \overline{K}$ as K is controllable.

$\Rightarrow s_n \in \overline{K}$, as required.

Marking Supervisory Controls - VI

Case 2) $\sigma \in \Sigma_c$

As $s_{n-1}\sigma \in L(V/\mathbf{G})$ (by **(8)**), and by defn of $L(V/\mathbf{G})$, we have:

$$\sigma \in V(s_{n-1}) \;\&\; s_{n-1}\sigma \in L(\mathbf{G})$$

$\sigma \in V(s_{n-1}) \& \sigma \in \Sigma_c \Rightarrow s_{n-1}\sigma \in \overline{K}$, by defn of V

$\Rightarrow s_n \in \overline{K}$, as required.

By cases 1 and 2, we have $s_n \in \overline{K}$.

Inductive Step complete.

By Base case, and Inductive Step, we can now conclude that $s \in \overline{K}$.

Part A complete.

Marking Supervisory Controls - VII

B) Show $\overline{K} \subseteq L(V/\mathbf{G})$

Let $s \in \overline{K}$. **(9)**

Must show implies $s \in L(V/\mathbf{G})$.

Will use induction on length of string.

Let $m = |s|$.

We thus know:

$$(\exists s_0, \ldots, s_m \in \Sigma^*) \quad (s_0 = \epsilon) \& (s_m = s) \& (s_n \leq s_{n+1}) \& (|s_n| = n), \quad n = 0, 1, \ldots, m-1 \tag{10}$$

$\Rightarrow s_n \in \overline{K}$, $n = 0, \ldots, m$, as $\overline{K}$ closed. **(11)**

Base case: Show that $s_0 \in L(V/\mathbf{G})$

We have $s_0 = \epsilon \in L(V/\mathbf{G})$ by definition of $L(V/\mathbf{G})$.

Marking Supervisory Controls - VIII

Inductive Step:

Assume $s_{n-1} \in L(V/\mathbf{G})$.
Must show implies $s_n \in L(V/\mathbf{G})$.

From **(10)**, we have:

$$(\exists \sigma \in \Sigma)\, s_{n-1}\sigma = s_n$$

As $s_n \in \overline{K}$ (by **(11)**), we have:
$s_{n-1}\sigma \in \overline{K}$ **(12)**

$\Rightarrow s_{n-1}\sigma \in L(\mathbf{G}))$, by **(1)**.

We have two cases: i) $\sigma \in \Sigma_u$ ii) $\sigma \in \Sigma_c$

Case i) $\sigma \in \Sigma_u$

$\Rightarrow \sigma \in V(s_{n-1})$ by defn of V

$\Rightarrow s_{n-1}\sigma \in L(V/\mathbf{G})$

Marking Supervisory Controls - IX

Case ii) $\sigma \in \Sigma_c$

$\Rightarrow \sigma \in V(s_{n-1})$ by defn of V, and by **(12)**.

$\Rightarrow s_{n-1}\sigma \in L(V/\mathbf{G})$

By cases i, and ii, we have $s_{n-1}\sigma = s_n \in L(V/\mathbf{G})$

Inductive Step complete.

By Base case, and Inductive Step, we can now conclude that $s \in L(V/\mathbf{G})$.

Part B complete.

By Parts A and B, we can conclude:

$$L(V/\mathbf{G}) = \overline{K} \qquad \mathbf{(13)}$$

Step 1 complete.

Marking Supervisory Controls - X

Step 2) Show that $L_m(V/\mathbf{G}) = K$.

By defn: $L_m(V/\mathbf{G}) := L(V/\mathbf{G}) \cap K$

Substituting $L(V/\mathbf{G}) = \overline{K}$ (by **(13)**) gives:

$$L_m(V/\mathbf{G}) = \overline{K} \cap K = K$$

Step 3) Show that V is nonblocking.

Sufficient to show that $\overline{L_m(V/\mathbf{G})} = L(V/\mathbf{G})$

Automatic as $L_m(V/\mathbf{G}) = K$ and $L(V/\mathbf{G}) = \overline{K}$ (by **(13)**).

By steps 1-3, we have constructed an MNSC V for $(K, \mathbf{G})$ such that $L_m(V/\mathbf{G}) = K$.

If part complete.

Marking Supervisory Controls - XI

(only if) Assume there exists a MNSC V for $(K, \mathbf{G})$ such that
$L_m(V/\mathbf{G}) = K$. **(14)**

Must show implies K is controllable with respect to $\mathbf{G}$.

Sufficient to show that: $\overline{K}\Sigma_u \cap L(\mathbf{G}) \subseteq \overline{K}$

As V is an MNSC, it is thus nonblocking.

$\Rightarrow \overline{L_m(V/\mathbf{G})} = L(V/\mathbf{G})$

$\Rightarrow \overline{K} = \overline{L_m(V/\mathbf{G})} = L(V/\mathbf{G})$, by **(14)**. **(15)**

Let $s \in \overline{K}$, $\sigma \in \Sigma_u$, and $s\sigma \in L(\mathbf{G})$. **(16)**

We thus have: $s\sigma \in \overline{K}\Sigma_u \cap L(\mathbf{G})$

Must show implies $s\sigma \in \overline{K}$

As $s \in \overline{K}$, we have $s \in L(V/\mathbf{G})$, by **(15)**.

Marking Supervisory Controls - XII

As $\sigma \in \Sigma_u$, we have $\sigma \in V(s)$, by defn of V.

Combining with $s\sigma \in L(\mathbf{G})$ (by **(16)**), and by defn of $L(V/\mathbf{G})$, we have: $s\sigma \in L(V/\mathbf{G})$

$\Rightarrow s\sigma \in \overline{K}$, by **(15)**.

We thus have shown K is controllable with respect to $\mathbf{G}$.

Only if complete.

QED

Supremal Controllable Sublanguages - §3.5

- Let $\mathbf{G} = (Q, \Sigma, \delta, q_o, Q_m)$ be a controlled DES with event set $\Sigma = \Sigma_c \dot{\cup} \Sigma_u$.
- Let $E \subseteq \Sigma^*$ be an arbitrary language.
- We will consider E to be a specification language for maximal legal behaviour.
- **Defn:** We define $\mathcal{C}(E)$ to be the *set of all sublanguages of E that are controllable for $\mathbf{G}$:*

$$\mathcal{C}(E) := \{K \subseteq E | \, K \text{ is controllable with respect to } \mathbf{G}\}$$

- Can be shown that $\mathcal{C}(E)$ is a poset with respect to subset inclusion.

Existence of Supremal Controllable Sublanguage

- **Proposition 3.5.1:** $\mathcal{C}(E)$ is nonempty and is closed under arbitrary unions. In particular, $\mathcal{C}(E)$ contains a (unique) supremal element [which we denote by sup$\mathcal{C}(E)$].

  Proof:

  As $\emptyset$ is controllable, we have $\emptyset \in \mathcal{C}(E)$.

  All that remains is to show: **A)** $\mathcal{C}(E)$ is closed under arbitrary unions and **B)** $\mathcal{C}(E)$ contains a (unique) supremal element.

  A) Show $\mathcal{C}(E)$ is closed under arbitrary unions.

  Let $K := \cup_{\alpha \in A} K_\alpha$ where $K_\alpha \in \mathcal{C}(E)$, is some collection of members of $\mathcal{C}(E)$, indexed by the set A.

  Sufficient to show that $K \in \mathcal{C}(E)$

Existence of sup$\mathcal{C}(E)$ - II

Since each $K_\alpha \in \mathcal{C}(E)$ ($\alpha \in A$), it's clear that $K \subseteq E$.

Must show that K is controllable with respect to $\mathbf{G}$.

Let $s \in \overline{K}\Sigma_u \cap L(\mathbf{G})$

$\Rightarrow (\exists s' \in \overline{K})(\exists \sigma \in \Sigma_u)\, s'\sigma = s$

Sufficient to show $s'\sigma \in \overline{K}$

$s' \in \overline{K} \Rightarrow s'u \in K$ for some $u \in \Sigma^*$.

$\Rightarrow s'u \in K_\alpha$ for some $\alpha \in A$

$\Rightarrow s' \in \overline{K_\alpha}$

$\Rightarrow s'\sigma \in \overline{K_\alpha}\Sigma_u \cap L(\mathbf{G})$

$\Rightarrow s'\sigma \in \overline{K_\alpha}$ as K_α is controllable.

$\Rightarrow s'\sigma u' \in K_\alpha$ for some $u' \in \Sigma^*$

Existence of sup$\mathcal{C}(E)$ - III

$\Rightarrow s'\sigma u' \in \cup_{\alpha \in A} K_\alpha = K$

$\Rightarrow s'\sigma \in \overline{K}$, as required.

We thus have K is controllable with respect to $\mathbf{G}$.

Part A complete.

B) Show $\mathcal{C}(E)$ contains a (unique) supremal element.

Sufficient to show existence of supremal element, as uniqueness would thus follow.

Let sup$\mathcal{C}(E) := \cup\{K \,|\, K \in \mathcal{C}(E)\}$

From Part A, we have: sup$\mathcal{C}(E) \in \mathcal{C}(E)$

Clearly, $(\forall K \in \mathcal{C}(E))\, K \subseteq$ sup$\mathcal{C}(E)$

Existence of sup$\mathcal{C}(E)$ - IV

All that remains is to show:

$$(\forall K' \in \mathcal{C}(E))\ ((\forall K \in \mathcal{C}(E))\ K \subseteq K') \Rightarrow \text{sup}\mathcal{C}(E) \subseteq K'$$

Let $K' \in \mathcal{C}(E)$.

Assume $(\forall K \in \mathcal{C}(E))\ K \subseteq K'$ **(1)**

Must show implies sup$\mathcal{C}(E) \subseteq K'$

Let $s \in$ sup$\mathcal{C}(E)$. Must show implies $s \in K'$.

$s \in$ sup$\mathcal{C}(E) \Rightarrow (\exists K \in \mathcal{C}(E))\, s \in K$ by defn of sup$\mathcal{C}(E)$

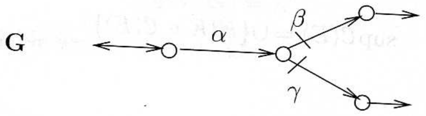
$\Rightarrow s \in K'$, by **(1)**

We thus conclude that sup$\mathcal{C}(E)$ is the supremal element.

Part B complete.
QED
We say that sup$\mathcal{C}(E)$ is maximally permissive.

Controllable Sublanguages and Intersection

- In general, $\mathcal{C}(E)$ is not closed under intersection.
- In example below, let $\Sigma = \{\alpha, \beta, \gamma\}$, $\Sigma_c = \{\beta, \gamma\}$.
- The languages $K_1 = \{\epsilon, \alpha\beta\}$ and $K_2 = \{\epsilon, \alpha\gamma\}$ are both controllable, but $K_1 \cap K_2 = \{\epsilon\}$ is not as $\alpha \notin \overline{K_1 \cap K_2}$.
- However, $\overline{K_1} \cap \overline{K_2} = \{\epsilon, \alpha\}$ is.
- As $\mathcal{C}(E)$ closed under arbitrary union but not always under intersection, it is only a complete *upper semilattice* with join operation.



Controllable Sublanguages Results

- **Proposition 3.5.2:** With respect to a fixed controlled DES $\mathbf{G}$ with alphabet Σ, the closed controllable sublanguages of an arbitrary language $E \subseteq \Sigma^*$ form a complete sublattice of the lattice of sublanguages of E.

- **Theorem 3.5.2:** Let $E \subseteq \Sigma^*$ and let $K = \text{sup}\mathcal{C}(E \cap L_m(\mathbf{G}))$. If $K \neq \emptyset$ there exists exists a MNSC V for $(K, \mathbf{G})$ such that $L_m(V/\mathbf{G}) = K$.

  Proof: By defn, $\text{sup}\mathcal{C}(E \cap L_m(\mathbf{G})) \subseteq E \cap L_m(\mathbf{G})$.

  $\Rightarrow K \subseteq L_m(\mathbf{G})$.

  As $\text{sup}\mathcal{C}(E \cap L_m(\mathbf{G}))$ by defn is controllable for $\mathbf{G}$, the results follow immediately from **Theorem 3.4.2.**

  QED

Implementation of Supervisory Controls as Automata - §3.6 and 3.7

- While a nice theoretical device, a supervisory control, V, represented as a map is not a practical implementation method.
- Instead, we want to use automata as our representations.
- Let V be a MNSC for the controlled DES $\mathbf{G} = (Q, \Sigma, \delta, q_o, Q_m)$ and language $K \subseteq L_m(\mathbf{G})$ with
$$L_m(V/\mathbf{G}) = K, \quad L(V/\mathbf{G}) = \overline{K} \qquad (\mathbf{1})$$
- Let **KDES** be a reachable automaton over Σ that represents K.
- This implies: $L_m(\textbf{KDES}) = K, \quad L(\textbf{KDES}) = \overline{K}$
- Clearly:
$$K = L_m(\textbf{KDES}) \cap L_m(\mathbf{G}), \quad \overline{K} = L(\textbf{KDES}) \cap L(\mathbf{G})$$

Implm. of SC as Automata

- With K defined as in **(1)**, let **SDES** be any DES such that

$$K = L_m(\textbf{SDES}) \cap L_m(\mathbf{G}), \quad \overline{K} = L(\textbf{SDES}) \cap L(\mathbf{G}) \qquad \mathbf{(2)}$$

- If **(2)** holds, we say **SDES** implements V.
- Note, this allows **SDES** to represent a superlanguage of $L_m(V/\mathbf{G})$ which may allow SDES to have a smaller state description.
- Possible as closed behaviour represents constraints from both V and the plant $\mathbf{G}$.

TCT Procedure supcon

- Let $\mathbf{G} = (Q, \Sigma, \delta, q_o, Q_m)$ be a controlled DES with event set $\Sigma = \Sigma_c \dot{\cup} \Sigma_u$.

- **Proposition: 3.6.1:** Let $E \subseteq \Sigma^*$ and $K :=$ sup$\mathcal{C}(E \cap L_m(\mathbf{G})) \neq \emptyset$. Let V be an MNSC such that $L_m(V/\mathbf{G}) = K$ (which exists by **Theorem 3.5.2**). Let **KDES** represent K. Then **KDES** implements V.

  Proof: Follows immediately from defn and **Theorem 3.5.2**.

- Let **EDES** represent E (ie, $L_m($**EDES**$) = E$), then:

  $$\textbf{KDES} = \text{supcon}(G, \textbf{EDES})$$

  We would thus have $L_m($**KDES**$) =$ sup$\mathcal{C}(E \cap L_m(\mathbf{G}))$ and **KDES** nonblocking.

- Note, as $L_m($**KDES**$) \subseteq L_m(\mathbf{G})$ by defn, we would thus have: **meet**$(G,$ **KDES**$) =$ **KDES**

Supervisor Results

- Let $S \subseteq \Sigma^*$ be an arbitrary language over Σ such that:

  S is controllable with respect to $\mathbf{G}$ **3a**

  $S \cap L_m(\mathbf{G}) \neq \emptyset$ **3b**

  $\overline{S \cap L_m(\mathbf{G})} = \overline{S} \cap L(\mathbf{G})$ **3c**

- **Proposition: 3.6.2:** Let **SDES** be any nonblocking DES over Σ such that $S := L_m(\mathbf{SDES})$ satisfies conditions **(3a)** and **(3c)**.

  Let $\emptyset \neq K := S \cap L_m(\mathbf{G})$ and let V be an MNSC such that $L_m(V/\mathbf{G}) = K$. Then **SDES** implements V. In particular,

  $$L_m(V/\mathbf{G}) = L_m(\mathbf{SDES}) \cap L_m(\mathbf{G}), \;\; L(V/\mathbf{G}) = L(\mathbf{SDES}) \cap L(\mathbf{G})$$

Supervisor Definitions

- In other words, the closed loop behavior of $\mathbf{G}$ under the control of V as per **proposition 3.6.2** is exactly equal to: **meet**$(G,$ **SDES**$)$.

- **Defn:** if points **3a** and **3c** are satisfied and **SDES** represents S, we say that **SDES** is a supervisor for $\mathbf{G}$. We also include under this definition the trivial case where $L_m(\mathbf{SDES}) \cap L_m(\mathbf{G}) = \emptyset$.

- **Defn:** If **SDES** represents S and S is controllable with respect to $\mathbf{G}$, we say that **SDES** is controllable with respect to $\mathbf{G}$.

- **Defn:** We say that **SDES** is a proper supervisor for $\mathbf{G}$ if:

  (i) **SDES** is trim (reachable and coreachable);

  (ii) **SDES** is controllable with respect to$\mathbf{G}$;

  (iii) $\overline{L_m(\mathbf{SDES}) \cap L_m(\mathbf{G})} = L(\mathbf{SDES}) \cap L(\mathbf{G})$

Supervisor Interpretation

- Let $\mathbf{S} = (X, \Sigma, \xi, x_o, X_m)$ implement V.
- We can consider $\mathbf{S}$ to be a state machine that accepts the sequence of symbols Σ generated by $\mathbf{G} = (Q, \Sigma, \delta, q_o, Q_m)$ as “forcing inputs” causing $\mathbf{S}$ to change state as per ξ, and thus track the behavior of $\mathbf{G}$.
- Control action is asserted on $\mathbf{G}$ as a state-output map $\psi : X \rightarrow \text{Pwr}(\Sigma)$ defined as:

$$\psi(x) := \{\sigma \in \Sigma |\, \xi(x, \sigma)!\}, \quad \text{for } x \in X$$

- Can imagine that while $\mathbf{S}$ is in state $x \in X$, it disables in $\mathbf{G}$ all controllable events that are not in $\psi(x)$.
- In other words, if $\mathbf{S}$ is in state $x \in X$ and $\mathbf{G}$ is in state $q \in Q$, then the next allowable events are:

$$\{\sigma \in \Sigma |\, \xi(x, \sigma)! \,\&\, \delta(q, \sigma)!\}$$

Supervisor Interpretation - II

- The closed loop behavior is denoted $\mathbf{S}/\mathbf{G}$ and defined to be:

$$\mathbf{S}/\mathbf{G} := (X \times Q, \Sigma, \xi \times \delta, (x_o, q_o), X_m \times Q_m)$$

where:

$$\xi \times \delta : X \times Q \times \Sigma \rightarrow X \times Q : (x, q, \sigma) \mapsto (\xi(x, \sigma), \delta(q, \sigma))$$

as long as $\xi(x, \sigma)!$ and $\delta(q, \sigma)!$

- In other words, $\mathbf{S}/\mathbf{G} = \textbf{meet}(G, \mathbf{S})$ would be equivalent with respect to language.

- **Note (1) :** Some software assumes everything combined using $||$ operator.

- **Note (2) :** $\mathbf{S}/\mathbf{G}$ represents the expected closed loop behavior. If $\mathbf{S}$ is NOT controllable for $\mathbf{G}$, then that means that the actual possible behavior is a super language of $L(\mathbf{S}/\mathbf{G})$.

TCT Procedure condat

- To check that $\mathbf{S}$ is controllable for $\mathbf{G}$, we use the TCT function **condat**.

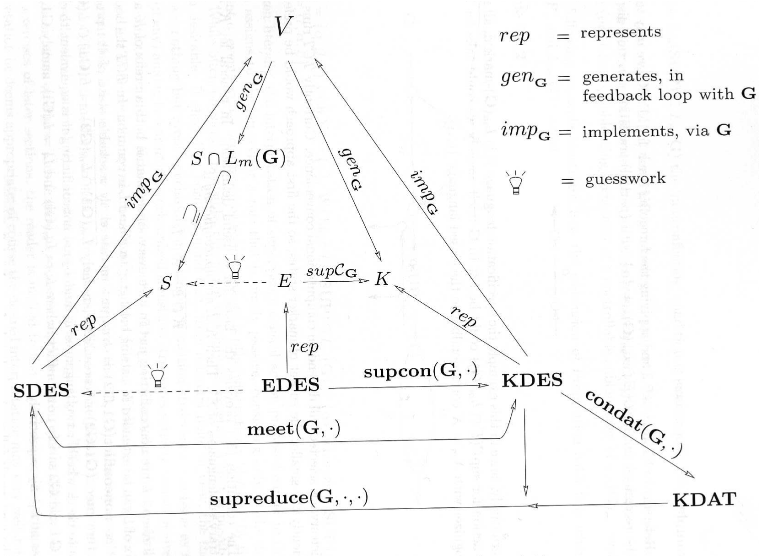
$$\text{SDAT} := \textbf{condat}(\mathbf{G}, \mathbf{S})$$

- The output of **condat** is a ".dat" file that tabulates all the states of $\mathbf{S}$ where event disablement occurs.

- If only controllable events (odd numbers) are listed, then $L(\mathbf{S})$ is controllable for $\mathbf{G}$.

- Actually, **condat** only checks that $L(\mathbf{S})\Sigma_u \cap L(\mathbf{G}) \subseteq L(\mathbf{S})$, and doesn't care if $\mathbf{S}$ is nonblocking.

Implementation Scheme



V
$gen_{\mathbf{G}}$
$imp_{\mathbf{G}}$
$S \cap L_m(\mathbf{G})$
$\cap$
S
E
$sup\mathcal{C}_{\mathbf{G}}$
K
rep
SDES
EDES
$\mathbf{supcon}(\mathbf{G}, \cdot)$
KDES
$\mathbf{condat}(\mathbf{G}, \cdot)$
$\mathbf{meet}(\mathbf{G}, \cdot)$
$\mathbf{supreduce}(\mathbf{G}, \cdot, \cdot)$
KDAT
rep = represents
$gen_{\mathbf{G}}$ = generates, in feedback loop with $\mathbf{G}$
$imp_{\mathbf{G}}$ = implements, via $\mathbf{G}$
= guesswork

Fig 3.6.1. Scheme for supervisory control implementation

Practical Considerations

- The text talks about $\mathbf{S}$ being a proper supervisor for $\mathbf{G}$ and verifying that $L_m(\mathbf{S})$ and $L_m(\mathbf{G})$ are nonconflicting.
- This would require performing the additional checks that $\mathbf{S}$ and $\mathbf{G}$ are trim.
- In practice, we use **condat** to verify that $L(\mathbf{S})\Sigma_u \cap L(\mathbf{G}) \subseteq L(\mathbf{S})$, and we use **nonconflict** function to verify that **meet**$(G, \mathbf{S})$ is nonblocking. If so, then we consider $\mathbf{S}$ a valid supervisor.
- We don't actually care if $\mathbf{S}$ and $\mathbf{G}$ are trim as the final result is the same.
- When people say that DES $\mathbf{S}$ is controllable for $\mathbf{G}$, what they usually mean is that $L(\mathbf{S})\Sigma_u \cap L(\mathbf{G}) \subseteq L(\mathbf{S})$, not that $\mathbf{S}$ is nonblocking and $L_m(\mathbf{S})$ is controllable for $\mathbf{G}$.

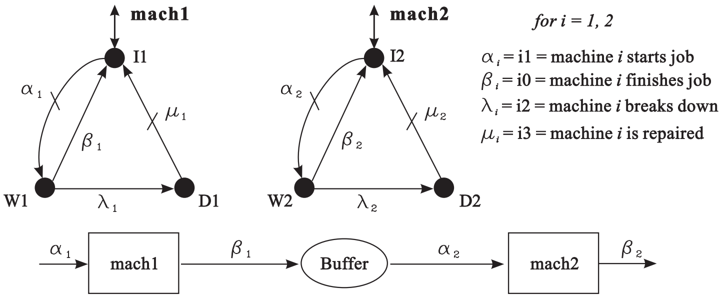
Small Factory Example

- Our plant is: $\mathbf{FACT} = \mathbf{sync}(\text{mach1}, \text{mach2})$
- **Specifications**

  For buffer with only one storage slot:

  1) Buffer must not overflow or underflow.

  2) If both machines down, machine 2 must be repaired first.

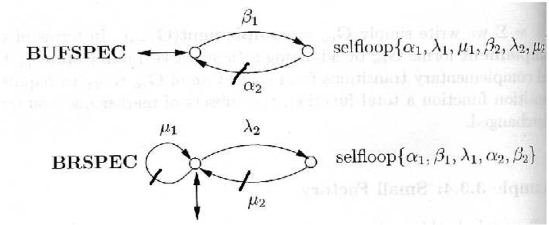


for i = 1, 2

α_i = i1 = machine *i* starts job
β_i = i0 = machine *i* finishes job
λ_i = i2 = machine *i* breaks down
μ_i = i3 = machine *i* is repaired

Small Factory: Synthesis

- We start by designing specifications that can be used to synthesize a supervisor using **supcon**.
- DES **BUFSPEC** represents the desired behavior to protect the buffer.
- DES **BRSPEC** represents the desired behavior for repair preference.


β_1
BUFSPEC
selfloop$\{\alpha_1, \lambda_1, \mu_1, \beta_2, \lambda_2, \mu_2$
α_2
μ_1
λ_2
BRSPEC
selfloop$\{\alpha_1, \beta_1, \lambda_1, \alpha_2, \beta_2\}$
μ_2

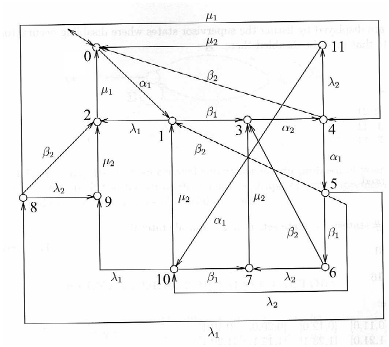
- Combine the two into one specification:

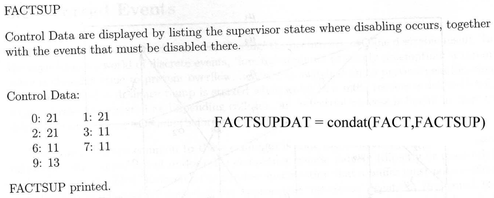
$$\textbf{SPEC} = \textbf{meet}(\mathbf{BUFSPEC}, \mathbf{BRSPEC})$$

Small Factory: Synthesis - II

- Use supcon to create supremal controllable nonblocking supervisor:

$$\textbf{FACTSUP} = \textbf{supcon}(\mathbf{FACT}, \mathbf{SPEC})$$




FACTSUP

Control Data are displayed by listing the supervisor states where disabling occurs, together with the events that must be disabled there.

Control Data:

```
0: 21    1: 21
2: 21    3: 11
6: 11    7: 11
9: 13
```

FACTSUP printed.

FACTSUPDAT = condat(FACT,FACTSUP)

Small Factory: SIMFTSUP

- The other option is to design a supervisor by hand and then verify that it is controllable and that the closed loop system is nonblocking.

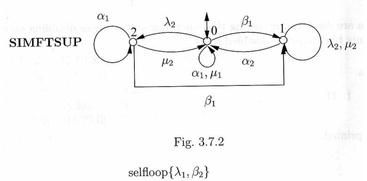


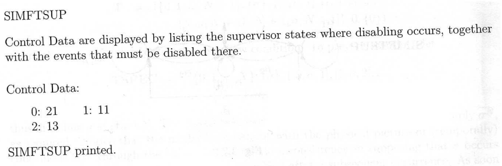
Fig. 3.7.2

selfloop$\{\lambda_1, \beta_2\}$

- We design **SIMFTSUP** using our knowledge of the system and design experience.

- To check controllability, do:

$$\textbf{SIMFTSUPDAT} = \textbf{condat(FACT,SIMFTSUP)}$$



SIMFTSUP

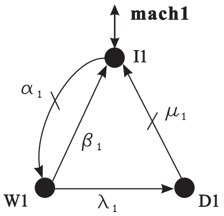
Control Data are displayed by listing the supervisor states where disabling occurs, together with the events that must be disabled there.

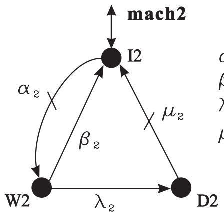
Control Data:

```
    0: 21      1: 11
    2: 13
```

SIMFTSUP printed.

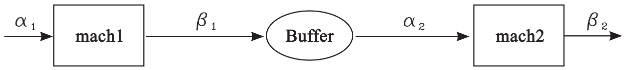
Small Factory and SIMFTSUP





for i = 1, 2

α_i = i1 = machine *i* starts job
β_i = i0 = machine *i* finishes job
λ_i = i2 = machine *i* breaks down
μ_i = i3 = machine *i* is repaired



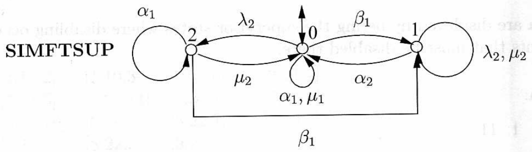


Fig. 3.7.2

selfloop$\{\lambda_1, \beta_2\}$

Small Factory: SIMFTSUP - II

- We next check to see if **SIMFTSUP**/**FACT** is nonblocking:

$$\textbf{nonconflict}(\textbf{FACT}, \textbf{SIMFTSUP}) = \text{TRUE}$$

- Now, we want to know if **SIMFTSUP** is optimal.
- We consider two supervisors to be equivalent for a given plant, if they produce the same closed loop behavior.
- If **SIMFTSUP** has same closed loop behavior as **FACTSUP**, then its optimal.
- To do this, we first need to compute the closed loop behavior for **SIMFTSUP**:

$$\textbf{SIMFTSUP}/\textbf{FACT} = \textbf{meet}(\textbf{FACT}, \textbf{SIMFTSUP})$$

- By construction, **FACTSUP** is its own closed loop behavior.

Small Factory: SIMFTSUP - III

- To use TCT's **isomorph** function, we have to make sure they are both minimal first.

$$\begin{aligned} \mathbf{MFACTSUP} &:= \mathbf{minstate}(\mathbf{FACTSUP}) \\ \mathbf{MCLSIMFTSUP} &:= \mathbf{minstate}(\mathbf{SIMFTSUP}/\mathbf{FACT}) \end{aligned}$$

- We now check that they are isomorphic:

$$\mathbf{isomorph}(\mathbf{MFACTSUP}, \mathbf{MCLSIMFTSUP}) = \text{TRUE}$$

- Thus **SIMFTSUP** (3 states, 16 trans) is optimal and provides the same control action with respect to **FACT**, as **FACTSUP** (12 states, 24 trans).

Controllability Algorithm

- Assume we are given plant $\mathbf{G} = (Q, \Sigma, \delta, q_o, Q_m)$, supervisor $\mathbf{S} = (X, \Sigma, \xi, x_o, X_m)$ and we want to determine if $\mathbf{S}$ is controllable for $\mathbf{G}$.
- How to check this?
- Remember that the cells of a generator represent the cells of $\equiv_c \wedge \equiv_m$.
- To check that $\mathbf{S}$ is controllable for $\mathbf{G}$, we need to check that $L(\mathbf{S})\Sigma_u \cap L(\mathbf{G}) \subseteq L(\mathbf{S})$.
- For $s \in L(\mathbf{S}) \cap L(\mathbf{G})$, let $[s] = x \in X, x = \xi(x_o, s)$.
- Let $\sigma \in \Sigma_u$ such that $s\sigma \in L(\mathbf{G})$.

Controllability Algorithm - II

- Clearly, $s\sigma \in L(\mathbf{S}) \Leftrightarrow (\forall s' \in [s])\, s'\sigma \in L(\mathbf{S})$, by definition of Nerode equivalence relation.
- Also, $s\sigma \in L(\mathbf{S}) \Leftrightarrow \xi(x, \sigma)!$ by definition of $L(\mathbf{S})$.
- This means we can convert a test on strings, to checks on the states of the DES.

Define Variables:

Let $y_o = (q_o, x_o)$.

$Y_{\mathsf{fnd}} = \emptyset$ // states reached by algorithm

$Y_{\mathsf{pend}} = \emptyset$ // states waiting to be examined

push(y_o, Y_{fnd})
push(y_o, Y_{pend})

Controllability Algorithm - III

```
while $(Y_{pend} \neq \emptyset)$ {
    $(q, x) = \mathsf{pop}(Y_{\mathsf{pend}})$
    for $\sigma$ in $\Sigma$ {
        if $(\delta(q, \sigma)!)$ then {
          if $\neg(\xi(x, \sigma)!)$ then {
              if $(\sigma \in \Sigma_u)$ then {
                return "uncontrollable at state" $(q, x)$, "for" $\sigma$
              }
          else {
              $y' = (\delta(q, \sigma), \xi(x, \sigma))$
              if $(y' \notin Y_{\mathsf{fnd}})$ then {
                push$(y', Y_{\mathsf{fnd}})$
                push$(y', Y_{\mathsf{pend}})$
              }
          }  // end else
        }  // end if
    } } // end for, then end while
```

Supcon Algorithm Overview

Assume we are given plant $\mathbf{G} = (Q, \Sigma, \delta, q_o, Q_m)$, specification $\mathbf{E} = (X, \Sigma, \xi, x_o, X_m)$, and we want to construct the supremal controllable nonblocking supervisor for $\mathbf{E}$.

1) Construct tmpSup := **meet**$(\mathbf{G}, \mathbf{E})$ using the **meet algorithm.**

2) While constructing the **meet**, use the controllability algorithm to determine which states, Y_{fail}, are uncontrollable.

3) Trim all states $y \in Y_{\mathsf{fail}}$ in a controllable manner as follows:

3a) Use δ^{-1} to determine which states have transitions leading to y.

3b) Delete all controllable transitions that lead to y.

3c) For all uncontrollable transitions leading to y, add the originating state y' ($\delta(y', \sigma) = y$ for some $y' \in Y$ and $\sigma \in \Sigma_u$) to Y_{fail}, and then delete the transition.

Supcon Algorithm Overview - II

3d) Delete y.

4) Use nonblocking algorithm to determine all remaining states that are not coreachable. Add them to Y_{fail}.

5) If $Y_{\mathsf{fail}} \neq \emptyset$, go to step 3.

6) Set **SupCont** $:=$ tmpSup. Stop

=====================

Read Sections 3.9 and 3.10 on own.