From Linear Systems to Discrete-Event Systems

W.M. Wonham
Systems Control Group
ECE Department
University of Toronto

2012.03.22

This talk is about

CYBERNETICS

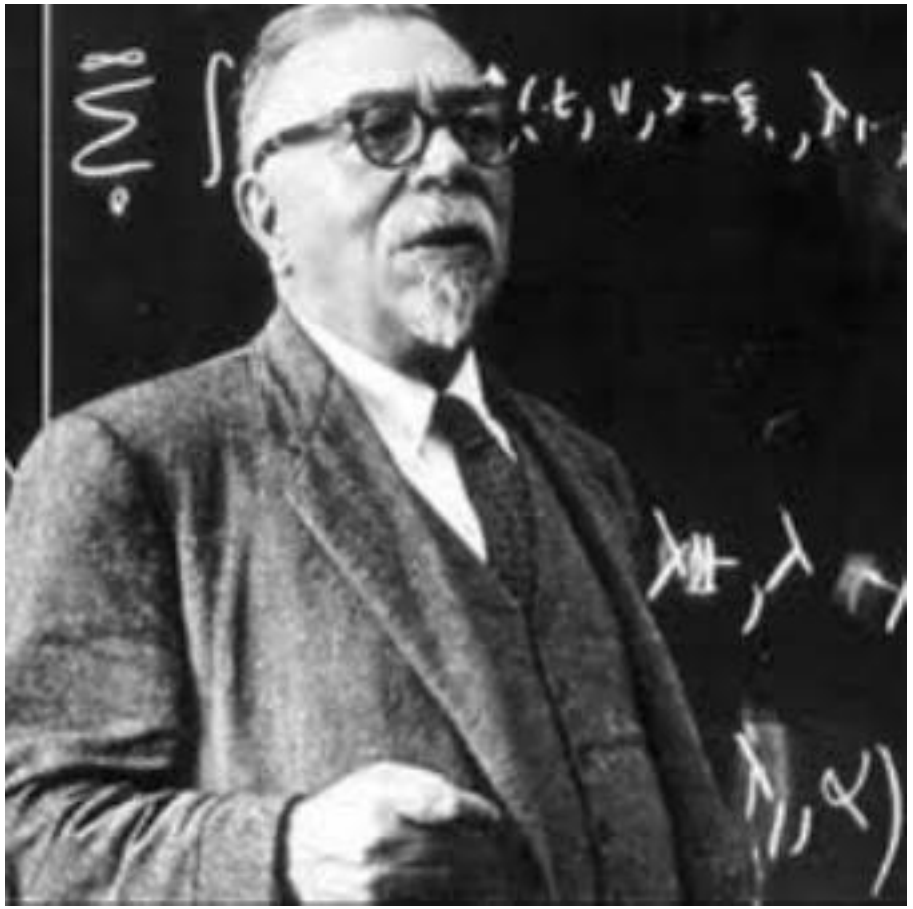
whatever that is …

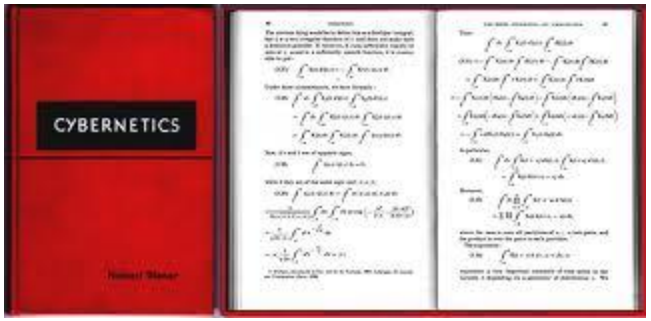
“KYBERNETES” = “STEERSMAN” (= “GOVERNOR”)

Homer’s *Iliad* and *Odyssey* (BCE 750)

“CYBERNETICS” ENTERS MATHEMATICAL SCIENCE

Norbert Wiener (1894-1964)





1948

Cybernetics: Control and Communication in the Animal and the Machine.

Theme: The universality of ***feedback*** in technology, physiology, psychology, sociology, economics …; together with the concomitant problems of ***stability***, ***noise filtering***, and ***prediction***.

“CYBERNETICS” ENTERS ENGINEERING SCIENCE

H.S. Tsien (Qian Xuesen, 1911-2009)

ENGINEERING CYBERNETICS

H. S. TSIEN

Daniel and Florence Guggenheim Jet Propulsion Center
California Institute of Technology
Pasadena, California

1954

“[E]ngineering cybernetics is an engineering *science* [which] aims to organize the design principles used in engineering practice into a discipline and thus … to emphasize the power of fundamental concepts.”

What is a Discrete-Event System?

- Structure with ‘states’ having duration in time, ‘events’ happening instantaneously and asynchronously.
- States: machine is idle, is operating, is broken down, is under repair.
- Events: machine starts work, completes work, breaks down, or completes repair.
- State space discrete and usually finite.
- State transitions ‘identified’ with events.

Summary

- Some history
- Supervisory Control Theory (SCT)
- Large systems (using IDDs)
- Hierarchy
- Extensions and Applications
- Conclusions

Discrete-Event Systems (c. 1980)

- Practical problems –
  inventory, traffic, logistics …
- Programming languages
  for modeling & simulation
- Queues, Markov chains, Petri nets
- Synchronization (semaphores, path
  expressions …)
- Process algebras (CSP, CCS)

Discrete-Event Systems Control ? (c.1980)

- Control problems ***implicit*** in the literature (enforcement of resource constraints, synchronization, ...)

But

- Emphasis on modeling, simulation, performance measurement, verification
- Little formalization of control synthesis
- Absence of control-theoretic ideas
- No standard model or approach to control

Systems Control Concepts (c. 1980)

- State space framework well-established:
  - Stability
  - Controllability
  - Observability
  - Optimality (Quadratic, $L_{various}$, H_{∞})
- Qualitative synthesis via
  controlled dynamic invariants
- Use of geometric constructs and
  partial order:
  - Controllability subspaces (c.s.)
    - supremal subspaces!

Needed (1980): DES Control Theory

- System model
  - Discrete in time and (usually) space
  - Asynchronous (event-driven)
  - Nondeterministic
    - support transitional choices
- Amenable to formal control synthesis
  - exploit control concepts
- Applicable: manufacturing, traffic, logistic,...

Proposed (1982): Supervisory Control Theory

(*Peter Ramadge & WMW*)

- Automaton representation
  - ***internal*** state descriptions for concrete modeling and computation
- Language representation
  - ***external*** i/o descriptions for implementation-independent concept formulation
- Simple control ‘technology’

Community Response

Anonymous Referees (1983-87)

- ***[Leading control journal]***

  "Automata have no place in control engineering."

  Reject!

- ***[Leading computer journal]***

  "Finite automata and regular languages are nothing new at best and trivial at worst."

  Reject!

- ***SIAM J. Control & Optimization***

  "So this is optimal control? Well..." **Accept**

Summary

- Some history
- Supervisory Control Theory (SCT)
- Large systems (using IDDs)
- Hierarchy
- Extensions and Applications
- Conclusions

FROM ‘STANDARD’ CONTROL TO SUPERVISORY CONTROL

- Standard dynamics: $dx/dt = f(t,x,u)$

$\rightarrow$

Supervisory control dynamics: automaton with labeled transitions (*events*), some of which are *controllable*

- Standard output: $y(t) = G[x(s),u(s)|s \leq t]$

$\rightarrow$

Supervisory control output: sequence of transition labels = *string* in a *language*

“AUTOMATON” = “SELF-MOVER”

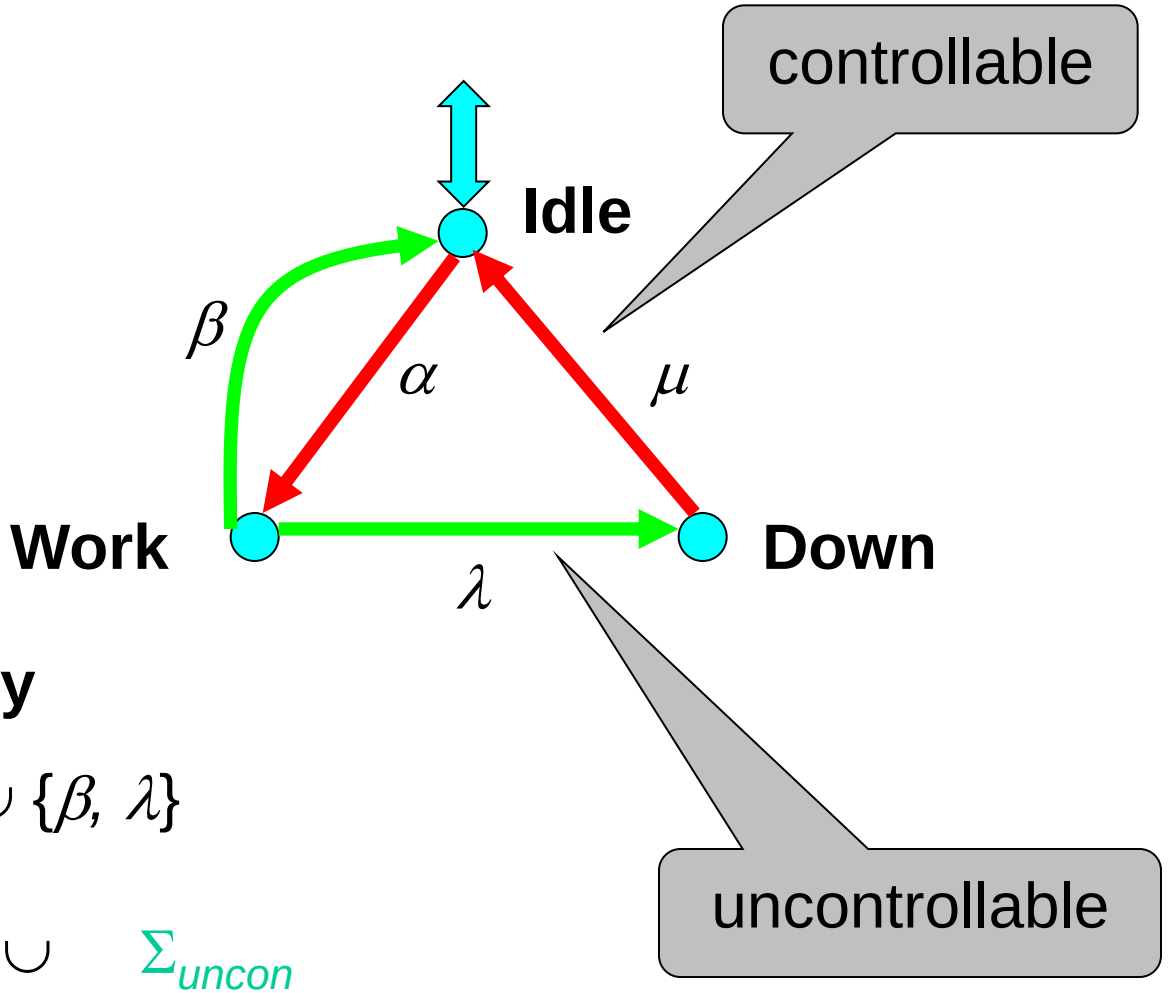
Homer’s *Iliad* - 18, lines 373-377

Twenty tripods [Hephaistos] crafted, to stand around … his house. At the base of each he placed golden wheels, so these self-movers [*hoi automatoi*] might enter the divine assembly, and return back to the house, a wonder to behold!

SCT Base Model

- **Automaton**

MACH



- **Control Technology**

$$\Sigma = \{\alpha, \mu\} \cup \{\beta, \lambda\}$$

$$= \Sigma_{con} \cup \Sigma_{uncon}$$

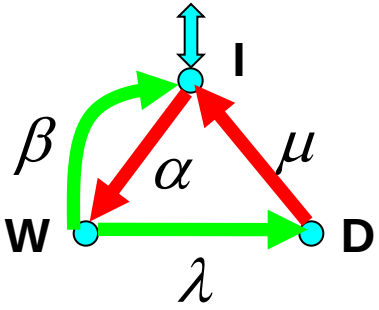
SCT Languages

- ***Closed and Marked Behaviors***

L(**MACH**) = all strings generable from initial state **I**

= $\{\varepsilon,\ \alpha,\ \alpha\beta,\ \alpha\lambda,\ \alpha\beta\alpha,\ \alpha\lambda\mu,\ \ldots\}$

= *closed behavior* of **MACH**



L_m(**MACH**) = all generable strings hitting some marker state

= $\{\varepsilon,\ \alpha\beta,\ \alpha\lambda\mu,\ \ldots\}$

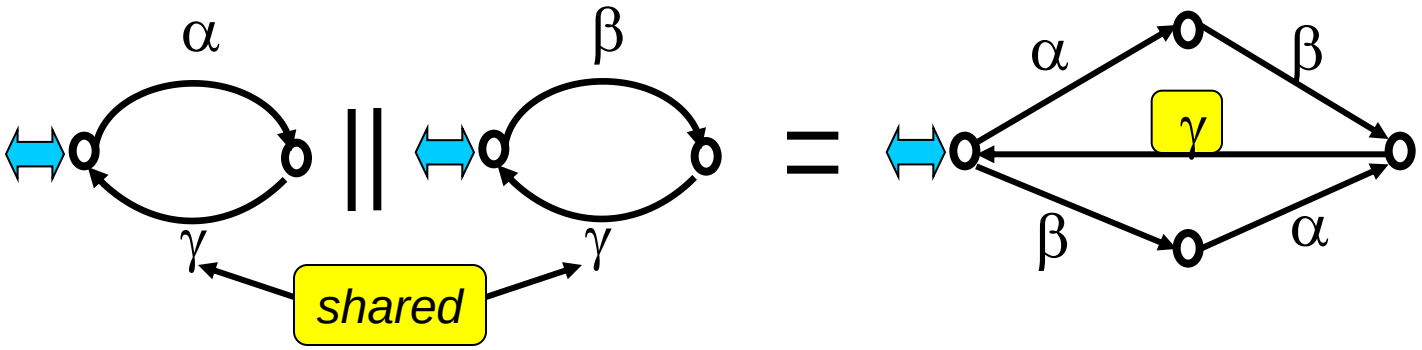
= *marked behavior* of **MACH**

prefix closure

- ***Liveness (Nonblocking):*** L(**MACH**) = $\overline{L_m\,(\textbf{MACH})}$

Synchronous Product

- Builds a more complex automaton



with more complex language

$$L(A_1 \parallel A_2) = \boldsymbol{P_1^{-1}} L(A_1) \cap \boldsymbol{P_2^{-1}} L(A_2)$$

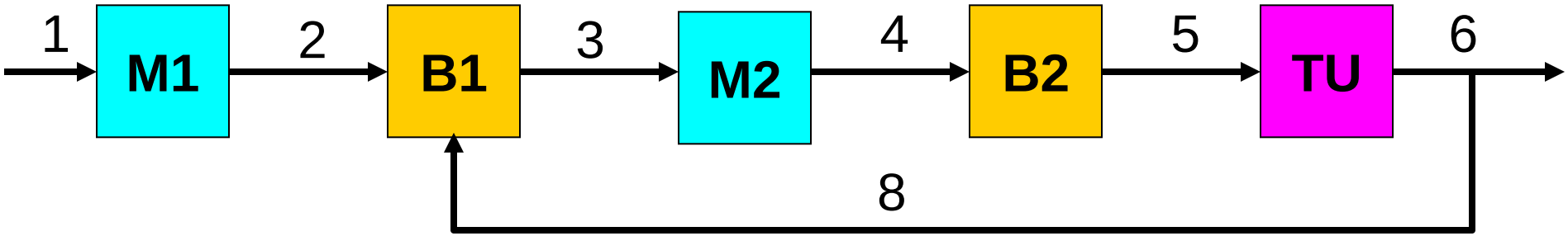
expressed by natural projections

$$\boldsymbol{P_i}: (\Sigma_1 \cup \Sigma_2)^* \rightarrow \Sigma_i^* \quad (i = 1,2)$$

SCT Complex Plant

- ***Complex*** plant
  = sync product of ***simple*** subplants

Transfer Line **TL** (*Al-Jaar & Desrochers*)

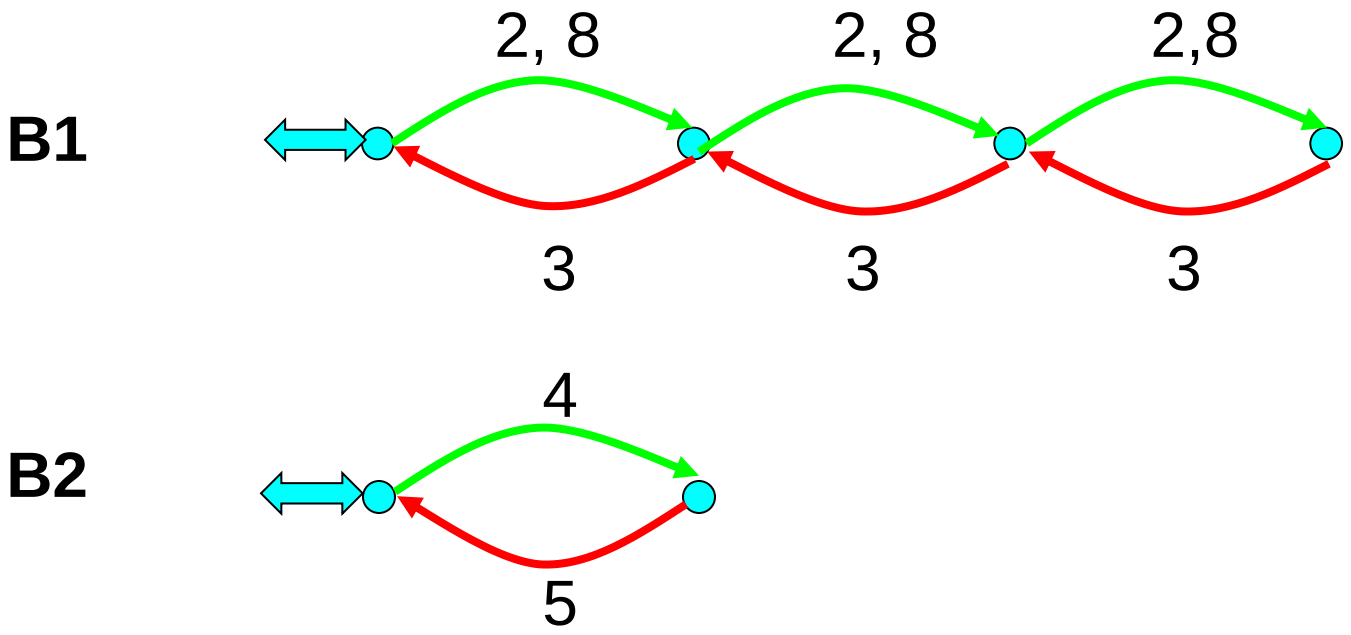


TL = M1 || M2 || TU

SCT Complex (Safety) Specification

- ***Complex*** specification

  = sync product of ***partial*** specifications



BUFFSPEC = B1 || B2

General Control Issues

- Is there a control that enforces both **safety**, and **liveness (nonblocking)**, and which is **maximally permissive** ?
- If so, can its design be **automated** ?
- If so, with **acceptable computing effort** ?

SCT Synthesis - Problem

E.g. for **TL**, let **ConTL** = '**TL** under control'
Must guarantee

1. ***Safety*:**

   $L_m(\mathbf{ConTL}) \subseteq L_m(\mathbf{BUFFSPEC})$

2. ***Liveness (nonblocking):***

   $\overline{L_m(\mathbf{ConTL})} = L(\mathbf{ConTL})$

3. ***Maximal permissiveness*:**

   $L_m(\mathbf{ConTL})$ = maximum

   subject to safety and liveness

SCT Synthesis - Solution

E.g. for **TL**:

1. Fundamental **definition**

   A sublanguage $K \subseteq L_m(\mathbf{TL})$ is ***controllable*** if

   $$\overline{K}\,\Sigma_{uncon} \cap L(\mathbf{TL}) \subseteq \overline{K}$$

   "Once in $\overline{K}$, you can't skid out on an uncontrollable event."

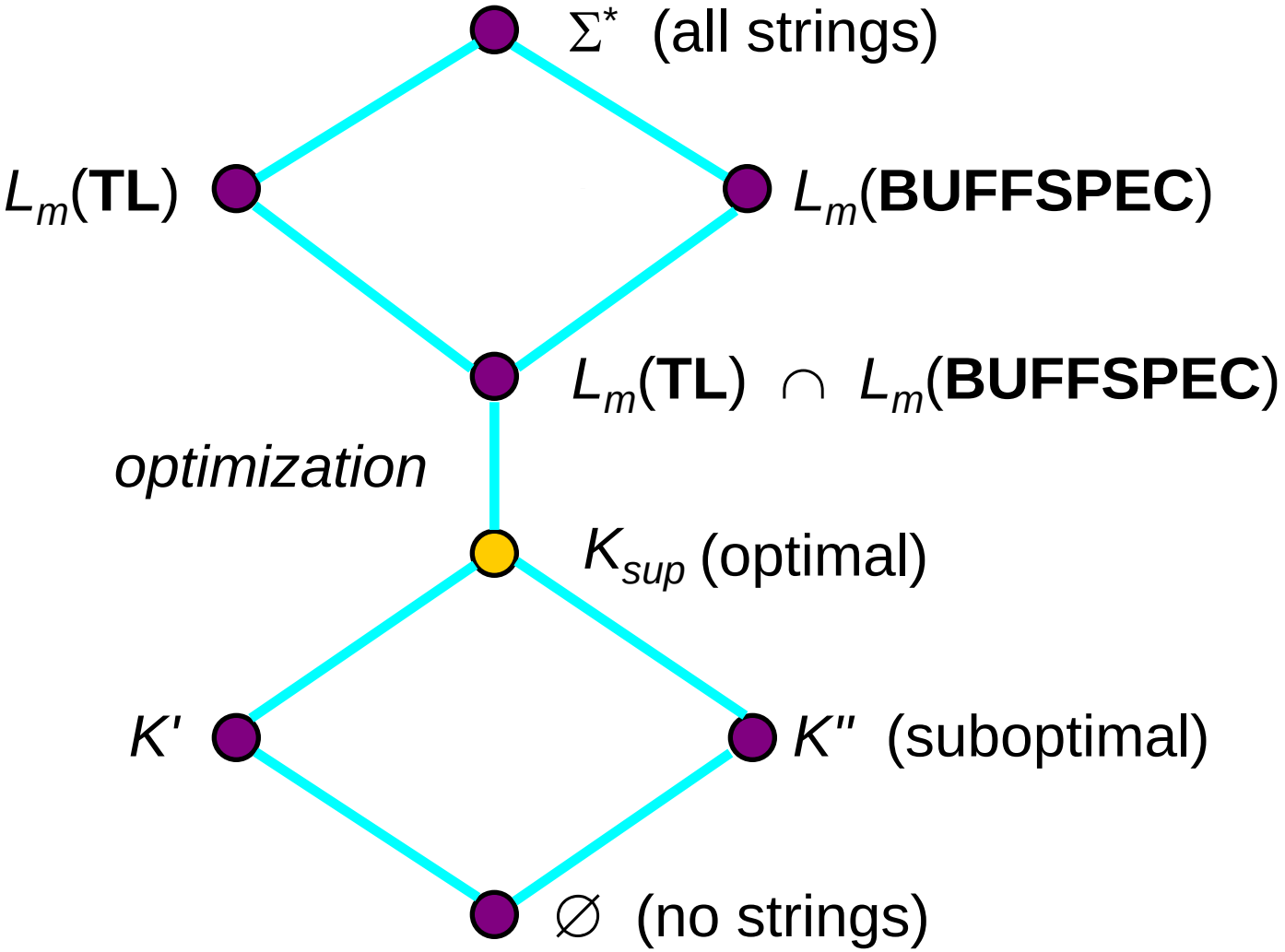
2. Fundamental **result**

   There exists a (unique) ***supremal*** controllable sublanguage

   $$K_{sup} \subseteq L_m(\mathbf{TL}) \cap L_m(\mathbf{BUFFSPEC})$$

   Furthermore K_{sup} can be effectively computed.

SCT Synthesis Lattice



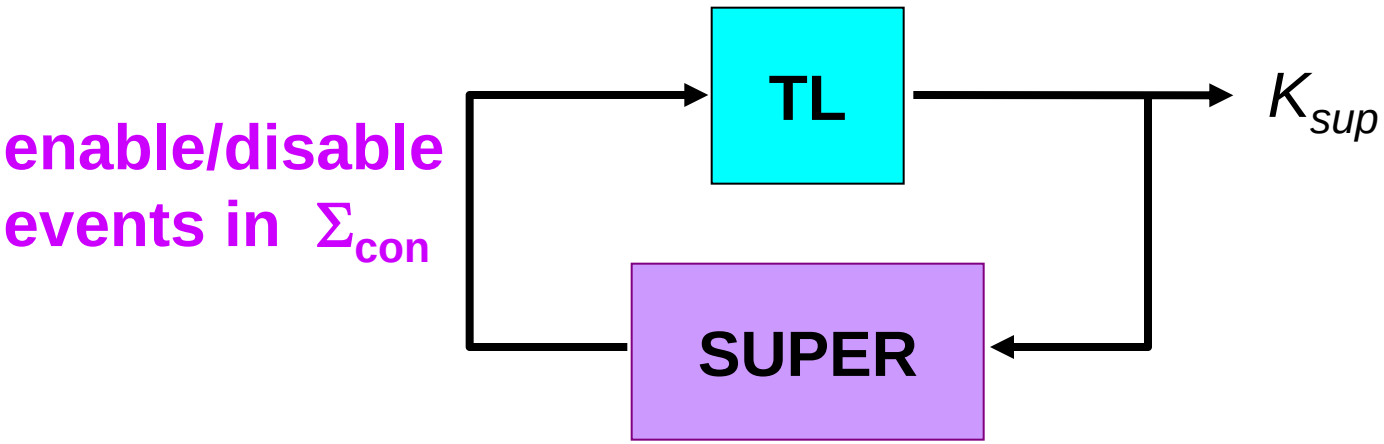
‘Monolithic’ SCT Implementation

- Given **TL** and **BUFFSPEC**, compute K_{sup}

$$K_{sup} = L_m(\textbf{SUPER})$$

$$\textbf{SUPER} = \textbf{supcon}\ (\textbf{TL}, \textbf{BUFFSPEC})$$

- Given **SUPER,** implement K_{sup}



Summary

- Some history
- Supervisory Control Theory (SCT)
- Large systems (using IDDs)
- Hierarchy
- Extensions and Applications
- Conclusions

Large DES

PLANT = **sync** (**PLANT.1**, … , **PLANT.m**)

SPEC = **sync** (**SPEC.1**, … , **SPEC.n**)

SUPER = **supcon** (**PLANT**, **SPEC**)

State size of **SUPER** ~ (*Constant*) $^{m+n}$

Exponential state space explosion !

‘**Extensional**’ listing of ‘flat’ transition structures is **impossible !**

What To Do ?

- In state representations,

  retain product structure

  PLANT state vector **x** = $[x_1, \ldots, x_m]$

  SPEC state vector **y** = $[y_1, \ldots, y_n]$

- Express **SUPER** as a *predicate*

  $\text{Pred}_{\text{sup}}(\mathbf{x}, \mathbf{y}, \sigma, \mathbf{x}', \mathbf{y}') = 0 \text{ or } 1$

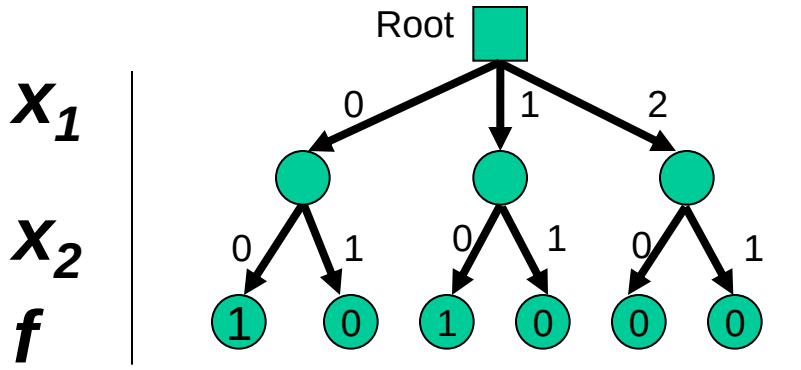
- Algorithmize representation of Pred_{sup} using *Integer Decision Diagrams (IDDs)*

Integer Decision Diagrams (IDDs)

- IDDs represent functions on finite sets

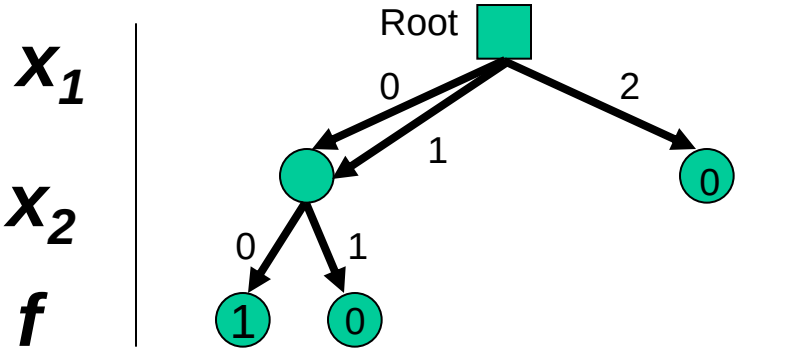
x_1	x_2	f
0	0	1
0	1	0
1	0	1
1	1	0
2	0	0
2	1	0

Order!



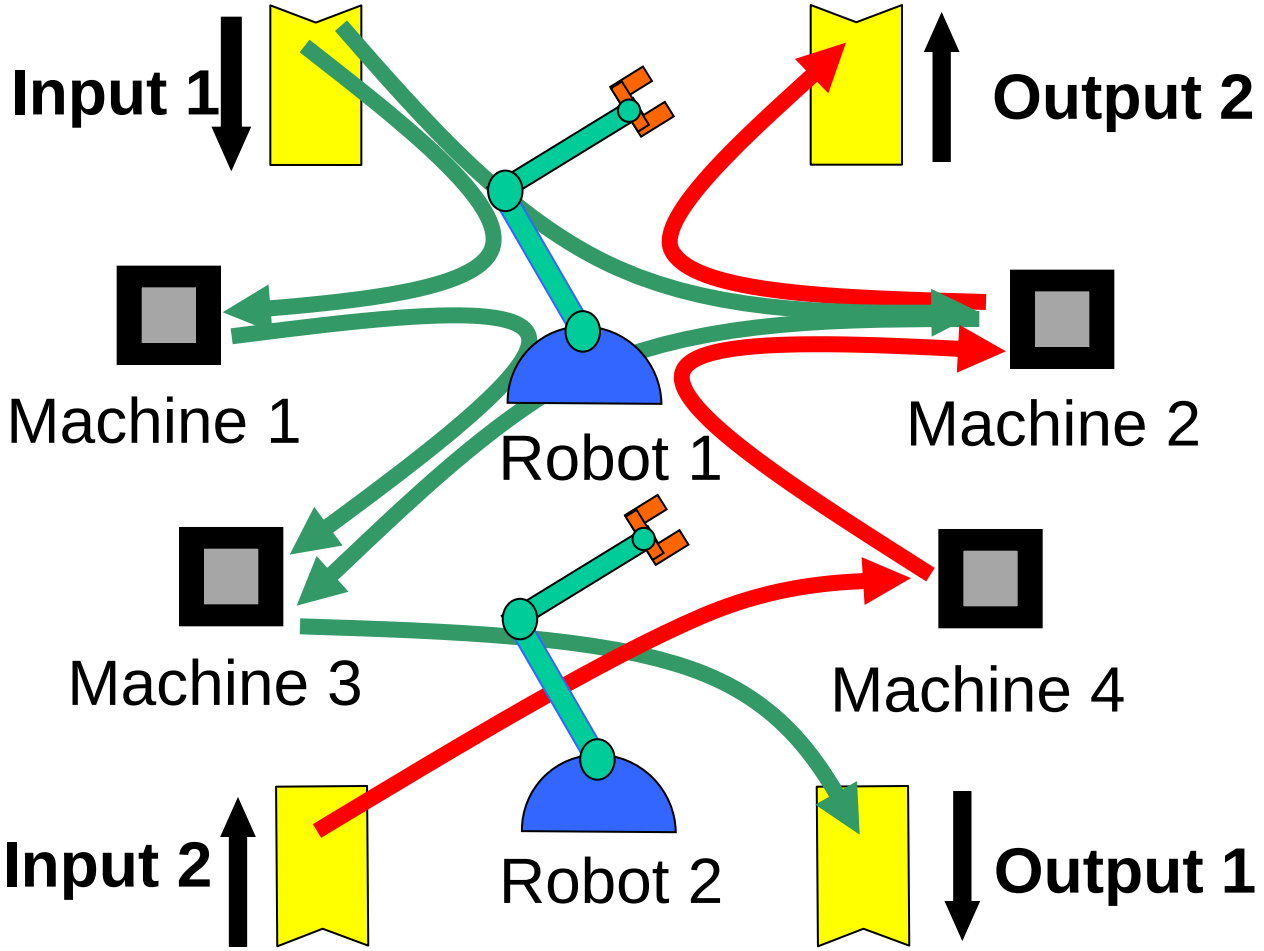
Reduce!

IDD



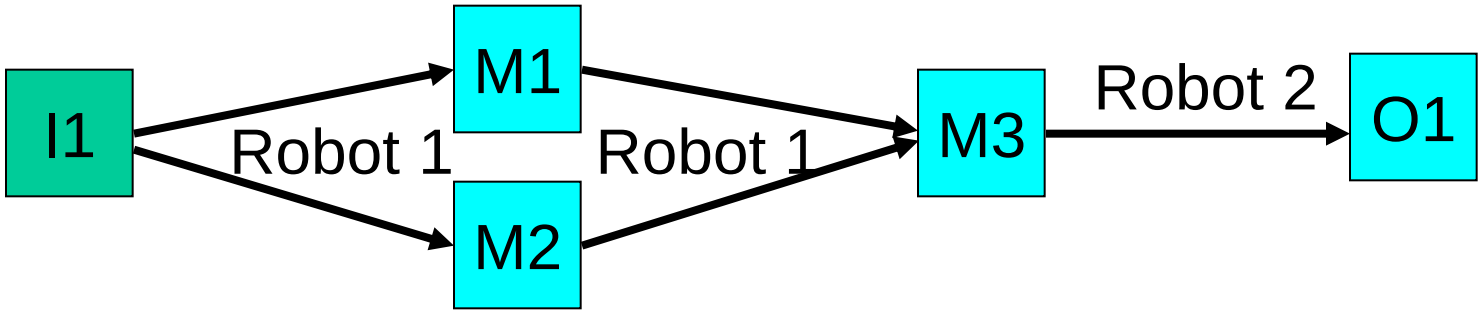
Manufacturing Workcell

(*Barkaoui & Ben Abdallah 1995, Seidl 2000*)

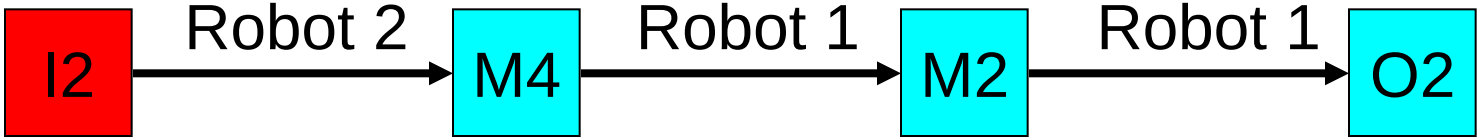


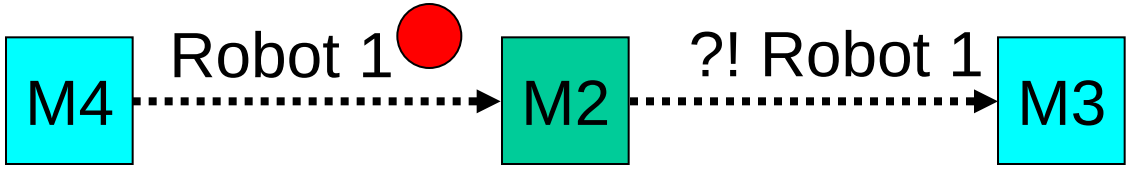
Workcell Control Issues

Green Production Sequence (‘***safety***’ specification)



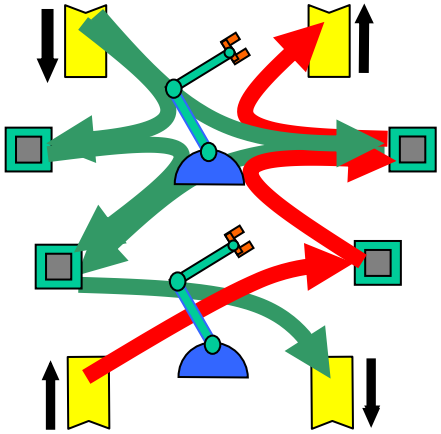
Red Production Sequence (‘***safety***’ specification)





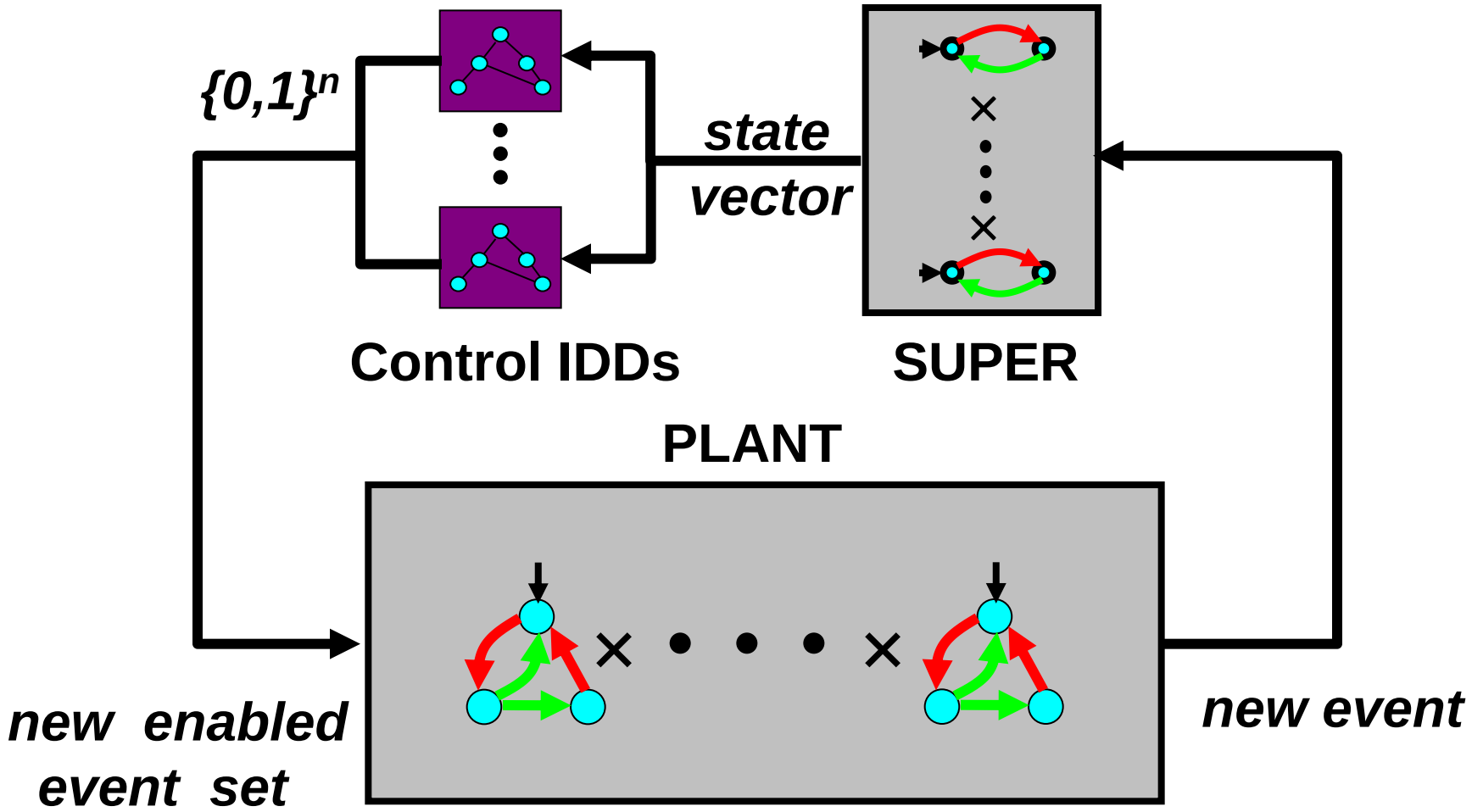
Blocking! (prohibit by ***nonblocking*** ‘***liveness***’ spec’n)

IDD Results: Workcell



K	State size	Node count	Time (sec)	Mem (MB)	Condat (KB)
1	205	77	1	1.0	1
4	1.9×10^{6}	194	2	1.6	3
10	5.8×10^{9}	620	10	2.9	19
30	3.4×10^{14}	3,600	201	11.	281
50	7.4×10^{16}	8,980	1,382	30.	1,123

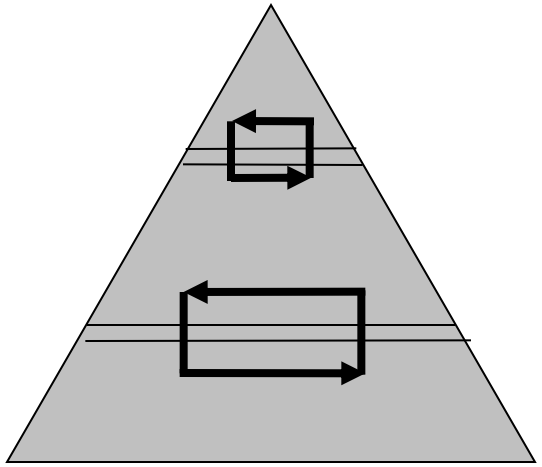
Supervisor Implementation



Summary

- Some history
- Supervisory Control Theory (SCT)
- Large systems (using IDDs)
- Hierarchy
- Extensions and Applications
- Conclusions

Architecture: Hierarchical Layering



Manager (slow)

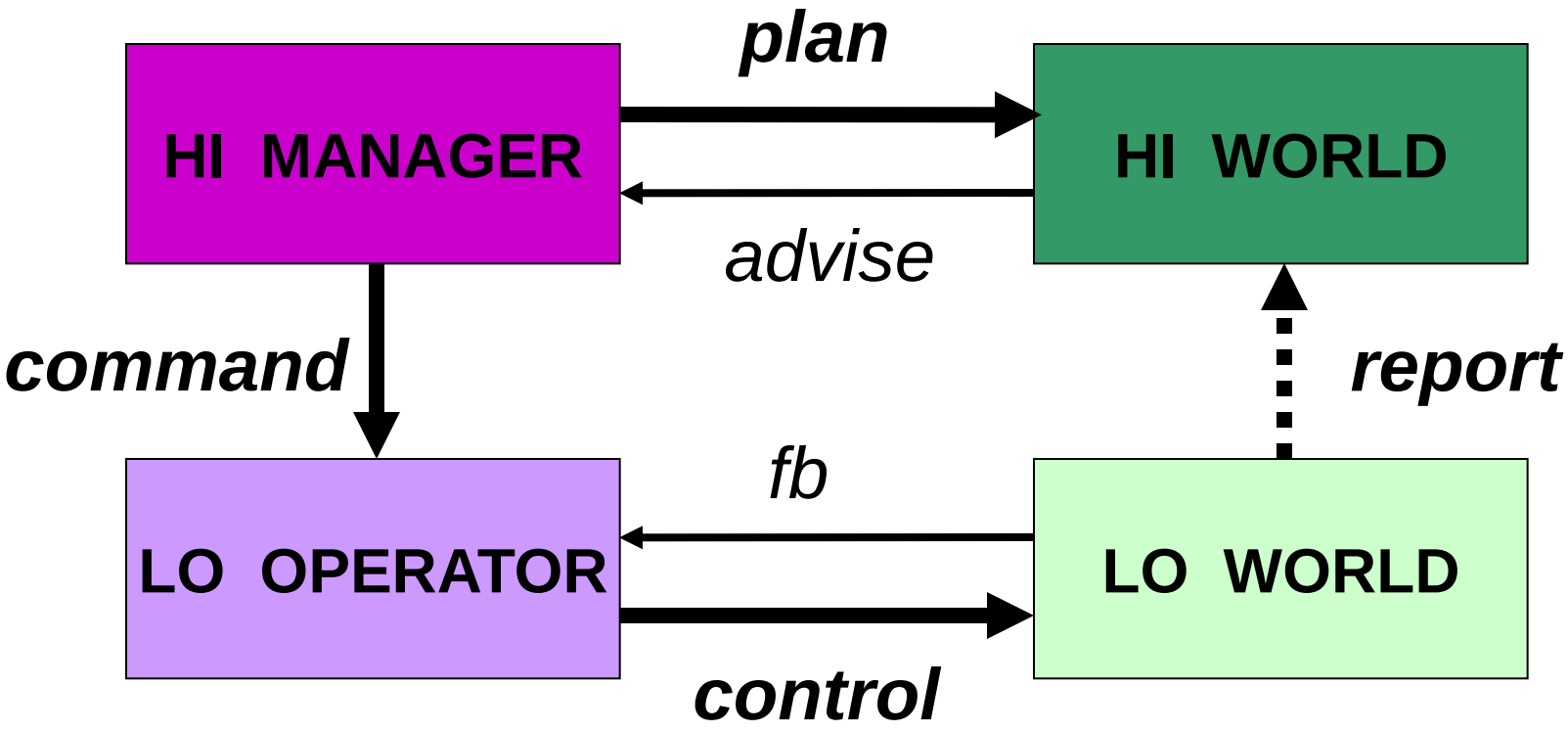
scope

Operator (fast)

- **Scope ~ # subordinates**
  time horizon
  bandwidth $^{-1}$
  frequency $^{-1}$ of significant events

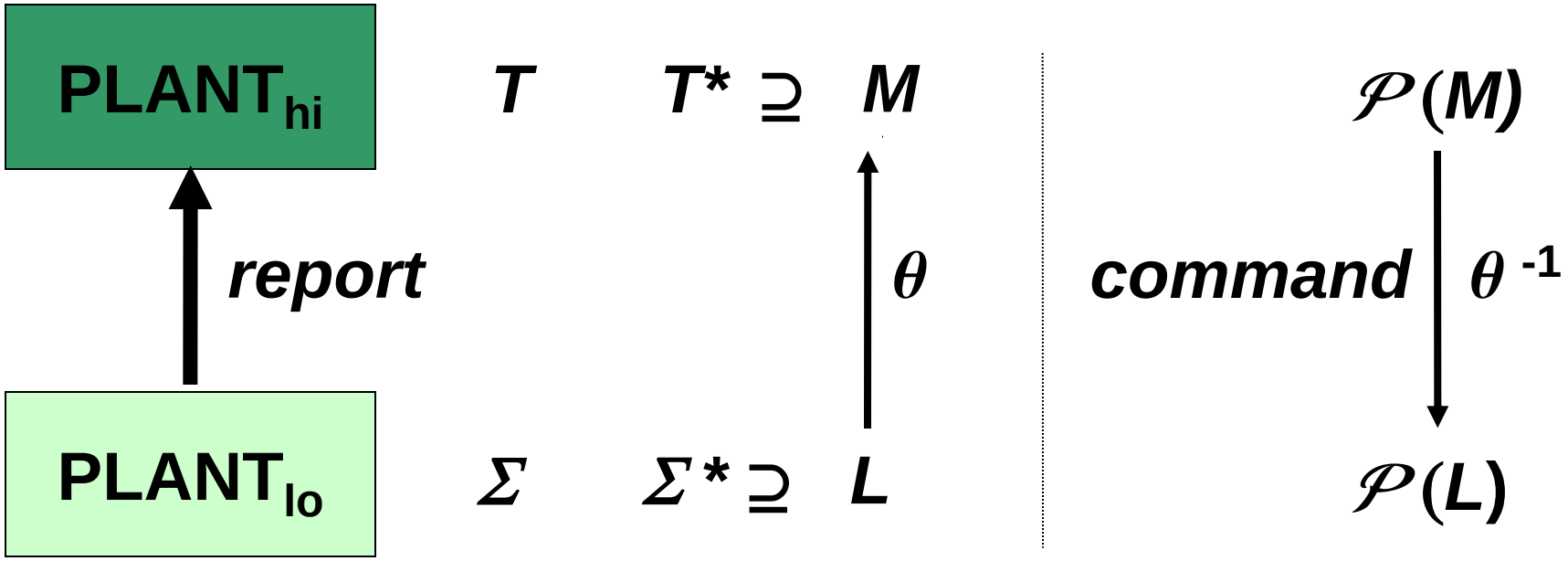
- **Scope ratio (adjacent levels) ~ 5:1**
  e.g. 20,000 employees ~ 7 levels

Hierarchical Consistency



$$plan \overset{?}{=} report \circ (control \circ command)$$

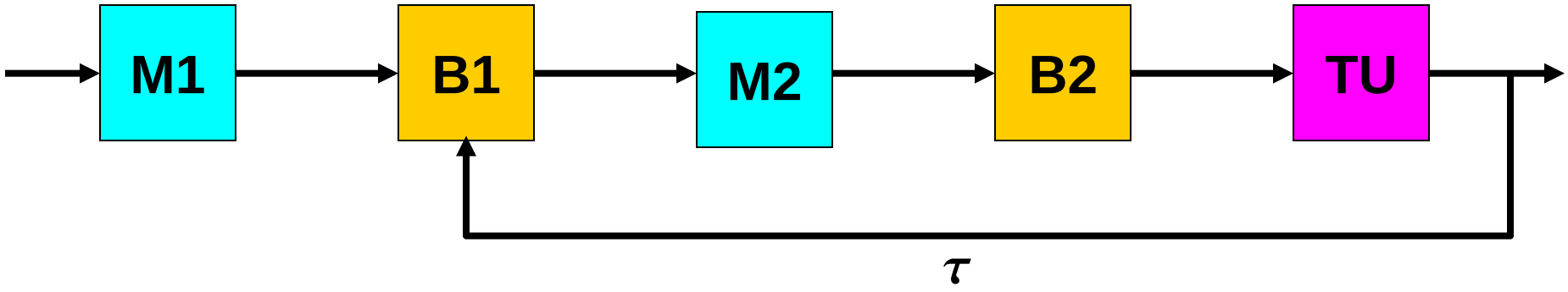
Report and Command



report is modeled by $\theta : L \rightarrow T^*$, $\theta(L) =: M$

command is modeled by $\theta^{-1} : \mathcal{P}(M) \rightarrow \mathcal{P}(L)$

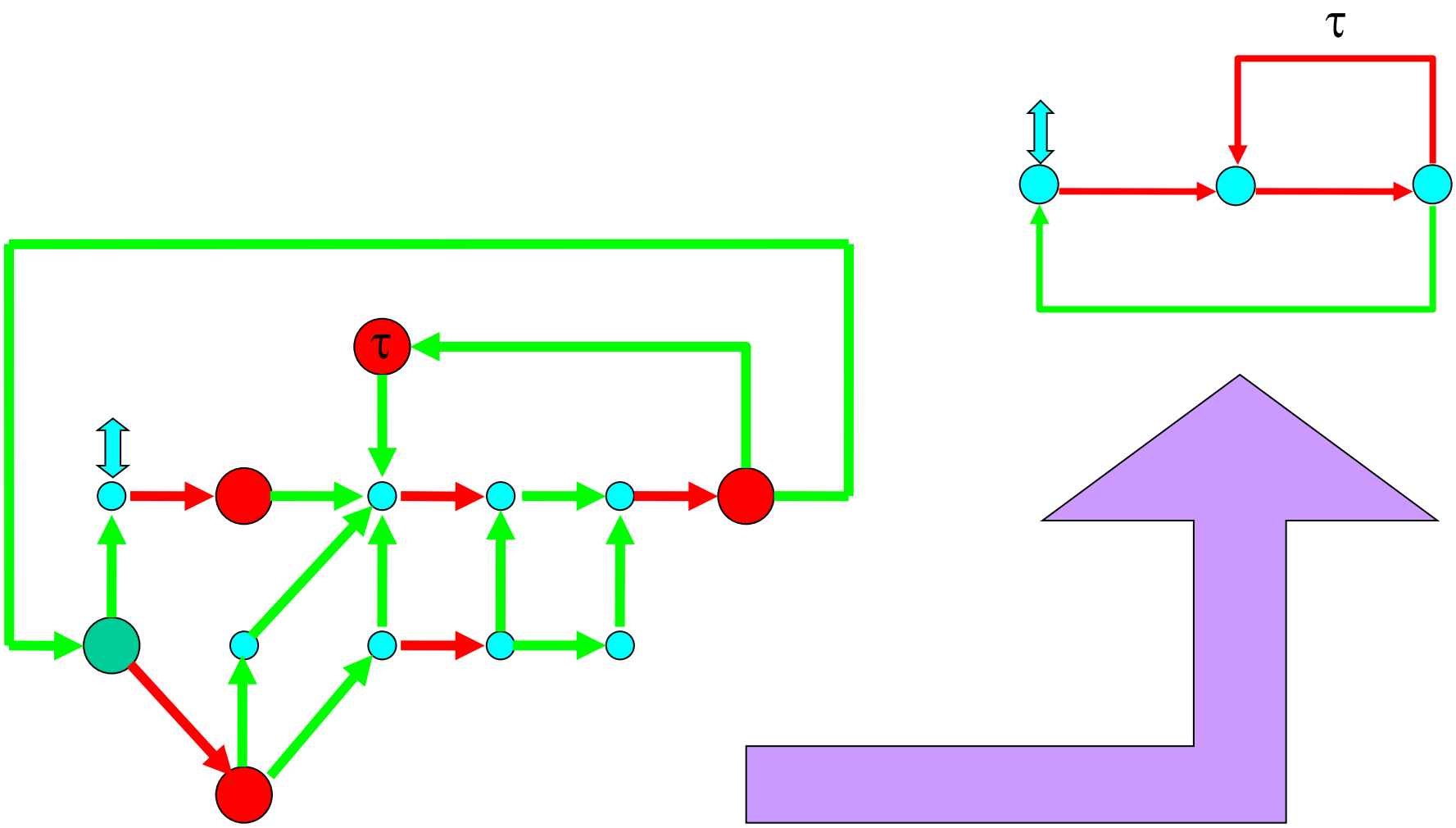
Hierarchical Transfer Line



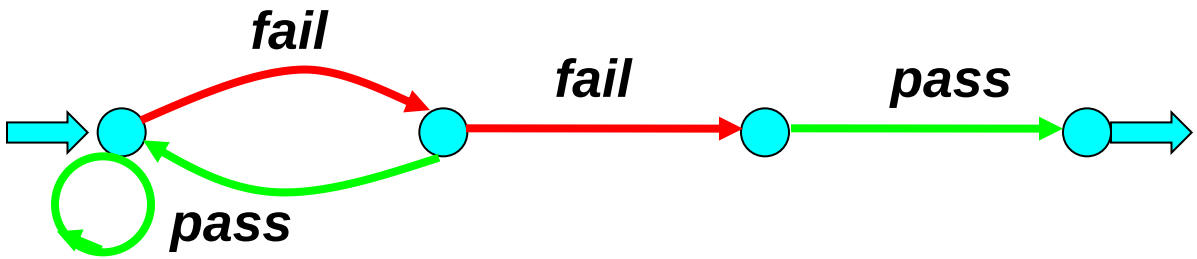
For hierarchical control, bring in
manager's hi-level alphabet ***T*** with events τ, τ', ...

Event τ = '**TU** returns faulty workpiece for reworking'

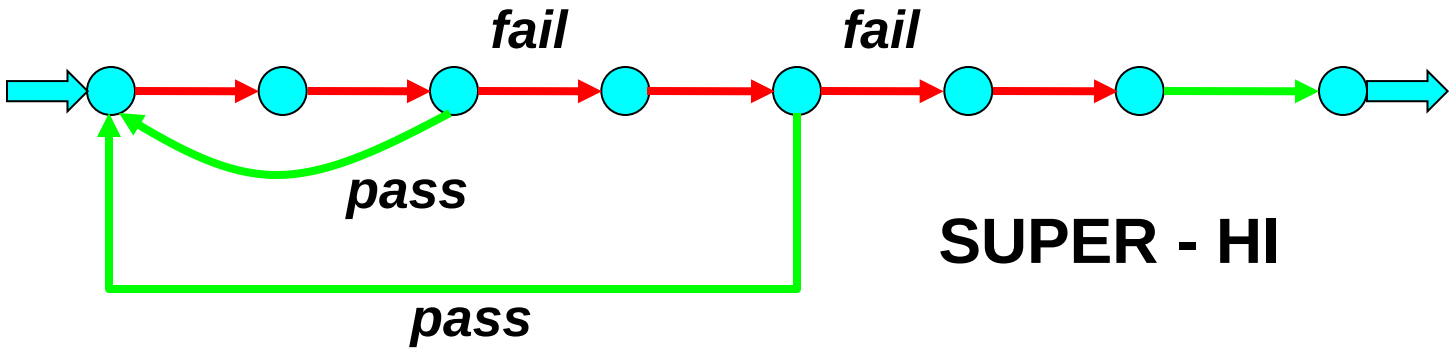
Hierarchical Transfer Line – ***LO*** to ***HI***



Hierarchical Transfer Line - ***HI***-Level Synthesis



SPEC - HI



SUPER - HI

Summary

- Some history
- Supervisory Control Theory (SCT)
- Large systems (using IDDs)
- Hierarchy
- Extensions and Applications
- Conclusions

Extensions

- General architectures: Heterarchical control, combining hierarchical, decentralized, and distributed control, with supervisor localization to create 'smart' agents
- Forced (preemptive) events
- Timed events (delays, deadlines, forcing)
- Liveness (= eventuality), temporal logic – infinite-string (Σ^{ω} - languages)
- Liveness (fairness, μ-calculus)
- Algebraically hybrid (?) –
  $X = Q_1 \times \ldots \times Q_k \times \mathbb{N}^n \times \mathbb{B}^m$
- Smart computation: (Timed) State Tree Structures with BDDs

Applications

- Communication protocol specification (*Rudie 1990*)
- Rapid thermal multiprocessor (*Hoffmann 1991*)
- Robotic agents (*Kosecka 1994*)
- AIP automated manufacturing system
  (*Brandin 1994*, *Leduc 2001*, *Ma 2003*)
- Telephone feature interaction (*Thistle 1995*)
- Chemical process control (*Sanchez 1996*, *Alsop 1996*)
- Truck dispatching (*Blouin 2001*)
- Telephone directory assistance call center (*Seidl 2004*)
- Electrical power flow control (*Afzalian 2009*)
- MRI scanner patient support system (*Theunissen 2010)*

Modular supervisors applied on a Patient Support System

P.A.H. Thijs, R.J.M. Theunissen, D.A. van Beek, and J.E. Rooda

Eindhoven University of Technology
Department of Mechanical Engineering, Systems Engineering Group
http://se.wtb.tue.nl

Embedded Systems Institute

Introduction

In the Darwin project on evolvability of MRI scanners, Supervisory Control Theory of Ramadge Wonham [1] is used to synthesize supervisors for controlling the Patient Support System (PSS).

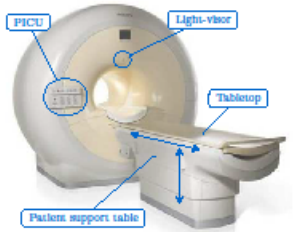


Figure 1: *Components of the Patient Support System*

The PSS is used to position a patient inside an MRI scanner. The system consists of a patient support table, a removable tabletop, a light-visor and a user interface (PICU). The uncontrolled system of the PSS consists of 6.3 billion states.

Objective

Supervision of the PSS using a monolithic event based supervisor is not possible due to state space explosion. Therefore, the goal of this research is to supervise the normal local behavior of the PSS using modular supervisors.

Modular supervision

For the synthesis of modular supervisors, the following steps are performed:

1. Partitioning of the uncontrolled system (27 automata) into 11 modules (F_1 - F_11). This partitioning is based on the functionality and/or components of the system.
2. Partitioning of the control requirements (57 automata) into 14 modules (E_1 - E_14). This partitioning is based on the functionality and/or components of the system and the partitioning made in the previous step.
3. Calculating the natural observer (FO_1 - FO_14) for every supervisor. This is done to reduce the state space of each modular supervisor.
4. Synthesizing the supervisor (SM_1 - SM_14) for every group of control requirements.
5. Checking the global nonconflicting property for the modular supervisors. As it is not guaranteed that the modular supervisors are not conflicting.

A part of the relation between plant modules, control requirement modules, natural observers and supervisors of the PSS are visualized in the figure below.

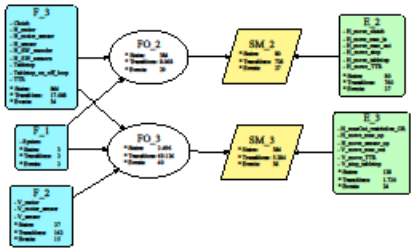


Figure 2: *Relation diagram of 2 supervisors*

Conclusion

To control the normal local behavior of the PSS, 14 modular supervisors are synthesized. The size of each supervisor is in the range of 20 – 2.000 states. The real-time implementation of these supervisors show that modular supervisory control can be used to supervise complex systems such as the PSS. The response time of the PSS controlled by a prototype implementation of the modular supervisors is equal to the response time of the current implementation of the control system.

References

[1] Wonham, W.M., *Supervisory control of discrete-event systems*, Department of Electrical & Computer Engineering, University of Toronto, 2007.

This work has been carried out as part of the DARWIN project at Philips Healthcare under the responsibilities of the Embedded Systems Institute (ESI). This project is partially supported by the Dutch Ministry of Economics Affairs under the BSIK program.

Supervisory control of MRI scanners

R.J.M. Theunissen, P.A.H. Thijs, R.R.H. Schiffelers, D.A van Beek and J.E. Rooda

Eindhoven University of Technology
Department of Mechanical Engineering, Systems Engineering Group
http://se.wtb.tue.nl

TU/e

Introduction

In the framework of the Darwin project on evolvability of MRI scanners, supervisory control theory is used to synthesize a supervisor for the patient support system (PSS), see Figure 1.

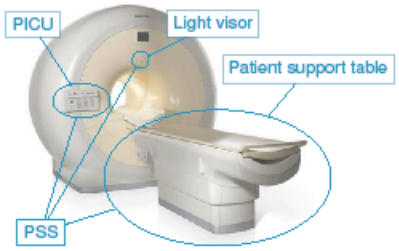


Figure 1: *MRI scanner*

The PSS is used to position a patient inside a MRI scanner. The system consists of a table and a removable tabletop. It has several sensors and actuators, see Figure 2. The PSS is connected to a user interface for manual control (PICU) and to the main controller of the MRI system for host based control.

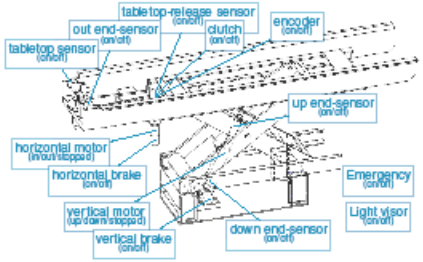


Figure 2: *Patient support table*

Evolvability

Supervisory control theory helps to improve systems evolvability; after changes in system requirements the following steps are taken:

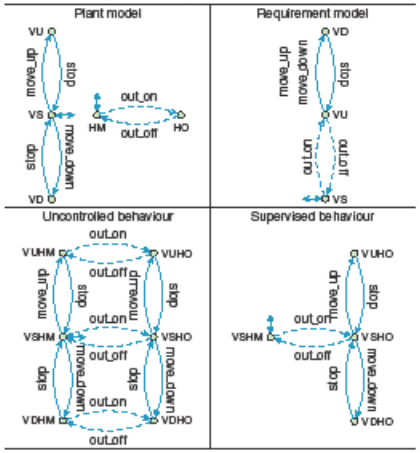
1. Update plant model
2. Update control requirements
3. Regenerate supervisor

To further improve the evolvability of the PSS supervisor, the models are:

- Small
- Loosely coupled

Supervisor specification example

The patient support table may only move vertically if it is horizontally maximally out:



Simulation and real-time control

Monolithic supervisor:

- Light visor and emergency behaviour excluded
- State space: 4.712 states, 33.684 transitions

Modular supervisor:

- Light visor and emergency behaviour included
- 9 components
- State space: 1.420 states, 11.656 transitions (monolithic ca. 1.500.000 states)

Monolithic and modular supervisors have both been tested by means of:

- Simulation with a hybrid χ model of the PSS
- Real-time control on the actual PSS

Results: First time right, no errors.

This work has been carried out as part of the DARWIN project at Philips Healthcare under the responsibilities of the Embedded Systems Institute (ESI). This project is partially supported by the Dutch Ministry of Economics Affairs under the BSIK program.

Conclusions

- **Achievements of SCT:**
  - * **Synthetic and general**
  - * **Results correct by construction and computable for 'large' systems**
  - * **Modular architecture, and smart computation for management of complexity**
  - * **Easy to teach and use (e.g. materials on Internet)**

- **Challenges for SCT:**
  - * **How to interpret and modify controller structure (e.g. IDDs ⇒ linear inequalities) ?**
  - * **How to find general laws of architecture ?**

APPENDIX

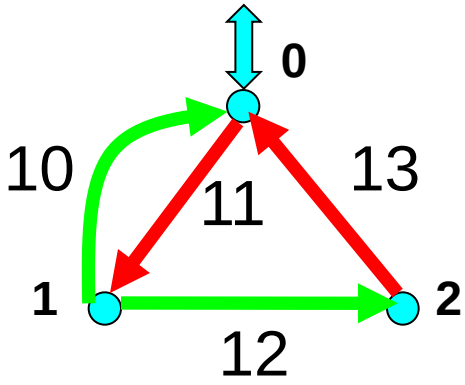
ADDITIONAL
TECHNICAL DETAILS

‘ROMANTIC’ VIEW – CE 1891

TCT **MACH**

MACH := (Q, Σ, δ, q_0, Q_m)

MACH = **Create** (**MACH**)



> name: **MACH**

> # states: 3 {*TCT* Q := {0,1,2}, q_0 := 0}

> marker state(s): 0 {*TCT* Q_m := {0}}

> transitions: [0,11,1], [1,10,0], [1,12,2], [2,13,0]
{*TCT* Σ := {10, 11, 12, 13}, δ : Q × Σ → Q transitions}

> quit <Ret> {*TCT* files **MACH.DES**}

Supervisor Reduction

- Monolithic supervisor **SUPER** is automaton representation of ***controlled behavior***

- Controlled behavior has ***state size***

$$||L_m(\mathbf{SUPER})|| = ||\mathrm{Sup}\mathcal{C}(L_m(\mathbf{PLANT}) \cap L_m(\mathbf{SPEC}))||$$
$$\leq ||L_m(\mathbf{PLANT})|| \times ||L_m(\mathbf{SPEC})||$$

- Heuristically, compute ***reduced, control-equivalent*** supervisor **SIMSUP**, often with

$$||L_m(\mathbf{SIMSUP})|| \ll ||L_m(\mathbf{SUPER})||$$

- E.g. for **TL** (below), $12 \leq 16 \times 4$, $3 \ll 12$

TCT TRANSFER LINE (TL)

M1 = Create (M1), M2 = Create (M2), TU = Create (TU)

TL = Sync (M1, M2, TU) {*synchronous product*}

B1 = Create (B1), B2 = Create (B2)

BUFFSPEC = Sync (B1, B2) {*synchronous product*}

SUPER (.DES) = SupCon (TL, BUFFSPEC) {*optimization*}

SUPER (.DAT) = ConDat (TL, SUPER(.DES)) {*control data*}

SIMSUP = SupReduce (TL, SUPER(.DES), SUPER(.DAT))
{*supervisor reduction*}

SIMSUP (.DAT) = ConDat (TL, SIMSUP) {*control data*}

Computing Effort vs. |Nodes|

- Computing time ~ $|\text{Nodes}|^{1.5}$ << |States|
- Memory usage ~ $|\text{Nodes}| \times K$
- For 'loosely coupled' practical systems

$$|\text{Nodes}| \sim N \times K^{C}$$

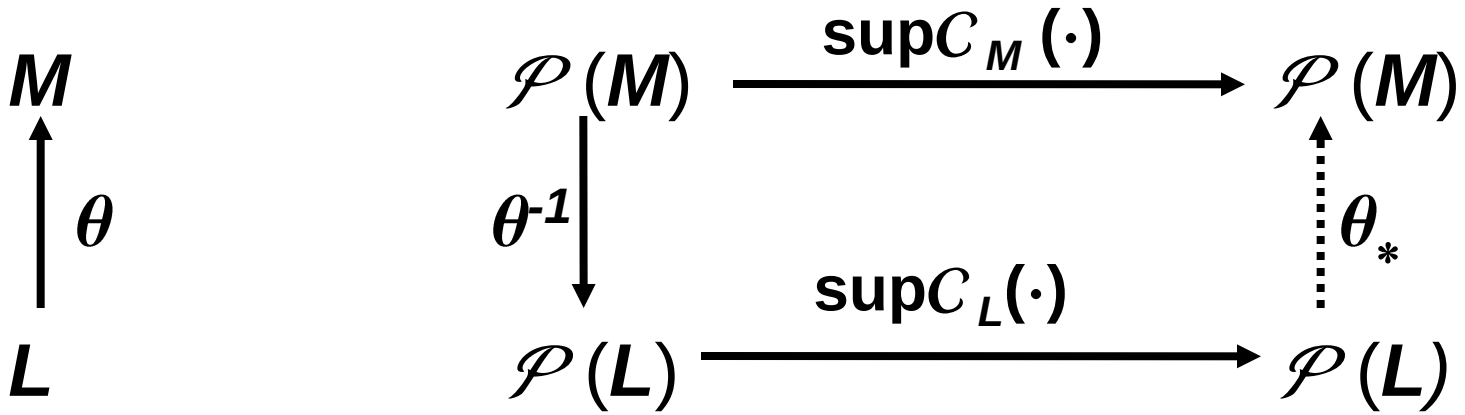
where N = number of system components (m+n)

K ~ state size of individual automata

C = coupling coefficient ~ 2

- |Nodes| ~ linear (***not exponential!)*** in N

Achieving Hierarchical Consistency



By design of T, θ *arrange*

"θ is an observer and preserves controllability"

Then diagram commutes, giving

hierarchical consistency