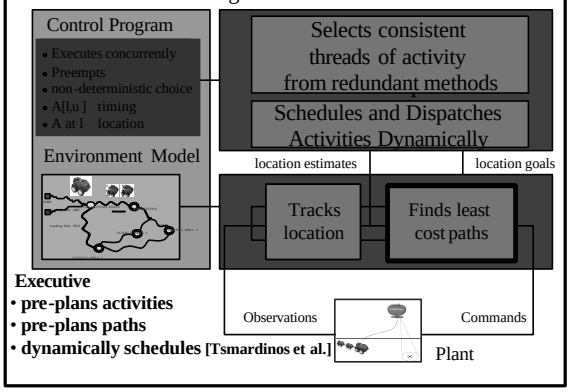
Reactive Planning of Hidden States in Large State Spaces Through Decomposition and Serialization

Brian C. Williams

Joint with
Seung H. Chung

16.412J/6.834J
March 16th, 2005

RMPL Model-based Program Kirk Model-based Executive

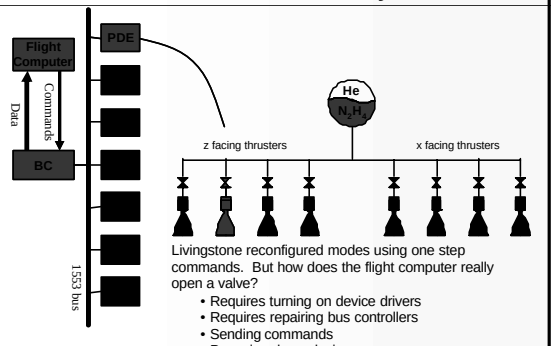


Executive
- **pre-plans activities**
- **pre-plans paths**
- **dynamically schedules [Tsmardinos et al.]**

Outline

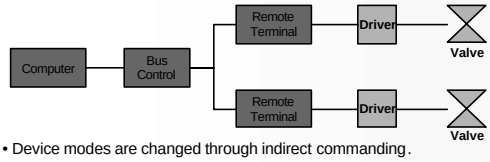
- The need for model-based reactive planning
- The Burton model-based reactive planner

DS 1 Attitude Control System



Livingstone reconfigured modes using one step commands. But how does the flight computer really open a valve?
- Requires turning on device drivers
- Requires repairing bus controllers
- Sending commands
- Powering down devices . . .

How do we reconfigure a valve?



- Device modes are changed through indirect commanding.
- Communication paths are established by reconfiguring other devices.
- The task of reconfiguring devices in the proper order generalizes state-space planning to handle indirect effects.
- To achieve reactivity all possible plans for all possible goal states should be pre-compiled (a generalization of universal plans).

⇨ To achieve compactness we decompose these universal plans according to a goal/sub-goal hierarchy.

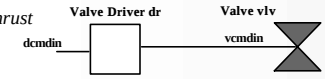
Mode Reconfiguration

Goal Interpreter

Reactive Planner

Example: Driver Valve Command Sequence

Goal: No thrust



Commands	Driver State	Valve State
ME:	**dr = off,**	**vlv = open**
GI:	**dr = off,**	**vlv = closed**
MRP dcmdin = on		
ME:	**dr = on,**	**vlv = open**
MRP dcmdin = close		
ME:	**dr = reset failure,**	**vlv = open**
MRP dcmdin = reset		
ME:	**dr = on,**	**vlv = open**
MRP dcmdin = off		
ME:	**dr = off,**	**vlv = open**

To achieve reactivity we eliminate all forms of search.

Model-based Reactive Planning

Achieved by:

1. Eliminate Indirect Control

   . . . through Compilation

2. Eliminate Search for Goal Ordering

   . . . through Reversibility and Serialization

3. Eliminate Search to find Suitable Transitions

   . . . by Constructing Hierarchical Polices

Model-based Reactive Planning

Achieved by:

1. Eliminate Indirect Control

   . . . through Compilation

2. Eliminate Search for Goal Ordering

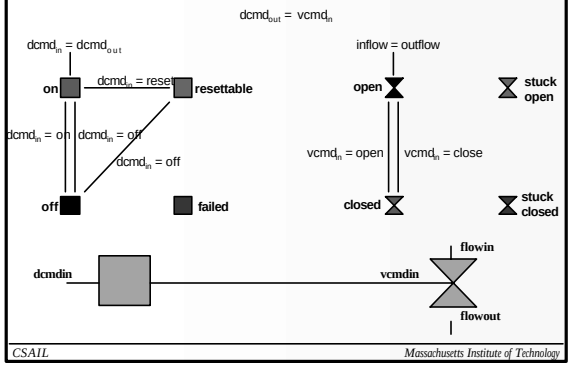
   . . . through Reversibility and Serialization

3. Eliminate Search to find Suitable Transitions

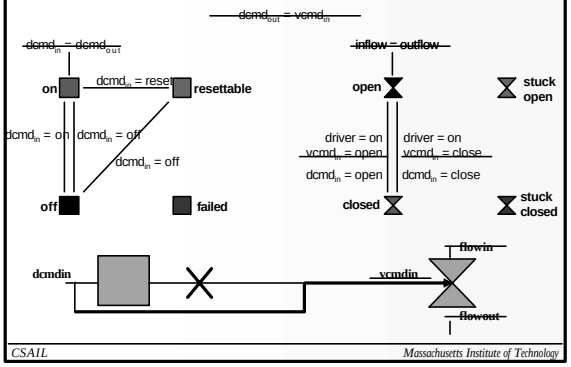
   . . . by Constructing Hierarchical Polices

CSAIL Massachusetts Institute of Technology

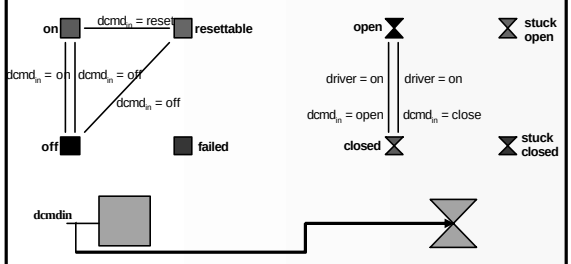
To Handle Indirect Control . . .



. . . Compile Out Constraints



. . . Compile Out Constraints



To Compile Out Constraints

- Eliminate intermediate variables.

⇨ Transitions are conditioned on mode and control variables

- Generate transitions as prime implicates:

$$\Phi_i \Rightarrow \text{next}(y_i = e_i)$$

where Φ_i is a conjunction of mode and control variable assignments.

⇨ Prime implicates for transitions enumerated using OpSAT

  – 40 seconds on SPARC 20 for 12,000 clause spacecraft model.

Model-based Reactive Planning

Achieved by:

1. Eliminate Indirect Effects

   . . . through Compilation

2. Eliminate Search for Goal Ordering

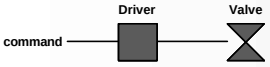
   . . . through Reversibility and Serialization

3. Eliminate Search to find Suitable Transitions

   . . . by Constructing Hierarchical Polices

Why Search is Needed

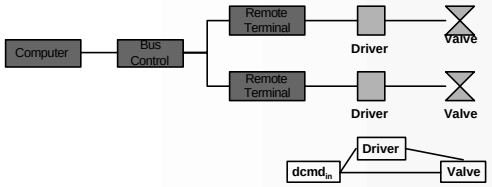
1) An achieved goal can be clobbered by a subsequent goal.



- Example
  – Current State: driver = on, valve = closed
  – Goal State: driver = off, valve = open
  – Achieving (driver = off) and then (valve = open) clobbers (driver = off)

➔ Achieve Valve goal before Driver goal

Observation: Engineered systems tend not to have loops



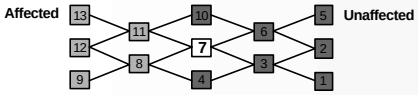
  – Define: Causal Graph *G* of compiled transition system *S*
    - vertices are state variables.
    - edge from v_i to v_j if v_j's transition is conditioned on v_i.
  – Requirement: The causal graph is acyclic.

➔ Work conjunctive goals upstream from outputs to inputs

Solution

- The only variables used to set some variable (y_7) is its ancestors ,

  ⇒ y_7 can be changed without affecting its descendants .

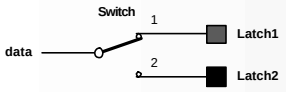


⇨ Safe to achieve goals in an upstream order.

- Simple check:
  – Number causal graph depth first
  – achieve goals in order of increasing depth first number.

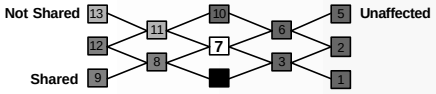
Why Search is Needed

2) Two goals can compete for the same variable in their subgoals.

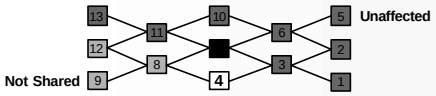


- Example
  - Latch1 and Latch2 compete for the position of Switch if latch goals achieved concurrently.

- Sibling goals (7,4) may both need shared ancestors.



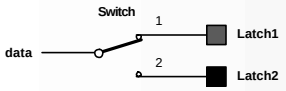
- But ancestors no longer needed once goal (7) is satisfied.



- Solution: Solve one goal before starting next sibling (Serialization).
- Feature: Generates first control action of plan first!

Why Search is Needed

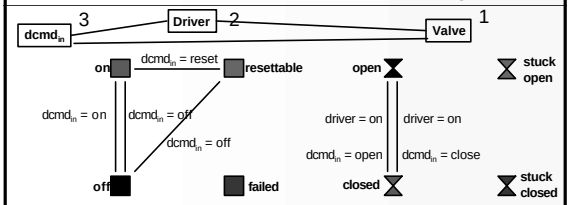
3) A state transition of a subgoal variable has irreversible effect.



- Example
  - Assume Switch can be used once,
  - Then Latch1 must be latched before Latch2.
- But irreversible effects aren't desirable for reactive planners

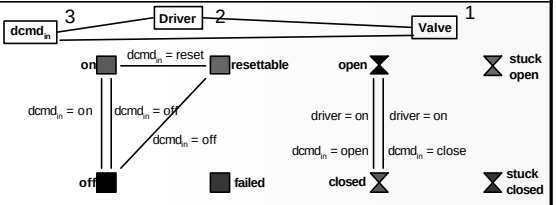
➔ Don't allow irreversible actions
➔. . . Except to repair failure modes

Solution: Mark Allowed Transitions/Assignments



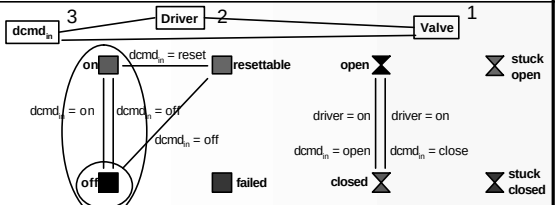
- Mark all control variable assignments allowed:

Solution: Mark Allowed Transitions/Assignments



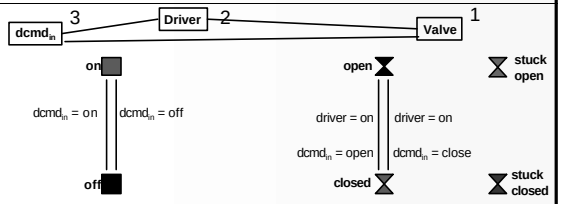
- Mark all control variable assignments allowed:
- For each mode variable v, in decreasing order of DF number:
  - Select each transition of v, whose guard has only allowed assignments.

Solution: Mark Allowed Transitions/Assignments



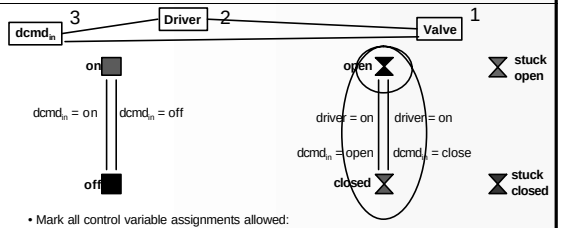
- Mark all control variable assignments allowed:
- For each mode variable v, in decreasing order of DF number:
  - Select each transition of v, whose guard has only allowed assignments.
  - Given current assignment v = I for v:
    - Find strongly connected component of selected transitions that contains I.
    - Mark assignments and transitions in SCC allowed.

Solution: Mark Allowed Transitions/Assignments



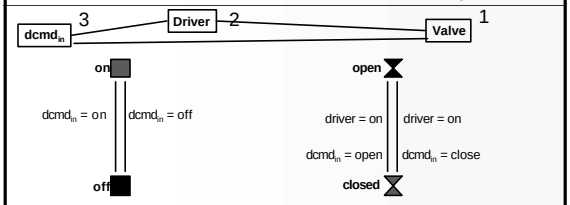
- Mark all control variable assignments allowed:
- For each mode variable v, in decreasing order of DF number:
  - Select each transition of v, whose guard has only allowed assignments.
  - Given current assignment v = l for v:
    - Find strongly connected component of selected transitions that contains l.
    - Mark assignments and transitions in SCC allowed.

Solution: Mark Allowed Transitions/Assignments



- Mark all control variable assignments allowed:
- For each mode variable v, in decreasing order of DF number:
  - Select each transition of v, whose guard has only allowed assignments.
  - Given current assignment v = l for v:
    - Find strongly connected component of selected transitions that contains l.
    - Mark assignments and transitions in SCC allowed.

Solution: Mark Allowed Transitions/Assignments



- Mark all control variable assignments allowed:
- For each mode variable v, in decreasing order of DF number:
  - Select each transition of v, whose guard has only allowed assignments.
  - Given current assignment v = l for v:
    - Find strongly connected component of selected transitions that contains l.
    - Mark assignments and transitions in SCC allowed.

Model-based Reactive Planning

Achieved by:

1. Eliminate Indirect Effects

   . . . through Compilation

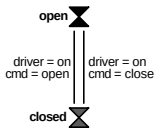
2. Eliminate Search for Goal Ordering

   . . . through Reversibility and Serialization

3. Eliminate Search to find Suitable Transitions

   . . . by Constructing Hierarchical Polices

Solution



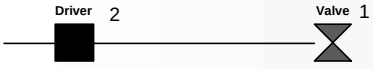
Current \ Goal	Open	Closed
Open	idle	driver = on cmd = close
Closed	driver = on cmd = open	idle
Stuck	fail	fail

- Convert automata into hierarchical policies, one per automaton
  - Policy selects first transition towards achieving each automata goal state, given current state.
  - Policy maps goals to subgoals and commands, in proper order
  - Ensures only reversible transitions are taken, by only using transitions marked allowed.

Plan by passing sub-goals up causal graph

Goal: Driver = off, Valve = closed

Current: Driver = off, Valve = open



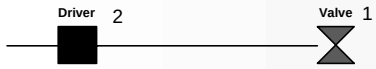
Current \ Goal	On	Off
On	idle	cmd = off
Off	cmd = on	idle
Resettable	cmd = reset	cmd = off

Current \ Goal	Open	Closed
Open	idle	driver = on cmd = close
Closed	driver = on cmd = open	idle
Stuck	fail	fail

Plan by passing sub-goals up causal graph

Goal: Driver = off, Valve = closed

Current: Driver = off, Valve = open



Current \ Goal	On	Off
On	idle	cmd = off
Off	cmd = on	idle
Resettable	cmd = reset	cmd = off

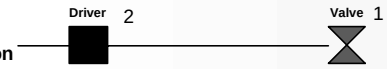
Current \ Goal	Open	Closed
Open	idle	**driver = on** cmd = close
Closed	driver = on cmd = open	idle
Stuck	fail	fail

Plan by passing sub-goals up causal graph

Goal: Driver = off, Valve = closed

Current: Driver = off, Valve = open

Send: cmd = on



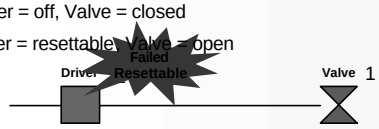
Current \ Goal	On	Off
On	idle	cmd = off
Off	**cmd = on**	idle
Resettable	cmd = reset	cmd = off

Current \ Goal	Open	Closed
Open	idle	**driver = on** cmd = close
Closed	driver = on cmd = open	idle
Stuck	fail	fail

Plan by passing sub-goals up causal graph

Goal: Driver = off, Valve = closed

Current: Driver = resettable, Valve = open



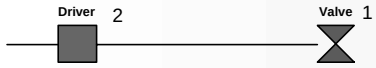
Current \ Goal	On	Off
On	idle	cmd = off
Off	cmd = on	idle
Resettable	cmd = reset	cmd = off

Current \ Goal	Open	Closed
Open	idle	**driver = on** cmd = close
Closed	driver = on cmd = open	idle
Stuck	fail	fail

Plan by passing sub-goals up causal graph

Goal: Driver = off, Valve = closed

Current: Driver = resettable, Valve = open



Current \ Goal	On	Off
On	idle	cmd = off
Off	cmd = on	idle
Resettable	**cmd = reset**	cmd = off

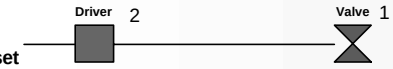
Current \ Goal	Open	Closed
Open	idle	**driver = on** cmd = close
Closed	driver = on cmd = open	idle
Stuck	fail	fail

Plan by passing sub-goals up causal graph

Goal: Driver = off, Valve = closed

Current: Driver = resettable, Valve = open

Send cmd = reset



Current \ Goal	On	Off
On	idle	cmd = off
Off	cmd = on	idle
Resettable	**cmd = reset**	cmd = off

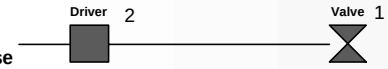
Current \ Goal	Open	Closed
Open	idle	**driver = on** cmd = close
Closed	driver = on cmd = open	idle
Stuck	fail	fail

Plan by passing sub-goals up causal graph

Goal: Driver = off, Valve = closed

Current: Driver = on, Valve = open

Send cmd = close



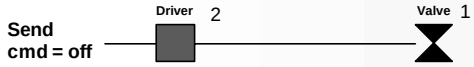
Current \ Goal	On	Off
On	idle	cmd = off
Off	cmd = on	idle
Resettable	cmd = reset	cmd = off

Current \ Goal	Open	Closed
Open	idle	driver = on **cmd = close** ☆
Closed	driver = on cmd = open	idle
Stuck	fail	fail

Plan by passing sub-goals up causal graph

Goal: Driver = off, Valve = closed

Current: Driver = on, Valve = closed



Current \ Goal	On	Off
On	idle	**cmd = off**
Off	cmd = on	idle
Resettable	cmd = reset	cmd = off

Current \ Goal	Open	Closed
Open	idle	driver = on cmd = close
Closed	driver = on cmd = open	idle
Stuck	fail	fail

Plan by passing sub-goals up causal graph

Goal: Driver = off, Valve = closed

Current: Driver = off, Valve = closed

Success

Current \ Goal	On	Off
On	idle	cmd = off
Off	cmd = on	idle
Resettable	cmd = reset	cmd = off

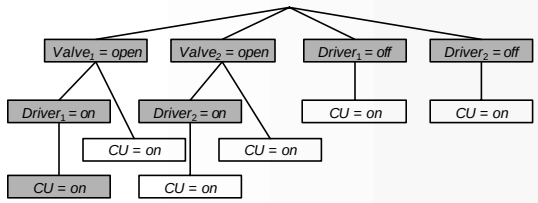
Current \ Goal	Open	Closed
Open	idle	driver = on cmd = close
Closed	driver = on cmd = open	idle
Stuck	fail	fail

Hierarchical, Model-based Reactive Planning

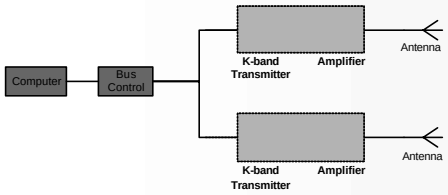
- Compile-time Analysis:
  - Compile-out interactions
  - Confirm schematics are loop free.
  - Depth first number variables.
- Periodic, Run-time Analysis:
  - Given initial state
    - Identify allowed transitions and assignments
  - Given autonomous jump to failure state
    - Identify allowed transitions and assignments
- Run-time Plan Execution:
  - Work conjunctive goals from outputs to inputs.
  - Achieve goals serially.
  - Only perform reversible transitions.
  - Lookup control actions and sub-goals in policies

Complexity of Reactive Planning

- Worst Case per action: Depth * Sub-goal branch factor
- Average Cost per action: Sub-goal branch factor

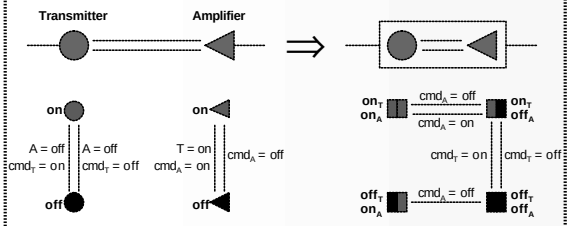


What If Plan is Not Serializable?



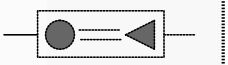
- What if causal graph *G* contains cycles?
- Solution:
  - Isolate the cyclic components (compute SCCs)
  - compose each cycle into a single component.
- New causal graph *G'* is acyclic,
- Goals of G' are serializable

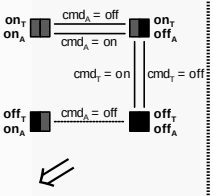
Composing Cyclic Components



Policy for Composed Components

- Problem: Composition grows exponential in space usage.
- Solution: Use BDD encoding (Seung Chung SM).





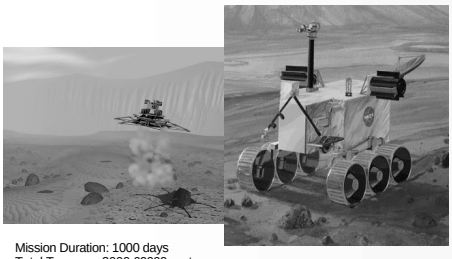
Current \ Goal	On_T, On_A	On_T, Off_A	Off_T, Off_A	Off_T, On_A
On_T, On_A	idle	cmd_A = off	cmd_A = off	fail
On_T, Off_A	cmd_A = on	idle	cmd_T = off	fail
Off_T, Off_A	cmd_T = on	cmd_T = on	idle	fail
Off_T, On_A	fail	fail	cmd_A = off	idle

Model-based Reactive Planning

1. Compile away constraints from the model
2. Compile away cyclic components
3. Plan serially pursuing causal graph upstream
4. Generate actions using hierarchical policies

Only performs reversible actions
Responds to failure at each step
Average cost per step = subgoal branching factor

Next Challenge: Mars Smart Lander (2009)



Mission Duration: 1000 days
Total Traverse: 3000-69000 meters
Meters/Day: 230-450
Science Mission: 7 instruments, sub-surface science package (drill, radar), in-situ sample "lab"

Technology Demonstration: (2005).

Model-based Programming of Embedded Systems

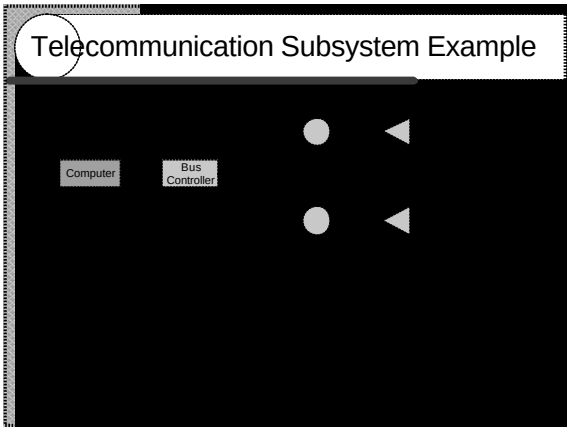
- To survive decades embedded systems orchestrate complex regulatory and immune systems.
- Future systems will be programmed with models, describing themselves and their environments.
- Runtime kernels will be agile, deducing and planning by solving optimization problems with propositional constraints.
- Model-based reactive planners respond quickly to failure, while using compile-time analysis of structure to respond quickly and concisely to indirect effects.

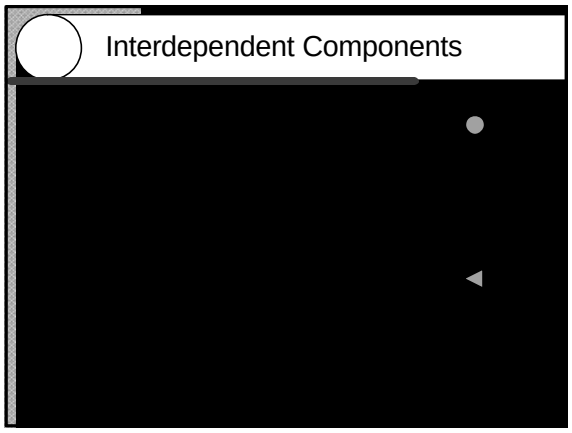
Appendix 1

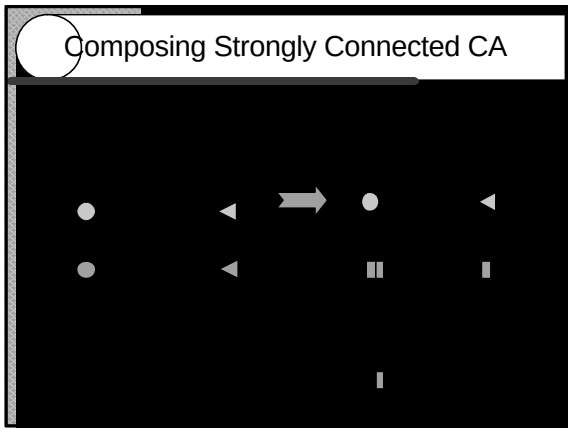
Handling Cycles In
The Causal Graph

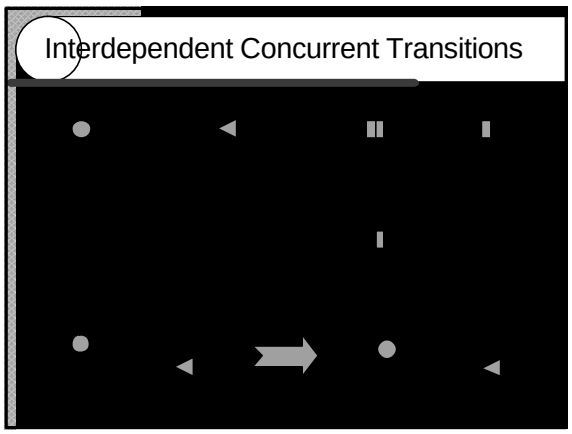
Slides by Seung Chung

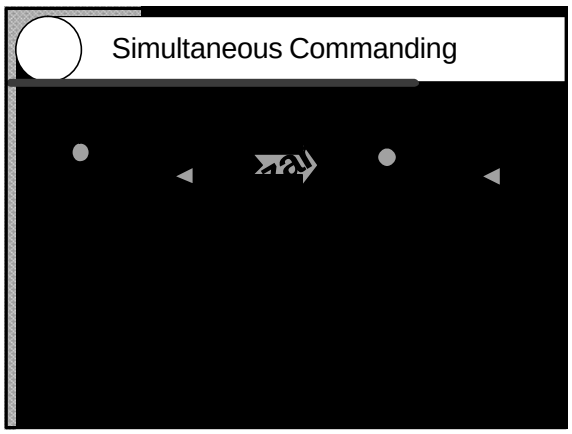
Telecommunication Subsystem Example

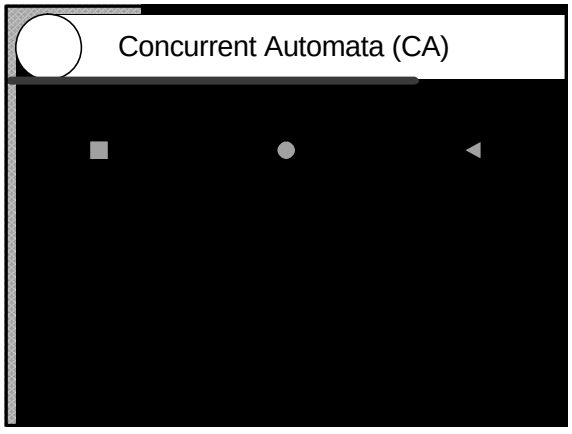


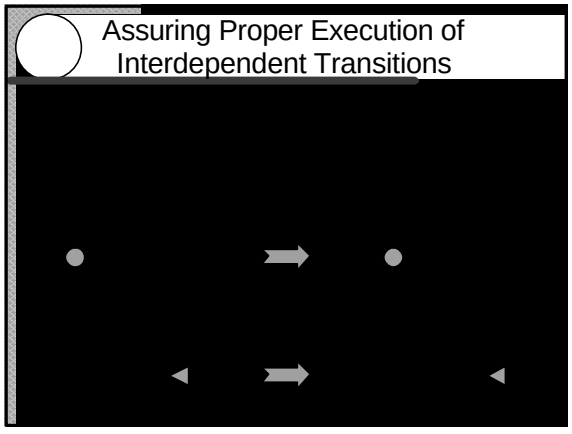

Interdependent Components


Composing Strongly Connected CA


Interdependent Concurrent Transitions


Simultaneous Commanding


Concurrent Automata (CA)


Assuring Proper Execution of Interdependent Transitions