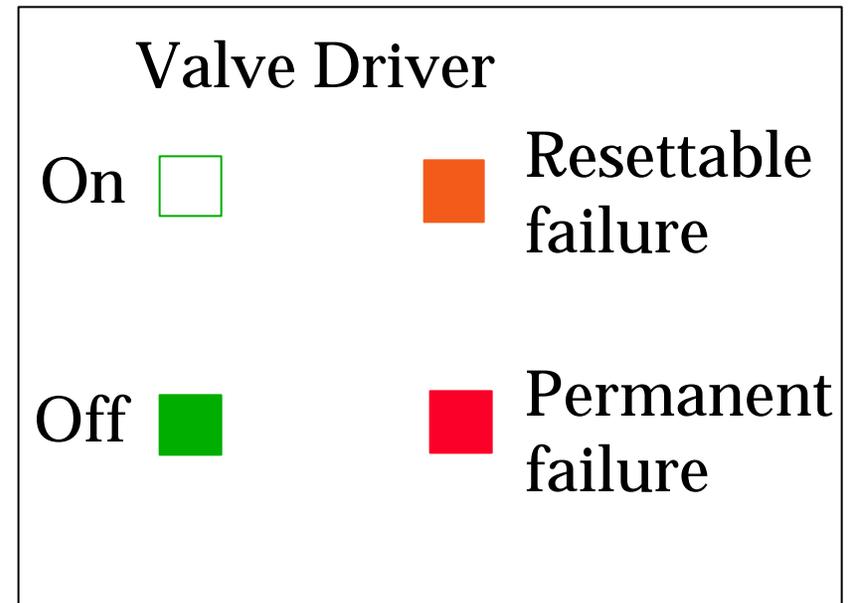
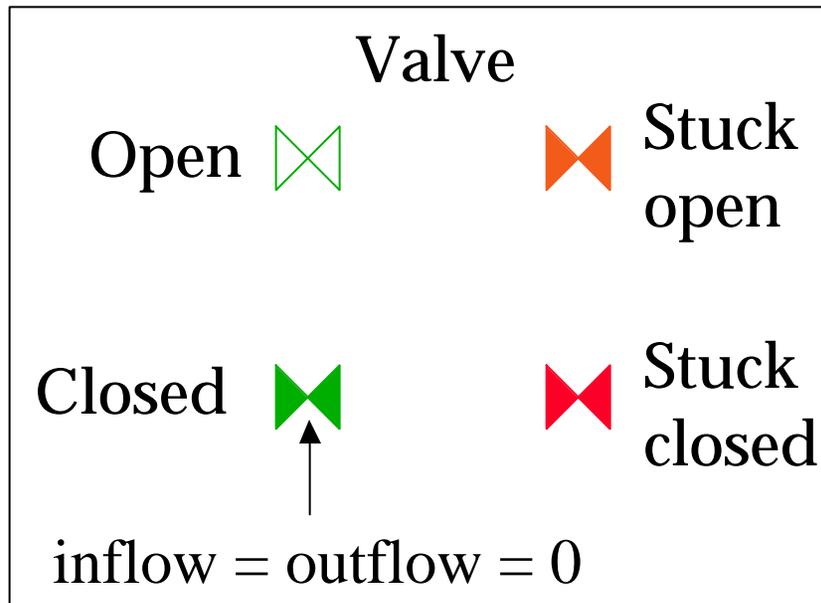
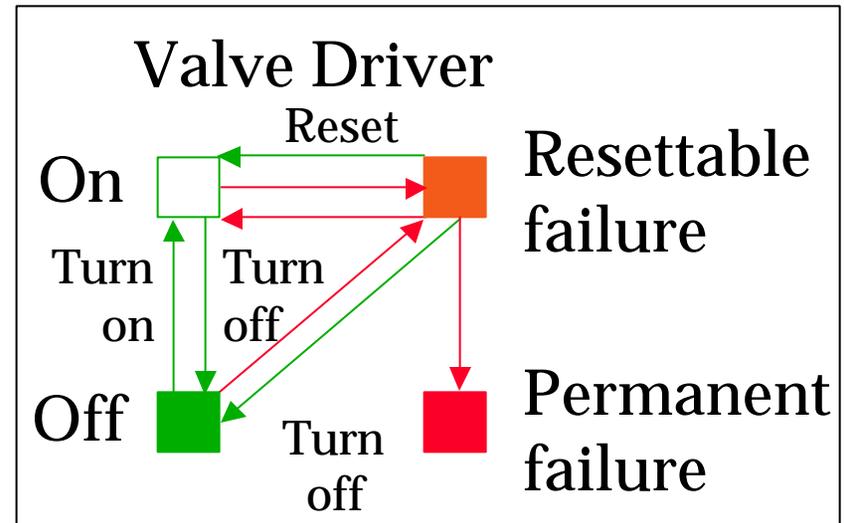
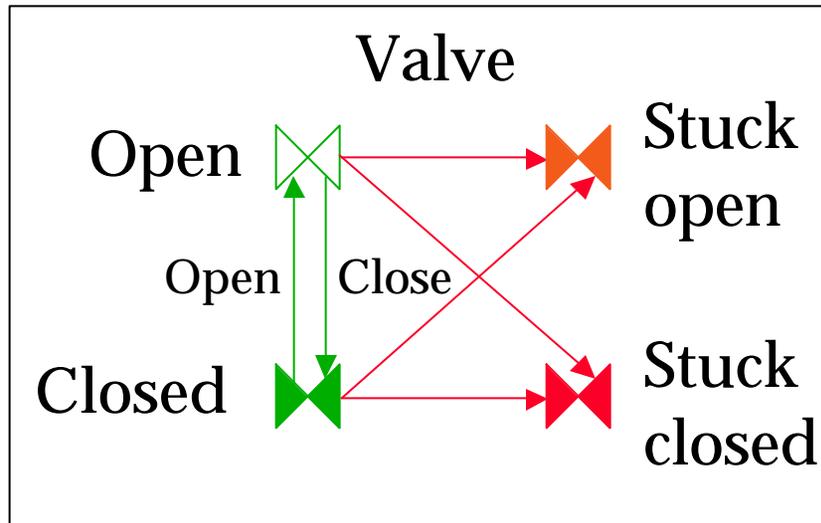

Incorporating Dynamics

Model-based Diagnosis



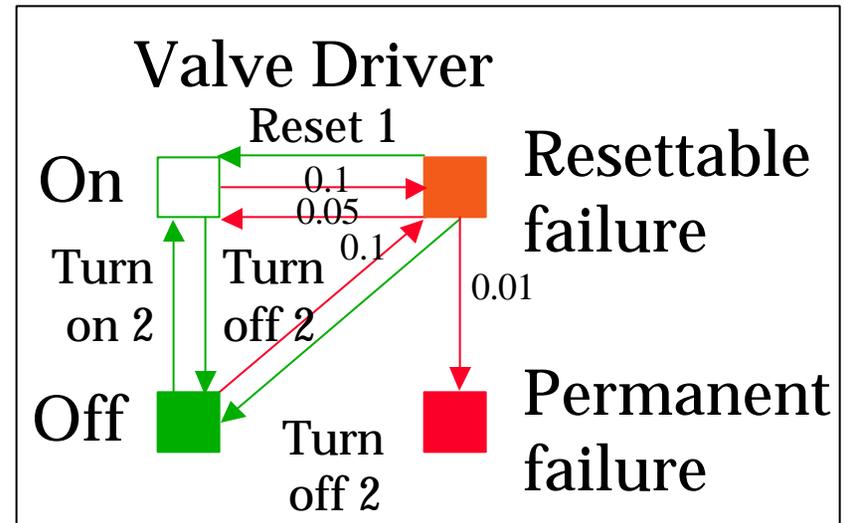
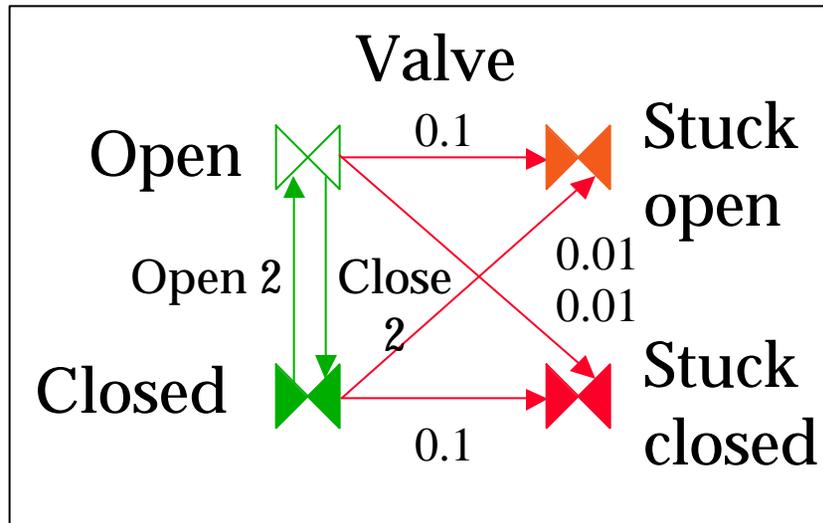
- Models behavior only *within* a component mode
- Does not explicitly model *transitions* between modes

Transition systems



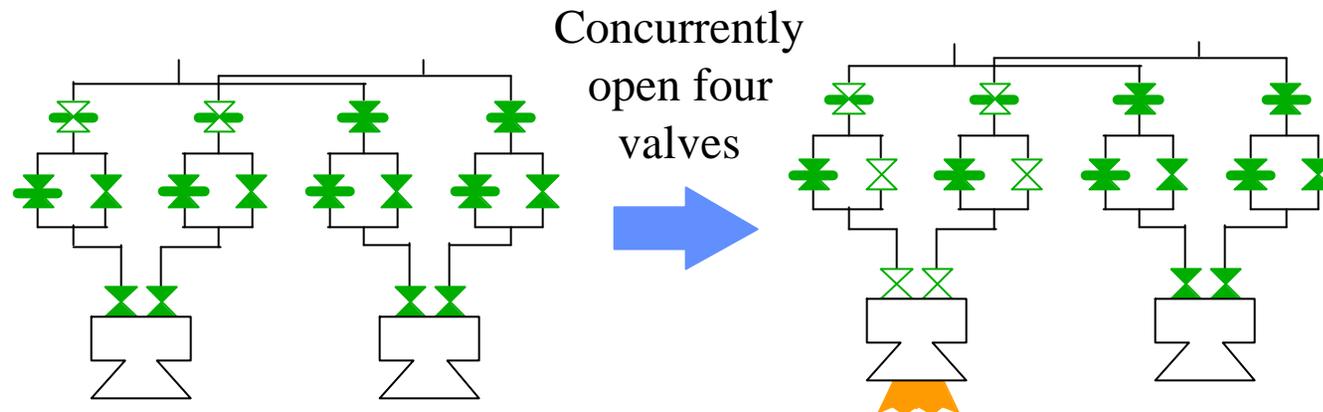
- Explicitly model mode transitions (including self-transitions)
 - commanded transitions with preconditions
 - failure (uncommanded) transitions
 - repair transitions
 - intermittency

Markov models



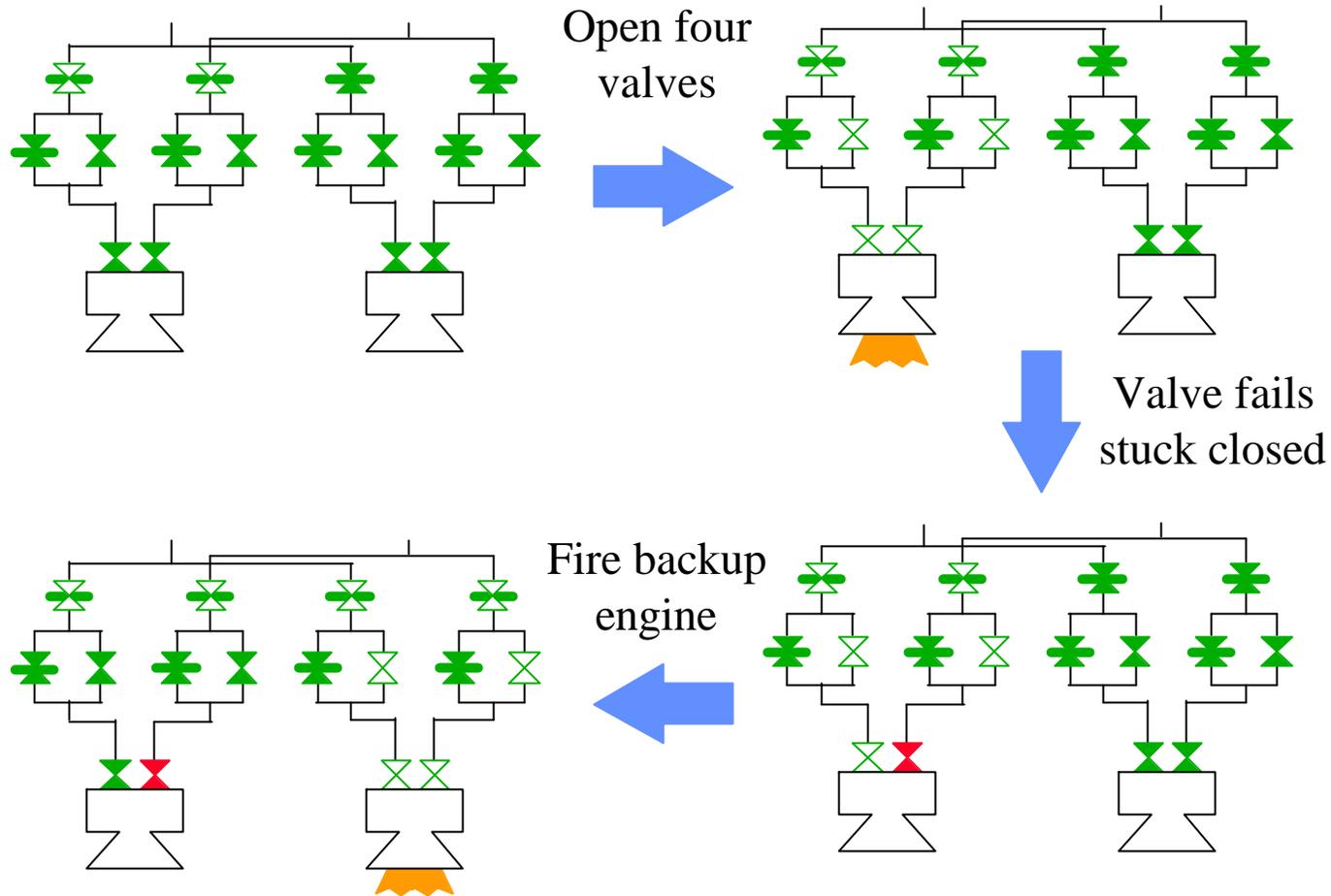
- Represent probability of uncommanded transitions and cost of commanded transitions
- Can model
 - Reliability
 - Optimal control

Concurrent transition systems



- Components within a system are modeled as *concurrent* transition system [Manna & Pnueli 92]
- ⇒ Each system transition consists of a single transition by *each* component transition system (possibly the idling transition)
- Can naturally model digital hardware, analog hardware using qualitative representations, and real-time software

Trajectories of concurrent transition systems



Nayak/Williams

SP2-99

AAAI-97 Tutorial SP2

Transition system models

- A system S is a tuple (Π, Σ, T)
- Π : set of *variables* ranging over finite domains
 - *state* variables (Π_s)
 - *control* variables (Π_c)
 - *dependent* variables (Π_d)
 - *observable* variables (Π_o)
- Σ : set of *feasible assignments*
 - Σ_s is the projection of Σ on Π_s
 - each element of Σ_s is a *state*
- **T : set of *transitions***
 - **each transition is a function $S \rightarrow S_s$**
 - **a single transition $t_n \in T$ is the *nominal* (commanded) transition**
 - **all other transitions are *failure* (uncommanded) transitions**

Specifying transition systems

- System $S = (\Pi, \Sigma, T)$ is specified using a *propositional temporal logic* formula ρ_S
- Propositions are of the form $y_k = e_k$
 - y_k is a variable in Π , and e_k is in y_k 's domain
- Feasible assignments Σ specified by a propositional formula ρ_Σ
 - specifies the set of assignments that satisfy ρ_Σ
- **Each transition $t \in T$ specified using a conjunction r_t of formulas of the form $r_{t_i} : F_{t_i} \text{ P } next(Y_{t_i})$**
 - F_{t_i} is a propositional formula
 - Y_{t_i} is of the form $y_k = e_k$ for state variable y_k
 - $t(a) = s$ if and only if for each r_{t_i} whenever assignment a satisfies F_{t_i} then s satisfies Y_{t_i}

Specifying a valve transition system

Same as for state-free systems:

- Variables $\Pi = \{mode, cmd, f_{in}, f_{out}, p_{in}, p_{out}\}$
 - *mode* ranges over $\{open, closed, stuck-open, stuck-closed\}$
 - *cmd* ranges over $\{open, close, no-cmd\}$
 - *f_{in}* and *f_{out}* range over $\{positive, negative, zero\}$
 - *p_{in}* and *p_{out}* ranging over $\{high, low, nominal\}$
- Specifying Σ with ρ_{Σ}
 - $mode = open \mathbf{P} (p_{in} = p_{out}) \wedge (f_{in} = f_{out})$
 - $mode = closed \mathbf{P} (f_{in} = zero) \wedge (f_{out} = zero)$
 - $mode = stuck-open \mathbf{P} (p_{in} = p_{out}) \wedge (f_{in} = f_{out})$
 - $mode = stuck-closed \mathbf{P} (f_{in} = zero) \wedge (f_{out} = zero)$

Specifying a valve transition system (cont.)

- Specifying the nominal transition $\tau_n \in T$
 - $mode = closed \wedge cmd = open \mathbf{P} next (mode = open)$
 - $mode = closed \wedge cmd \circ open \mathbf{P} next (mode = closed)$
 - $mode = open \wedge cmd = close \mathbf{P} next (mode = closed)$
 - $mode = open \wedge cmd \circ close \mathbf{P} next (mode = open)$
 - $mode = stuck-open \mathbf{P} next (mode = stuck-open)$
 - $mode = stuck-closed \mathbf{P} next (mode = stuck-closed)$
- Specifying failure transitions
 - $\tau_1 : mode = closed \mathbf{P} next (mode = stuck-closed)$
 - $\tau_2 : mode = closed \mathbf{P} next (mode = stuck-open)$
 - $\tau_3 : mode = open \mathbf{P} next (mode = stuck-open)$
 - $\tau_4 : mode = open \mathbf{P} next (mode = stuck-closed)$

Concurrent transition system models

⇒ Concurrent transition systems are simply *composed* out of the component transition systems

- Let $S = (\Pi, \Sigma, T)$ be a system consisting of a set component transition systems $C_i = (\Pi_i, \Sigma_i, T_i)$
- $\Pi \supseteq \Pi_i$ for each Π_i
 - Π may include additional variables
- Each feasible assignment in Σ , when restricted to Π_i , is in Σ_i
 - Σ may enforce additional feasibility constraints, *e.g.*, to model component connections
 - ρ_Σ entails ρ_{Σ_i} (by simply including it as a conjunct)
- Transitions in T are cross-products of component transitions
 - each transition in T performs exactly one transition from each T_i
 - ρ_T is equivalent to the conjunction of the ρ_{T_i}

Configuration Manager

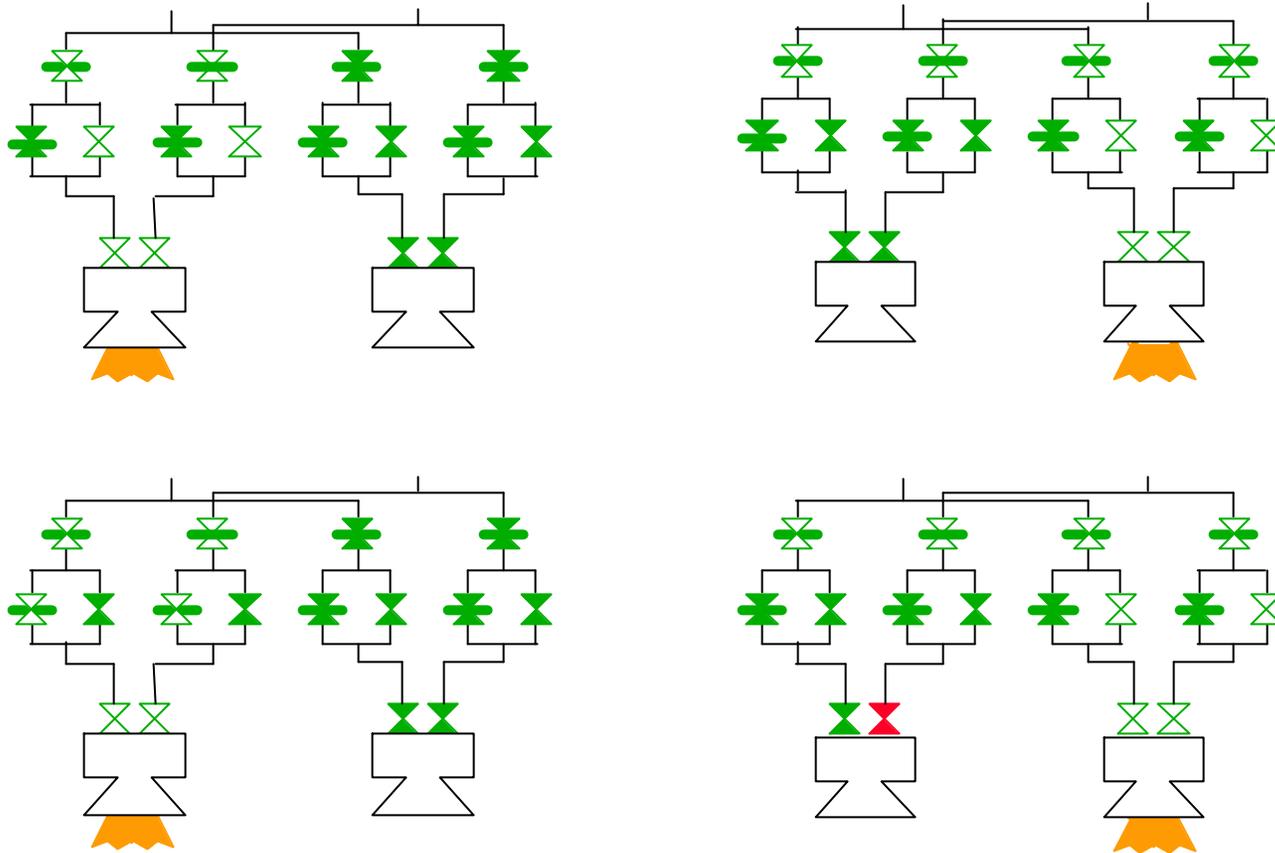
Input

- S is a system (Π, Σ, T)
- $\gamma: g_0, g_1 \dots$, called goal configurations, is a sequence of propositional formulae on Π
- $O: o_0, o_1 \dots$, is a sequence of observations

Output

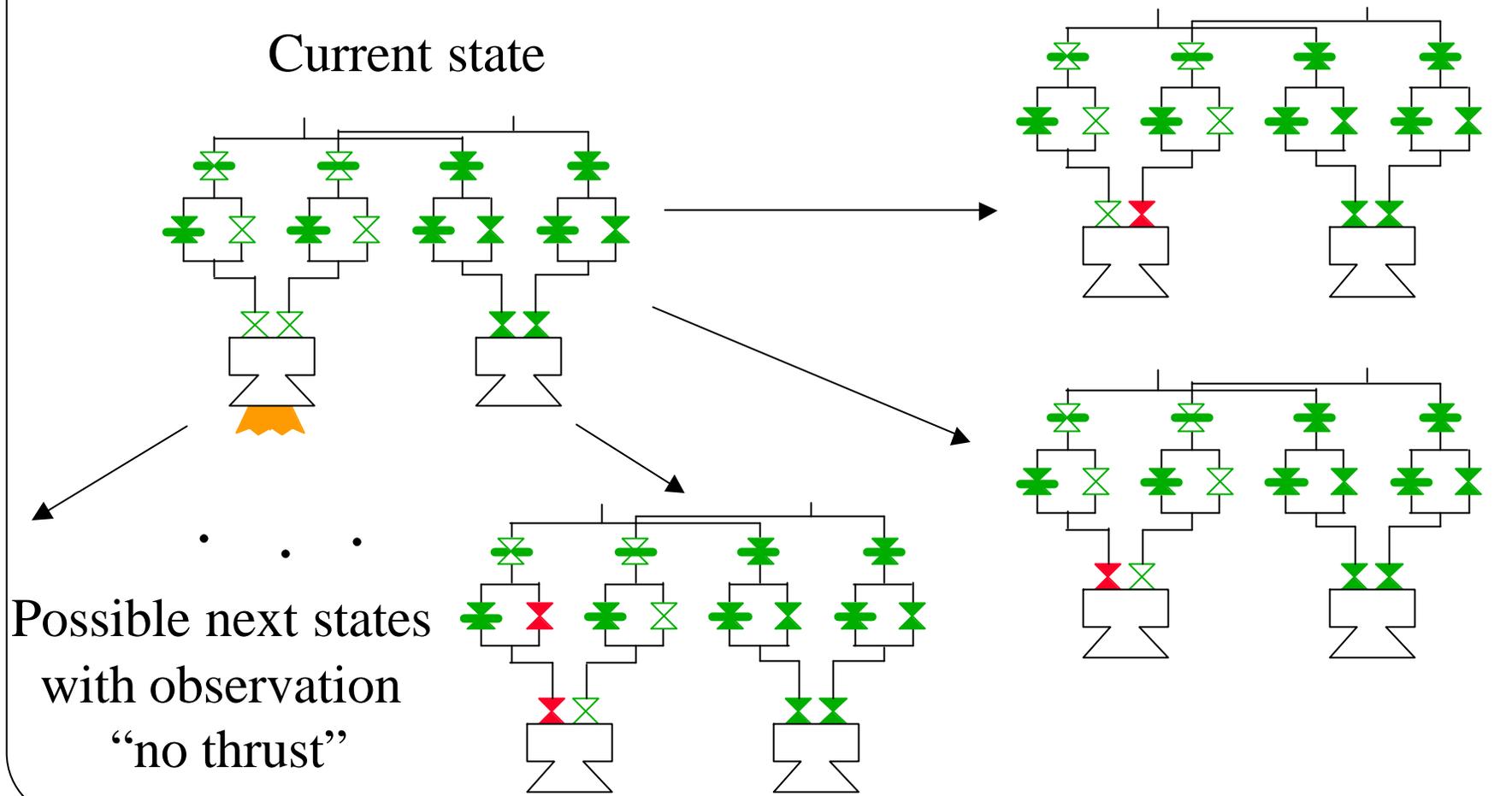
- $\mathbf{m}: \mathbf{m}_0, \mathbf{m}_1 \dots$, a sequence of values for all control variables so that S evolves through a *configuration trajectory* $\sigma: s_0, s_1 \dots$ such that
 - for every assignment a_i that agrees with s_i, o_i , and \mathbf{m}
 - **if $s_{i+1} = t_n(a_i)$, then s_{i+1} satisfies goal configuration g_i**

Configuration goal: Fire main engine



- Specified using a propositional logic formula

Mode Identification



Nayak/Williams

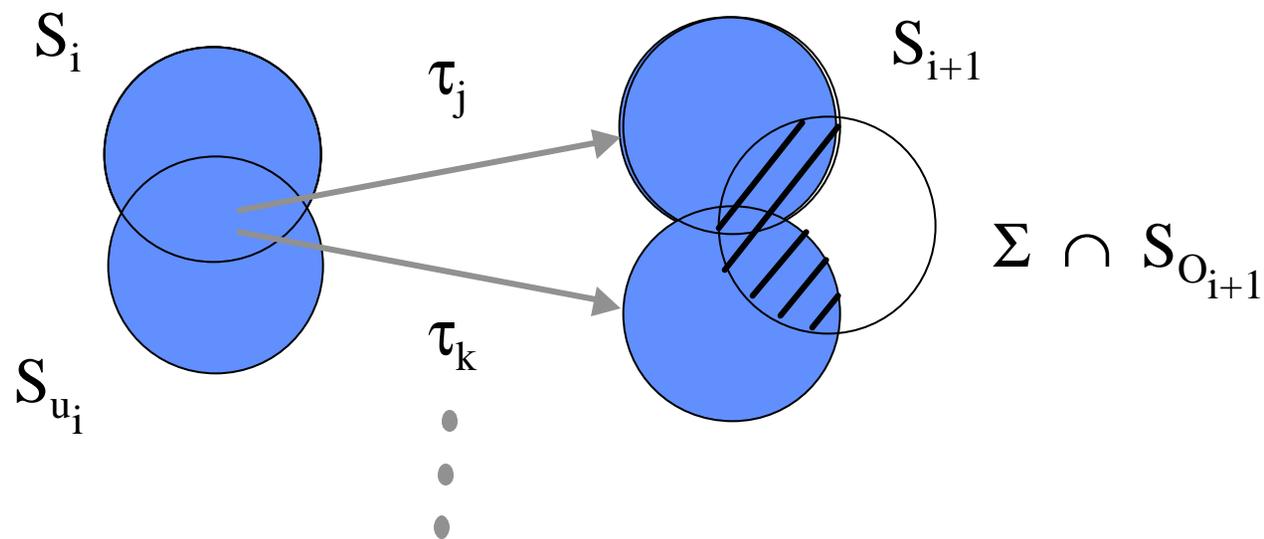
SP2-107

AAAI-97 Tutorial SP2

Characterizing MI

- Find possible next states, given current state commands and next state observations.

$$S_{i+1} = \left(\bigcup_j \tau_j(S_i \cap S_{m_i}) \right) \cap \Sigma \cap S_{O_{i+1}}$$



Characterizing MI

- Possible next states given current state commands and next state observations

$$S_{i+1} = \left(\bigcup_j t_j(S_i \cap S_{m_i}) \right) \cap \Sigma \cap S_{O_{i+1}}$$

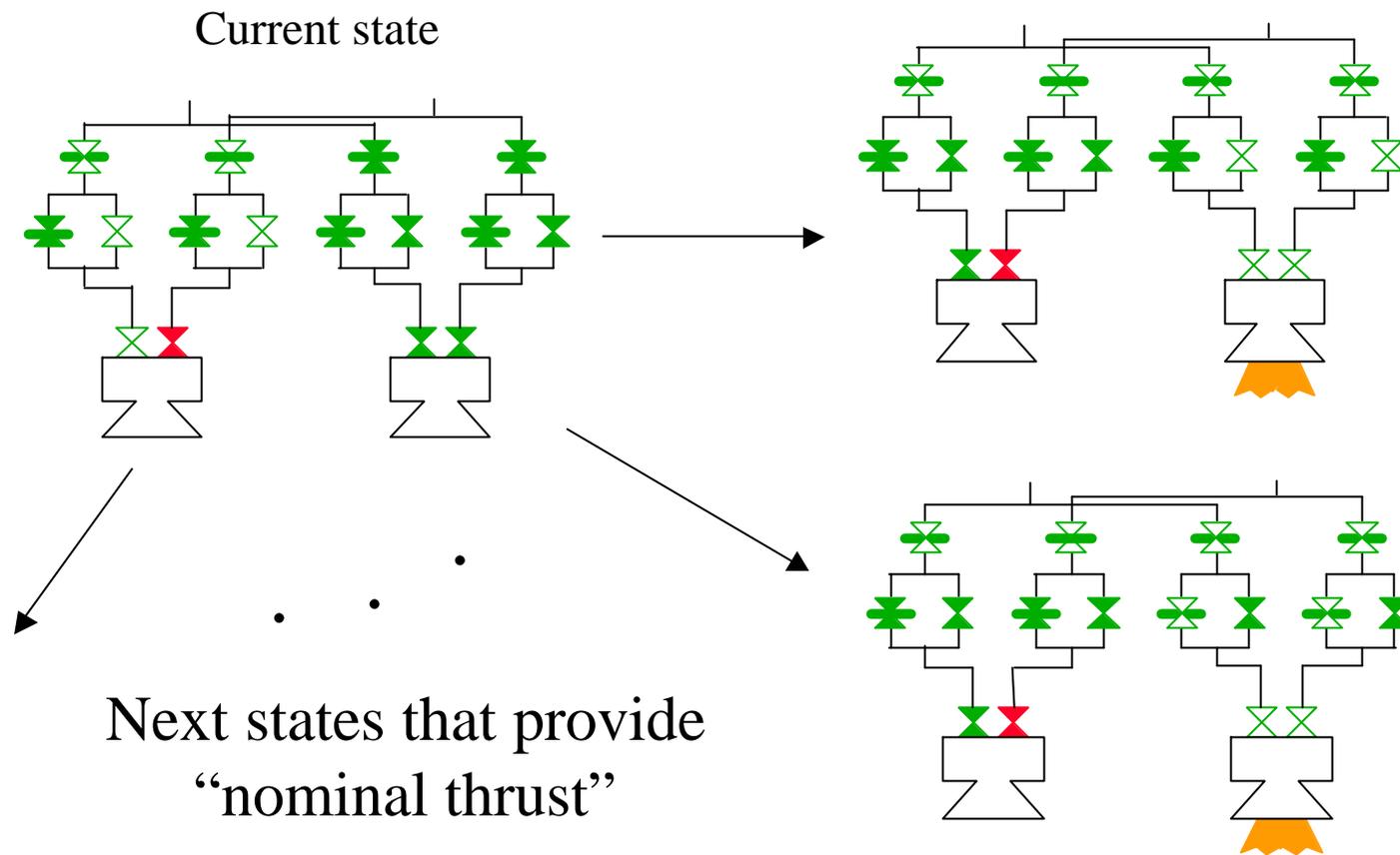
- Characterization of the possible next states

$$r_{S_{i+1}} \equiv \bigvee_j \left(t_j \left(r_{S_i} \wedge r_{S_{m_i}} \text{ entails } \Phi_{jk} \Psi_{jk} \right) \wedge r_{\Sigma} \wedge r_{O_{i+1}} \right)$$

where transition t_j is specified by a conjunction of formulas $\Phi_{jk} \Rightarrow next(\Psi_{jk})$

- Focus on likely transitions

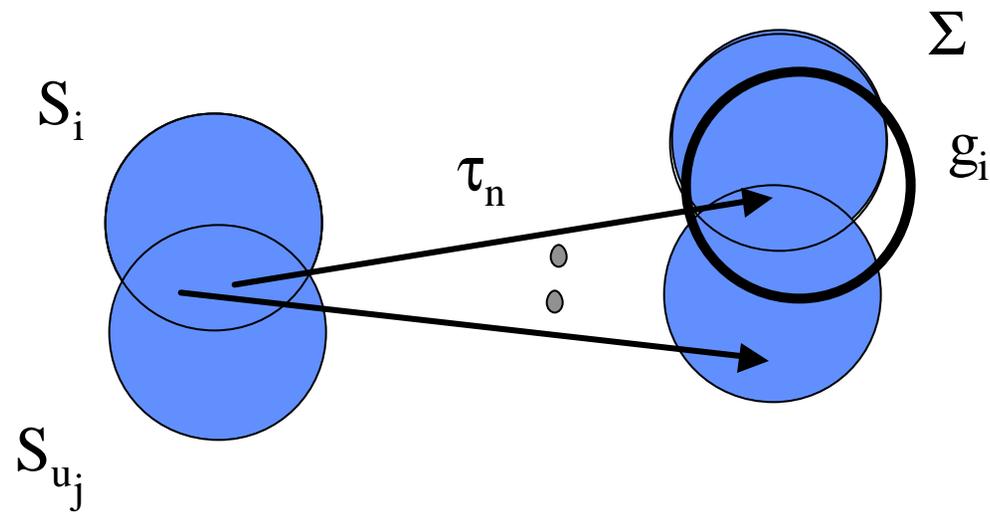
Mode Reconfiguration: Reachability in the next state



Characterizing MR

- Find possible commands that achieve the current goal in the next state.

$$M_i = \left\{ m_j \mid t_n (S_i \cap S_{m_j}) \cap \Sigma \subseteq g_i \right\}$$



Characterizing MR

- Possible commands that achieve the current goal in the *next* state for all predicted trajectories

$$M_i = \left\{ m_j \mid t_n (S_i \cap S_{m_j}) \cap \Sigma \subseteq g_i \right\}$$

- Characterization of the possible commands

$$M_i = \left\{ m_j \mid r_{S_i} \wedge r_{m_j} \text{ is consistent and} \right. \\ \left. \left(r_{S_i} \wedge r_{m_j} \overset{\wedge}{\text{entails}} \Phi_{nk} \Psi_{nk} \right) \wedge r_{\Sigma} \wedge r_{g_i} \text{ entails } r_{g_i} \right\}$$

- Focus on optimal control actions

Statistically Optimal Configuration Management

Statistical Mode Identification

- $p(\mathbf{t}_j | o_i) = p(o_i | \mathbf{t}_j) p(\mathbf{t}_j) / p(o_i)$ Bayes Rule
 $\propto p(o_i | \mathbf{t}_j) p(\mathbf{t}_j)$
- $p(o_i | \mathbf{t}_j)$ is approximated from the model
 - $p(o_i | \mathbf{t}_j) = 1$ if $\mathbf{t}_j(a_{i-1})$ entails o_i
 - $p(o_i | \mathbf{t}_j) = 0$ if $\mathbf{t}_j(a_{i-1})$ is inconsistent with o_i
 - $p(o_i | \mathbf{t}_j) = ?$ otherwise

Optimal Mode Reconfiguration

$$\mathbf{m} = \operatorname{argmin} c(\mathbf{m}') \text{ s.t. } \mathbf{m}' \text{ in } M_i$$

MI and MR as combinatorial optimization

MI

- Variables V : transitions taken by each component
- Feasibility f : consistency of resulting state with observations
- Cost c : derived from transition probabilities

MR

- Variables V : control variables of the system
- Feasibility f : the resulting state must entail the goal
- Cost c : derived from cost of control actions