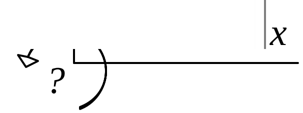
Mode Estimation Evolution

Oliver Martin
Tsoline Mikaelian

MERS Summer Seminar

- July 10, 2003 -

- Purpose of Estimation
  - Maintain accurate knowledge of the state of the system throughout time
- Hovercraft example
  - States:
    - Translation {x, y}
    - Attitude {?}
  - Sensors are only approximations, at best, due to noisy measurements

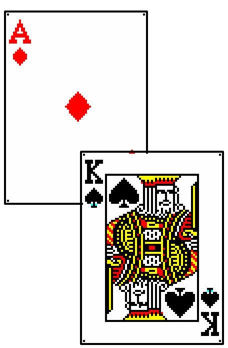


Estimation Fundamentals: Stationary Systems

- Goal
  - Find the most likely state, $\hat{s} \in S$, given an observation, $o \in O$

$$\hat{s} = \arg\max_{i=1..n} P(s_i \mid o)$$

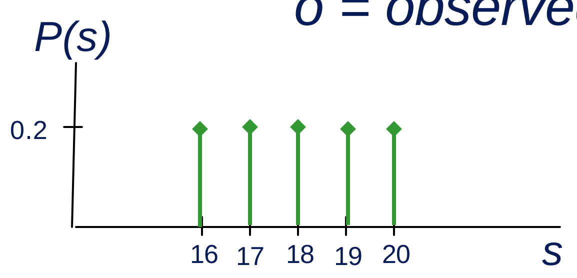
- Card Counting Example
  - Estimate the number of cards that have been played after 3 hands of blackjack
    - One Player
    - One Dealer

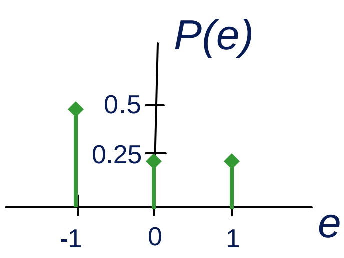


s = # of cards dealt

e = counting error

o = observed value = s + e





Card Counting Example Continued

- Joint PMF, $P(s \wedge o)$

$$P(\bigcirc) > P(\circ)$$

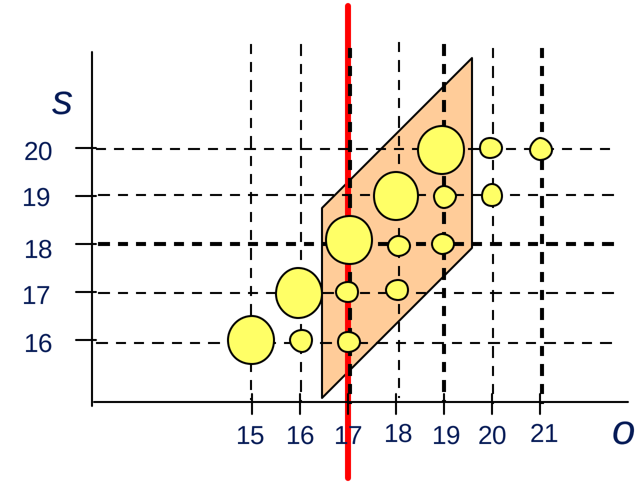
- Determine conditional probability of being in a state given an observation,

$$P(s \mid o) = \frac{P(s \wedge o)}{P(o)}$$

- Observe, o = 17

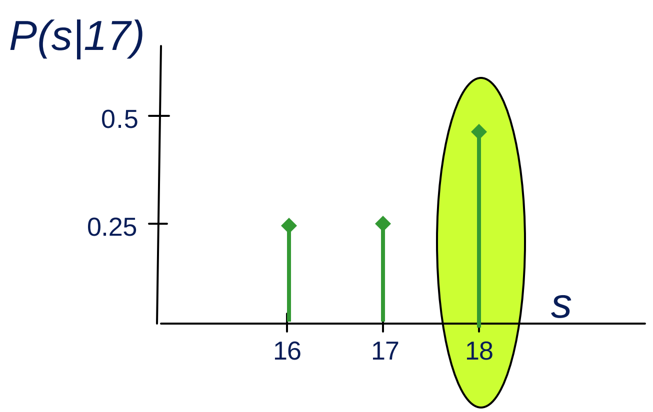
$$\hat{s} = \arg\max_{i=16..18} P(s_i \mid 17)$$

$$\hat{s} = 18$$



o = s + e

$$\hat{s} = \arg\max_{i=1..n} P(s_i \mid o)$$



Card Counting Example Continued

- Joint PMF, $P(s \wedge o)$

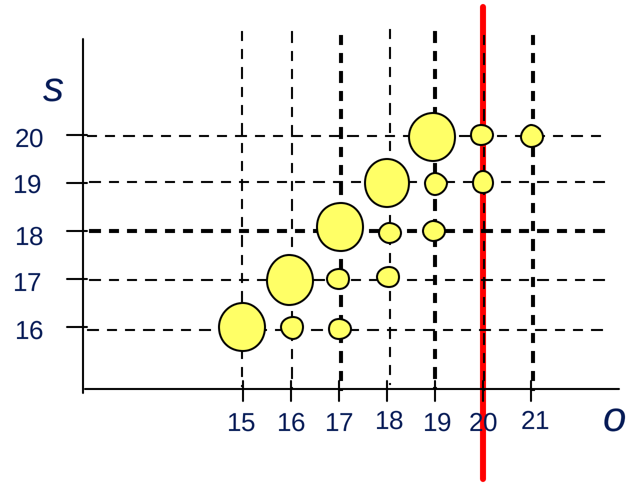
$$P(\bigcirc) > P(\circ)$$

- Determine conditional probability of being in a state given an observation,

$$P(s \mid o) = \frac{P(s \wedge o)}{P(o)}$$

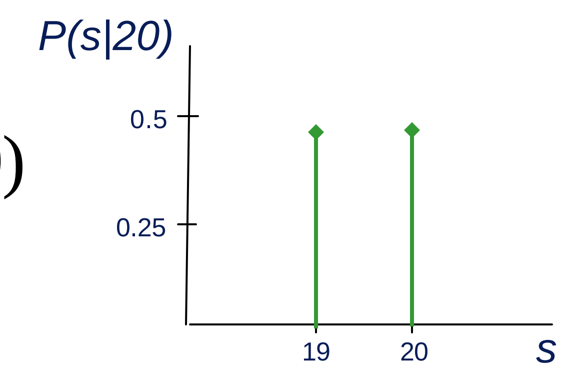
- Observe, o = 20

$$\hat{s} = \arg\max_{i=19,20} P(s_i \mid 20)$$



o = s + e

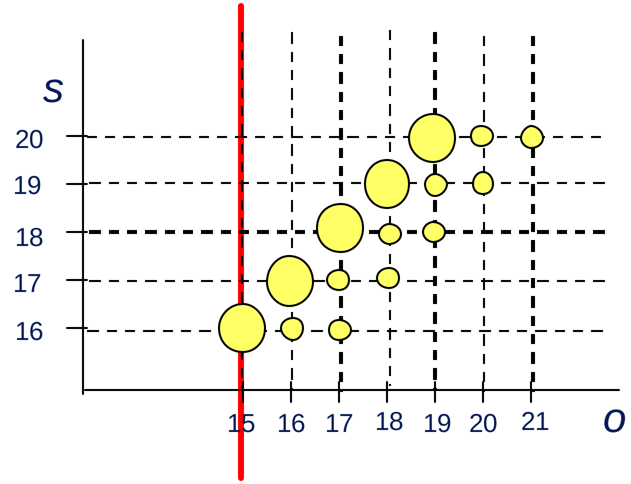
$$\hat{s} = \arg\max_{i=1..n} P(s_i \mid o)$$



Card Counting Example Continued

- Joint PMF, $P(s \wedge o)$

$$P(\bigcirc) > P(\circ)$$



$o = s + e$

- Determine conditional probability of being in a state given an observation,

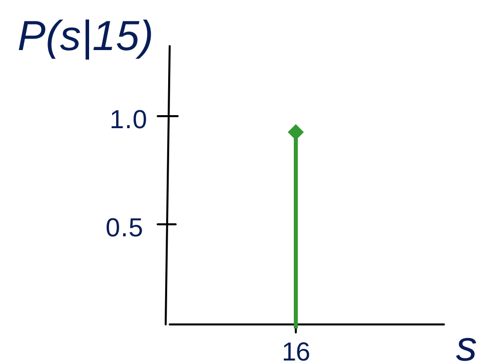
$$P(s \mid o) = \frac{P(s \wedge o)}{P(o)}$$

$$\hat{s} = \arg\max_{i=1..n} P(s_i \mid o)$$

- Observe, o = 15

$$\hat{s} = \arg\max_{i=16} P(s_i \mid 15)$$

$$\hat{s} = 16$$



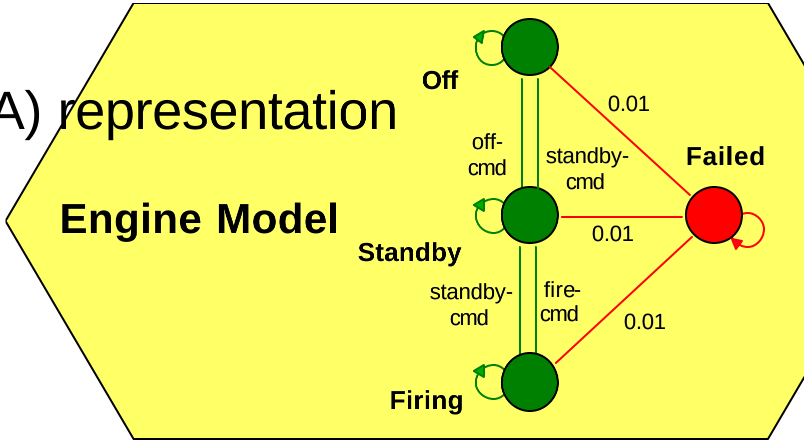
Estimation for Non-Stationary Systems

- Continuous Dynamics Model
  - Differential Equations

$$\ddot{\vec{x}} = \frac{1}{m}\left[A\left(\vec{x}_c - \vec{x}\right) + B\left(\dot{\vec{x}}_c - \dot{\vec{x}}\right)\right]$$

$$\begin{bmatrix} \dot{w}_x \\ \dot{w}_y \\ \dot{w}_z \end{bmatrix} = \begin{bmatrix} (I_y - I_z) w_y w_z / I_x + M_x / I_x \\ (I_z - I_x) w_x w_z / I_y + M_y / I_y \\ (I_x - I_y) w_x w_y / I_z + M_z / I_z \end{bmatrix}$$

  - Continuous Estimation: Kalman Filter
    - Propagate and Update
- Concurrent Constraint Automata (CCA) representation
  - Discrete Transitions



  - Mode Estimation: Belief State Update
    - Propagate (a priori probability)

$$P(s_j^{t+1} \mid o^{<0,t>}, m^{<0,t>}) = \sum_{i=1}^{n} P(s_i^t \mid o^{<0,t>}, m^{<0,t-1>}) P(s_j^{t+1} \mid s_i^t, m^t)$$

    - Update (posteriori probability)

$$P(s_j^{t+1} \mid o^{<0,t+1>}, m^{<0,t>}) = P(s_j^{t+1} \mid o^{<0,t>}, m^{<0,t>}) \frac{P(o^{t+1} \mid s_j^{t+1})}{\sum_{i=1}^{n} P(s_i^{t+1} \mid o^{<0,t>}, m^{<0,t>}) P(o^{t+1} \mid s_i^{t+1})}$$

Belief State Update Formulation

- Estimate the State, s

$$\hat{s} = \arg\max_{i=1..n} P(s_i \mid o) \quad \text{where} \quad P(s \mid o) = \frac{P(s \wedge o)}{P(o)}$$ *- Conditional Probability*

- Joint PMFs/PDFs can be difficult to compute → Bayes Rule

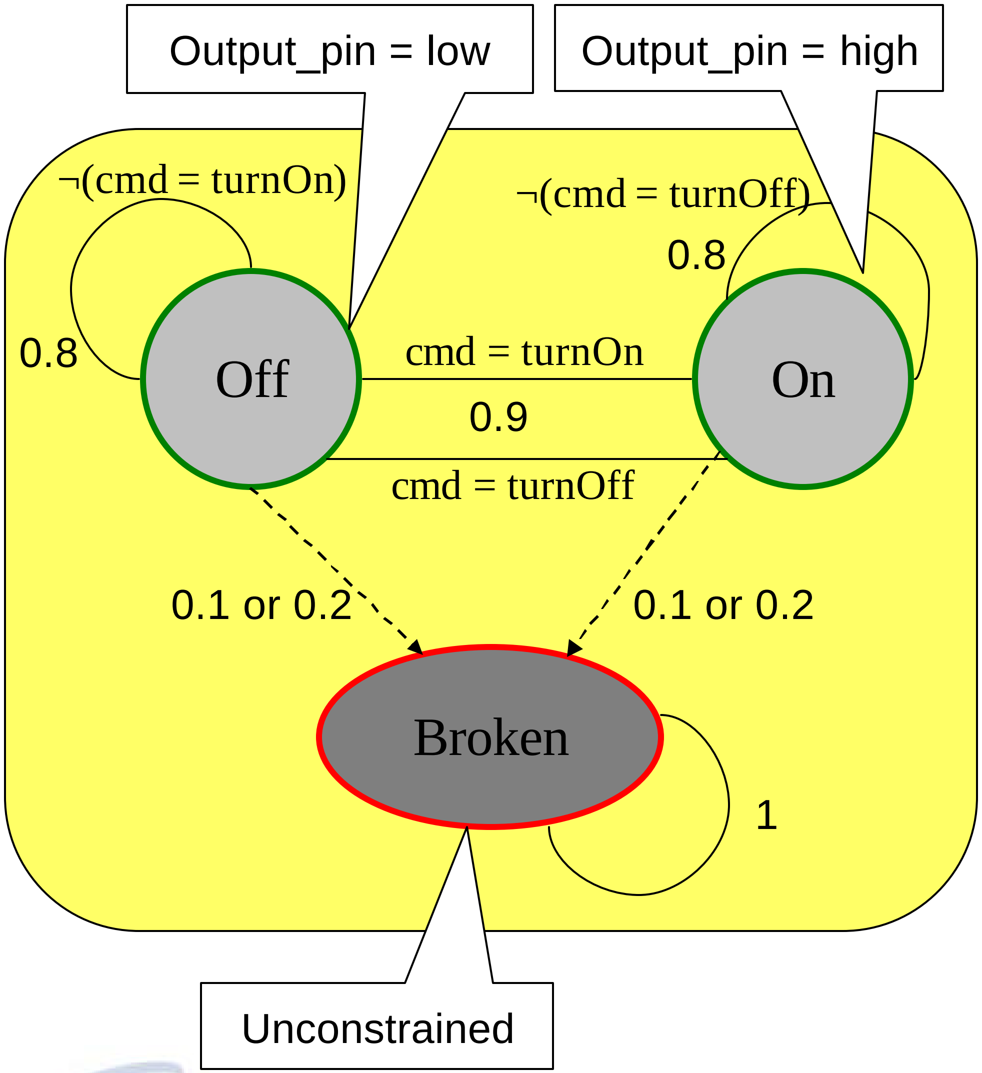
$$P(s \mid o) = \frac{P(s \wedge o)}{P(o)} = \frac{P(s)P(o \mid s)}{P(o)}$$ *- Conditional Probability*

$$P(s \mid o) = \frac{P(s)P(o \mid s)}{P(o)} = \frac{P(s)P(o \mid s)}{\sum_{o \in O} P(s)P(o \mid s)}$$ *- Law of Total Probability*

- Update Step in Belief State Update Equations

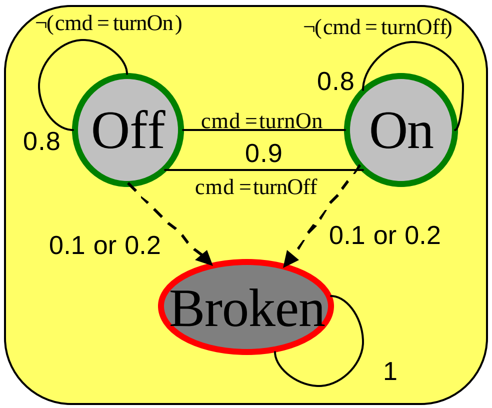
$$P(s_j^{t+1} \mid o^{<0,t+1>}, m^{<0,t>}) = P(s_j^{t+1} \mid o^{<0,t>}, m^{<0,t>}) \frac{P(o^{t+1} \mid s_j^{t+1})}{\sum_{i=1}^{n} P(s_i^{t+1} \mid o^{<0,t>}, m^{<0,t>}) P(o^{t+1} \mid s_i^{t+1})}$$

Non-Stationary System Example



- State Variables, Π^s
  - *Switch*
    - *{On, Off, Broken}*
- Command Variables, Π^c
  - *cmd*
    - *{on, off, no-command}*
- Observable Variables, Π^o
  - *Output_pin*
    - *{low, high}*

Non-Stationary System Example



- Initially: {$P(Off^0)=0.7$, $P(Broken^1)=0.3$}
- System command = turn On
- Propagate States

$$P(s_j^{t+1} \mid o^{<0,t>}, m^{<0,t>}) = \sum_{i=1}^{n} P(s_i^t \mid o^{<0,t>}, m^{<0,t-1>}) P(s_j^{t+1} \mid s_i^t, m^t)$$

  – On

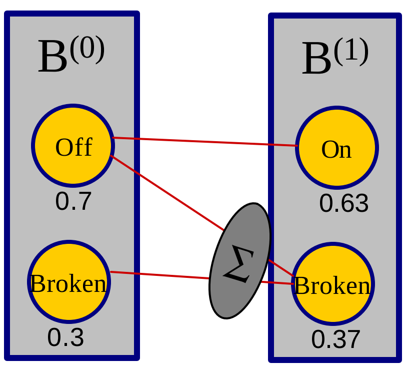
$$P(on^1 \mid cmd = turnOn^0) = P(off^0) \cdot P(on^1 \mid off^0, cmd = turnOn^0)$$

$$P(on^1 \mid cmd = turnOn^0) = 0.7 \cdot 0.9$$

  – Broken

$$P(broken^1 \mid cmd = turnOn^0) = P(off^0) \cdot P(broken^1 \mid off^0, cmd = turnOn^0) + P(broken^0) \cdot P(broken^1 \mid broken^0, cmd = turnOn^0)$$

$$P(broken^1 \mid cmd = turnOn^0) = 0.3 * 1 + 0.7 * 0.1$$



- Update States

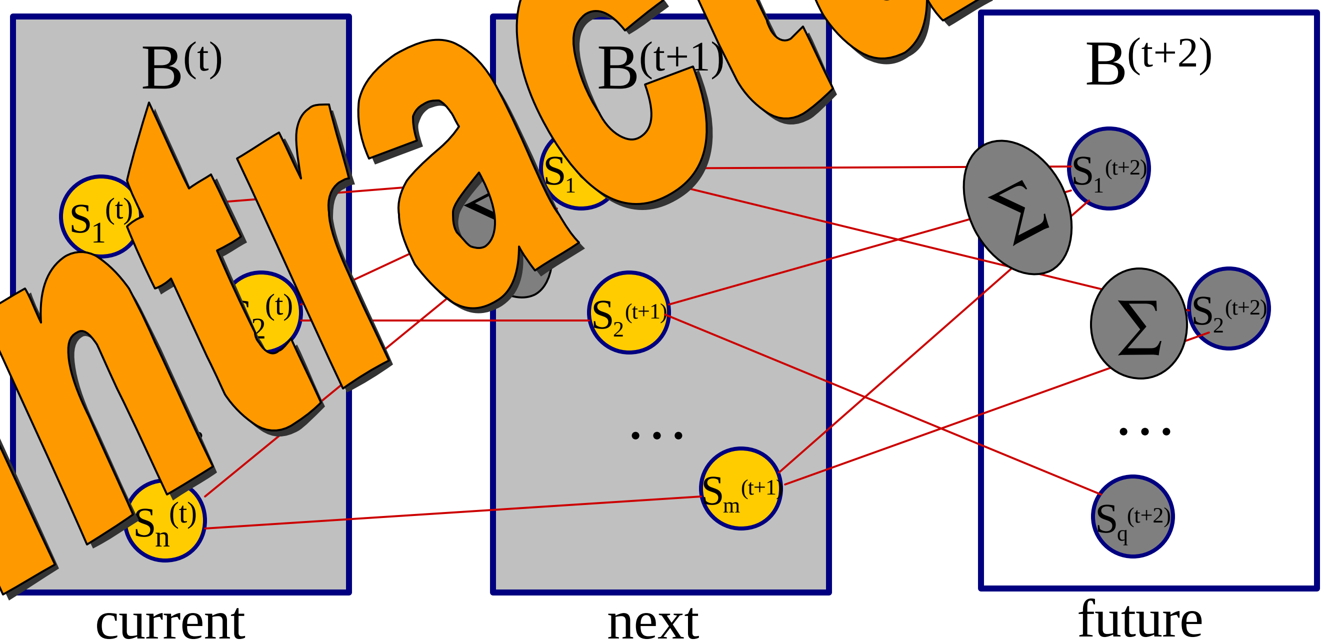
$$P(s_j^{t+1} \mid o^{<0,t+1>}, m^{<0,t>}) = P(s_j^{t+1} \mid o^{<0,t>}, m^{<0,t>}) \frac{P(o^{t+1} \mid s_j^{t+1})}{\sum_{i=1}^{n} P(s_i^{t+1} \mid o^{<0,t>}, m^{<0,t>}) P(o^{t+1} \mid s_i^{t+1})}$$

Full Belief State

- Requires:
  - P, $B^{(t)}$, $p^{(t\bullet)}$, $\mu^{(t)}$, $o^{(t+1)}$
- Computes:
  - $B^{(t+1)}$, $p^{(t+1\bullet)}$

- Size-S...
  - c: # of components
  - n: # of modes

Intractable

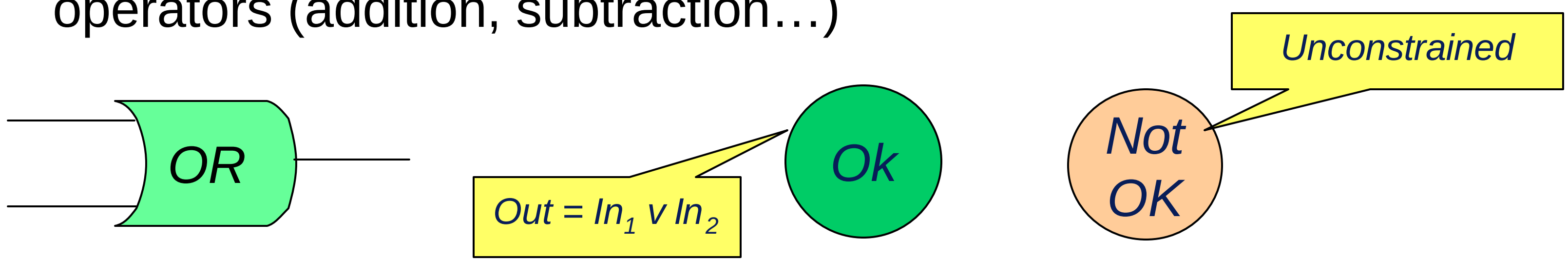


Mode Estimation Overview

- ~~Introduction~~
- Previous Mode Estimation Algorithms
  - General Diagnostic Engine (GDE)
  - Sherlock
  - Mine-Me
  - Livingstone
- Current Mode Estimation Algorithms
  - Single State Trajectory (Titan 1.0)
  - Most Likely Trajectories (Titan 1.1)
- Future Mode Estimation Algorithms
  - **Compiled Mode Estimation (Titan 2.0)**
  - **N-Step Estimation**

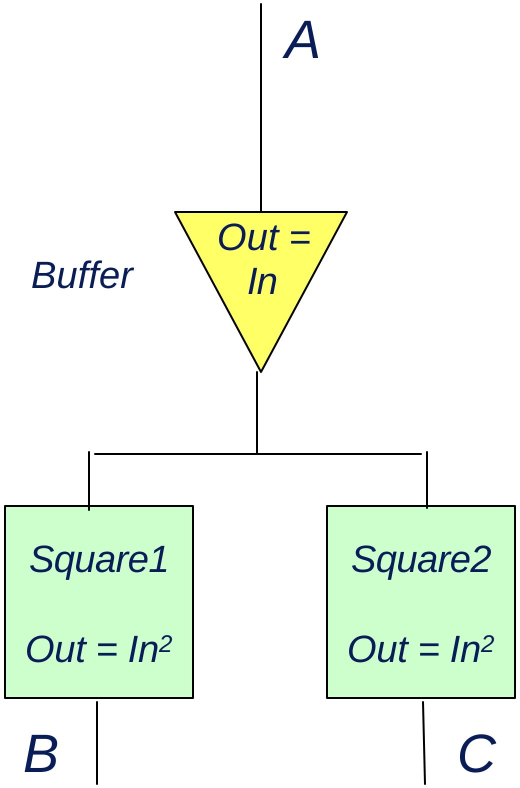
General Diagnostic Engine (GDE)

- Developed by deKleer and Williams in 1987
- Components modes limited to “ok” and “not okay”
  - Constrained based on observations
  - Useful for logic systems (and, or, not…) and mathematical operators (addition, subtraction…)



- Uses Divide and Conquer to generate diagnoses

GDE Example



Observe: A = 2, B = 5, C = 5

- Conflict Recognition

  Conflict is a partial set of assignments to mode variables which are inconsistent with observations

  - Ex:

$$\neg[(buffer = ok) \wedge (square1 = ok)]$$

$$\neg[(buffer = ok) \wedge (square2 = ok)]$$

- Candidate Generation
  - Constituent Diagnoses

$$[\neg(buffer = ok) \vee \neg(square1 = ok)]$$

$$[\neg(buffer = ok) \vee \neg(square2 = ok)]$$

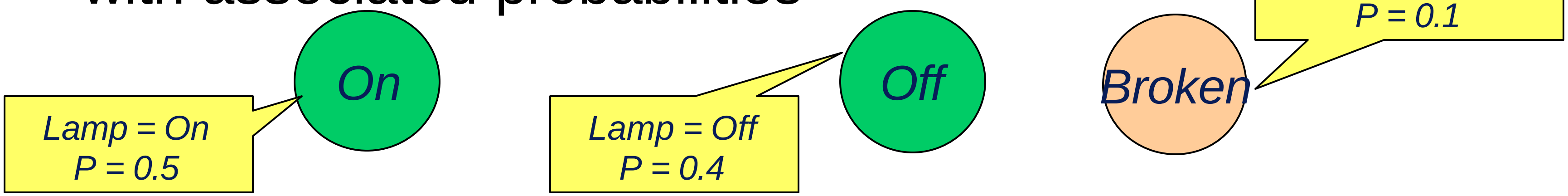
  - Kernel Diagnoses

$$[(buffer = not - okay)]$$

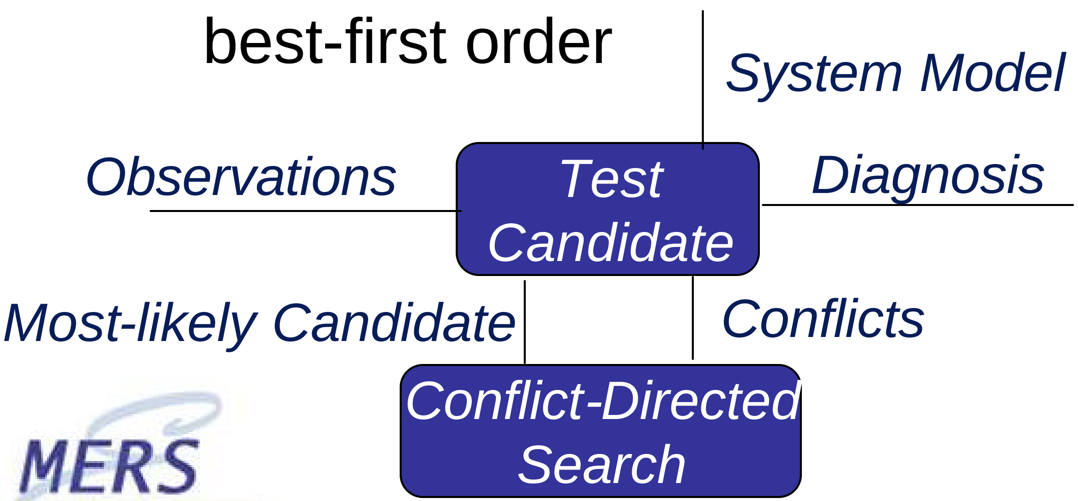
$$[(square1 = not - okay) \wedge (square2 = not - okay)]$$

Sherlock

- Developed by deKleer in 1989
- Similar to GDE but specifies nominal and fault modes with associated probabilities



- Added conflict directed search to prioritize diagnoses in best-first order



$$P(C_i \mid o) = \frac{P_i(o \mid C_i) \cdot P(C_i)}{\sum_j P_j(o \mid C_j) \cdot P(C_j)}$$

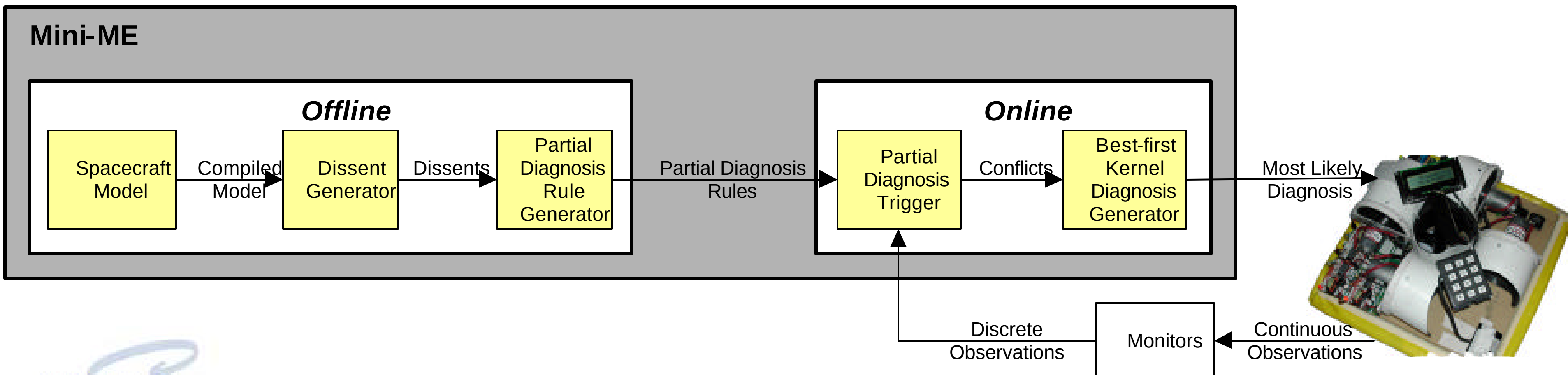
where

$$P(C_i) = \prod_i P(x_i = v)$$

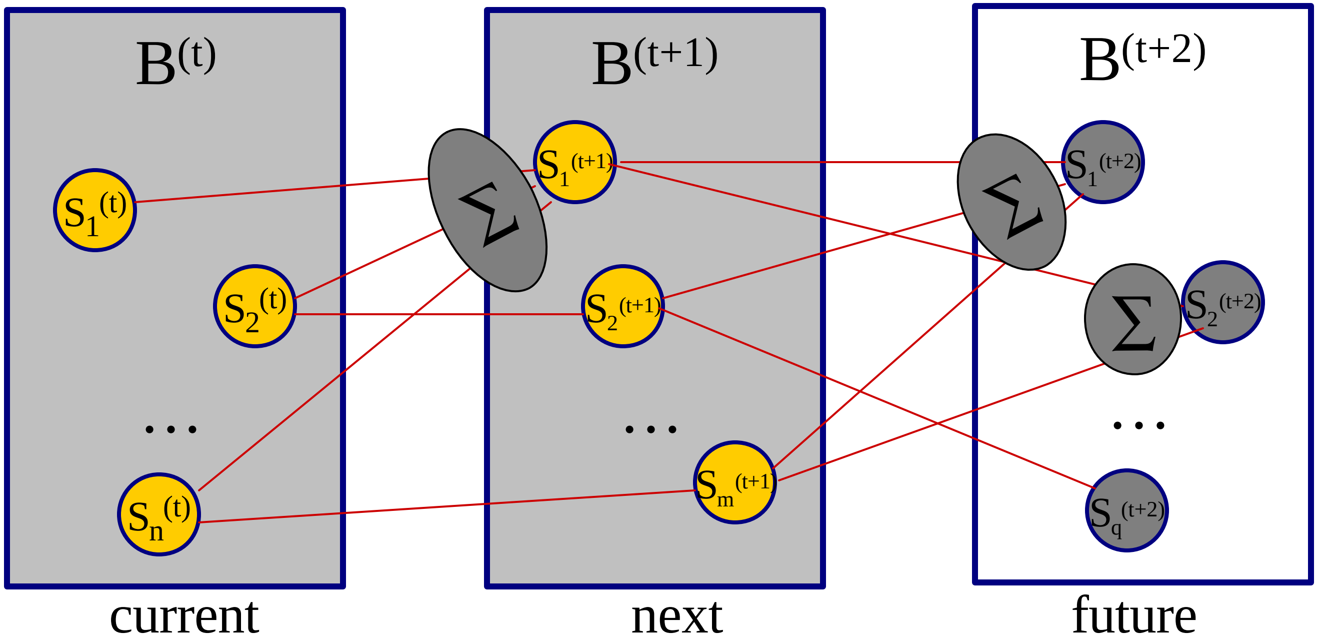
Mini-Me?

Miniature Mode Estimation (Mini-ME)

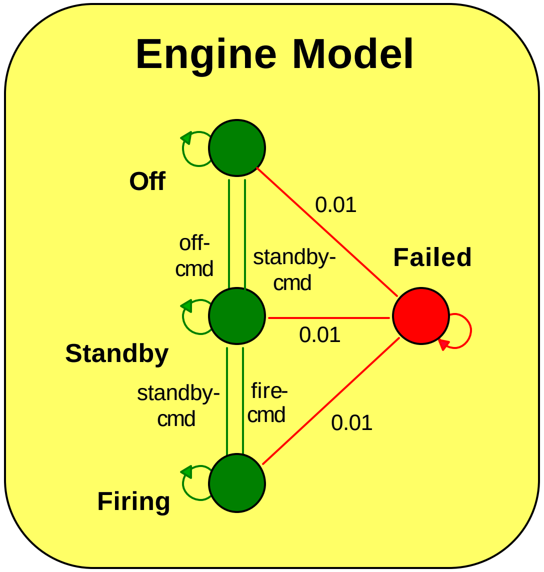
- Developed by Chung and Williams in 2001
- Same component models as Sherlock with nominal and fault modes and associated probabilities
- Increased performance by removing the need for satisfiability and online conflict generation through compilation



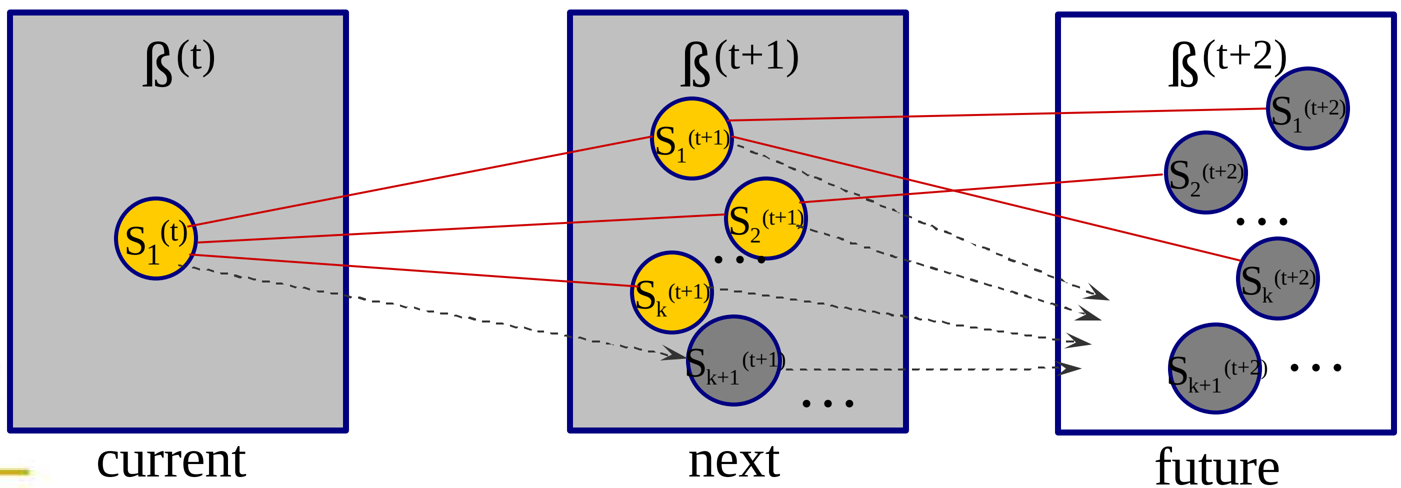
Moving towards a Full Belief State



Livingstone Mode Estimation



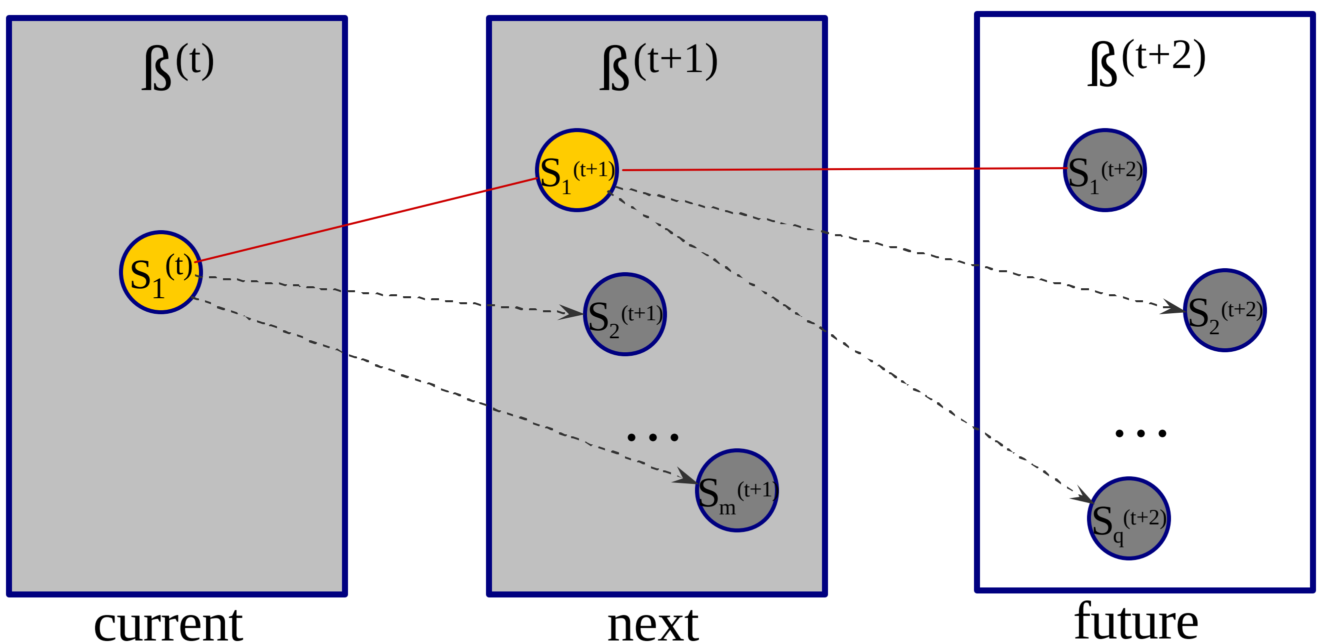
- Developed by Williams in 1996
- Same strategy as Sherlock
  - Conflict directed search to generate estimates in best-first order (any-time)
- Two key differences:
  - Uses CCA model representation with transitions
  - Tracks an Approximate Belief State over time



Single State Trajectory

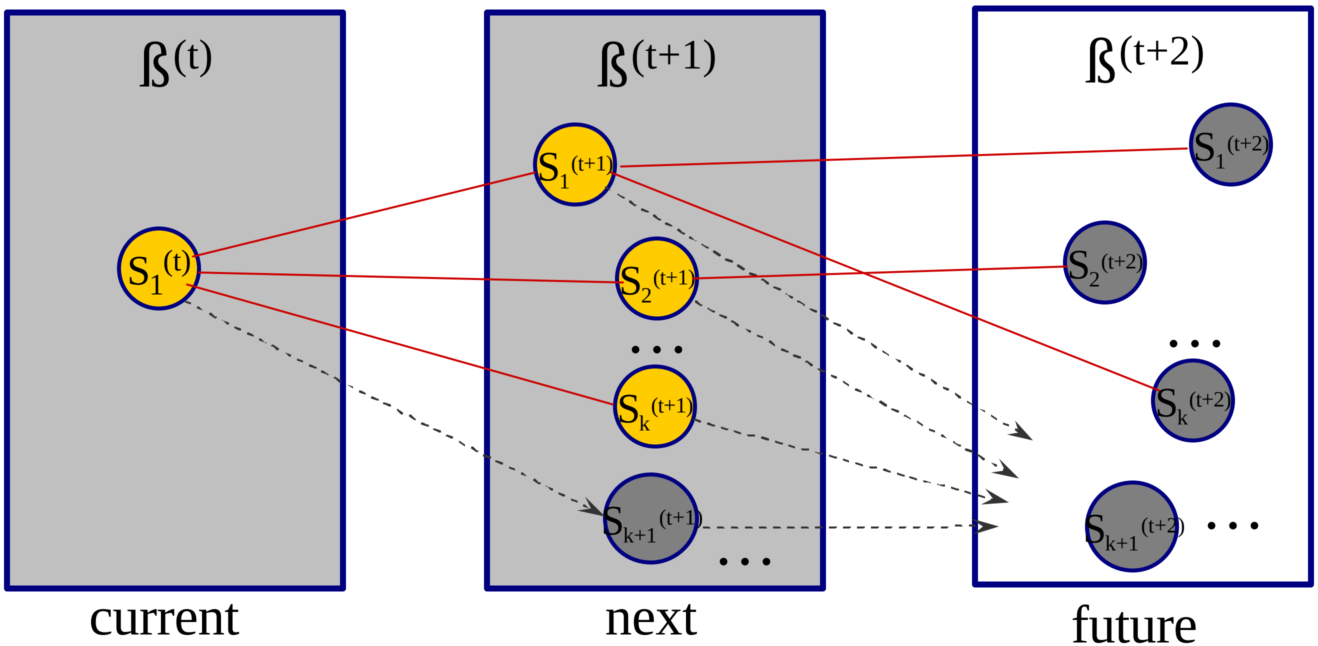
- Titan 1.0 Mode Estimation
  - Computes the *single* most likely state trajectory, s_1 Î ß, that is consistent with observations, $o^{(t+1)}$, and commands, $\mu^{(t)}$

- Enumerated in Best-First order using OPSAT



Most Likely Trajectories

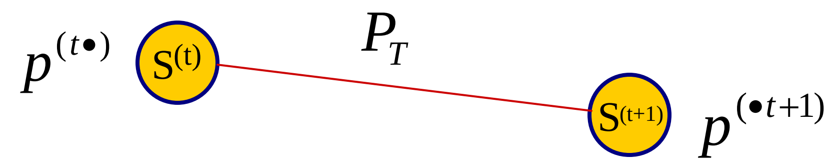
- Titan 1.1 Mode Estimation
  - Computes the *K* most likely state trajectories, $s_{1..K}$ Î ß, that is consistent with observations, $o^{(t+1)}$, and commands, $\mu^{(t)}$



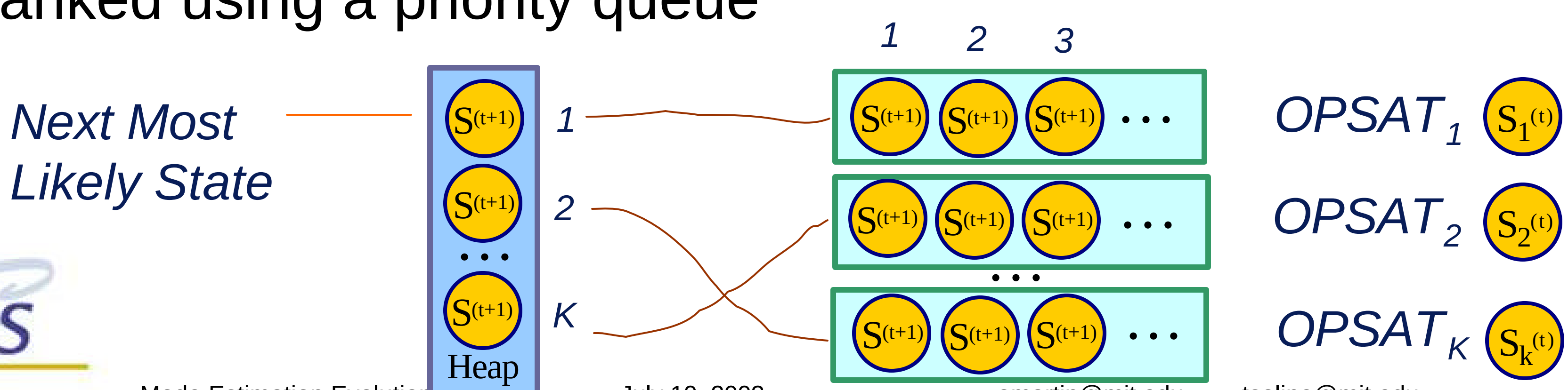
Each trajectory is considered unique and are not summed together when they converge to the same state

Probabilistically Incomplete

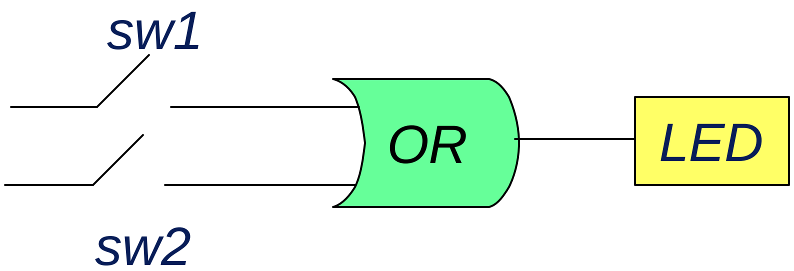
Most Likely Trajectory Calculation



- Probability Calculations
  - a priori $P(s_j^{t+1} \mid o^{<0,t>}, m^{<0,t>}) = \sum_{i=1}^{n} P(s_i^t \mid o^{<0,t>}, m^{<0,t-1>}) P(s_j^{t+1} \mid s_i^t, m^t)$
  - posteriori $P(s_j^{t+1} \mid o^{<0,t+1>}, m^{<0,t>}) = P(s_j^{t+1} \mid o^{<0,t>}, m^{<0,t>}) \dfrac{P(o^{t+1} \mid s_j^{t+1})}{\sum_{i=1}^{n} P(s_i^{t+1} \mid o^{<0,t>}, m^{<0,t>}) P(o^{t+1} \mid s_i^{t+1})}$
  - Normalize over K estimates*
- Ranked using a priority queue



Demo Results



Cmd1 = *turnOn-sw1*
Observation1: *LED light=off*
Cmd2 = *turnOn-sw2*
Observation2: *LED light=off*

Single Trajectory (previous implementation)

Sw1 = Broken & Sw2 = Broken
(Probability of Switch breaking is highest for each time-step)

Multiple Trajectories (new implementation)

Sw1 = Broken & others = nominal

Sw2 = Broken & others = nominal

OR-gate = Broken & others = nominal

OR-gate = not Broken & others = nominal

Sw1 = broken & Sw2 = broken & Or-gate = not broken

After Cmd2 and Obs2

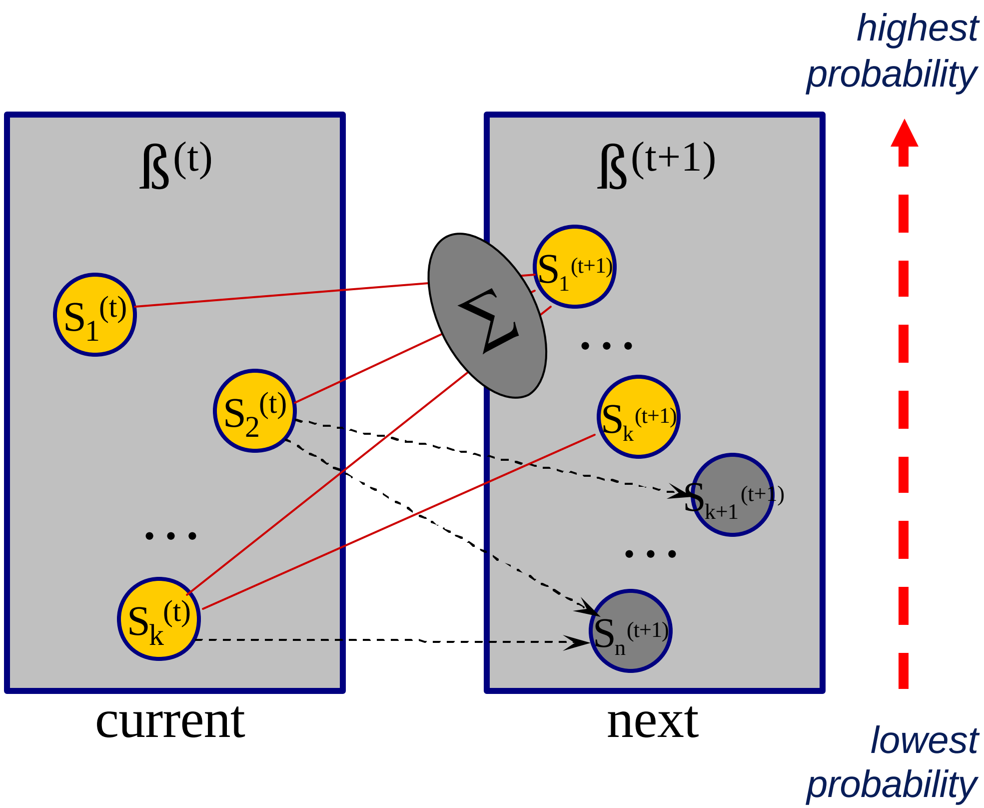
⋮

Compiled Mode Estimation

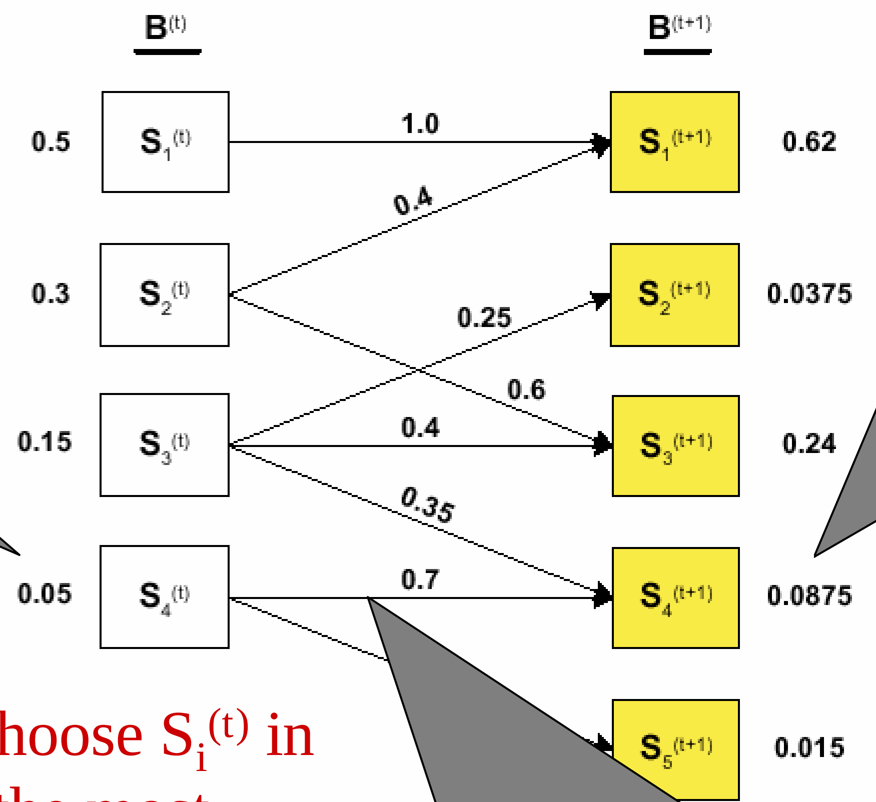
- Computes an approximate belief state, s_i *Î* ß, that is consistent with observations, $o^{(t+1)}$, by enumerating mode estimates in best-first order

- Decreases the trajectory search space by compiling out model constraints offline
  - *Mini-ME*

- Uses Conflict-Directed A* (CDA*) to find most likely trajectories, eliminating the need for a satisfiability engine
  - Sherlock/Livingstone



Dynamic Mode Estimate Generation



1. Choose a previous mode estimate, $S_i^{(t)}$

- generate algorithm

2. Choose the most likely transition from $S_i^{(t)}$ to $S_j^{(t+1)}$

- Conflict Directed A*

3. Determine all transitions from within $ß^{(t)}$ to $S_j^{(t+1)}$ to calculate $P^{(\bullet t+1)}[S_j]$

- rank algorithm

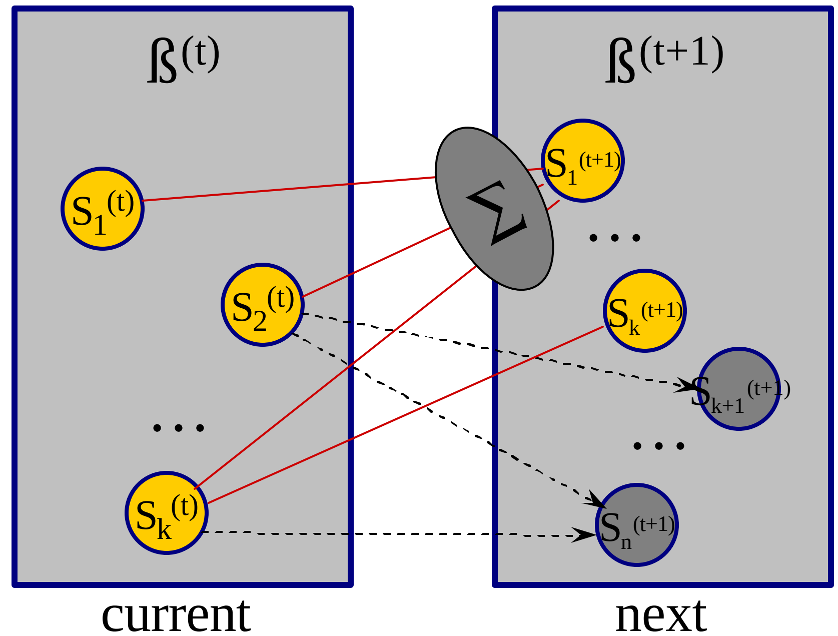
How do we choose $S_i^{(t)}$ in order to find the most likely $S_j^{(t+1)}$?

Generation Algorithm

- Determines a current state estimate which will generate the next most likely state estimate

- Modified A* Search

$$Cost[s_j^{t+1}] = \overbrace{p(s_j^{t+1}, s_i^{t} \mid m^{<1,t>}, o^{<1,t>})}^{\text{Uniform Cost}} + \overbrace{R}^{\text{Greedy}}$$



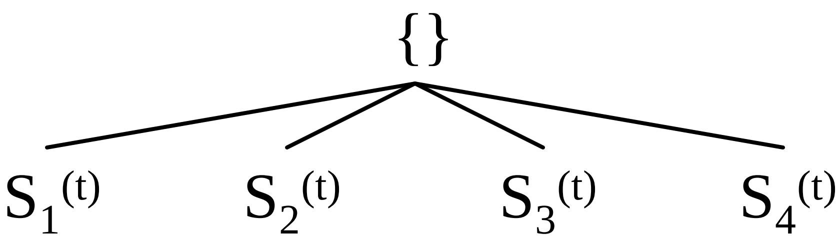
where

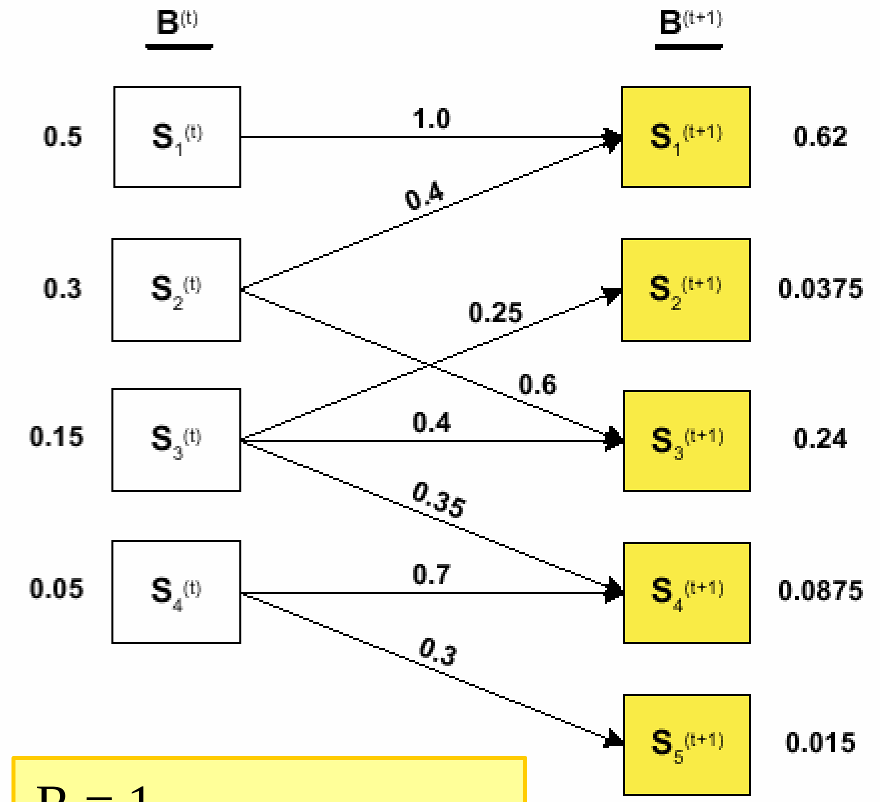
$$p(s_j^{t+1}, s_i^{t} \mid m^{<1,t>}, o^{<1,t>}) = p(s_j^{t+1} \mid s_i^{t}, m^{t})\, p(s_i^{t} \mid m^{<1,t-1>}, o^{<1,t>})$$

$$R = 1 - \sum_{s_j \in S^{t+1}} p(s_j^{t+1} \mid m^{<1,t>}, o^{<1,t>})$$

Initialize Search Tree

- Previous mode estimates are placed in order of highest posterior probability





R = 1

C_1= C_2= C_3= C_4=

Enumerated:

Iteration #1

- Expand most likely trajectory from most probable previous mode estimate
  → $S_1^{(t)}$

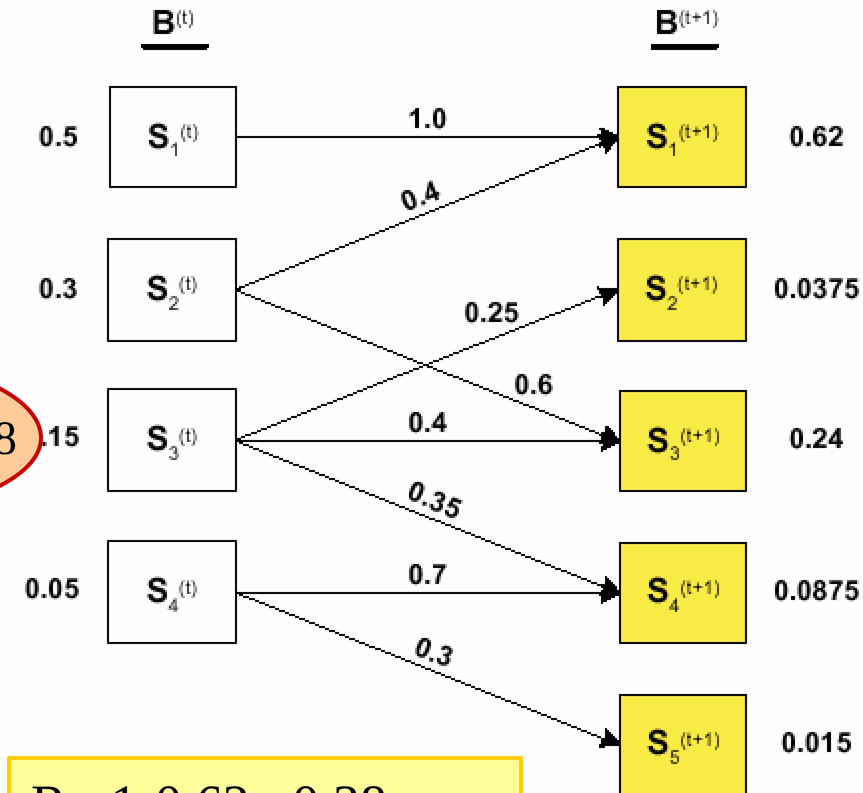
C_1=0.5*1.0+0.38

{}

$S_1^{(t)}$ $S_2^{(t)}$ $S_3^{(t)}$ $S_4^{(t)}$

$S_1^{(t+1)}$

C_1=0.88 C_2= C_3= C_4=



R =1-0.62= 0.38

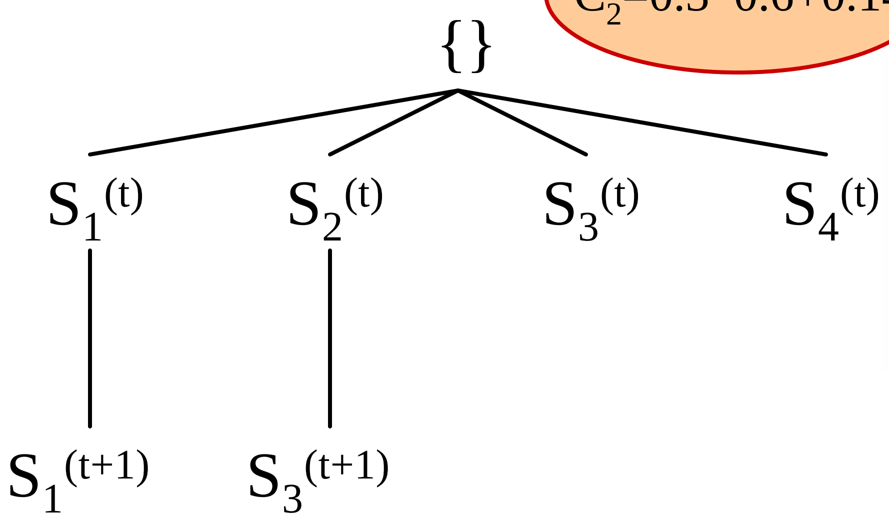
Enumerated:
$S_1^{(t+1)}$

Iteration #2

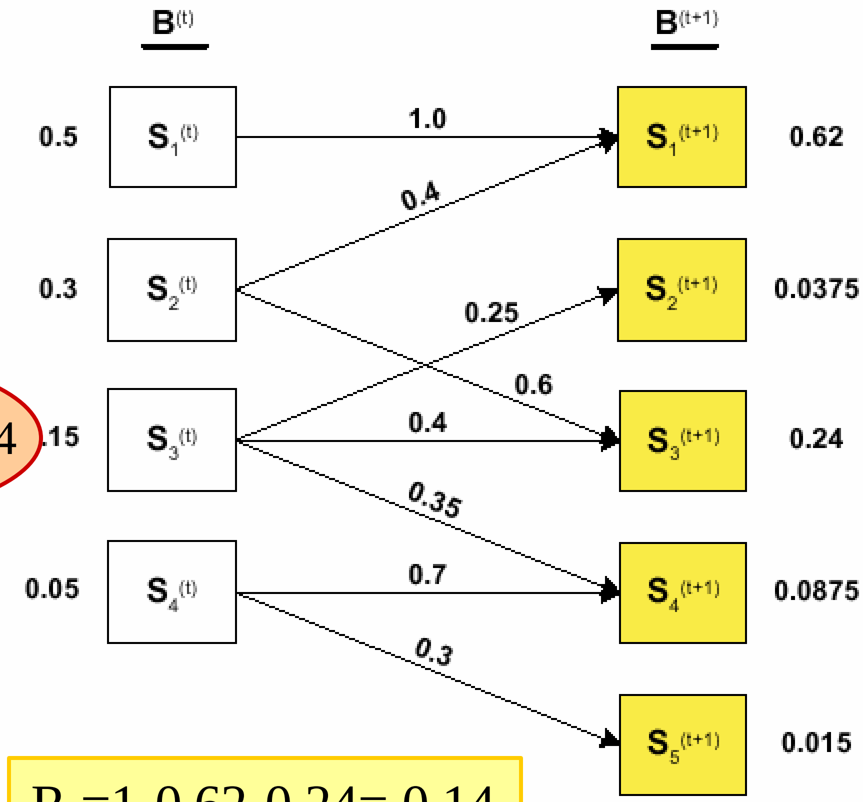
- Continue to expand next state in $B^{(t)}$ but <u>not highest cost</u>

  $\rightarrow S_2^{(t)}$

$C_2=0.3*0.6+0.14$



C_1=0.64 C_2= 0.32 C_3= C_4=



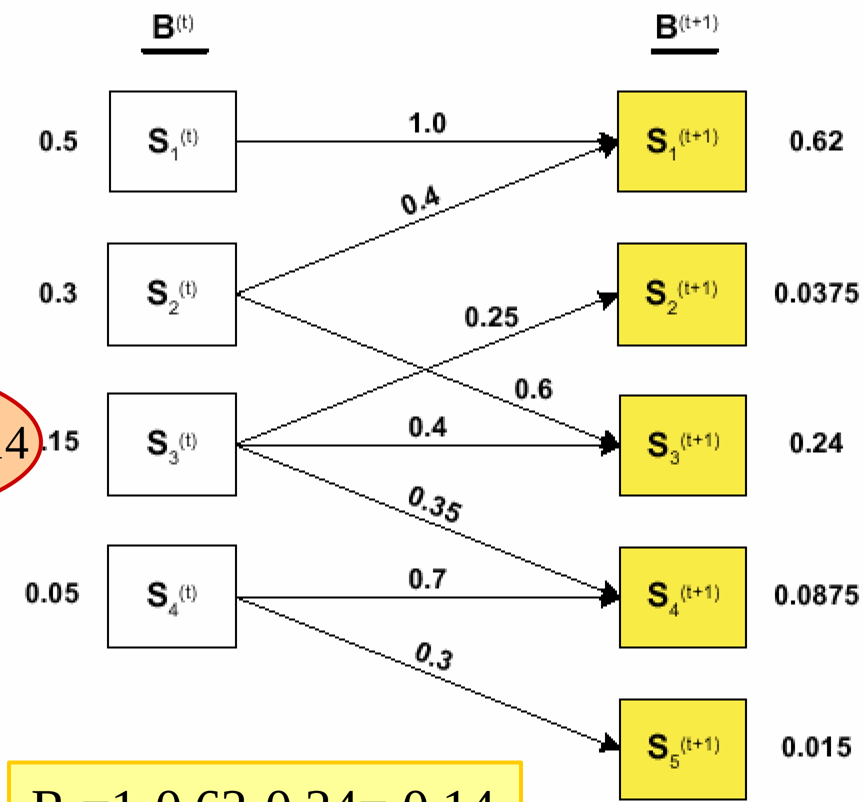
R =1-0.62-0.24= 0.14

Enumerated:
$S_1^{(t+1)}$, $S_3^{(t+1)}$

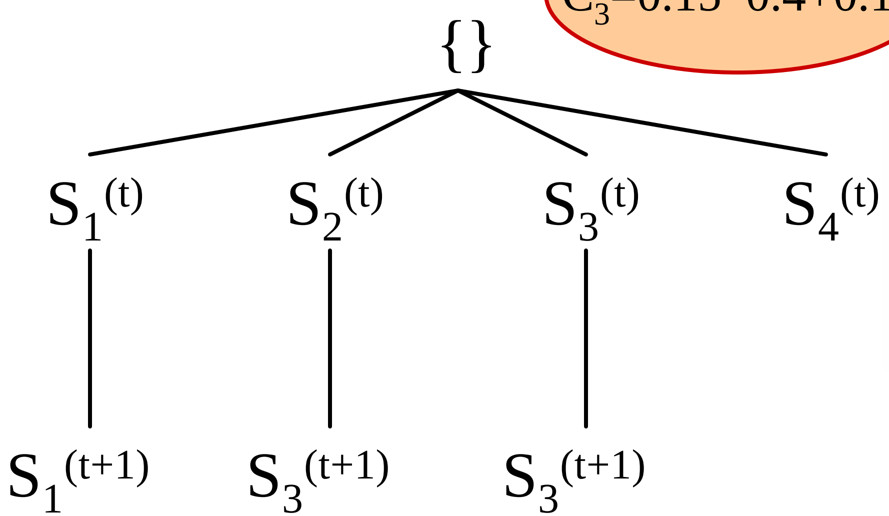
Iteration #3

- Continue to expand next state in $B^{(t)}$ but <u>not highest cost</u>

  → $S_3^{(t)}$



C_3=0.15*0.4+0.14



C_1= 0.64 C_2= 0.32 C_3= 0.20 C_4=

R =1-0.62-0.24= 0.14

Enumerated:
$S_1^{(t+1)}$, $S_3^{(t+1)}$

Iteration #4

- Continue to expand next state in $B^{(t)}$ but <u>not highest cost</u>

  → $S_4^{(t)}$

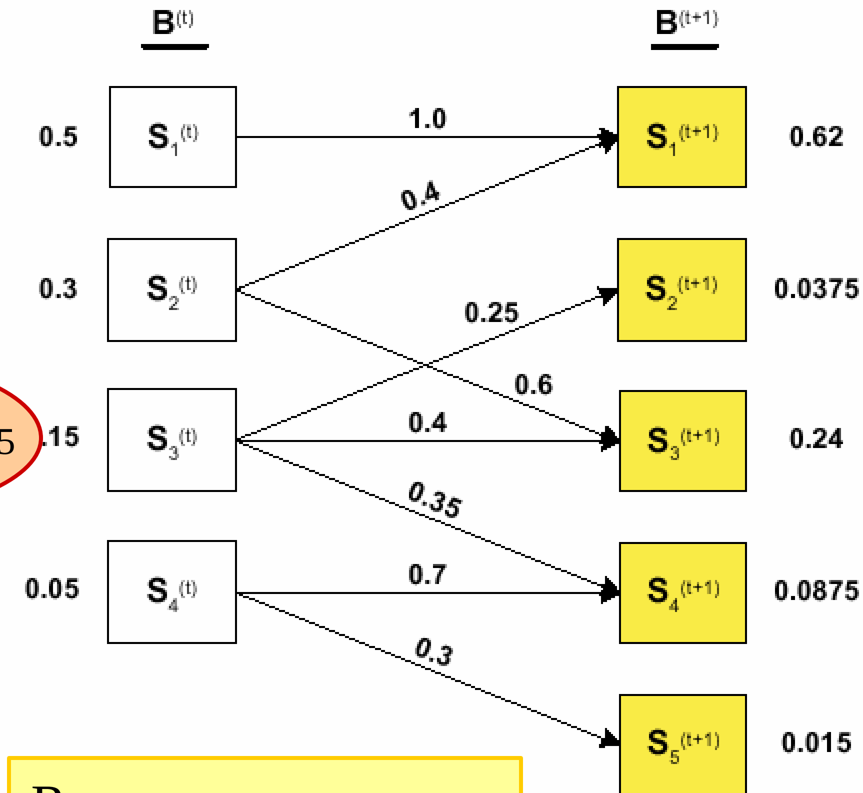
C_4=0.05*0.7+0.0525

{}

$S_1^{(t)}$ $S_2^{(t)}$ $S_3^{(t)}$ $S_4^{(t)}$

$S_1^{(t+1)}$ $S_3^{(t+1)}$ $S_3^{(t+1)}$ $S_4^{(t+1)}$

C_1= 0.5525 C_2= 0.2325 C_3= 0.1125 C_4= 0.0875



R =1-0.62-0.24-0.0875= 0.0525

Enumerated:

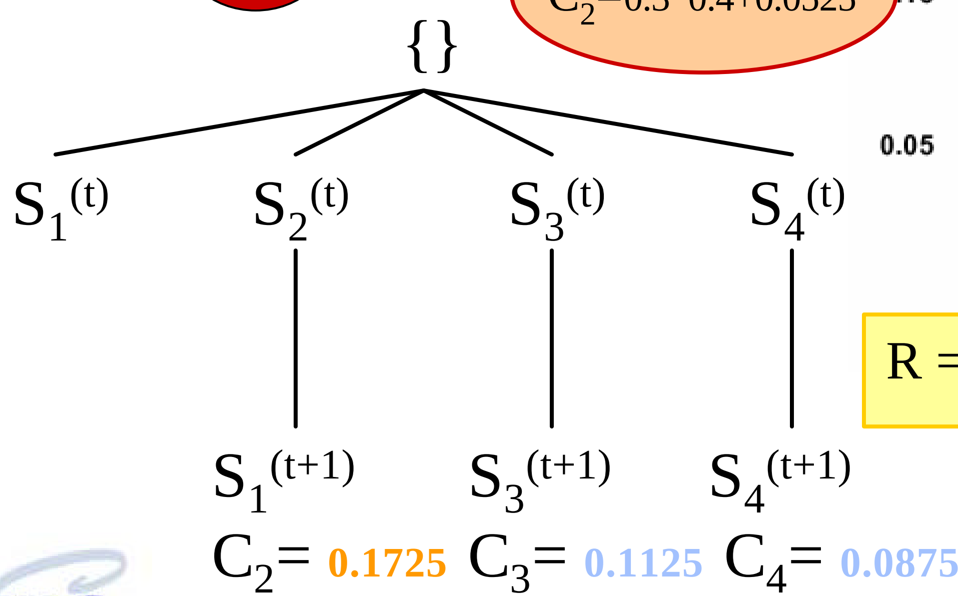
$S_1^{(t+1)}$, $S_3^{(t+1)}$, $S_4^{(t+1)}$

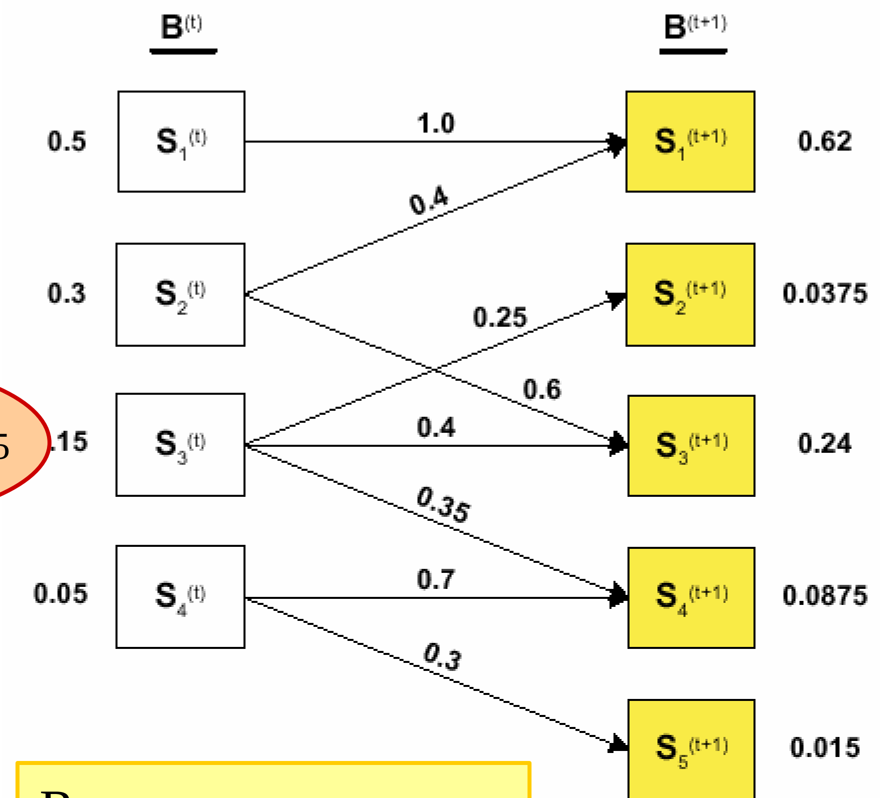
Iteration #5

- Expand most likely trajectory from the <u>highest cost</u> node

$\rightarrow \cancel{S_1^{(t)}} \rightarrow S_2^{(t)}$

C_2=0.3*0.4+0.0525





R =1-0.62-0.24-0.0875= 0.0525

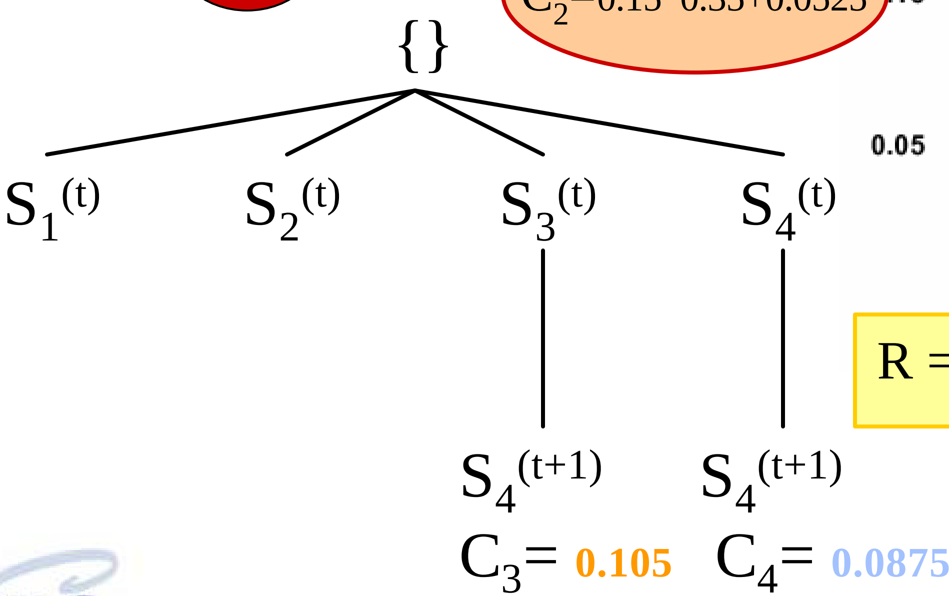
Enumerated:
$S_1^{(t+1)}$, $S_3^{(t+1)}$, $S_4^{(t+1)}$

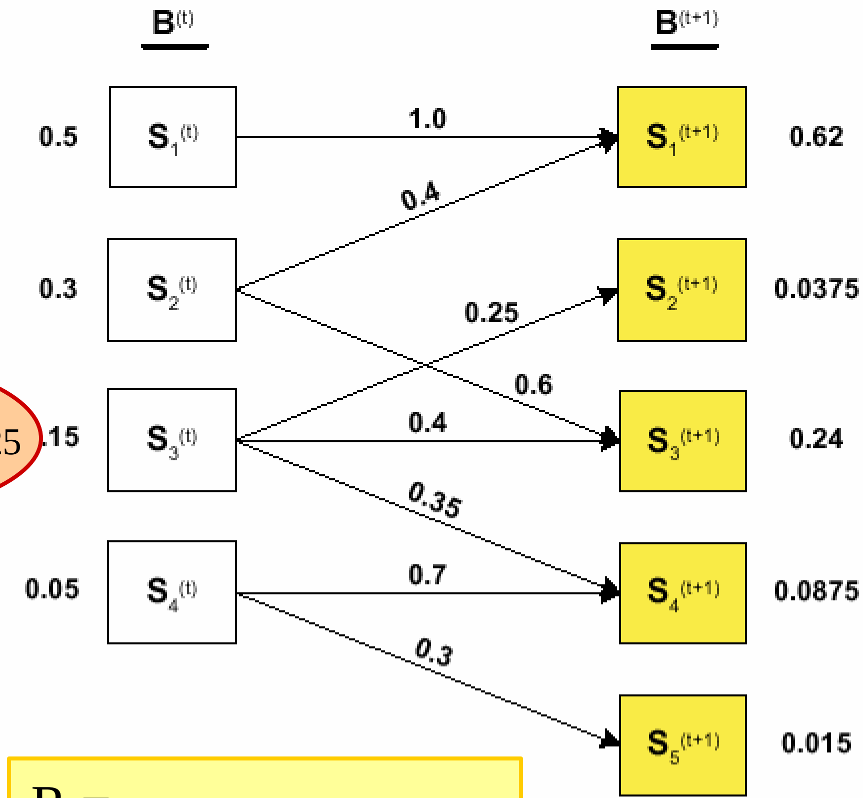
Iteration #6

- Expand most likely trajectory from the <u>highest cost</u> node

$\rightarrow$ $S_2^{(t)}$ $\rightarrow$ $S_3^{(t)}$

C_2=0.15*0.35+0.0525





R =1-0.62-0.24-0.0875= 0.0525

Enumerated:

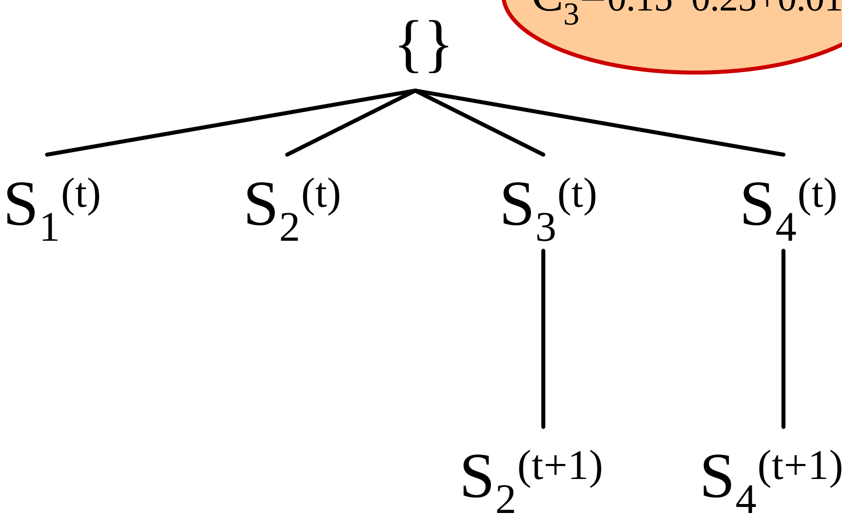
$S_1^{(t+1)}$, $S_3^{(t+1)}$, $S_4^{(t+1)}$

Iteration #7

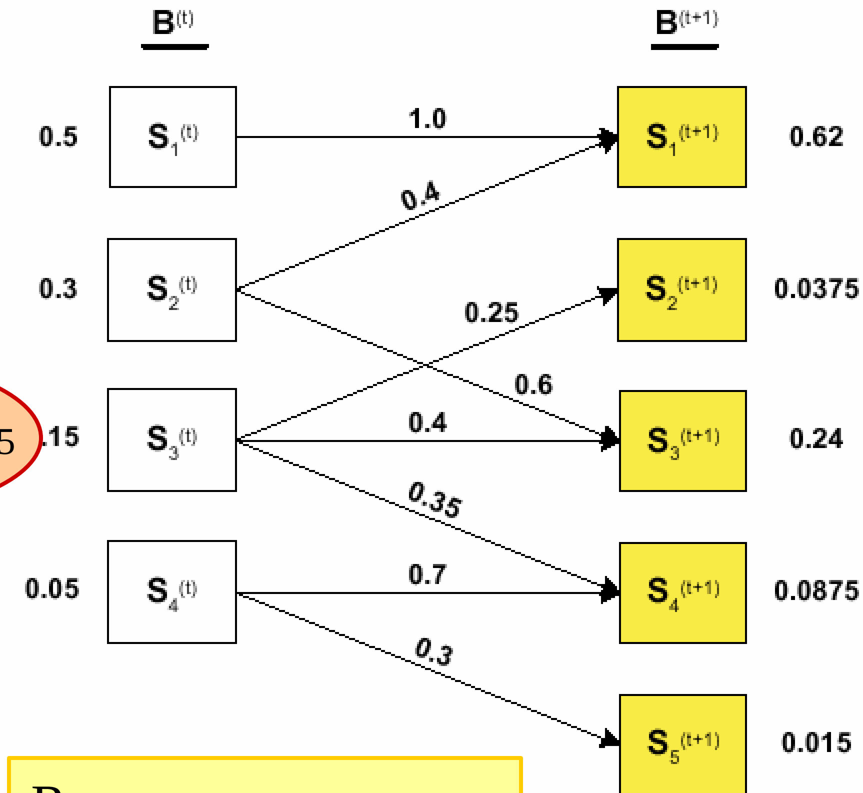
- Expand most likely trajectory from the <u>highest cost</u> node

  → $S_3^{(t)}$

$C_3 = 0.15*0.25+0.015$



$C_3 =$ **0.0525** $C_4 =$ **0.05**



R = 1-0.62-0.24-0.0875-0.0375
= 0.015

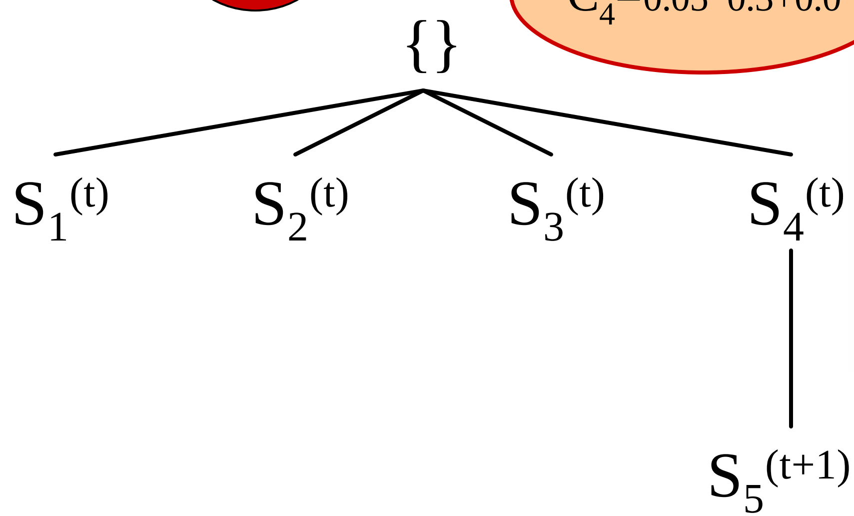
Enumerated:
$S_1^{(t+1)}$, $S_3^{(t+1)}$, $S_4^{(t+1)}$, $S_2^{(t+1)}$

Iteration #8

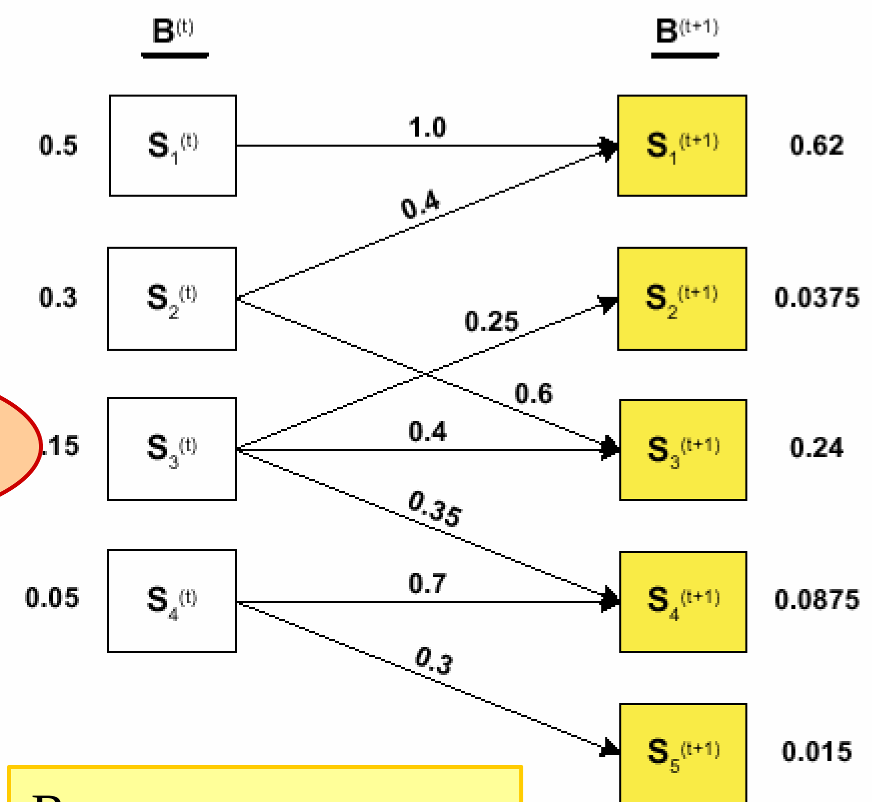
- Expand most likely trajectory from the highest cost node

$\rightarrow \cancel{S_3^{(t)}} \rightarrow S_4^{(t)}$

$C_4=0.05*0.3+0.0$



$C_4=$ **0.015**

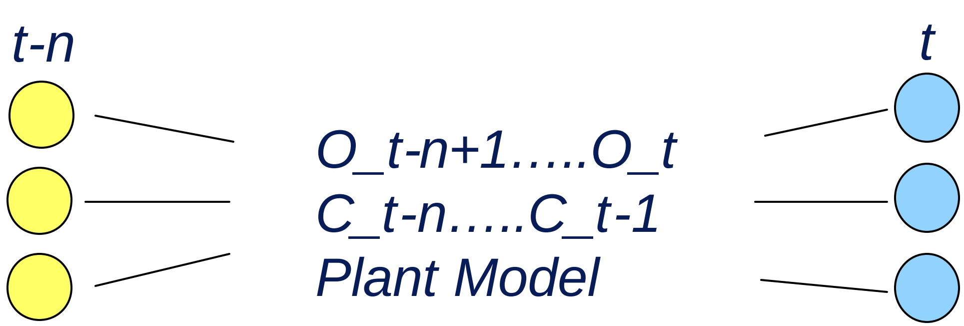


R = 1-0.62-0.24-0.0875-0.0375 - 0.015 = 0.0

Enumerated:
$S_1^{(t+1)}$, $S_3^{(t+1)}$, $S_4^{(t+1)}$, $S_2^{(t+1)}$, $S_5^{(t+1)}$

N-Step Mode Estimation

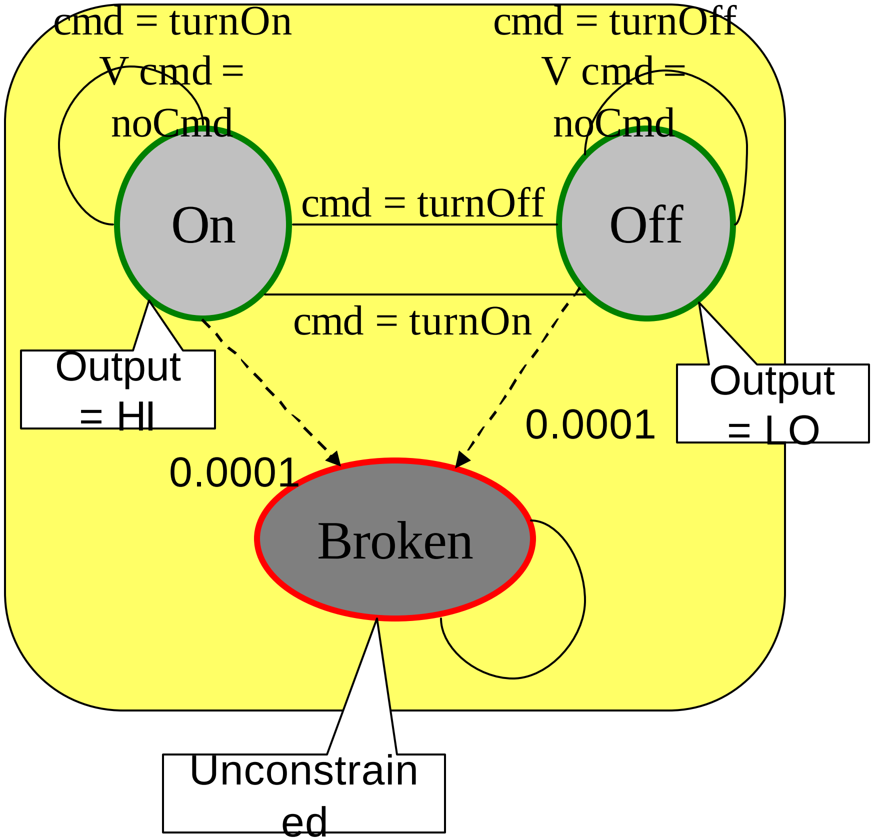
- Motivation:
  - Better estimates obtained by considering history of observations and commands in situations where a low probability trajectory becomes highly likely after additional evidence
- Use a moving window of N previous observations and commands to estimate the next states



Approach

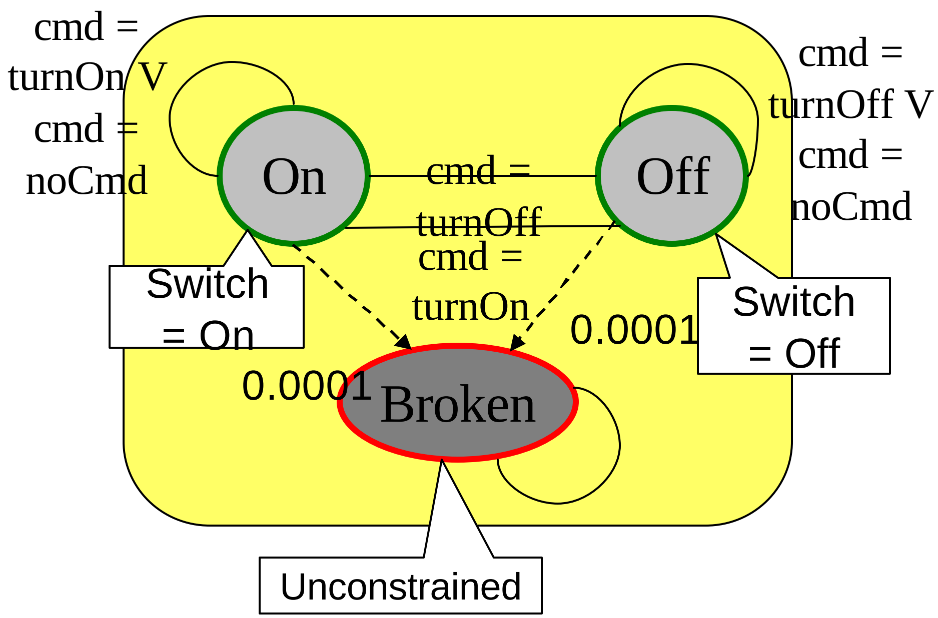
- Encode as CSP
  - Map CCA to constraints expressed in propositional logic
  - Types of constraints:
    - Modal constraints
    - Transition constraints
    - Non-deterministic transitions
    - Component interconnection constraints
    - Mutex constraints- interleaving of transitions
- Construct a Trellis Diagram
- Solve for best trajectories using OPSAT

Modal Constraints



M_Switch(Switch=On)=(output=HI)
M_Switch(Switch=Off)=(output=LO)

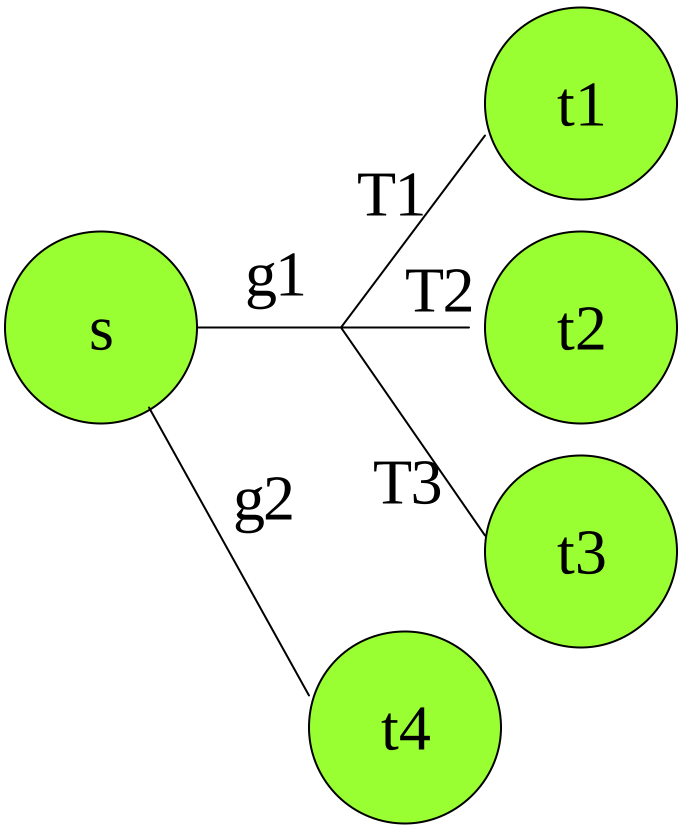
Transition Constraints



P(Tn_Switch)=0.99
P(Tf_Switch)=0.01

Tn_Switch(Switch=On)={(cmd=turnOff,Off), (cmd=$\overline{\text{turnOff}}$, On)}
Tn_Switch(Switch=Off)={(cmd=turnOn,On), (cmd=$\overline{\text{turnOn}}$, Off)}
Tf_Switch(Switch=On)={(True, Broken)}
Tf_Switch(Switch=Off)={(True,Broken)}

Non-deterministic Transitions

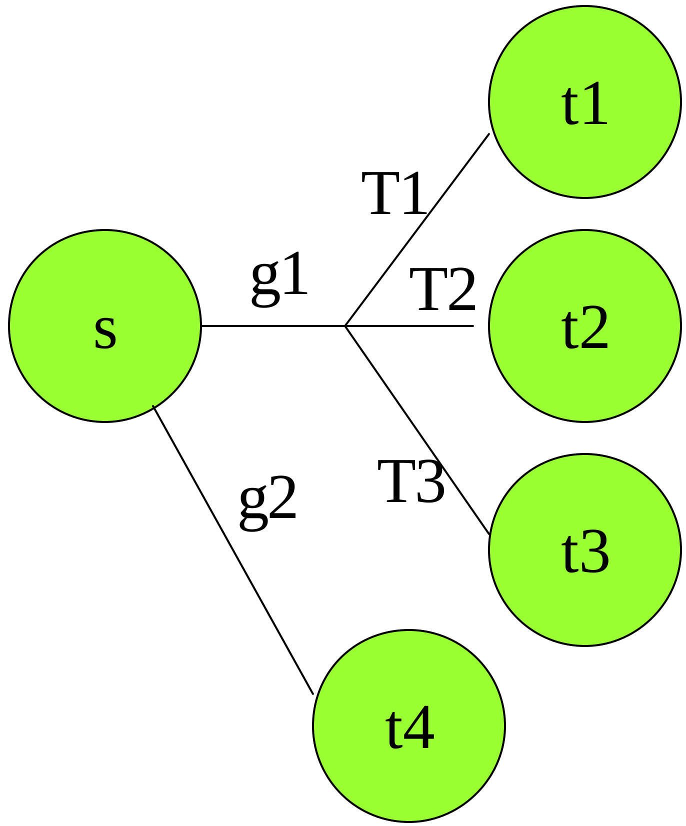


- Take only 1 transition at a time: g1 and g2 must be mutually exclusive, i.e. $\boxed{\sim(g1\&g2)}$

- g1 and g2 are mutually exhaustive $\boxed{g1 \vee g2}$

- RMPL compiler should check that these conditions on guards are satisfied

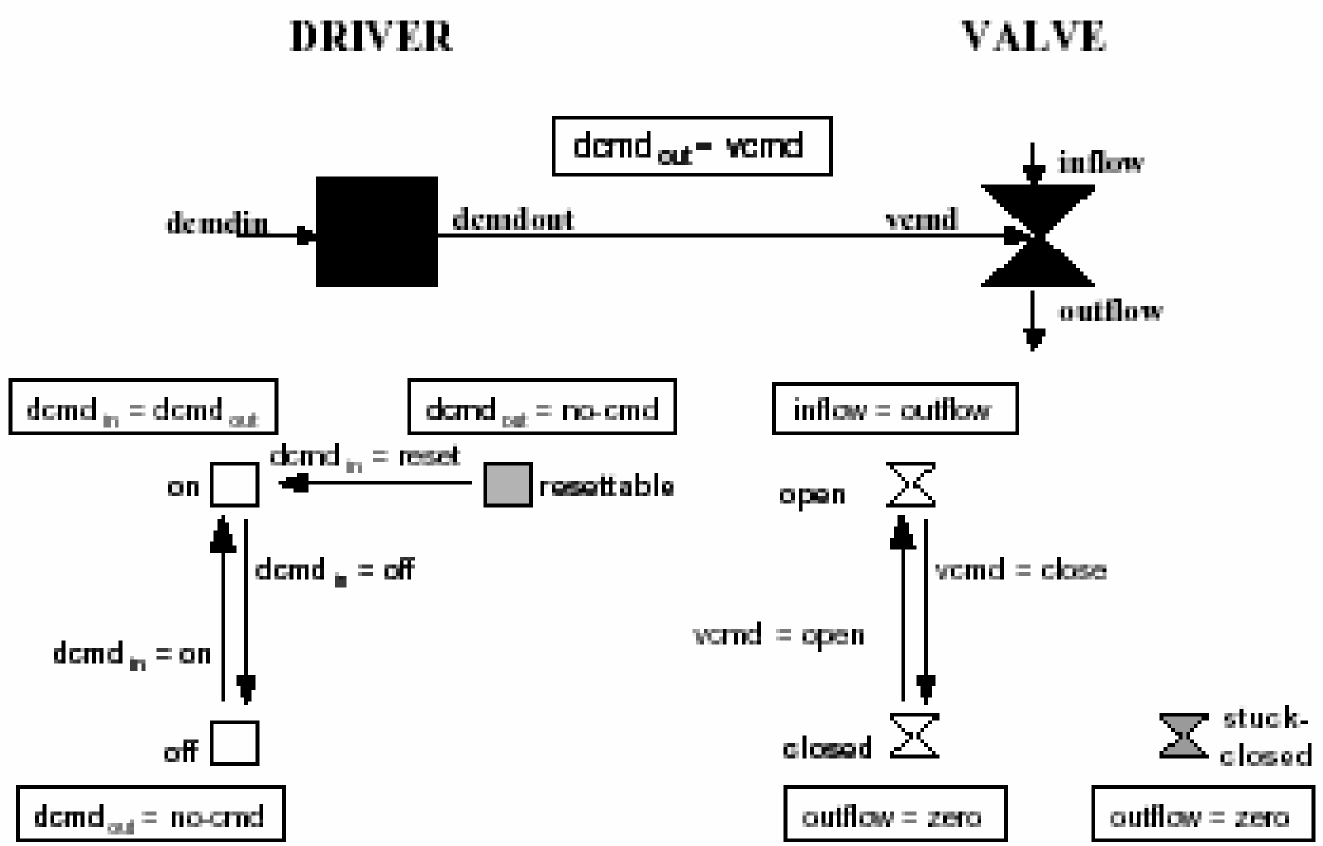
Non-deterministic Transitions



Tnd_component(x=s)={(g1,t1), (g1,t2), (g1,t3), ($\overline{\text{g1}}$,s)}

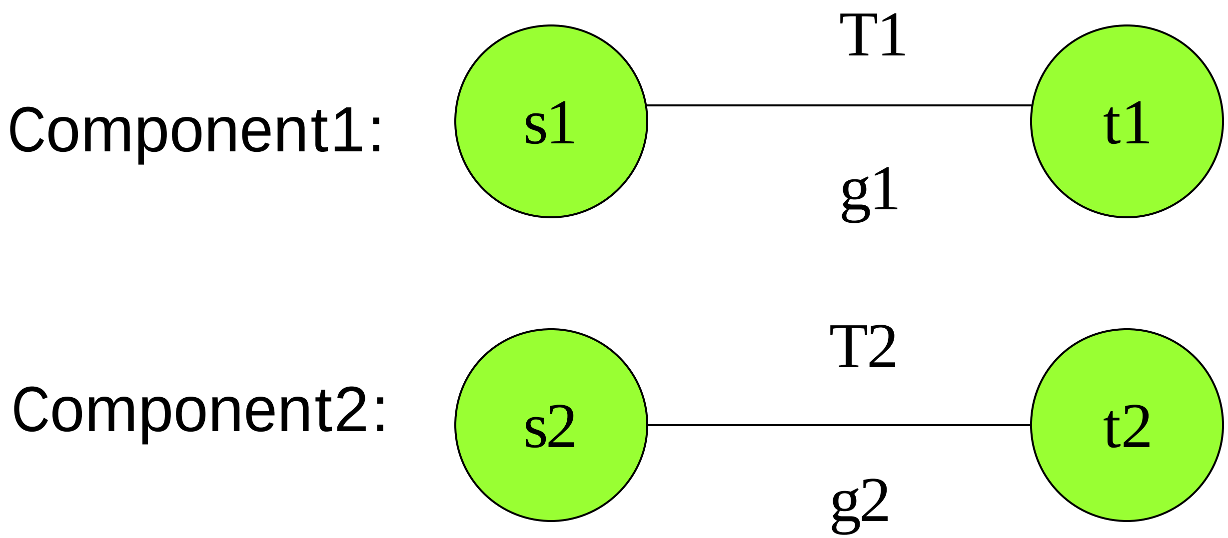
Tnd_component(x=s)={(g2, t4), ($\overline{\text{g2}}$,s)}

Component Interconnection Constraints



$$\mathbb{I} \quad = \quad (dcmd_{out} = vcmd).$$

Mutex Constraints



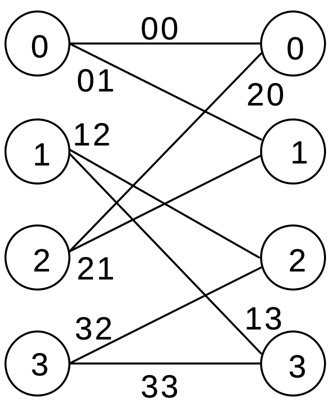
Interference: one of the effects of one transition is the negation of a precondition of the other

(M1_t=s1 & g1 & M2_t+1=t2) => (V1=T1 & V2=T2)
(M2_t=s2 & g2 & M1_t+1=t1) => (V1=T1 & V2=T2)

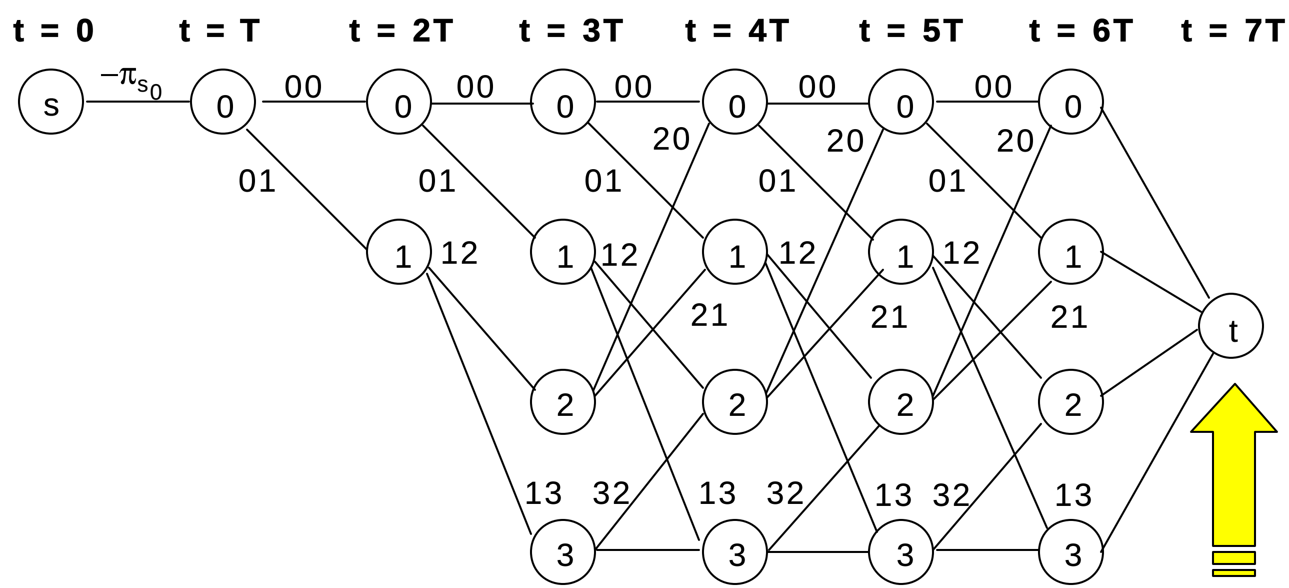
Trellis Diagram

- Construct a Trellis Diagram

Consider the following state diagram:



We can represent it by an equivalent state-time pair graph as follows:



Finish with a new end node with an empty transition from each state in the N^{th} + 1 step

Algorithm for Constructing Trellis Diagram

$$GenerateTrellis(\{s^{(0)}\}, N, \{m^{(0)},\ldots,m^{(N-1)}\}, \{o^{(0)},\ldots,o^{(N-1)}\}) ::$$

$$\quad For\ \ i = 1..N$$

$$\quad\quad \{s^{(i)}\} := ExpandTrellis(P, \{s^{(i-1)}\}, m^{(i-1)}, o^{(i-1)})$$

$$ExpandTrellis(P, \{s^{(i-1)}\}, m^{(i-1)}, o^{(i-1)}) ::$$

$$\quad For\ \ every\ \ s_i^{(t)} \in \{s^{(t)}\} :$$

$$\quad\quad 1.\ Compute\ \ constraint\ \ store$$

$$\quad\quad\quad C_M(s_i^{(t)}) := I \wedge \bigwedge_{x_i = v \in s_i^{(t)}} M_i(x_i = v)$$

$$\quad\quad 2.\ Return\ all\ \ feasible\ \ next\ \ states$$

$$\quad\quad\quad return \{ \bigcup_{x_i = v \in s_i^{(t)}} \overline{T}(x_i = v, C_M, m^{(t)}, o^{(t)}) \}$$

$$\quad\quad 3.\ Add\ \ feasible\ \ transitions \bigcup_{x_i = v \in s_i^{(t)}} t_i(x_i = v)$$

$$\quad\quad\quad to\ Dom(V^{(t)})$$

Algorithm for Constructing Trellis Diagram

$\overline{T}:$

$For\ every\ t_i(x_i = v) \in T_i(x_i = v):$

$\quad Identify\ transition\ pairs\ (l, v')\ in\ t_i(x_i = v)$

$\quad such\ that:$

$$C_M(s_i^{(t)}) \wedge m^{(t)} \wedge o^{(t)} \models l$$

$\quad return\ the\ next\ mode\ assignment$

Frame as OCSP

An OCSP $(\mathbf{x}, f, C)$ is a problem of the form "arg max $f(\mathbf{x})$ subject to $C(\mathbf{x})$," where $\mathbf{x}$ is a vector of decision variables, $C(\mathbf{x})$ is a set of propositional state constraints, and $f(\mathbf{x})$ is a multi-attribute utility function that is *mutually preferentially independent (MPI)*. $f(\mathbf{x})$ is MPI if the value of each $x_i \in \mathbf{x}$ that maximizes f is independent of the values assigned to the remaining variables.

Frame as OCSP

$$Decision\ \ variables\ \ are:$$

$$\underset{\sim}{x} = [V^{(0)},\ldots,V^{(N-1)}]$$

$$Dom(V^{(j)}) = \bigcup_{i=1..k} [\bigcup_{x_i = v \in s_i^{(j)}} t_i(x_i = v)]$$

$$where\ \ t_i(x_i = v) = \{(l_i, v'_i)\}\ \ such\ \ that$$

$$o^{(j)} \wedge m^{(j)} \wedge C_M(s_i^{(j)}) \models l_i \quad for\ \ each\ \ component$$

$$Constraints:$$

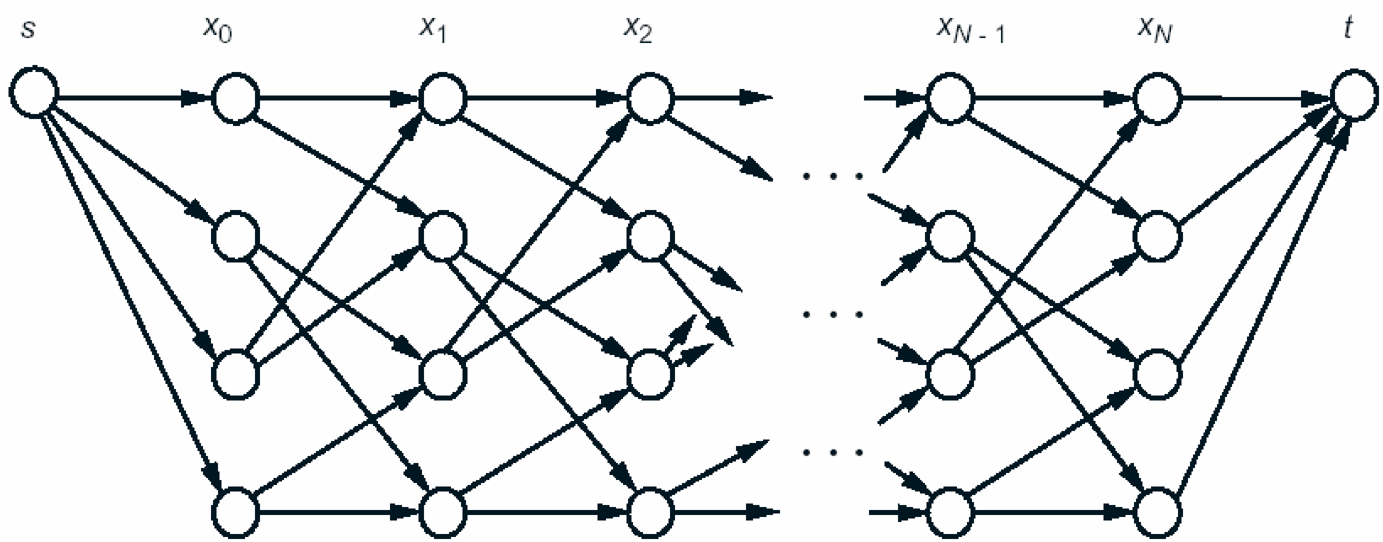
$$C^{(j)} := (V^{(j)}) = \bigcup_{x_i = v \in s_i^{(j)}} [t_i(x_i = v) = \{(l_i, v'_i)\}] =>$$

$$\bigvee_i (V' = s_i^{(j+1)} \wedge C_M(s_i^{(j+1)}))$$

$$where\ \ V' = \{v'_i\}$$

Frame as OCSP

- Trajectory estimation problem: Given the observation sequence $Z_N = \{z_1, z_2, \ldots, z_N\}$, what is the "most likely" state transition sequence $\hat{X}_N = \{\hat{x}_0, \hat{x}_1, \ldots, \hat{x}_N\}$ [one that maximizes $p(X_N \mid Z_N)$ over all $X_N = \{x_0, x_1, \ldots, x_N\}$].



Frame as OCSP

- We have

$$p(X_N \mid Z_N) = \frac{p(X_N, Z_N)}{p(Z_N)}$$

where $p(X_N, Z_N)$ and $p(Z_N)$ are the unconditional probabilities of occurrence of (X_N, Z_N) and Z_N

- Maximizing $p(X_N \mid Z_N)$ is equivalent with maximizing $\ln(p(X_N, Z_N))$

Frame as OCSP

$$p(X_N, Z_N) = p_{x_o} \prod_{k=1}^{N} p_{x_{k-1}, x_k} r(z_k; x_{k-1})$$

So the objective function is to maximize:

$$\ln(p_{x_o}) + \sum_{k=1}^{N} \ln(p_{x_{k-1}, x_k} r(z_k; x_{k-1}))$$

Over all possible sequences $\{x_o, x_1, \ldots, x_N\}$