Autonomy Practices:
Decision-making Architectures

KISS Workshop on
Engineering Resilient Space Systems

courtesy of JPL

Brian C. Williams, MIT
July 30th, 2012

Publications at: mers.csail.mit.edu

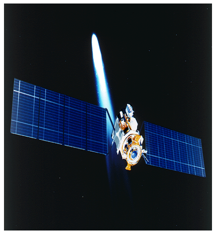
Desiderata

Provide a programming paradigm that enables robotic systems:

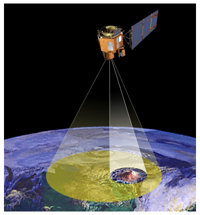
- to be commanded simply and intuitively;
- to adapt to uncertainty and failure;
- to communicate at a cognitive level;
- to work fluidly with humans, and
- to manage risk taking effectively.

Space Autonomy Architectures

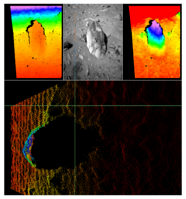
- Autonomous Operation
  - Cassini AACS, Remote Agent, MDS, Titan



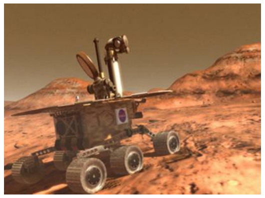
- Autonomous Science
  - Autonomous Sciencecraft Experiment



- Autonomous Navigation
  - ClarAty, MERS DIME & Gestalt

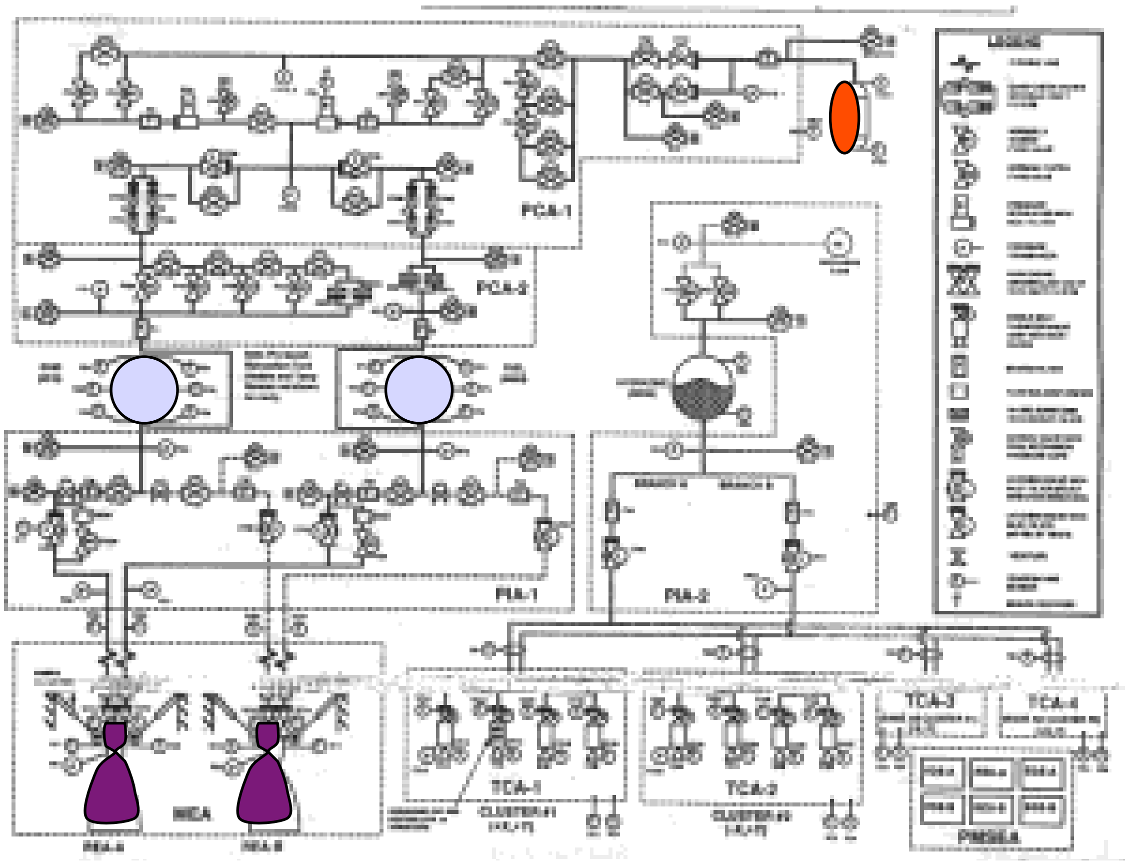


- Mixed Initiate Interaction
  - MapGen + descendants



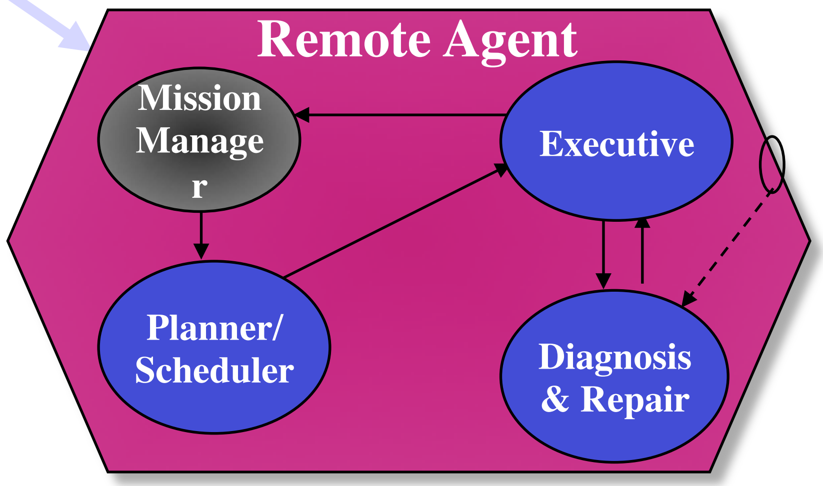
Outline

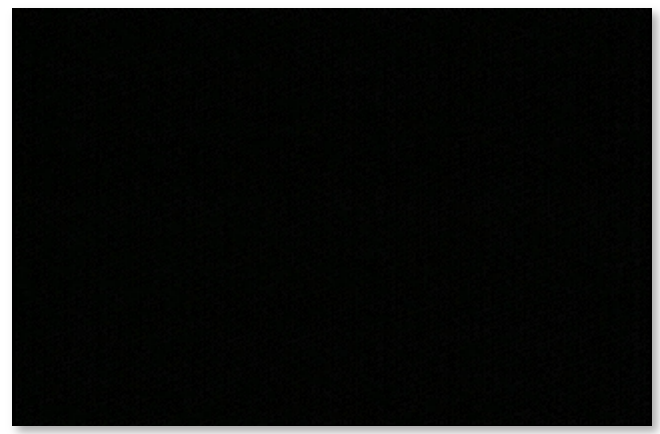
- Robust, Goal-directed Execution
- Plan Dispatching
- Diagnosis and Mode Estimation
- Plan Generation


PCA-1
PCA-2
MEA
TCA-1
TCA-2
TCA-3
TCA-4

Remote Agent on Deep Space One

Goals





1. Commanded by giving goals
2. Reasoned from commonsense models
3. Closed loop on goals

[Muscettola et al, AIJ 00;
Williams & Nayak, AAAI 95]

Remote Agent Experiment on Deep Space 1 – May, 1999

May 17-18th experiment: Mission-level Fault Protection

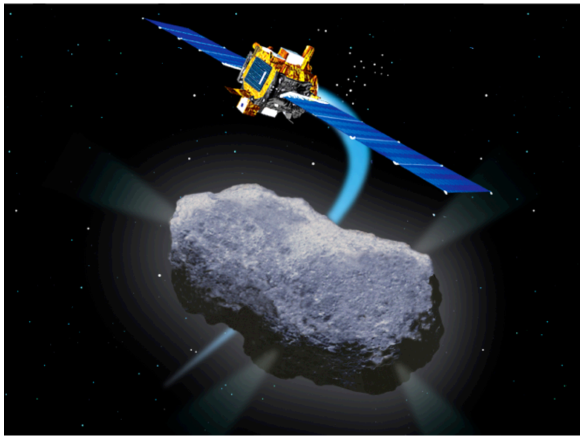
- Generate plan for course correction and thrust
- **Diagnose camera as stuck on**
  - **Power constraints violated, abort current plan & replan**
- Perform optical navigation
- Perform ion propulsion thrust

May 21th experiment: Engineering-level Fault Protection

- **Diagnose faulty device and**
  - **Repair by issuing reset.**
- **Diagnose switch sensor failure.**
  - **Determine harmless, and continue plan.**
- **Diagnose thruster stuck closed and**
  - **Repair by switching to alternate method of thrusting.**
- Back to back planning

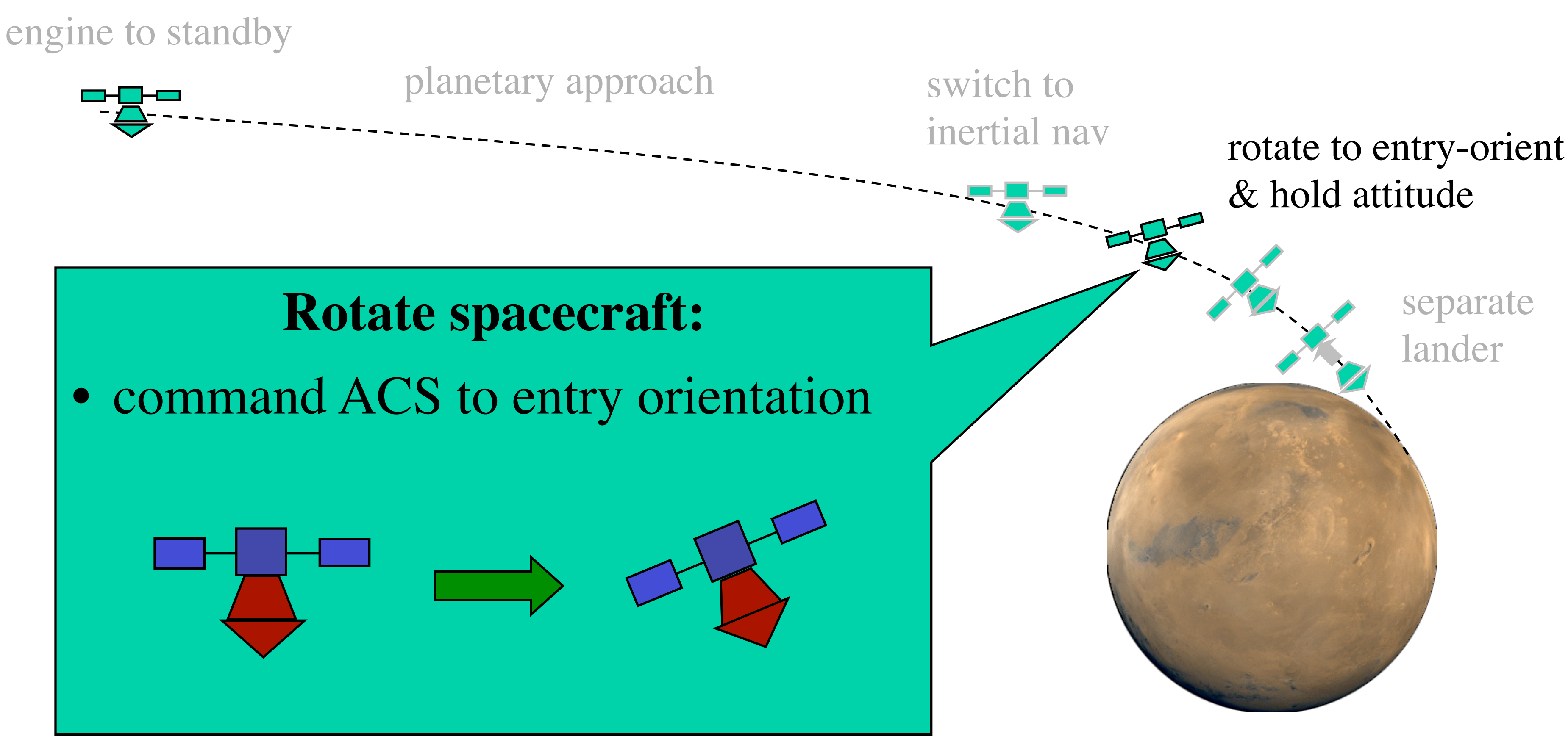
Model-based Autonomy

– An embedded programming languages elevated to the goal-level through operations on hidden state (RMPL).
– A language executive that achieves robustness by reasoning over constraint-based models (Titan).
– Interfaces that support natural human interaction fluidly and at the cognitive level.



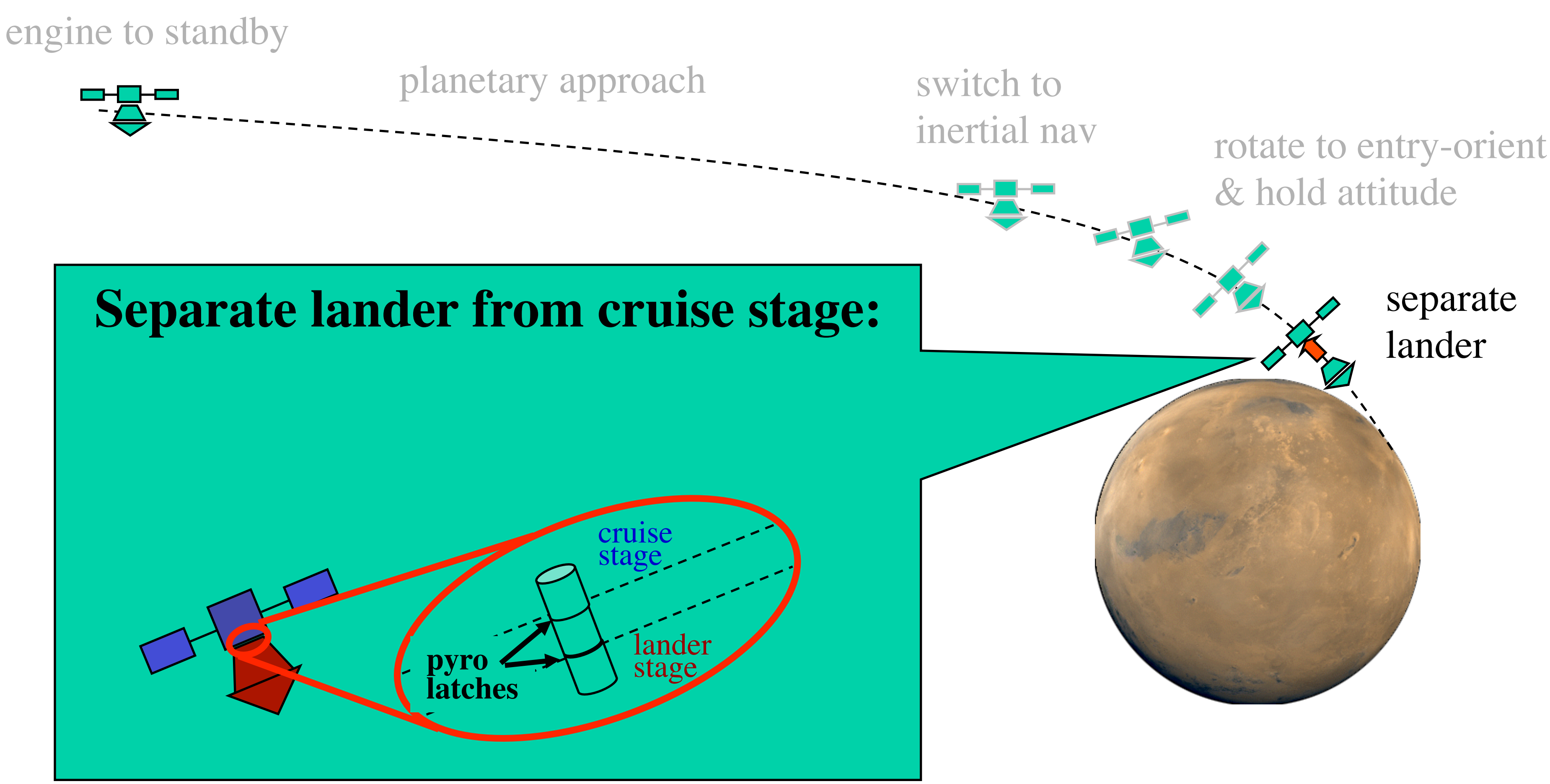
Commanding with Goals: System Engineers Specify Missions in Terms of Evolving States

[Titan Executive, Williams et al., IEEE Procs 03]



Commanding with Goals: System Engineers Specify Missions in Terms of Evolving States

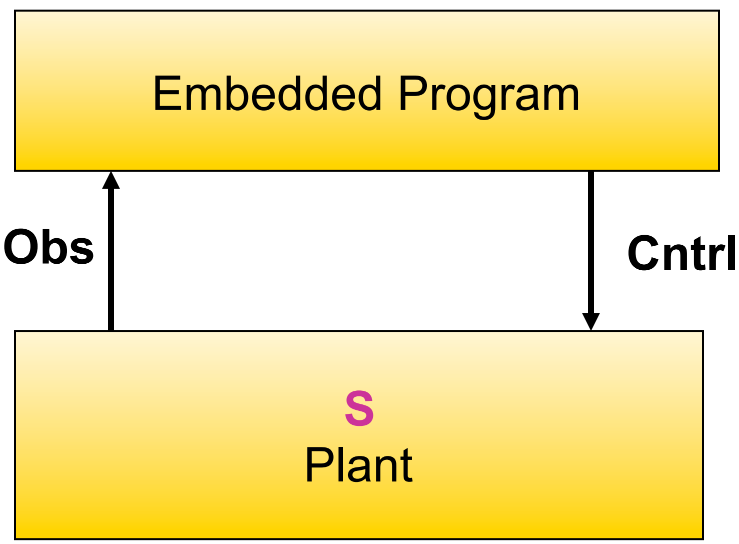
[Titan Executive, Williams et al., IEEE Procs 03]



Autonomous Systems are Commanded in Terms of Evolving Goal States

Embedded programs evolve actions by interacting with plant sensors and actuators:

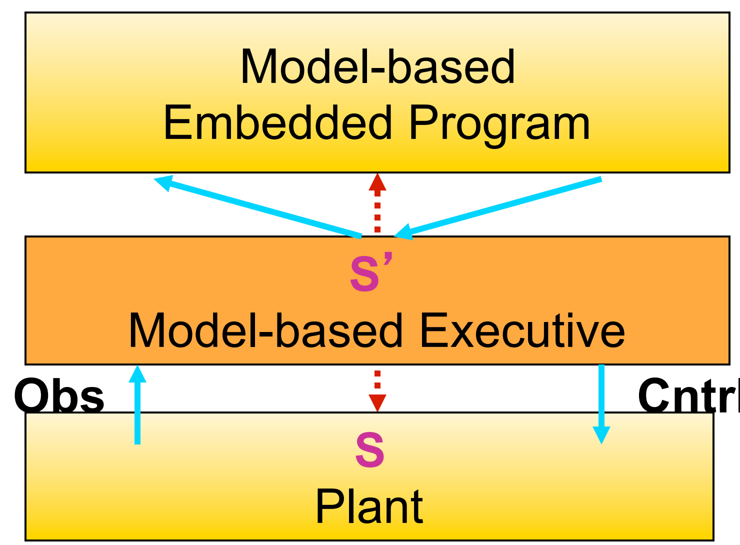
- Read sensors
- Set actuators



Programmer maps between state and sensors/actuators.

Model-based programs evolve abstract states through direct interaction:

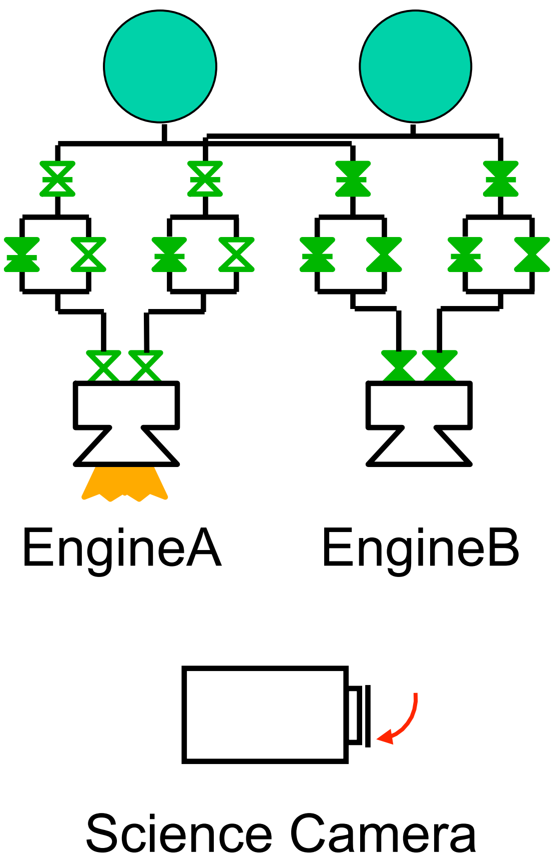
- Read abstract state
- Write abstract state



Model-based executive maps between state and sensors/actuators.

Model-based Programs
Specify and Execute Evolving States

Turn camera off and engine on

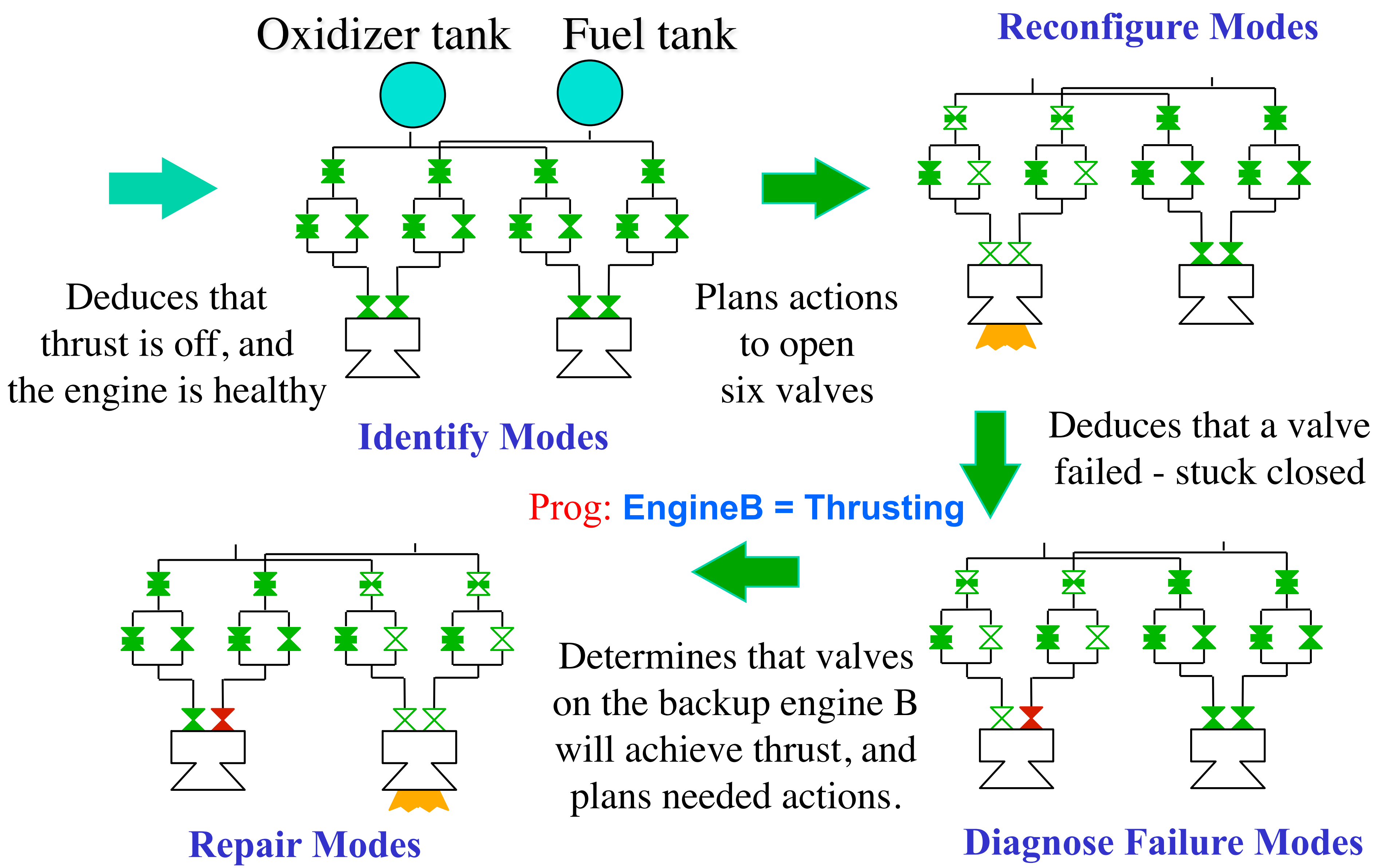



```
OrbitInsert()::

 do-watching (EngineA = Thrusting OR
              EngineB = Thrusting)
    parallel {
       EngineA = Standby;
       EngineB = Standby;
       Camera = Off;
       do-watching (EngineA = Failed)
          {when-donext (EngineA = Standby) AND
                        Camera = Off)
             EngineA = Thrusting};
       when-donext (EngineA = Failed AND
                    EngineB = Standby AND
                    Camera = Off)
          EngineB = Thrusting}
```

**[Titan Executive,
Williams et al., IEEE Procs 03]**

The program assigns **EngineA = Thrusting**, and the model-based executive . . . .



Behaviors Generated from a Plant Model

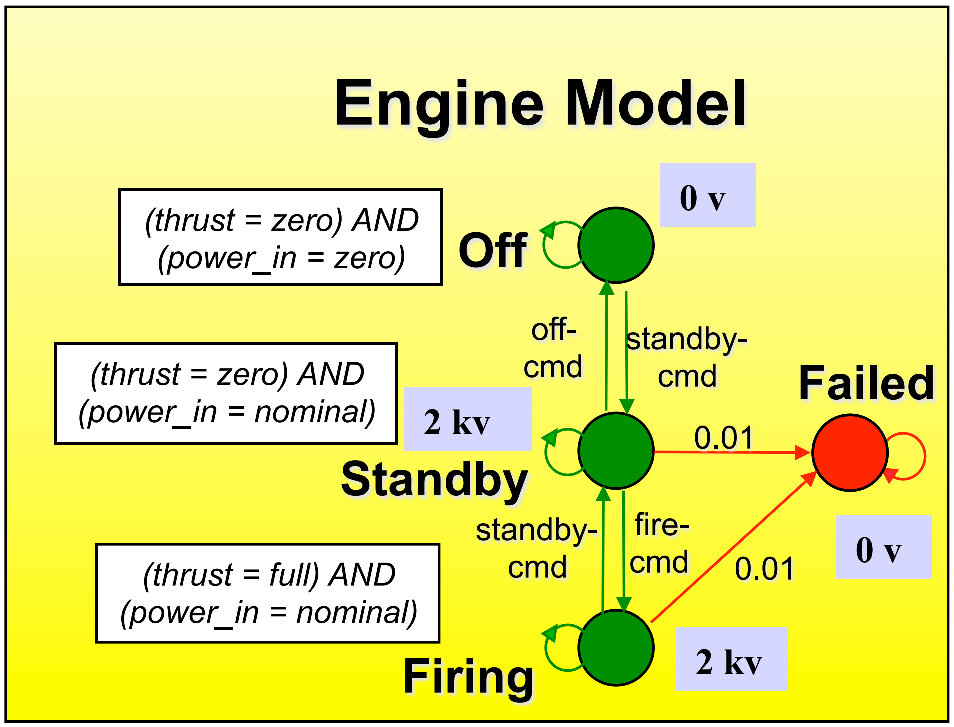
[Titan Executive,
Williams et al., IEEE Procs 03]

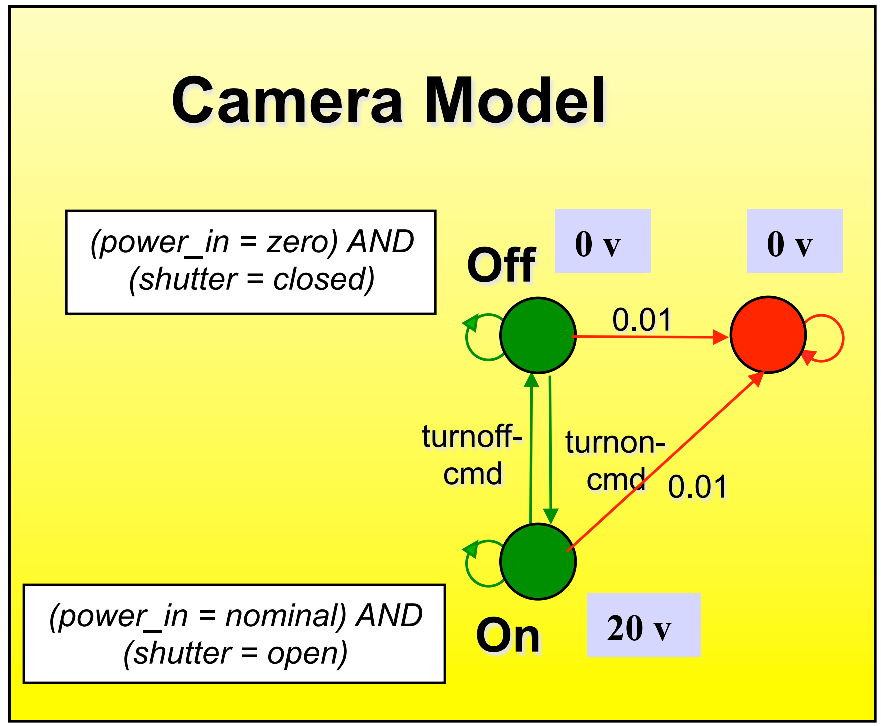
component modes…

described by finite domain constraints on variables…

deterministic and probabilistic transitions

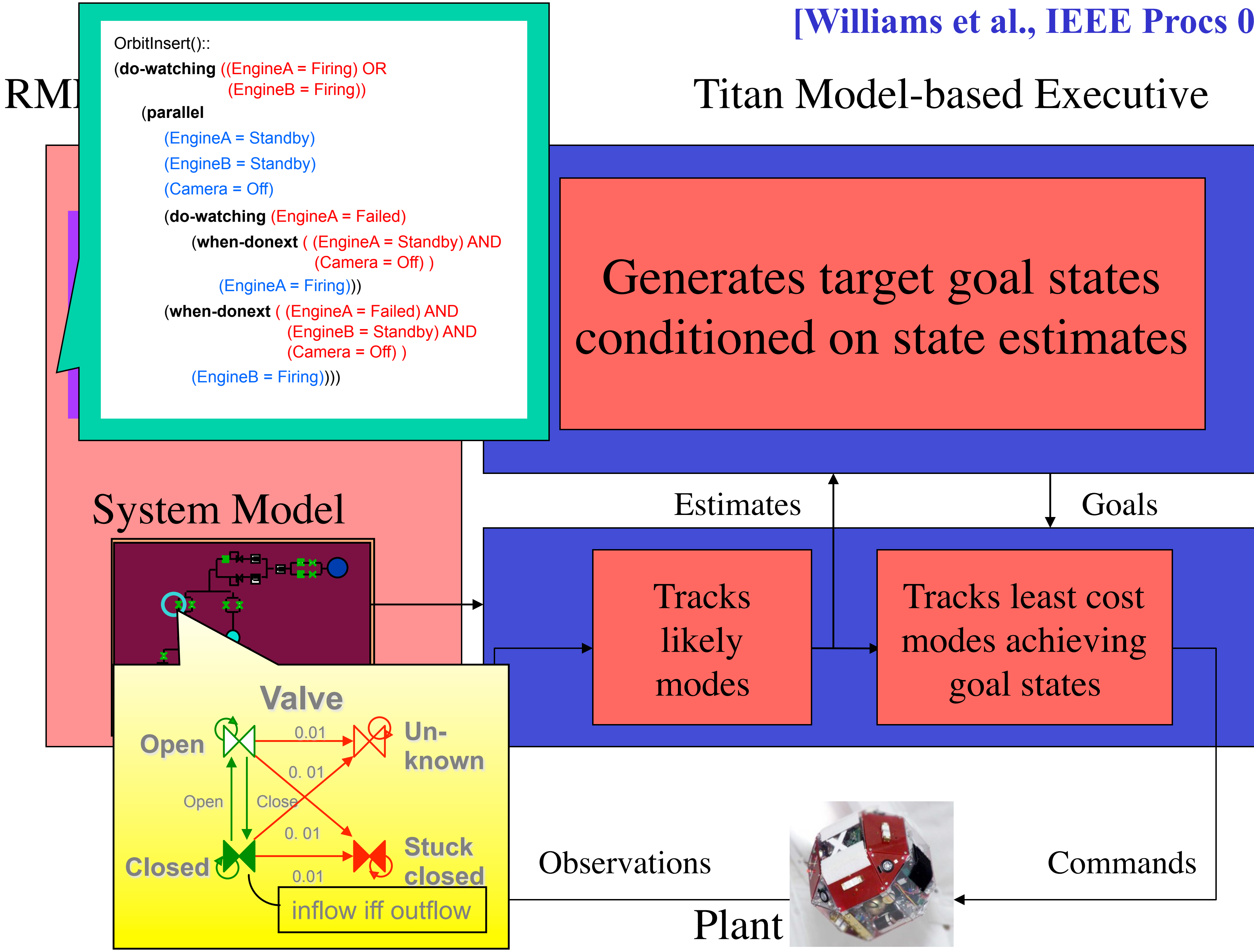
cost/reward

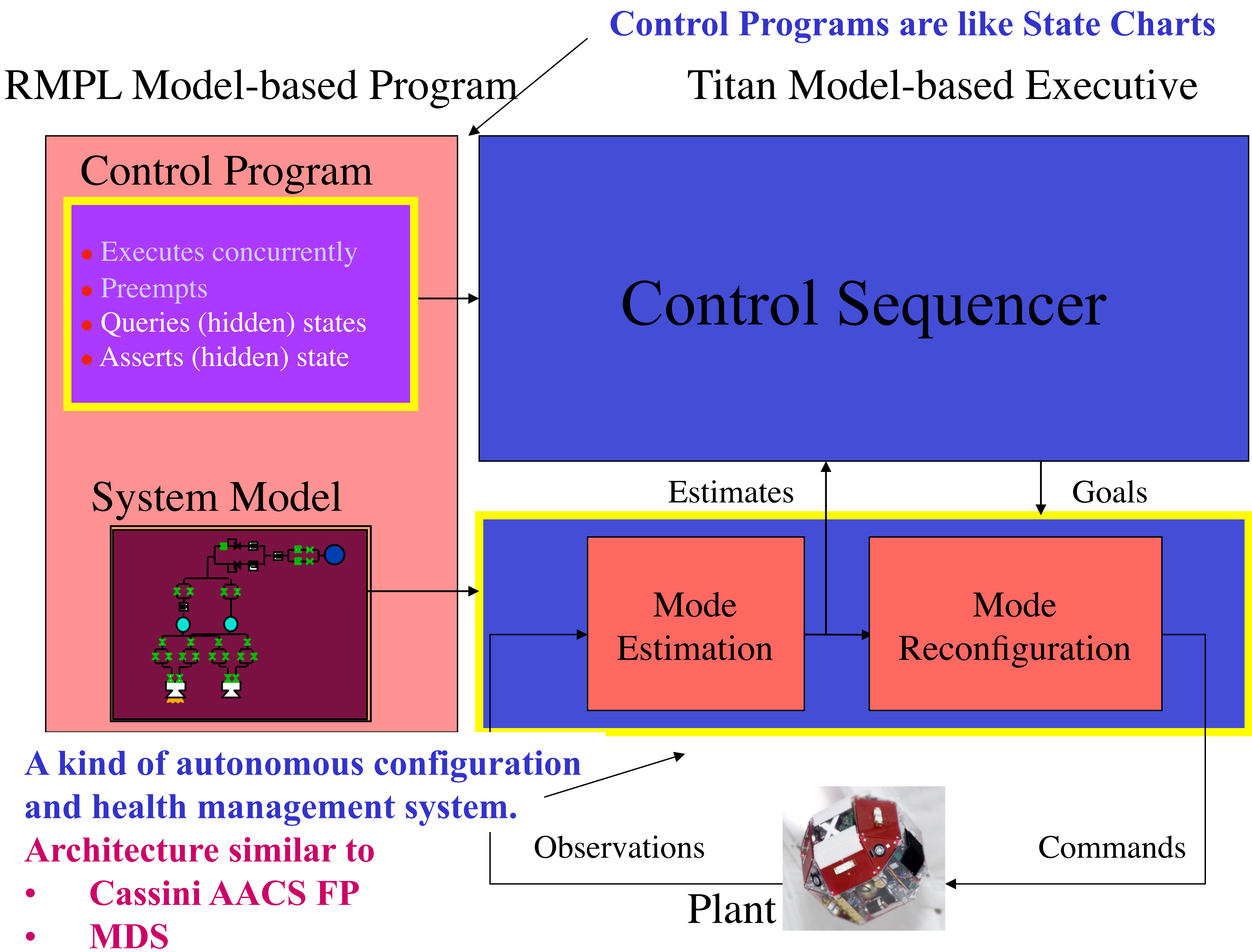




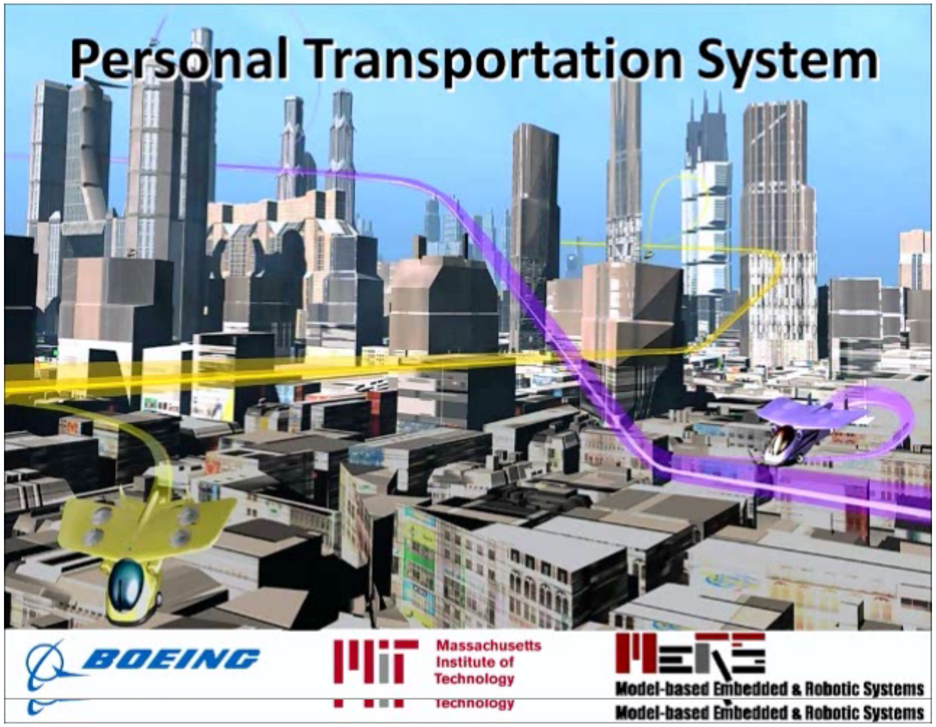
one per component … operating concurrently

[Williams et al., IEEE Procs 03]

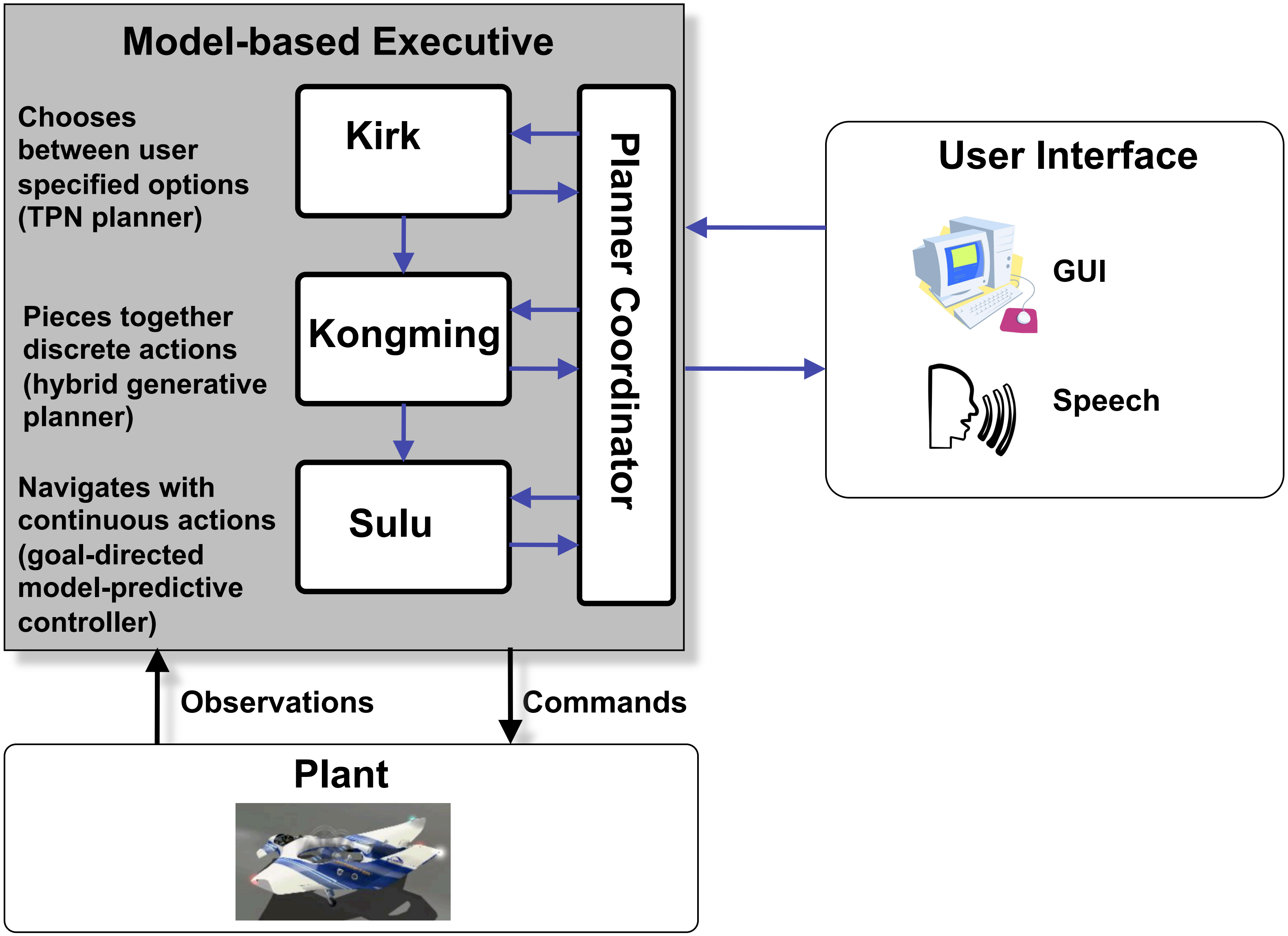



Control Programs are like State Charts
RMPL Model-based Program
Titan Model-based Executive
Control Program
Executes concurrently
Preempts
Queries (hidden) states
Asserts (hidden) state
Control Sequencer
System Model
Estimates
Goals
Mode Estimation
Mode Reconfiguration
A kind of autonomous configuration and health management system.
Architecture similar to
• Cassini AACS FP
• MDS
Observations
Commands
Plant

Navigation, Risk and Mixed Interaction: Personal Transportation System

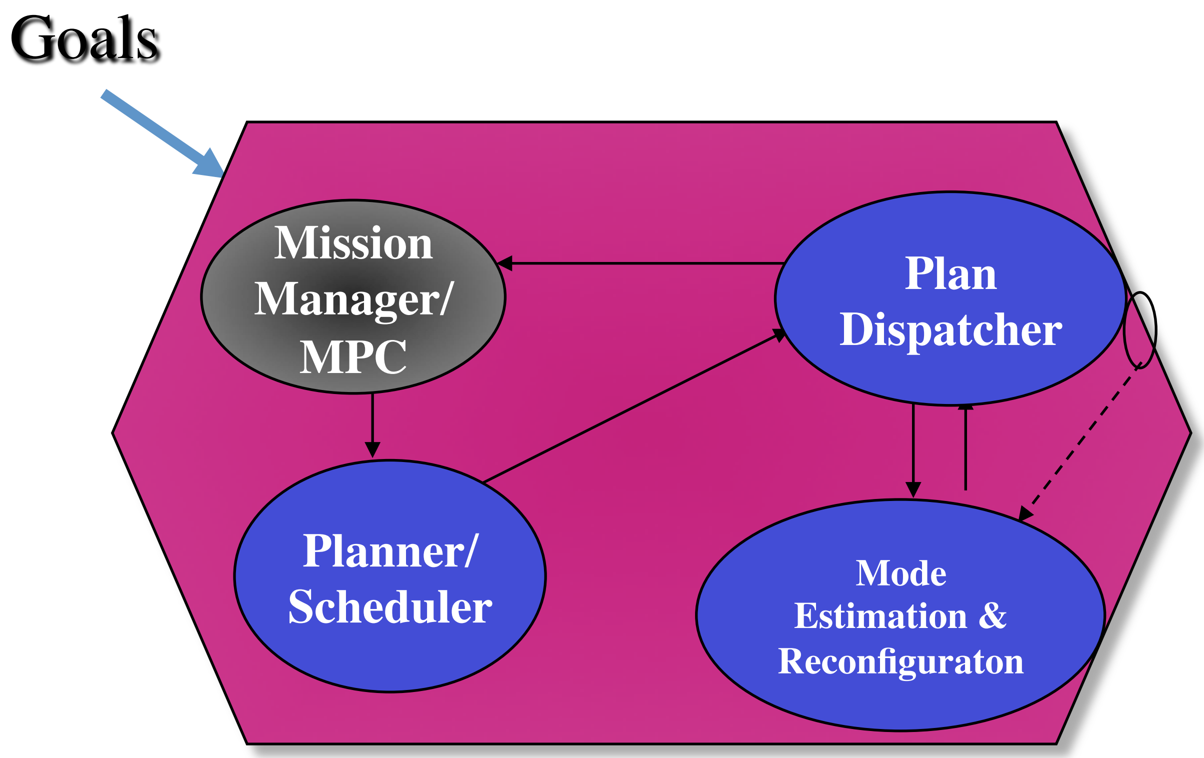


Complements of Branko Sarh, Boeing Research


Model-based Executive
Chooses between user specified options (TPN planner)
Kirk
Pieces together discrete actions (hybrid generative planner)
Kongming
Navigates with continuous actions (goal-directed model-predictive controller)
Sulu
Planner Coordinator
User Interface
GUI
Speech
Observations
Commands
Plant

Model-based Executives

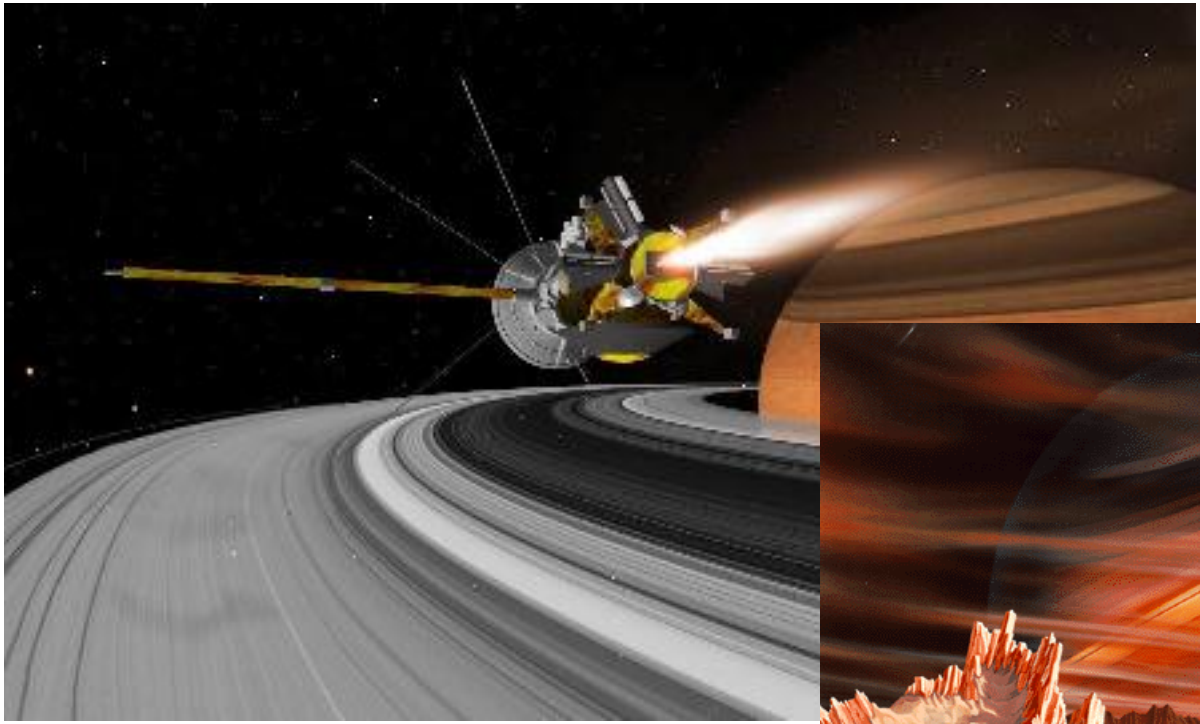
1. Commanded through time evolved goals.
2. Reasons from commonsense models.
3. Closes loop on goals.
4. Model-based programs specify goals and models.

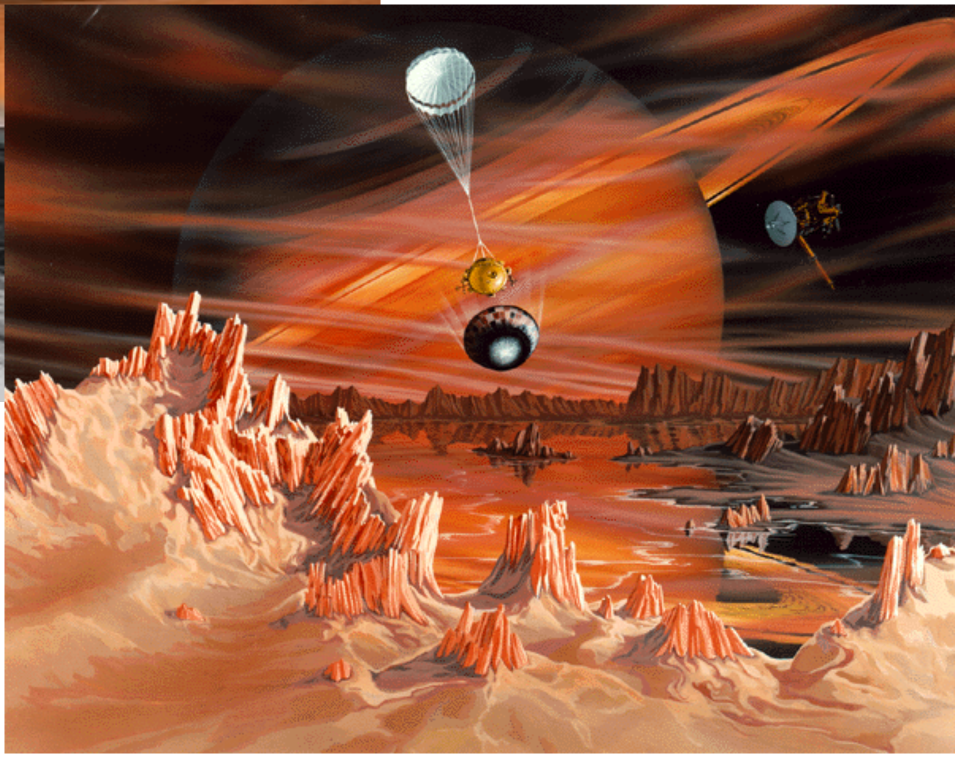


Outline

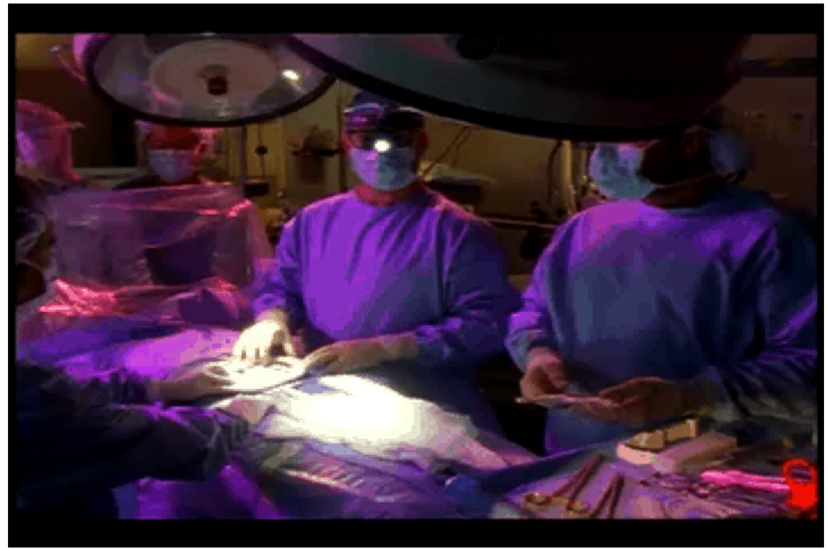
- Robust, Goal-directed Execution
- Plan Dispatching
- Diagnosis and Mode Estimation
- Plan Generation

Supports Time Critical Missions





Supports Robot & Human Coordination



An effective Scrub Nurse:

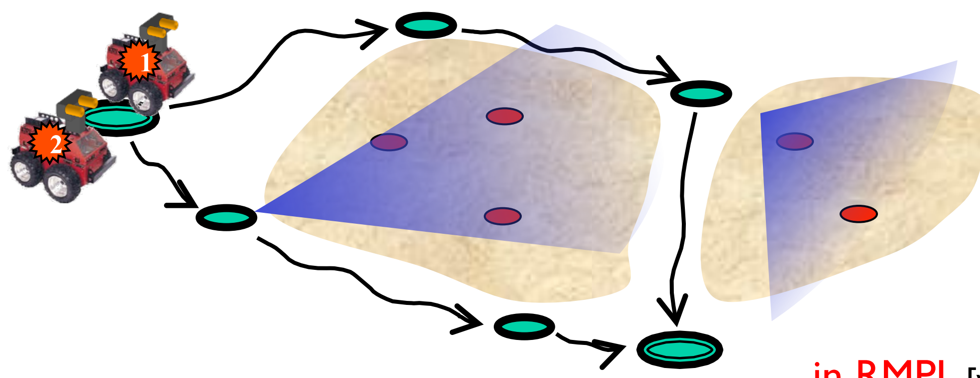
- works hand-to-hand, face-to-face with surgeon,
- assesses and anticipates needs of surgeon,
- provides assistance and tools in order of need,
- responds quickly to changing circumstances,
- responds quickly to surgeon’s cues and requests.

[Shah, Conrad and Williams, ICAPS 09]

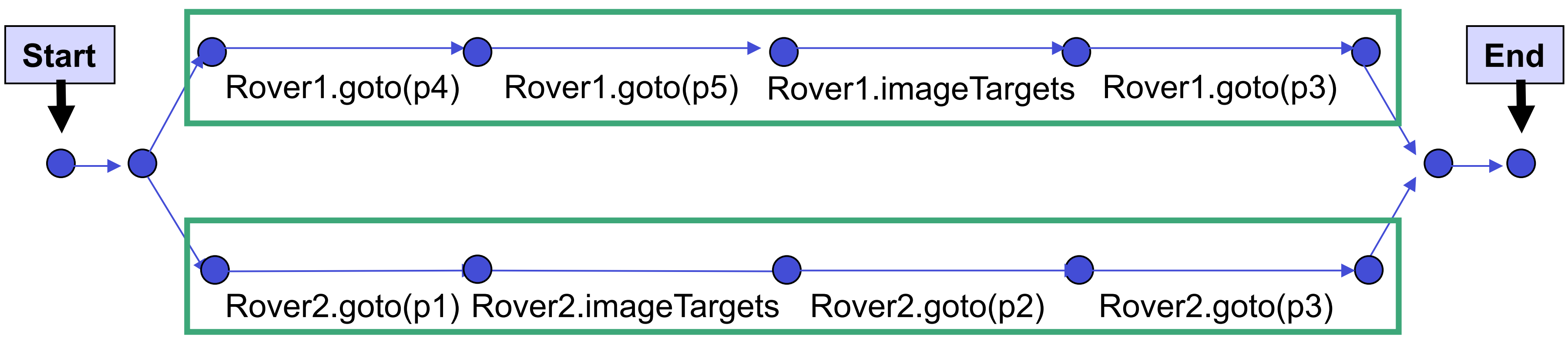
Robust Plan Execution

Model-based Embedded & Robotic Systems

```
imageScienceTargets(Rover1, Rover2)
{Parallel
  {Sequence
    [5,10] Rover1.goto(p4);
    [5,10] Rover1.goto(p5);
    [2,5] Rover1.imageTargets();
    [5,10] Rover1.goto(p3);
  },
  {Sequence
    [5,10] Rover2.goto(p1);
    [5,10]Rover2.imageTargets();
    [2,5] Rover2.goto(p2);
    [5,10] Rover2.goto(p3);
  }
}
```



in RMPL [williams et al]



Agents adapt to temporal disturbances in a coordinated manner by scheduling the start of activities on the fly.

[Muscettola, Morris, Tsamardinos, KR 98]

Outline: To Execute a Temporal Plan

Schedule Off-line

1. Describe Temporal Plan
2. Test Consistency
3. Schedule Plan

offline

online

4. Execute Plan

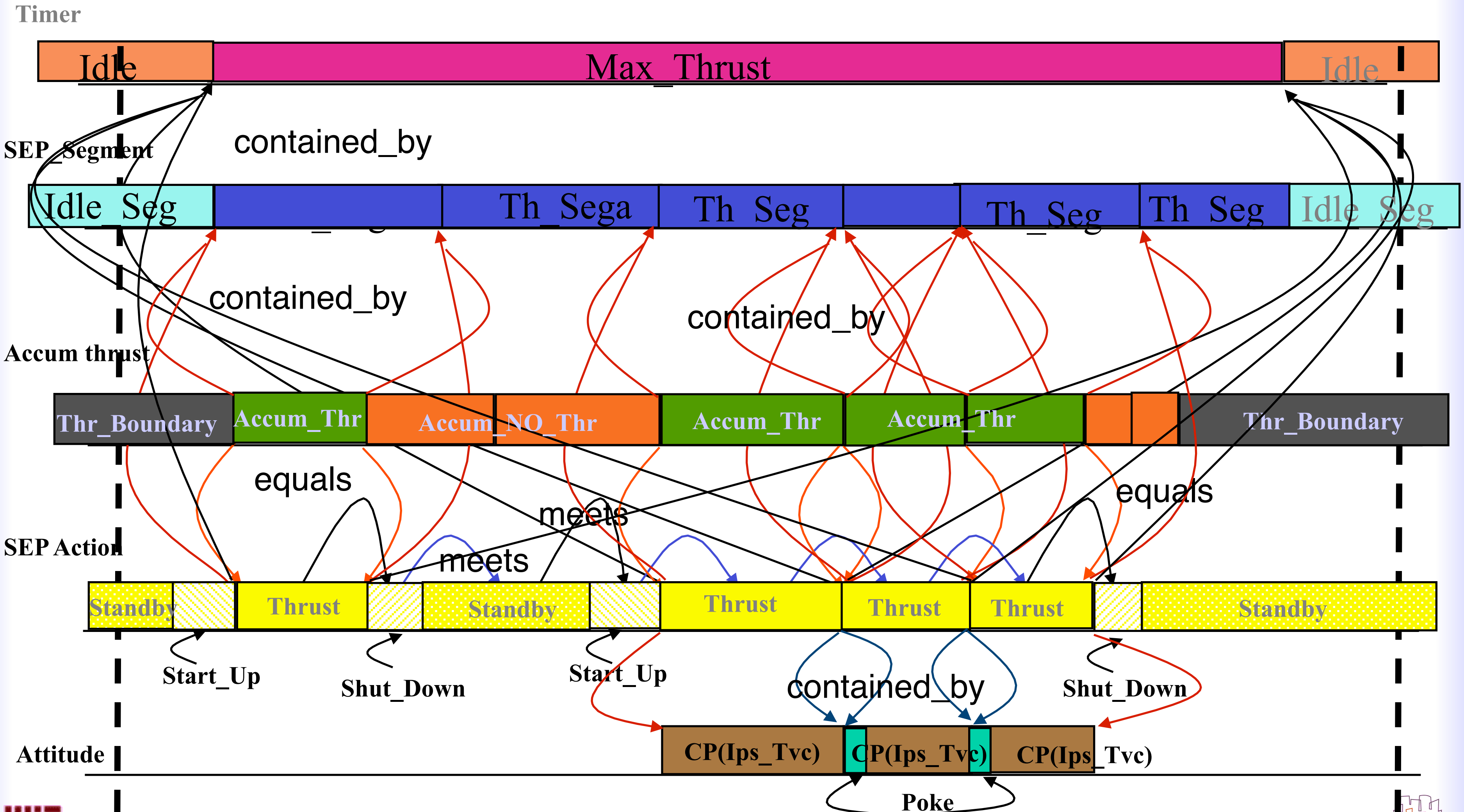
Schedule Online

1. Describe Temporal Plan
2. Test Consistency
3. Reformulate Plan
4. Dynamically Schedule Plan

Describe Temporal Plan

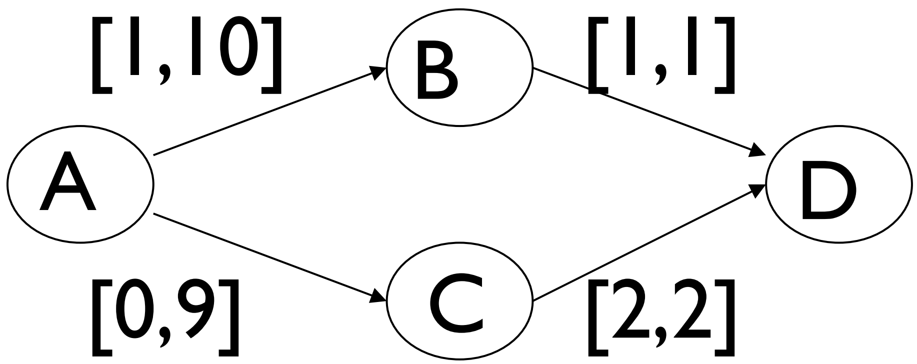
Model-based Embedded & Robotic Systems

Example: Deep Space One Remote Agent Experiment



Scheduling a Simple Temporal Network (STN)

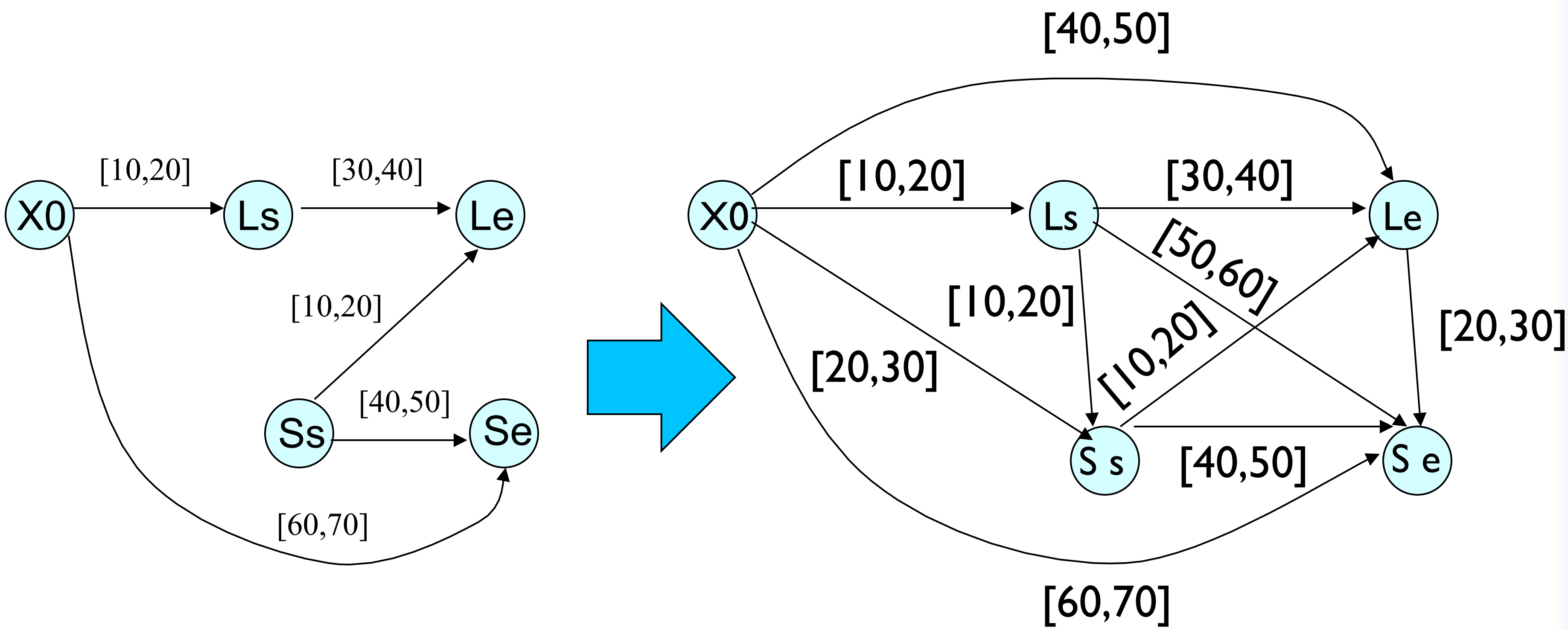
Input: An STN <X, C> where $C_j = <<X_k, X_i><a_j,b_j>>$



Output: An assignment to X satisfying C.

A=0, B=2, C=1, D=3

Scheduling without Search



Idea: Expose Implicit Constraints in STN

- Input: STN.
- Output: "Decomposable" (Implied) STN -> Schedule.

Scheduling without Search

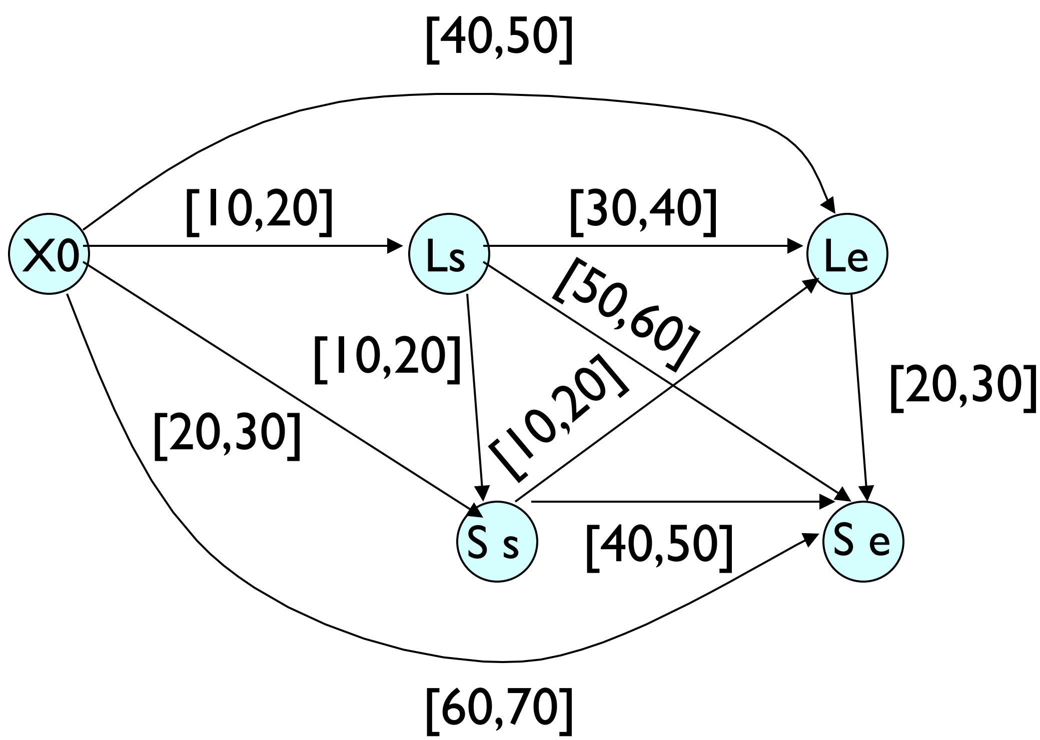
Input: Decomposable STN (APSP D-Graph)

Output: Schedule (Assignment to X, consistsent with STN)

Property: Can assign variables in any order, without backtracking.

Key ideas

- **Incrementally tighten feasible intervals, as commitments are made.**



Scheduling without Search

Input: Decomposable STN (APSP D-Graph)

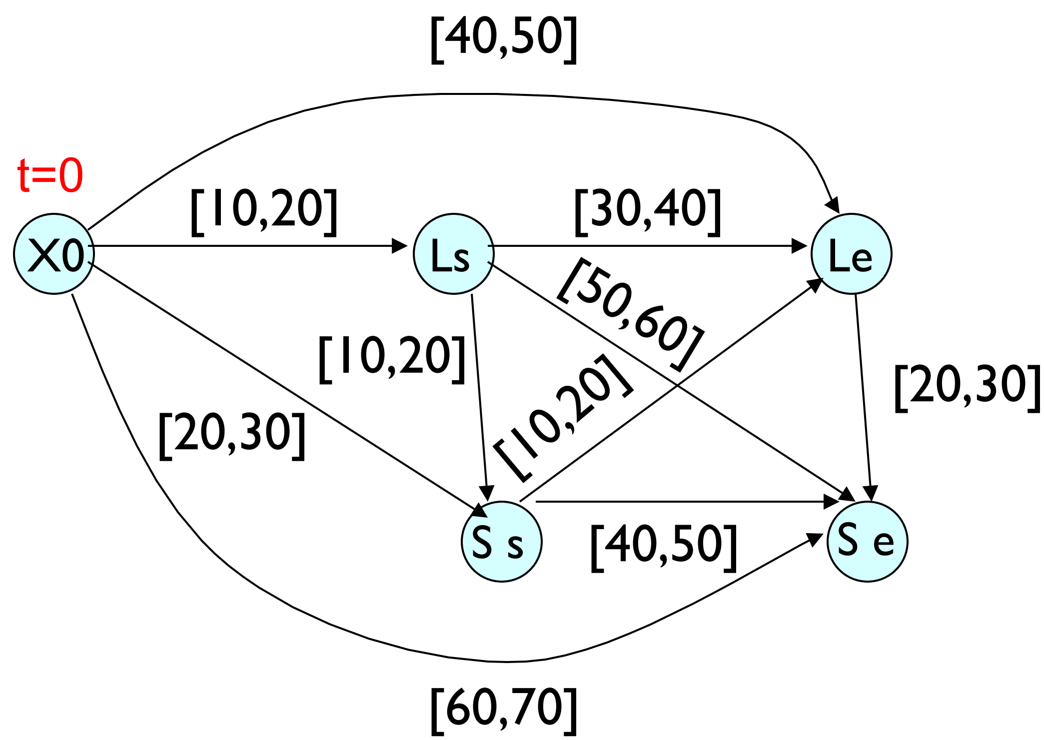
Output: Schedule (Assignment to X, consistsent with STN)

Property: Can assign variables in any order, without backtracking.

Key ideas

- **Incrementally tighten feasible intervals, as commitments are made.**

- Select value for X0



Scheduling without Search

Model-based Embedded & Robotic Systems

Input: Decomposable STN (APSP D-Graph)

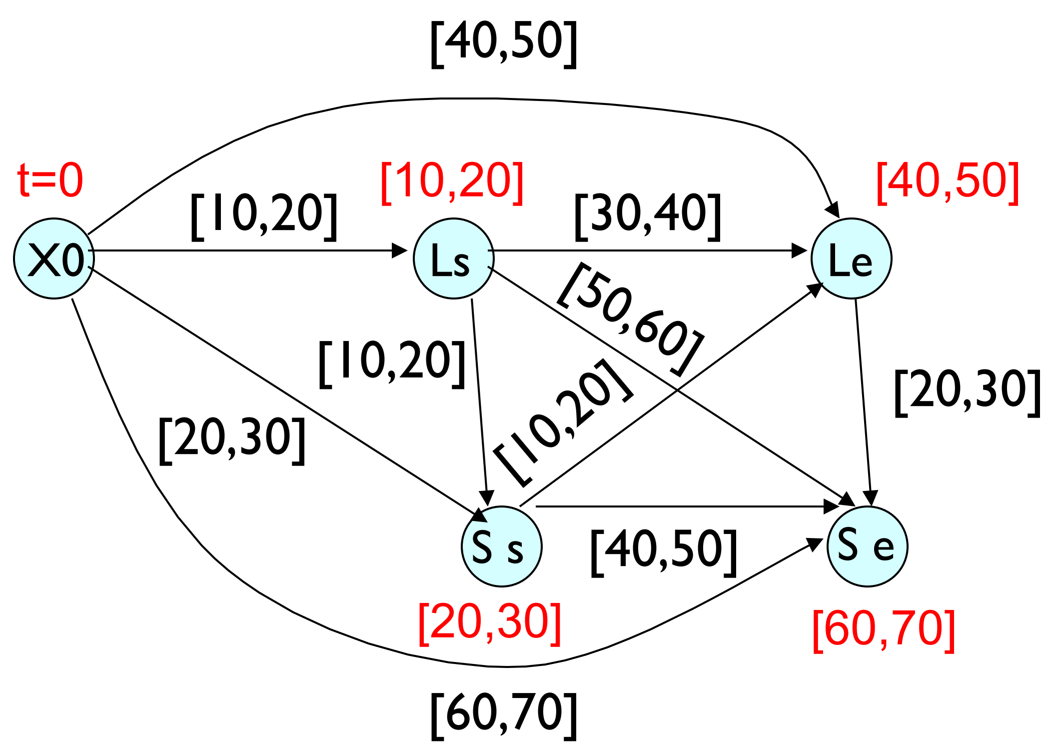
Output: Schedule (Assignment to X, consistsent with STN)

Property: Can assign variables in any order, without backtracking.

Key ideas

- **Incrementally tighten feasible intervals, as commitments are made.**

- Select value for X0



Scheduling without Search

Input: Decomposable STN (APSP D-Graph)

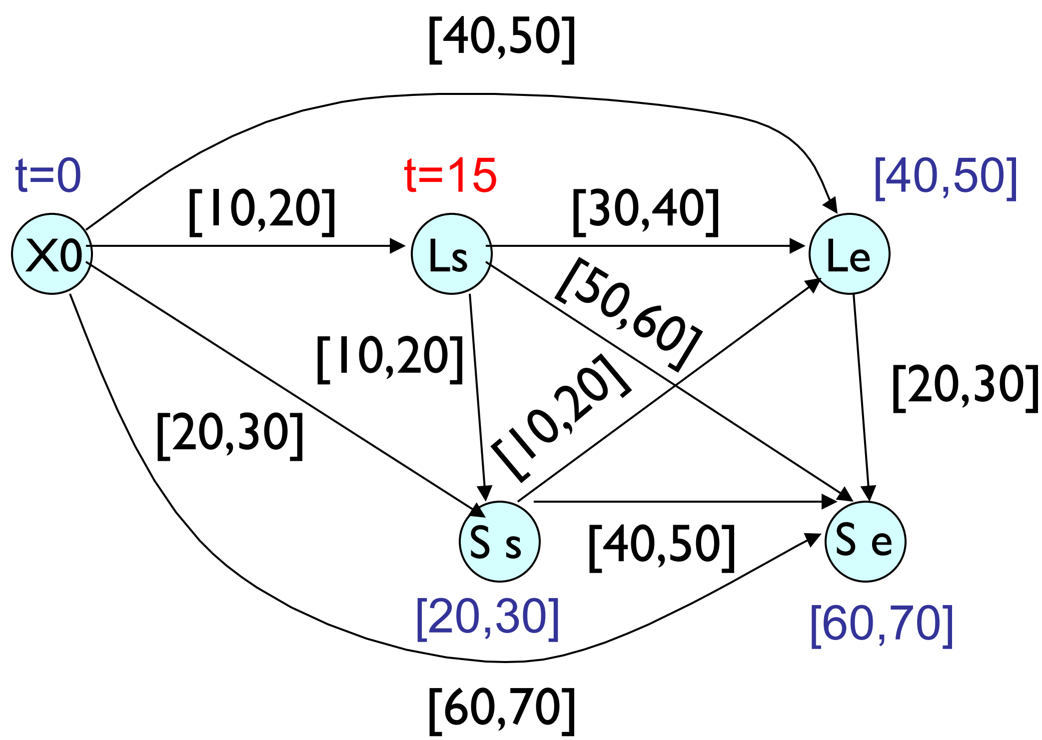
Output: Schedule (Assignment to X, consistsent with STN)

Property: Can assign variables in any order, without backtracking.

Key ideas

- **Incrementally tighten feasible intervals, as commitments are made.**

- Select value for X0
- Select value for Ls, consistent with X0



Scheduling without Search

Input: Decomposable STN (APSP D-Graph)

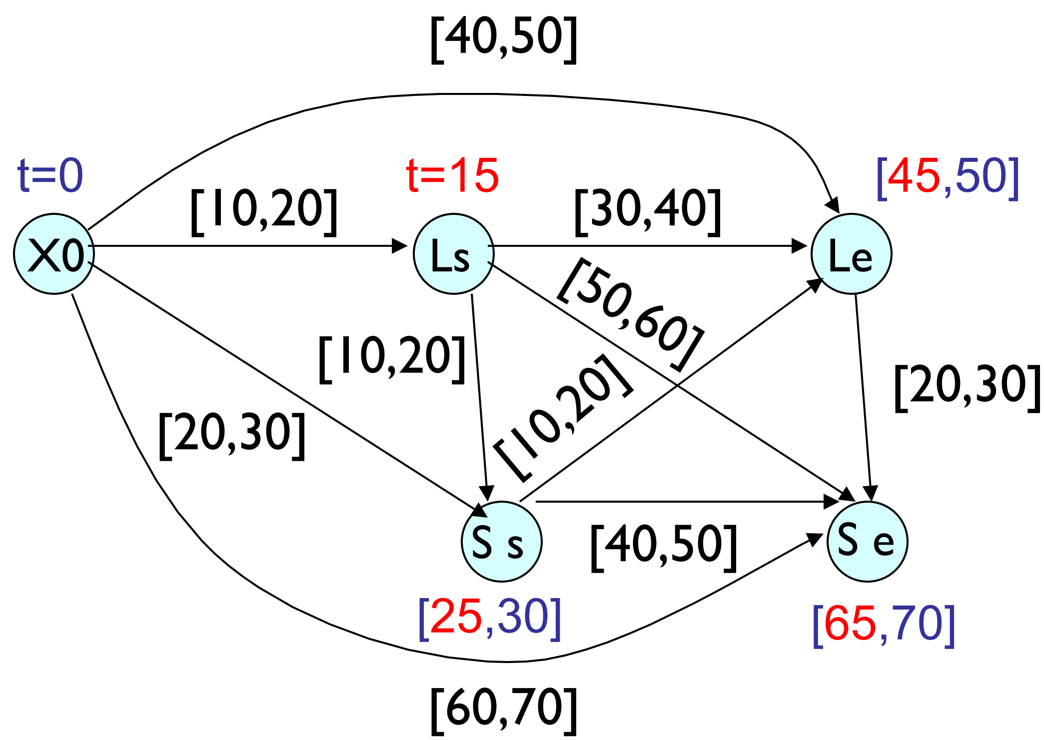
Output: Schedule (Assignment to X, consistsent with STN)

Property: Can assign variables in any order, without backtracking.

Key ideas

- **Incrementally tighten feasible intervals, as commitments are made.**

- Select value for X0
- Select value for Ls, consistent with X0



Scheduling without Search

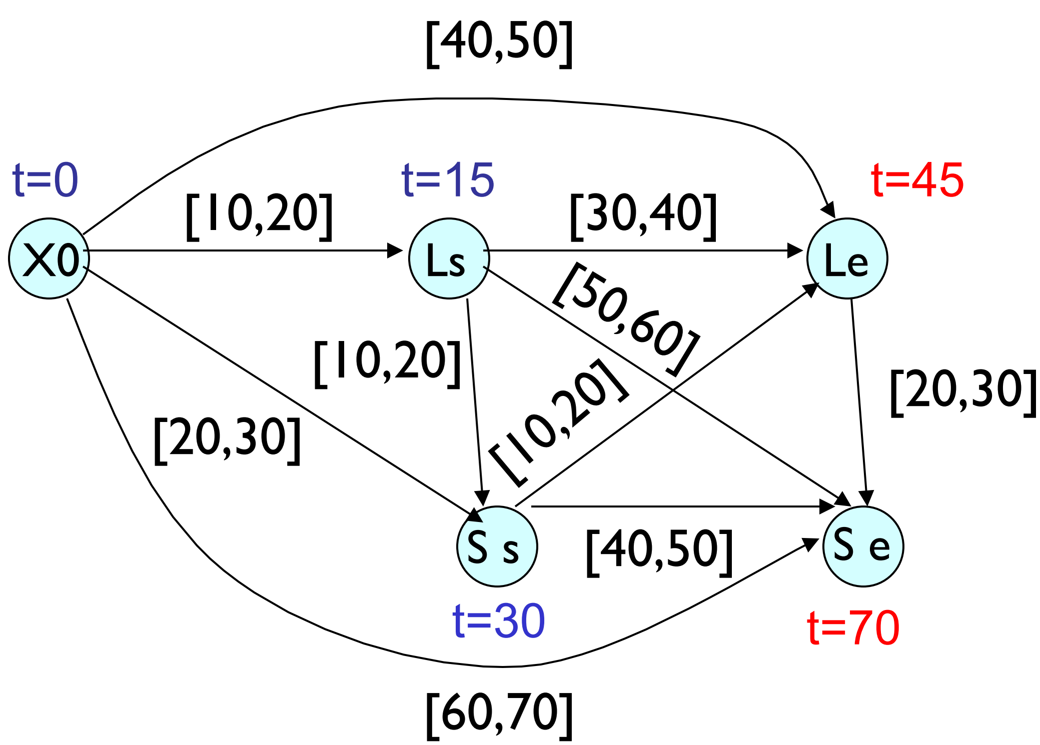
Input: Decomposable STN (APSP D-Graph)

Output: Schedule (Assignment to X, consistsent with STN)

Property: Can assign variables in any order, without backtracking.

Key ideas

- **Incrementally tighten feasible intervals, as commitments are made.**



- Select value for X0
- Select value for Ls, consistent with X0
- Select value for Le, consistent with X0, Ls
- Select value for Ss, consistent with X0, Ls, Le
- Select value for Se…

Flexible Execution

Schedule Off-line

1. Describe Temporal Plan
2. Test Consistency
3. Schedule Plan

offline

online

4. Execute Plan

Problem: delays and fluctuations in task duration can cause plan failure.

Observation: Least commitment temporal plans leave room to adapt.

Flexible Execution adapts through dynamic scheduling [Muscettola et al]

- Assigns time to event when executed.

Flexible Execution

Schedule Off-line

1. Describe Temporal Plan
2. Test Consistency
3. Schedule Plan

offline

online

4. Execute Plan

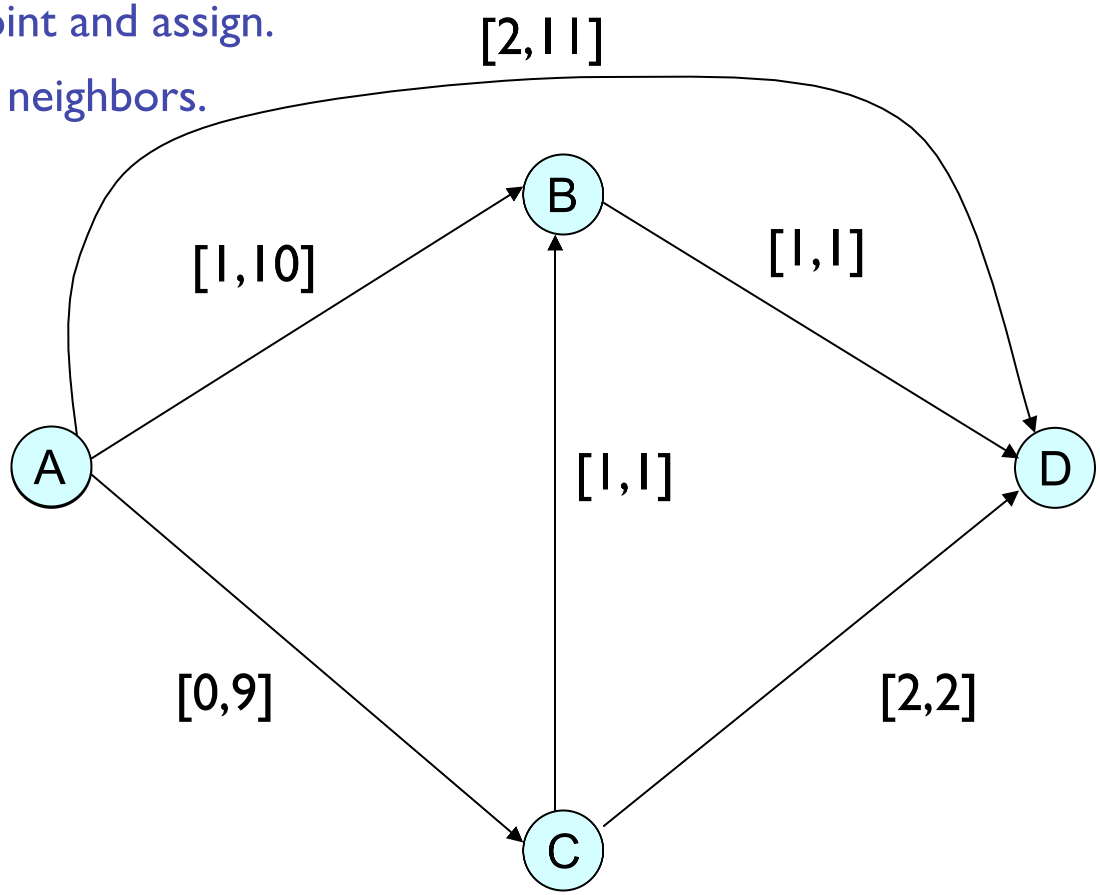
Schedule Online

1. Describe Temporal Plan
2. Test Consistency
3. Reformulate Plan
4. Dynamically Schedule Plan

Dynamic Scheduling by Decomposition?

Consider a Simple Example

- Select executable timepoint and assign.
- Propagate assignment to neighbors.



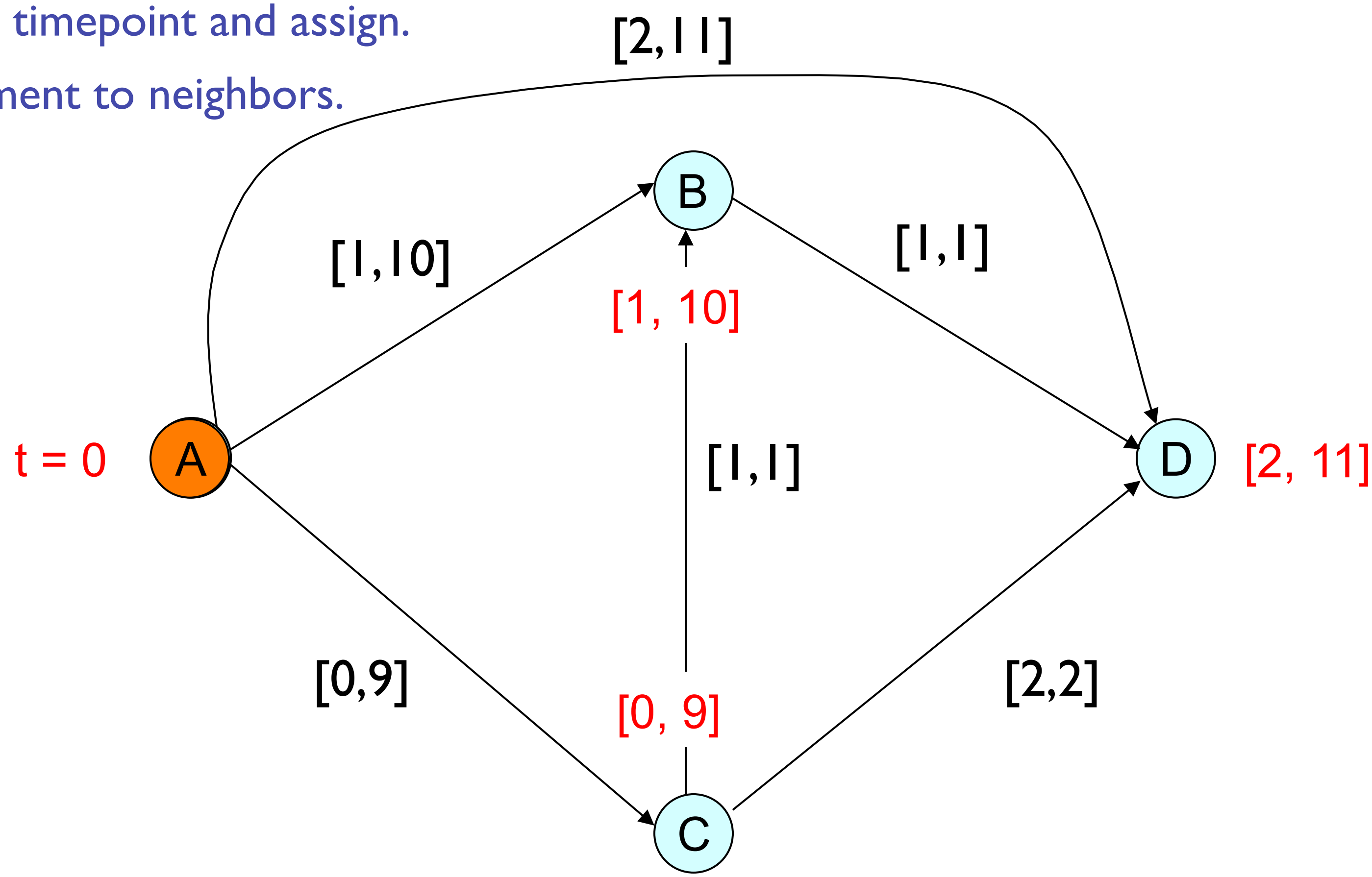
[Muscettola, Morris, Tsamardinos KR98]

Dynamic Scheduling by Decomposition?

Model-based Embedded & Robotic Systems

Consider a Simple Example

- Select executable timepoint and assign.
- Propagate assignment to neighbors.

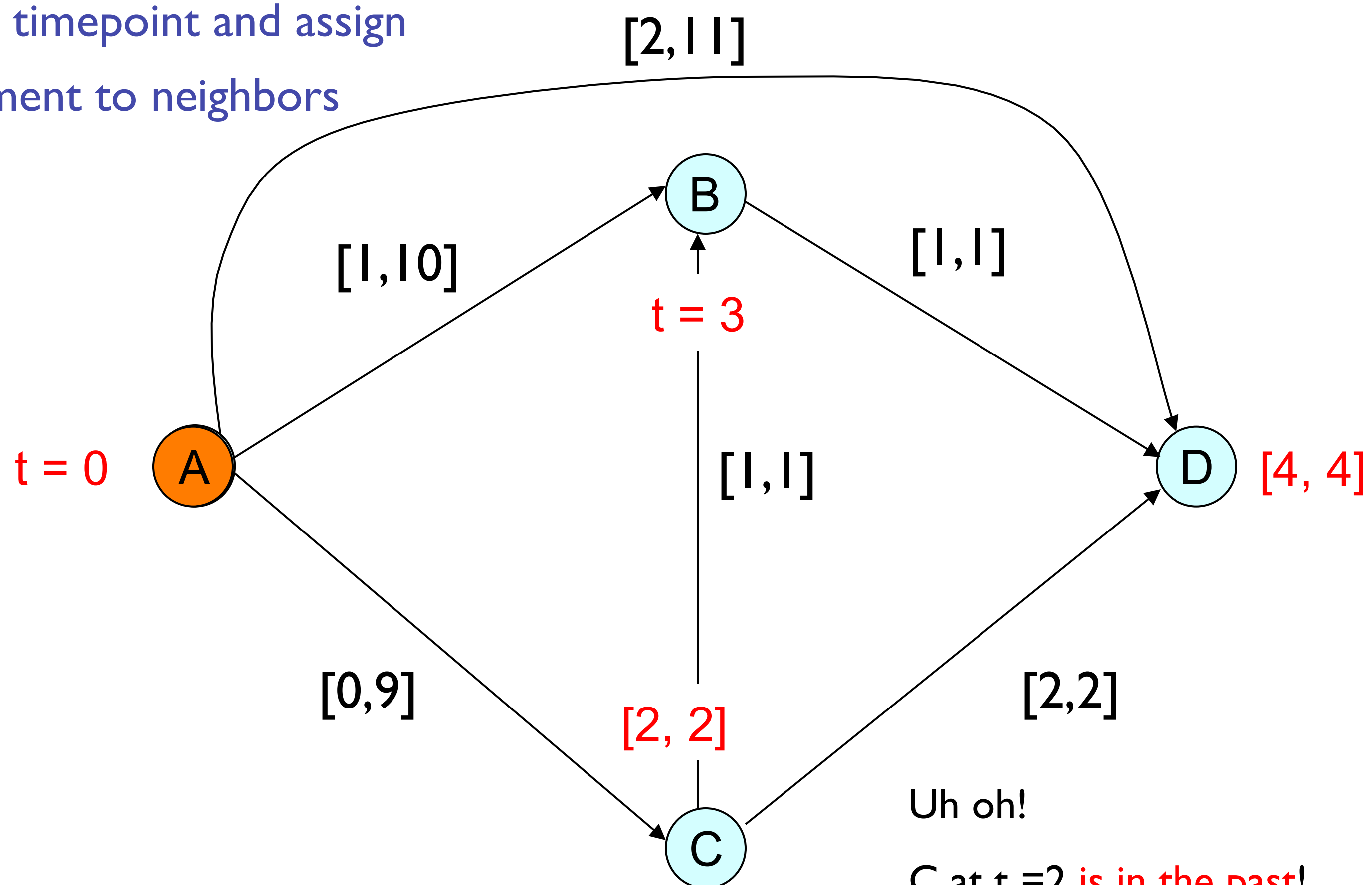


[Muscettola, Morris, Tsamardinos KR98]

Dynamic Scheduling by Decomposition?

Consider a Simple Example

- Select executable timepoint and assign
- Propagate assignment to neighbors



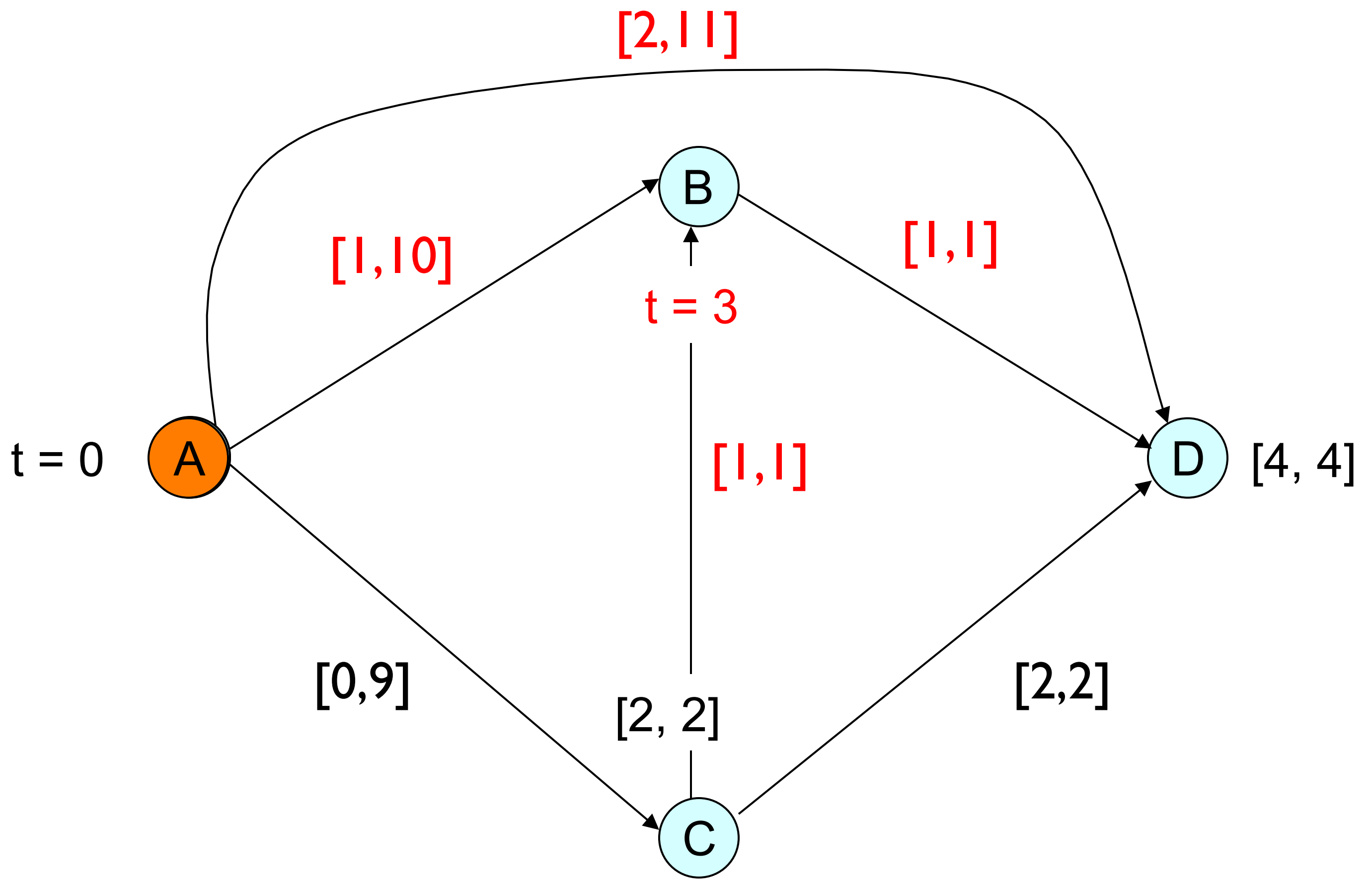
Uh oh!

C at t =2 is in the past!

[Muscettola, Morris, Tsamardinos KR98]

Dynamic Scheduling by Decomposition?

- Fix by scheduling according to implied orderings.
  - A <= C < B < D

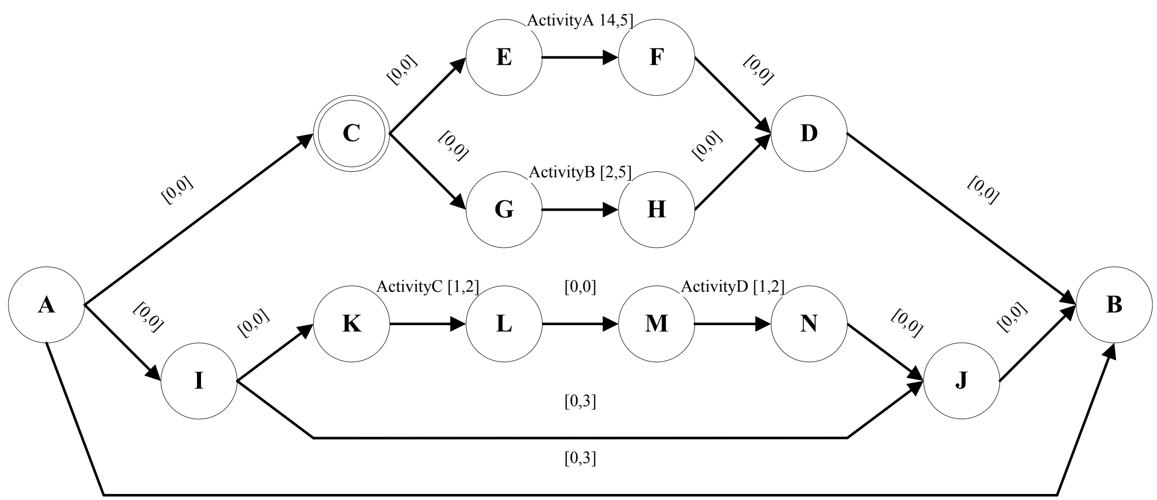


[Muscettola, Morris, Tsamardinos KR98]

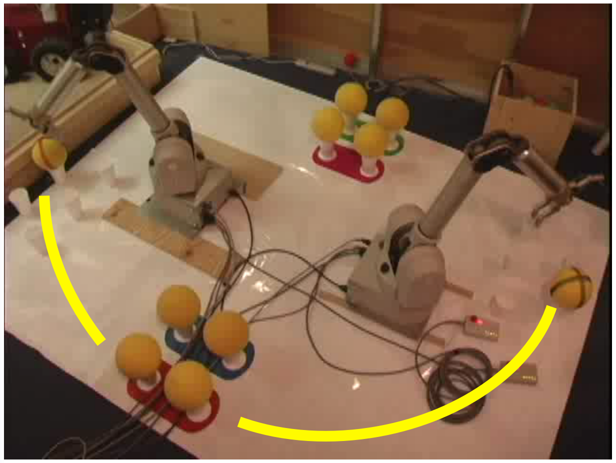
Generalizations of Dynamic Execution

1. Dynamically choose a) when, b) by whom, c) which method, and d) how.
2. Achieve consistency wrt models of uncontrollable events.

- Temporal Plan Networks (under Uncertainty)
- Disjunctive Temporal Networks (under Uncertainty)



Multi-Robot Teamwork



Agents choose and schedule activities.

(Someone) Remove one ball from red bin.

Remove one ball from red bin.

 OR 

L[32,39] V R[42,55]

t_{start}

Swap black striped ball

- Right Robot picks up and offers ball.
- Robots perform hand-to-hand swap.

Swap red striped ball

- Left Robot picks up and offers ball.
- Robots perform hand-to-hand swap.

Remove one ball from red bin.

Remove one ball from blue bin.

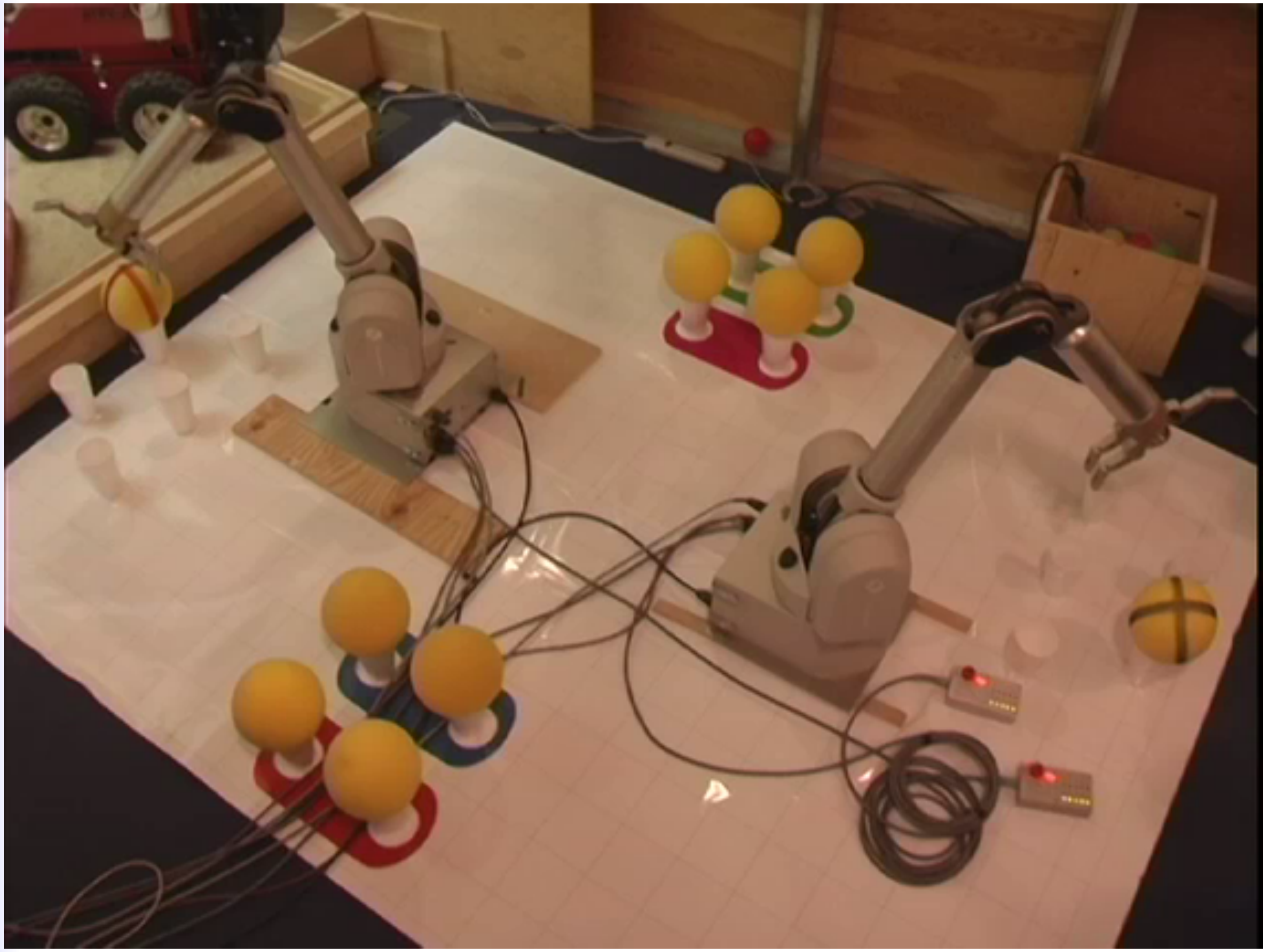
Remove one ball from pink bin.

Remove one ball from green bin.

t_{finish}

Multi-Robot Teamwork

Model-based Embedded & Robotic Systems



- Off-nominal.
- Partner adapts in response to teammate's failure.

Kim, Williams, Abramson IJCAI 2001; Shah and Williams ICAPS 2009; Conrad, Shah and Williams ICAPS 2010

Leader & Assistant

Assistant waits to see what Leader will do before acting.

Leader

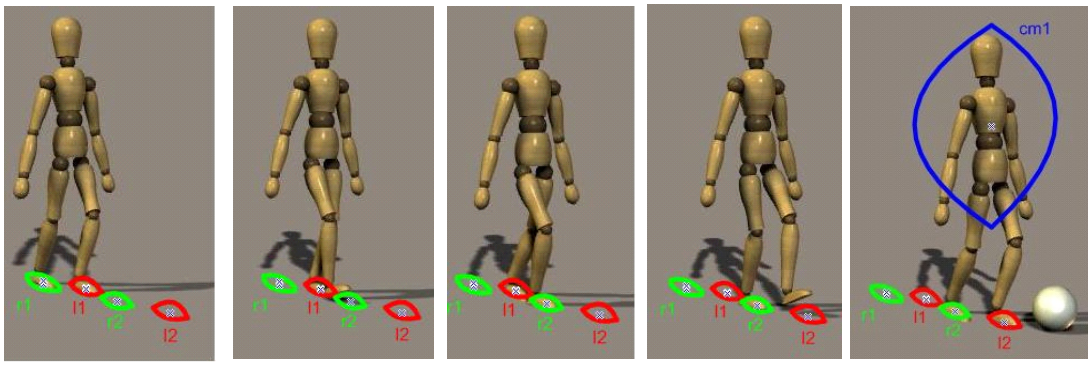
Assistant

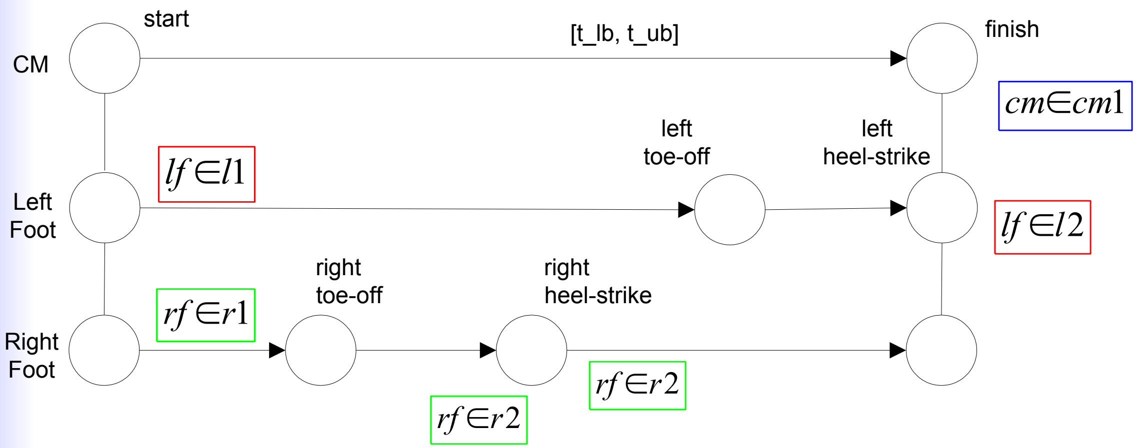
Shah and Williams ICAPS 2010

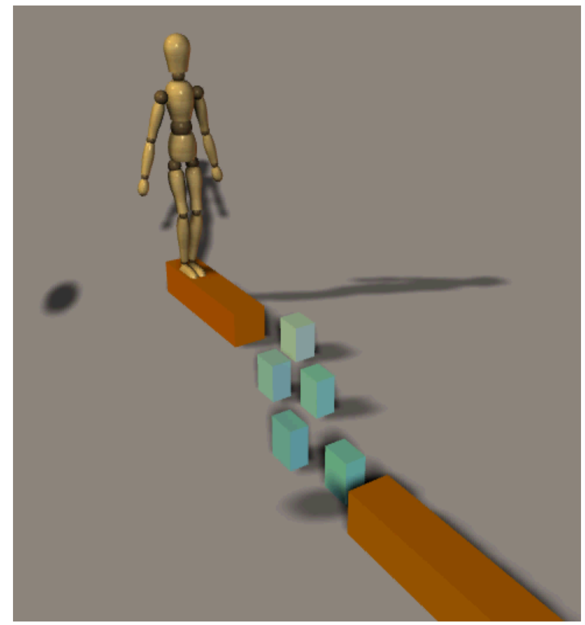
Idea: model leader durations and assignments as uncontrollable (TPNU).

Achieve safety by adaptively executing plans on qualitative poses

Input: Qualitative State Plan

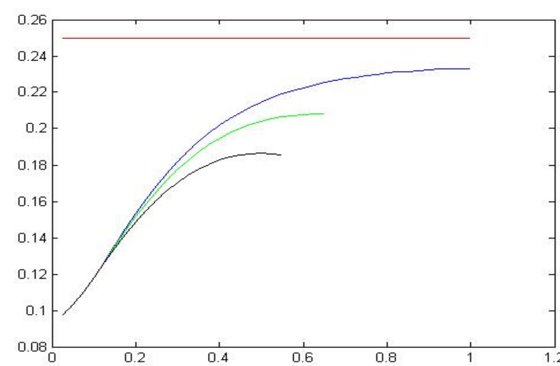






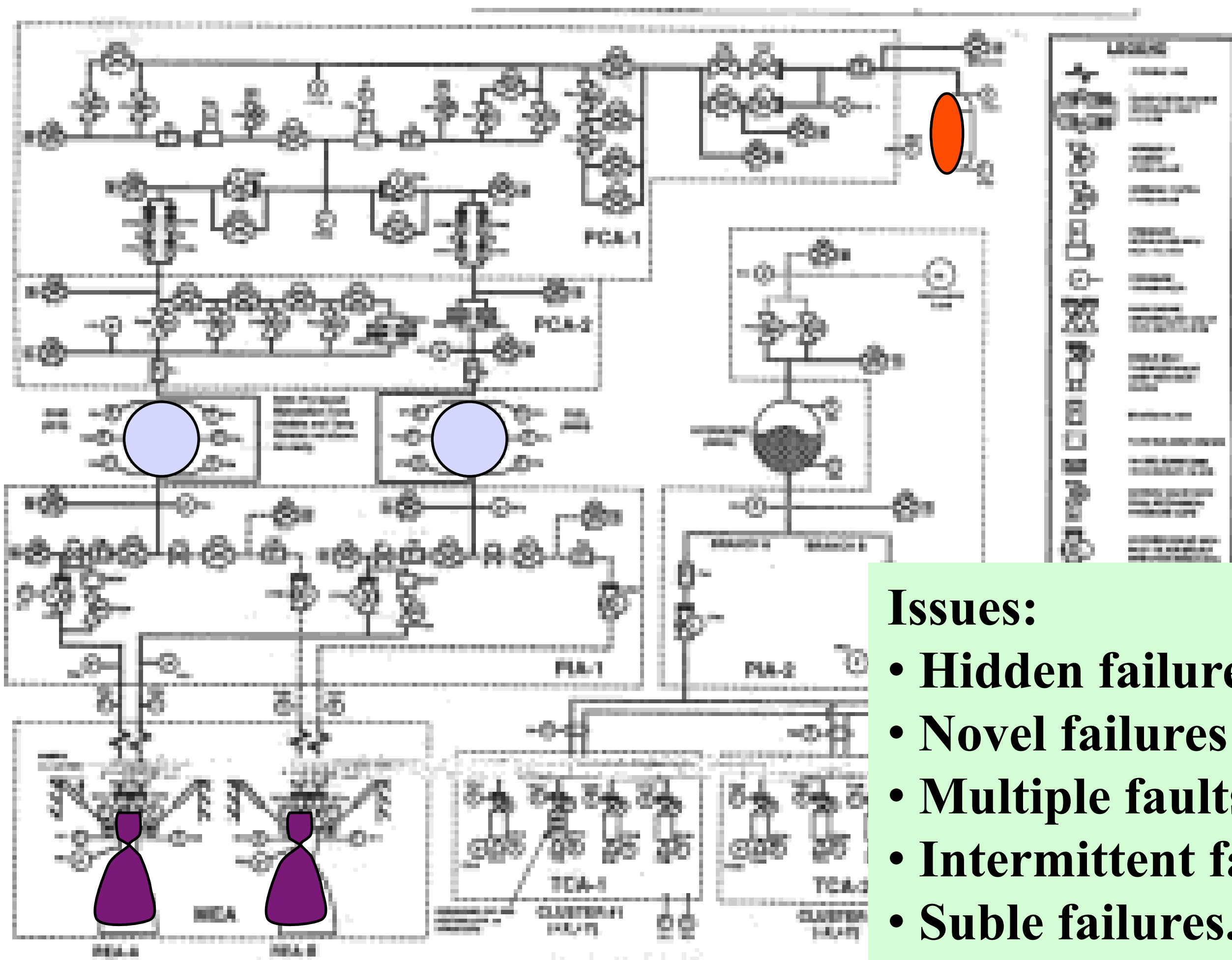
Lateral CM with push disturbance

- **Blue – 40 N**
- **Green – 35 N**
- **Black – 25 N**
- **Red – Max allowed displacement**



Outline

- Robust, Goal-directed Execution
- Plan Dispatching
- Diagnosis and Mode Estimation
- Plan Generation



Issues:

- **Hidden failures**
- **Novel failures**
- **Multiple faults**
- **Intermittent failures**
- **Suble failures.**
- **HW / SW interactions**

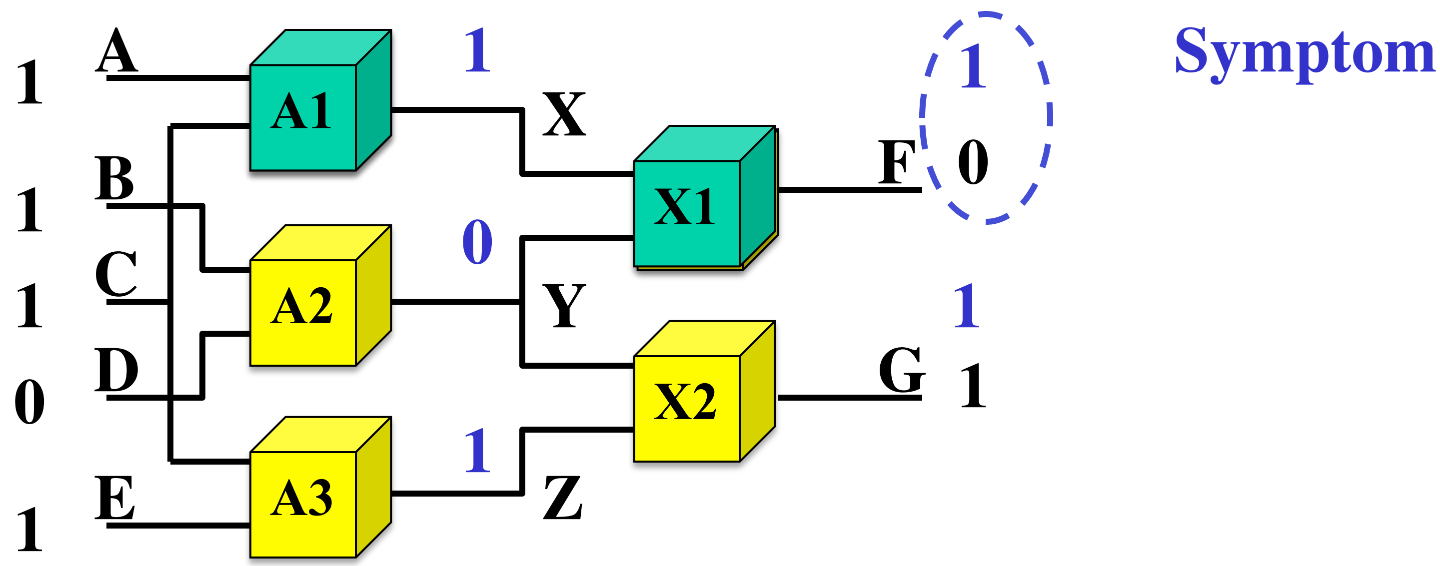
Mode Estimation

- Mode estimates and kernels.
- By divide and conquer (GDE).
- Likely estimates and Conflict-directed A*.
- Mode reconfiguration.
- Estimating probabilistic (hybrid) constraint automata.

Model-based Diagnosis

Input: Observations of a system with symptomatic behavior, and a model Φ of the system.

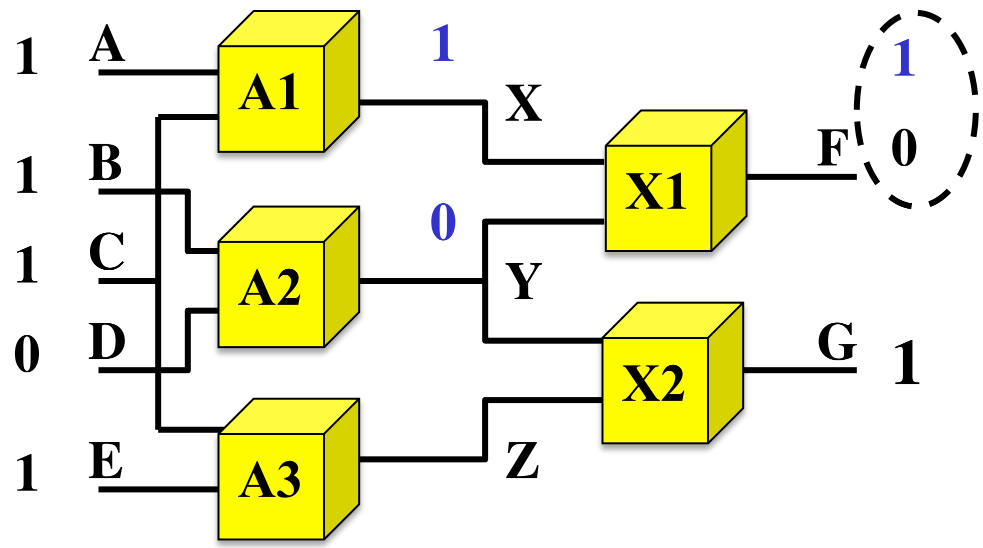
Output: Diagnoses that account for the symptoms.



How Should Diagnoses Account for Novel Failures?

Consistency-based Diagnosis: Given symptoms, find diagnoses that are consistent with symptoms.

Suspending Constraints: For novel faults, make no presumption about faulty component behavior.



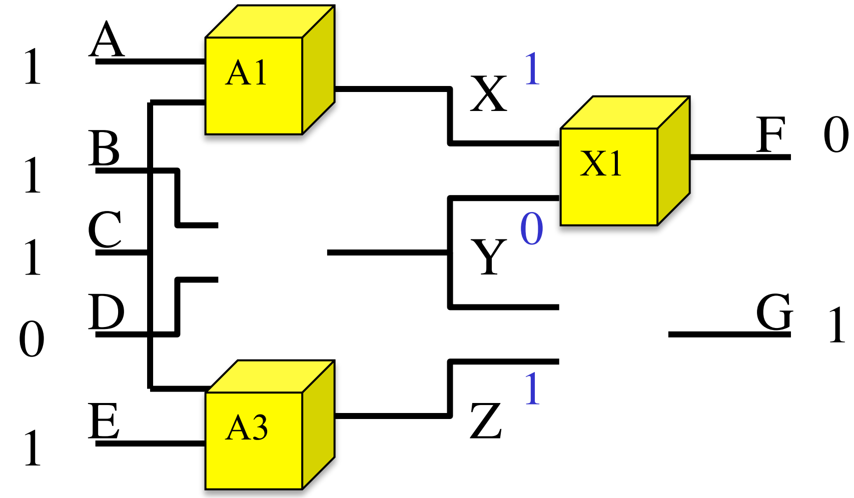
Symptom

[deKleer & Brown, 83]
[Davis, 84]
[Geneserth, 84]

Multiple Faults: Identify all Combinations of Consistent “Unknown” Modes

And(i):

- G(i):
  Out(i) = In1(i) AND In2(i)
- U(i): *No Constraint*

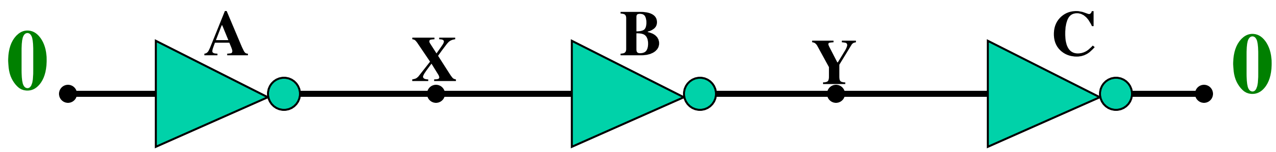


Diagnosis = {A1=G, A2=U, A3=G, X1=G, X2=U}

- Candidate: Assignment of G or U to each component.
- Diagnosis: Candidate consistent with model and observations.

Incorporating (Failure) Modes: Mode Estimation

Sherlock
[de Kleer & Williams, IJCAI 89]



Idea: Include Nominal, Fault and Unknown Modes

Inverter(i):

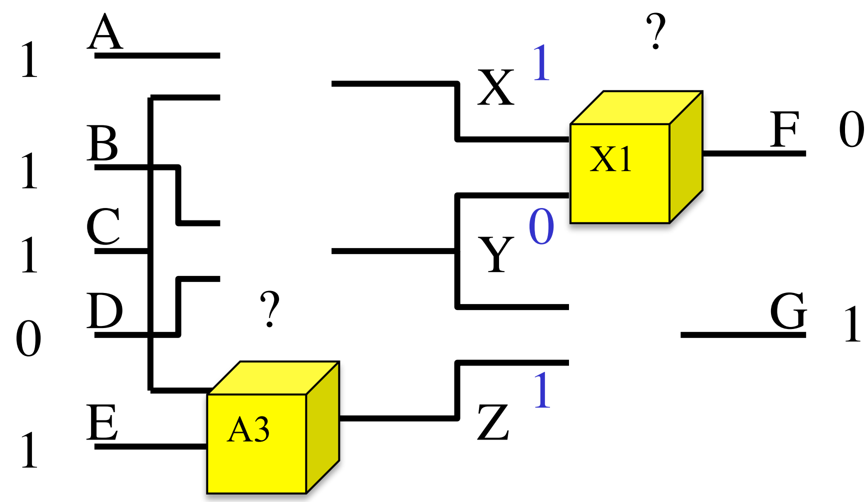
- G(i): Out(i) = not(In(i))
- S1(i): Out(i) = 1
- S0(i): Out(i) = 0
- U(i):

- Isolates unknown faults.
- Explains known faults.

Compact Encoding: Partial Diagnoses

Partial Diagnosis

{A1=U, A2=U, X2=U}



Partial Diagnosis:

A partial mode assignment M,

that "removes all symptoms."

- All full extensions of M are diagnoses.

Extensions (Diagnoses):

{A1=U, A2=U, A3=G, X1=G, X2=U}

{A1=U, A2=U, A3=G, X1=U, X2=U}

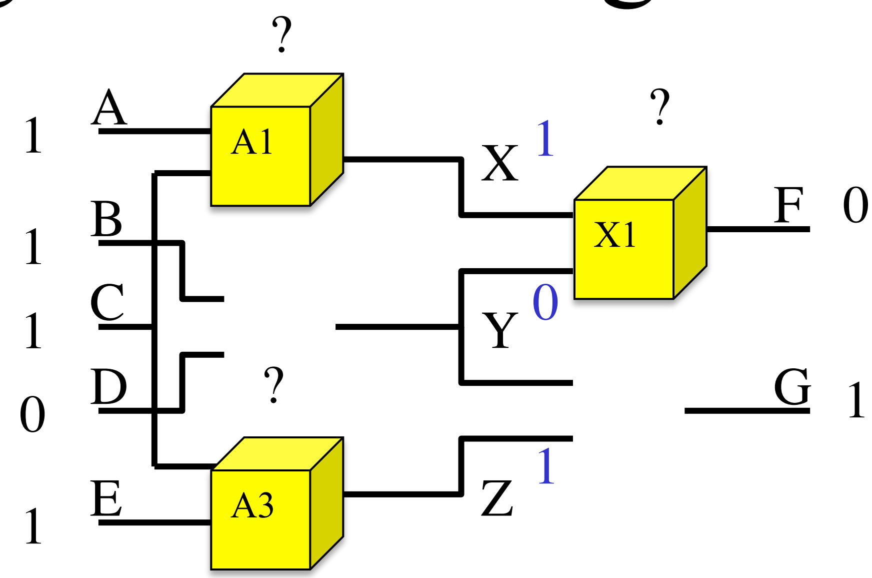
{A1=U, A2=U, A3=U, X1=G, X2=U}

{A1=U, A2=U, A3=U, X1=U, X2=U}

Compact Encoding: Kernel Diagnoses

Kernel Diagnosis

{A2=U, X2=U}



Partial Diagnosis:

A partial mode assignment M,

that "removes all symptoms".

- All full extensions of M are diagnoses.

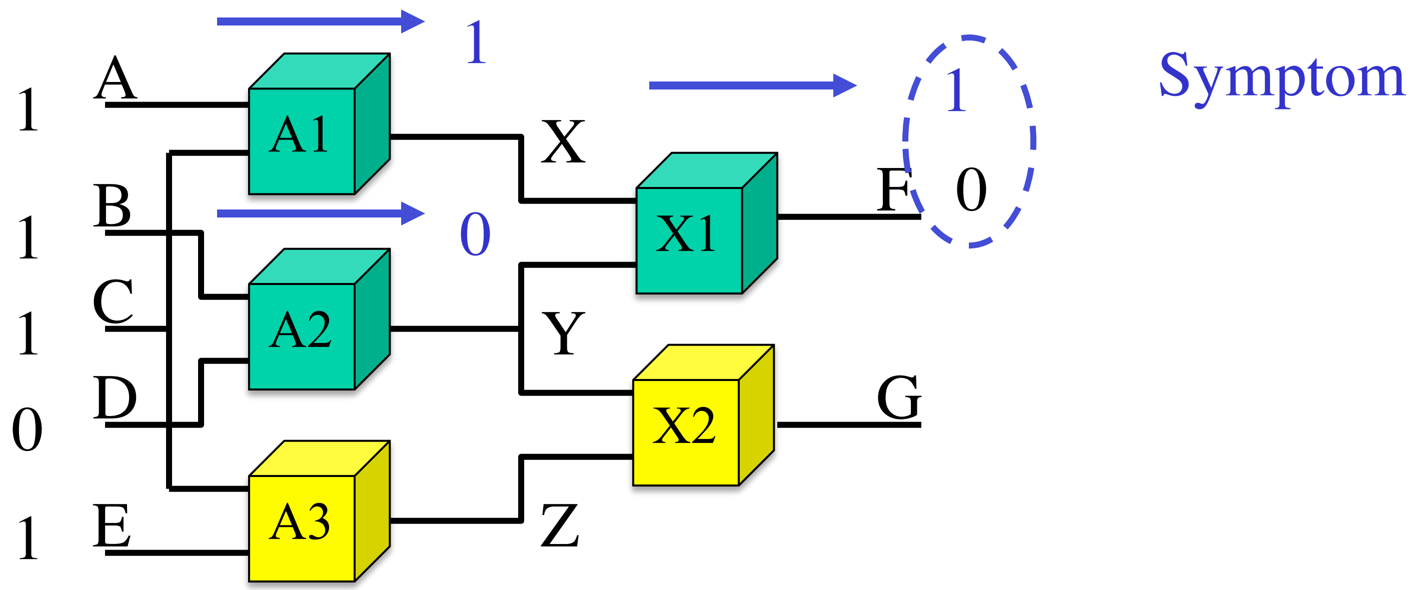
Kernel Diagnosis:

A partial diagnosis K,
no subset of which is a partial diagnosis.

Mode Estimation

- Mode estimates and kernels.
- By divide and conquer (GDE).
- Likely estimates and Conflict-directed A*.
- Mode reconfiguration.
- Estimating probabilistic (hybrid) constraint automata.

Conflicts Explain How to Remove Symptoms



Symptom:

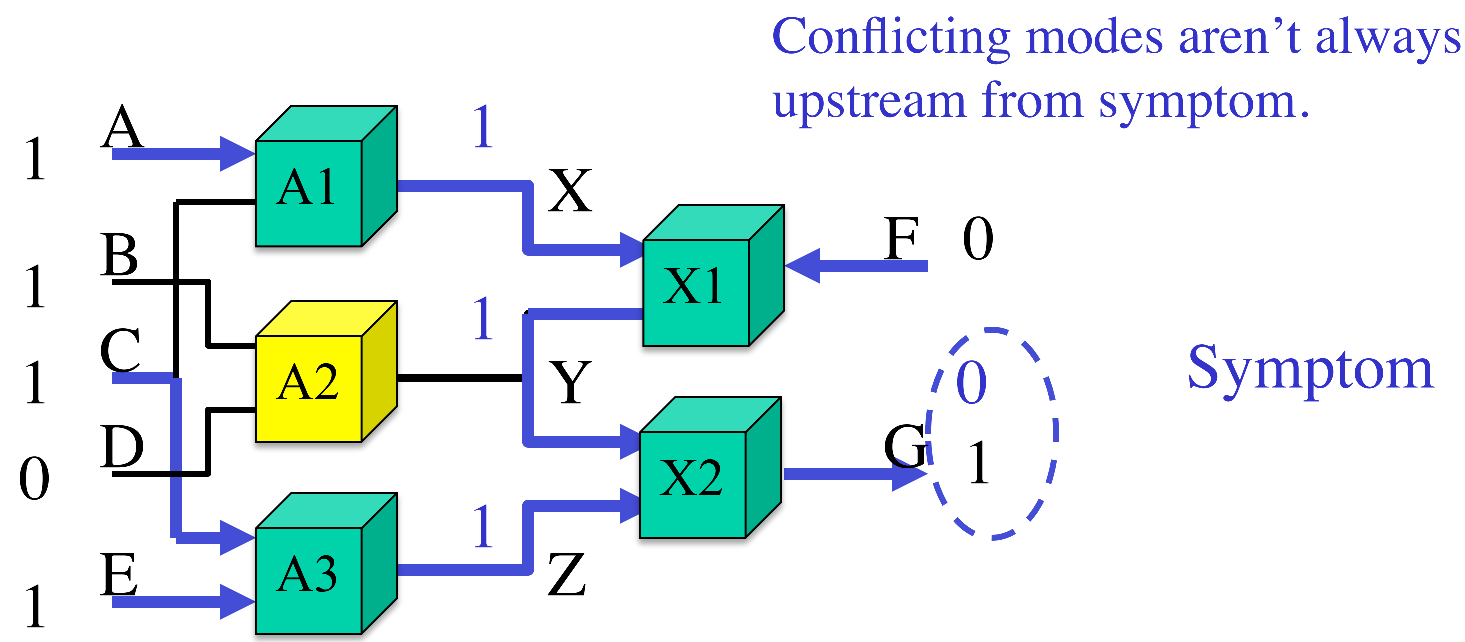
F is observed 0, but predicted to be 1 if A1, A2 and X1 are okay.

Conflict 1: {A1=G, A2=G, X1=G} is inconsistent.

→ One of A1, A2 or X1 must be broken.

Conflict: An inconsistent partial assignment to mode variables X.

Second Conflict



Symptom: G is observed 1, but predicted 0.

Conflict 2: {A1=G, A3=G, X1=G, X2=G} is inconsistent.

→ One of A1, A3, X1 or X2 must be broken.

Candidate Generation: Generate Kernels From Conflicts

{A1=G, A2=G, X1=G} Conflict 1.

{A1=G, A3=G, X1=G, X2=G} Conflict 2.

{A1=U, A2=U, X1=U} diagnoses for Conflict 1.

{A1=U, A3=U, X1=U, X2=U} diagnoses for Conflict 2.

Kernel Diagnoses =

"Smallest" sets of modes that remove all conflicts.

Candidate Generation: Generate Kernels From Conflicts

{A1=G, A2=G, X1=G} Conflict 1.

{A1=G, A3=G, X1=G, X2=G} Conflict 2.

{A1=U, A2=U, X1=U} diagnoses for Conflict 1.

{A1=U, A3=U, X1=U, X2=U} diagnoses for Conflict 2.

Kernel Diagnoses = {A1=U}

1. Compute cross product.
2. Remove supersets.
   - Old subset New.
   - New subset Old.

"Smallest" sets of modes that remove all conflicts.

Candidate Generation: Generate Kernels From Conflicts

{A1=G, A2=G, X1=G} Conflict 1.

{A1=G, A3=G, X1=G, X2=G} Conflict 2.

{A1=U, A2=U, X1=U} constituents of Conflict 1.

{A1=U, A3=U, X1=U, X2=U} constituents of Conflict 2.

Kernel Diagnoses = {X1=U}
{A2=U, X2=U}
{A2=U, A3=U}
{A1=U}

1. Compute cross product.
2. Remove supersets.
   - Old subset New.
   - New subset Old.

"Smallest" sets of modes that remove all conflicts.

Mode Estimation

- Estimates and kernels.
- By divide and conquer (GDE).
- Likely estimates and Conflict-directed A*.
- Mode reconfiguration.
- Estimating probabilistic (hybrid) constraint automata.

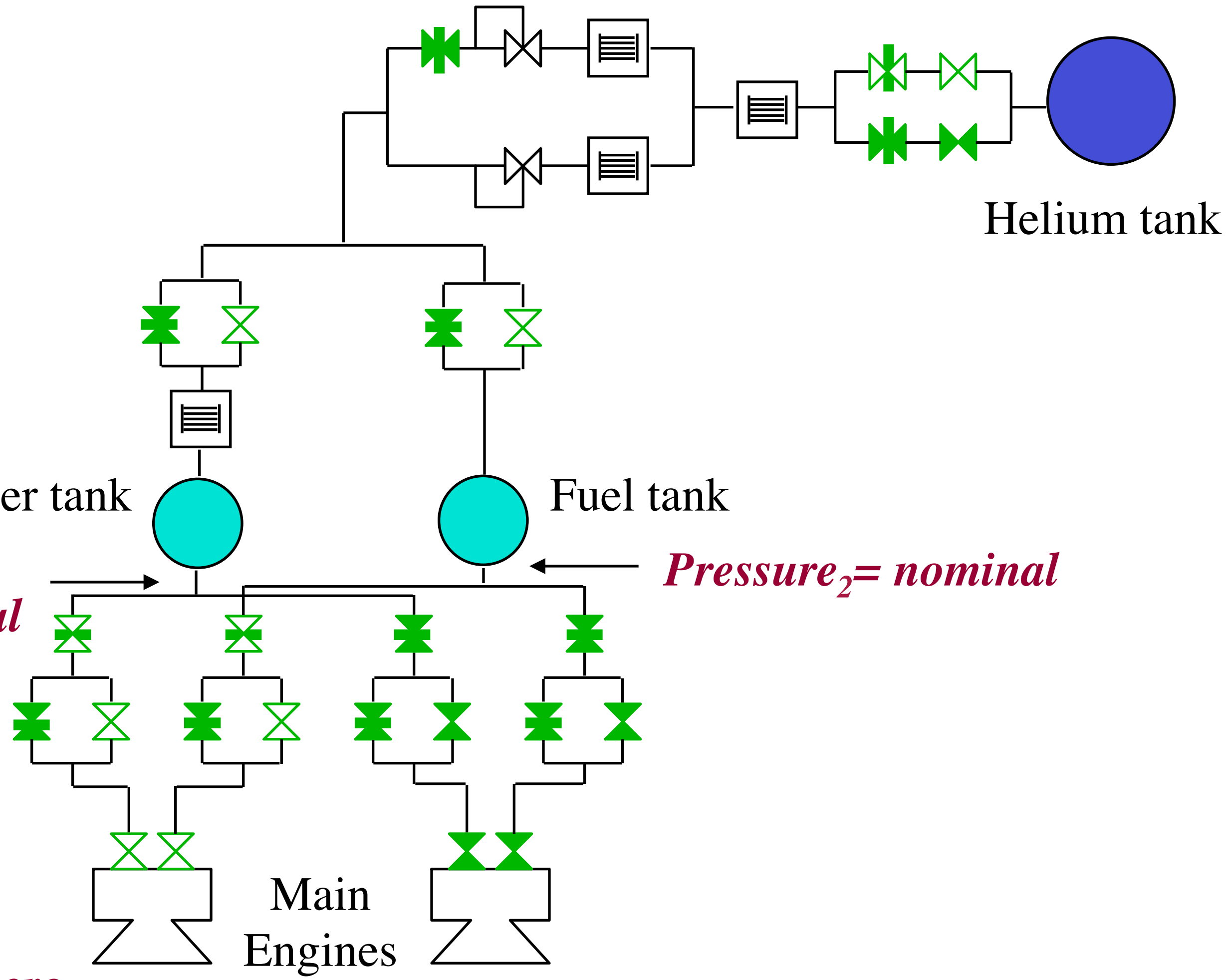
Mode Estimation as Conflict-directed Best First Search

When you have eliminated the impossible, whatever remains, however improbable, must be the truth.

- Sherlock Holmes. The Sign of the Four.

1. Generate most likely hypothesis.
2. Test hypothesis.
3. If inconsistent, learn reason for inconsistency (a conflict).
4. Use conflicts to leap over similarly infeasible options to next best hypothesis.

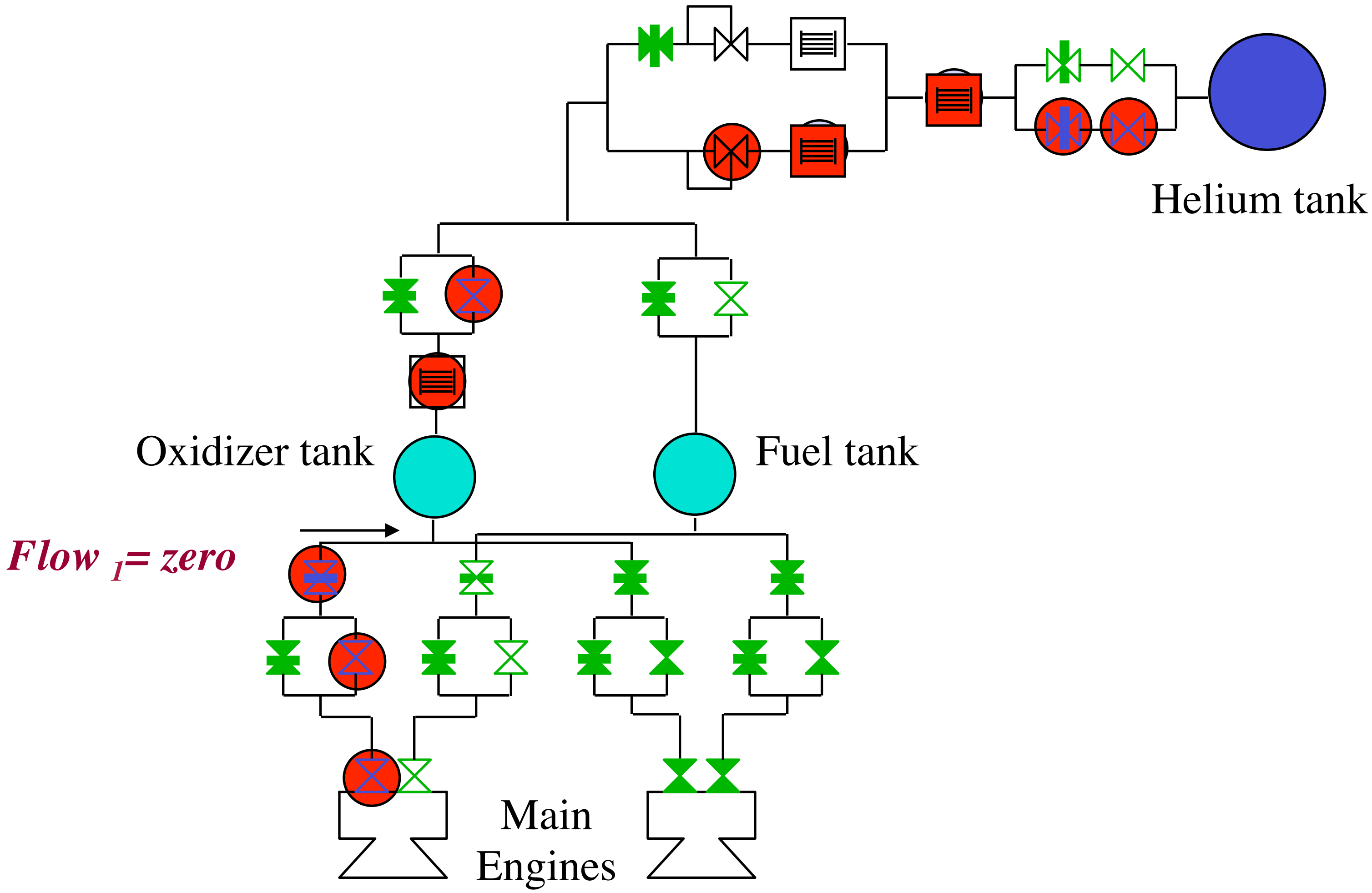
Compare Most Likely Hypothesis to Observations



Flow$_1$ = zero
Pressure$_1$ = nominal
Pressure$_2$= nominal
Acceleration = zero

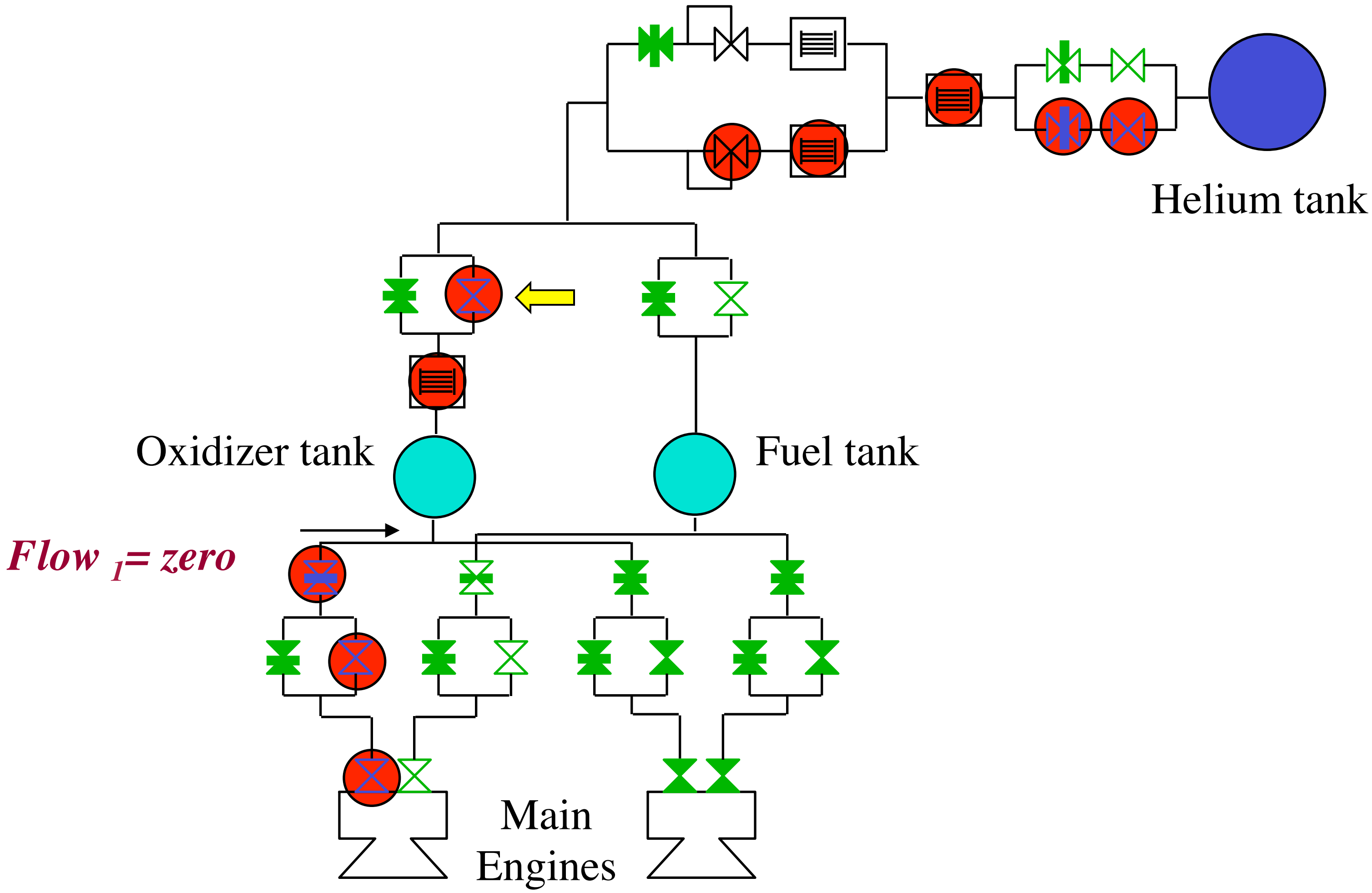
It is most likely that all components are okay.

Isolate Conflicting Information



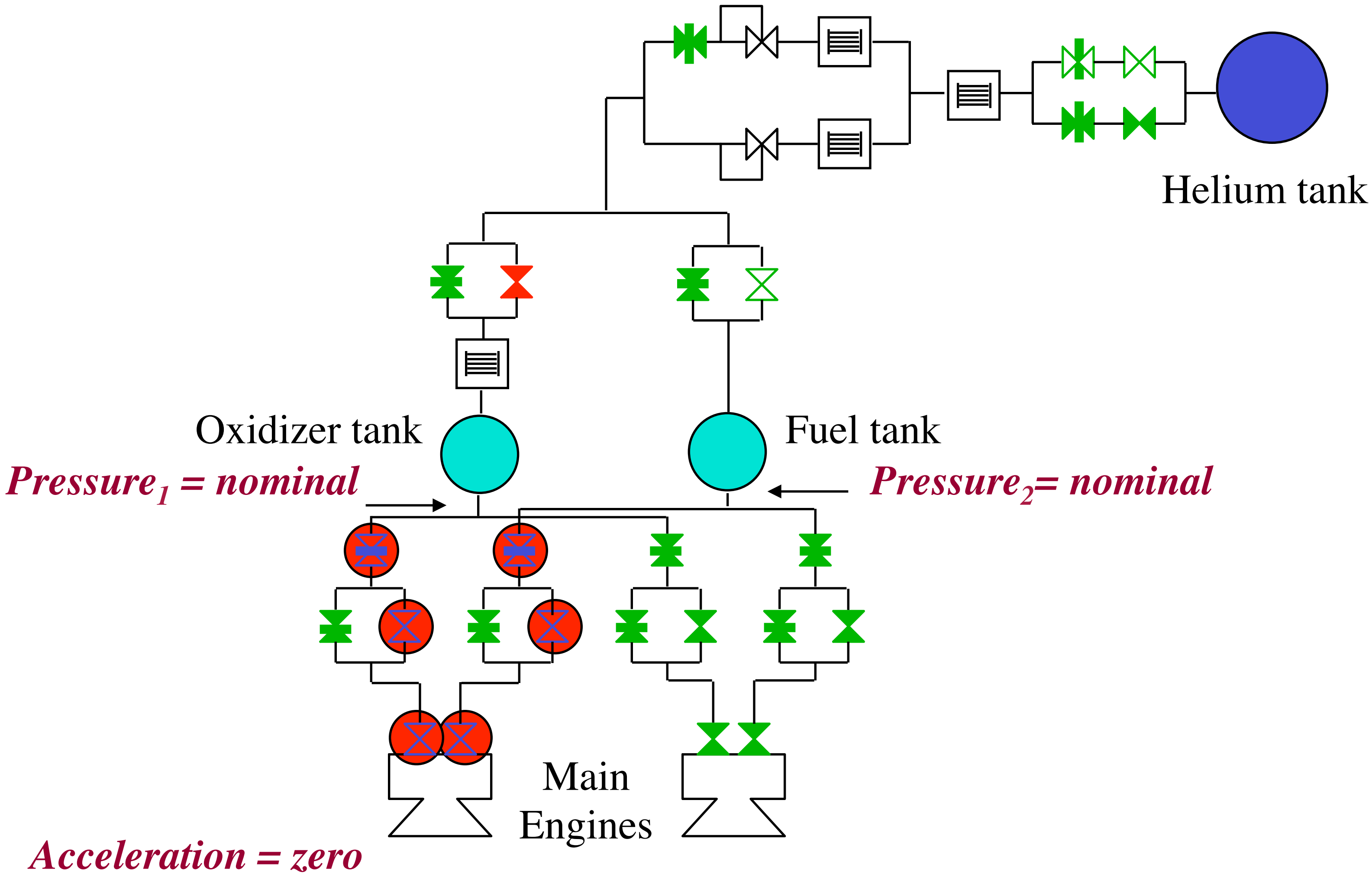
The red component modes *conflict* with the model and observations.

Leap to the Next Most Likely Hypothesis that Resolves the Conflict



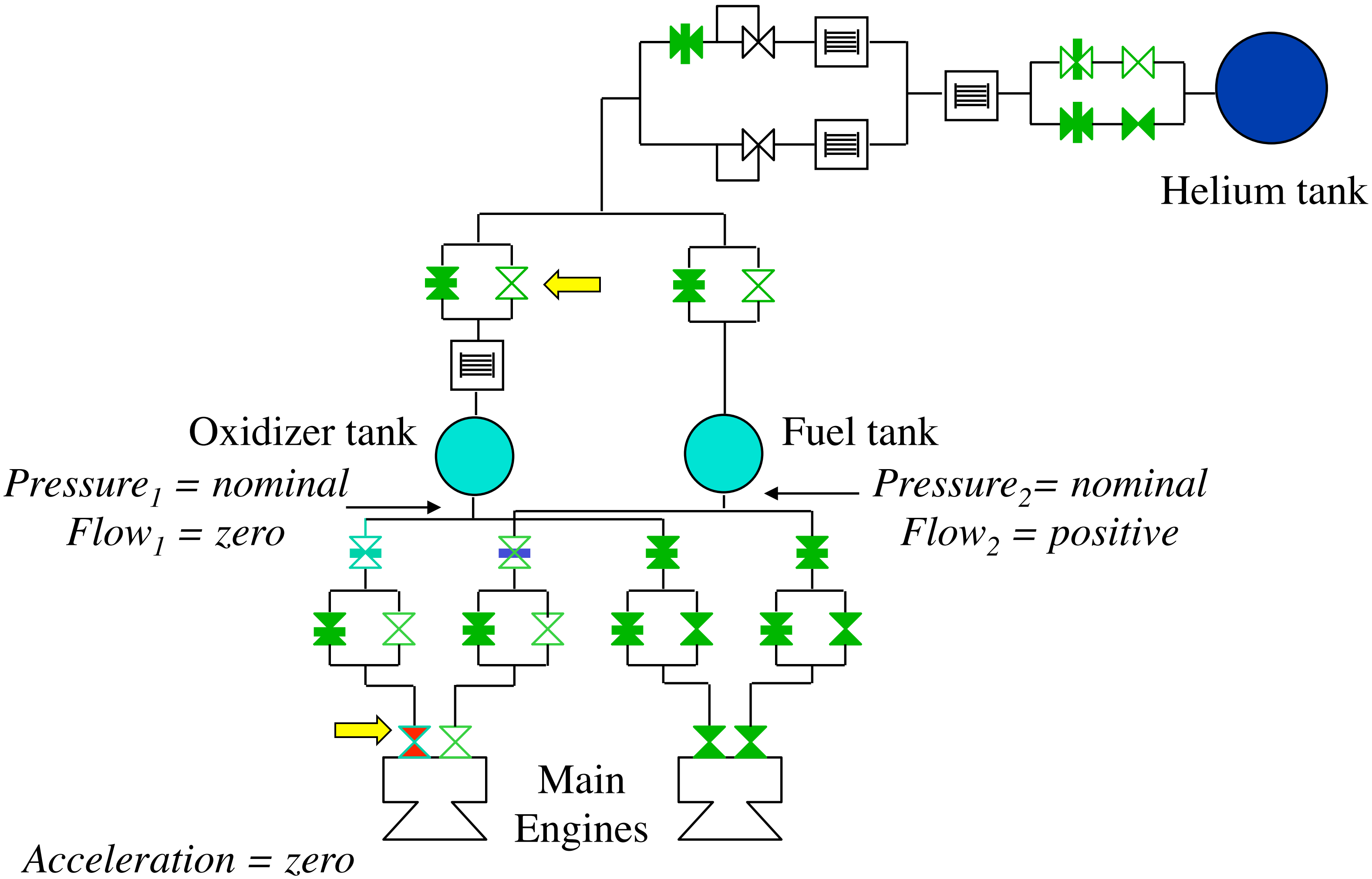
The next hypothesis must remove the conflict.

New Hypothesis Exposes Additional Conflicts



Another conflict, try removing both.

Final Hypothesis Resolves all Conflicts

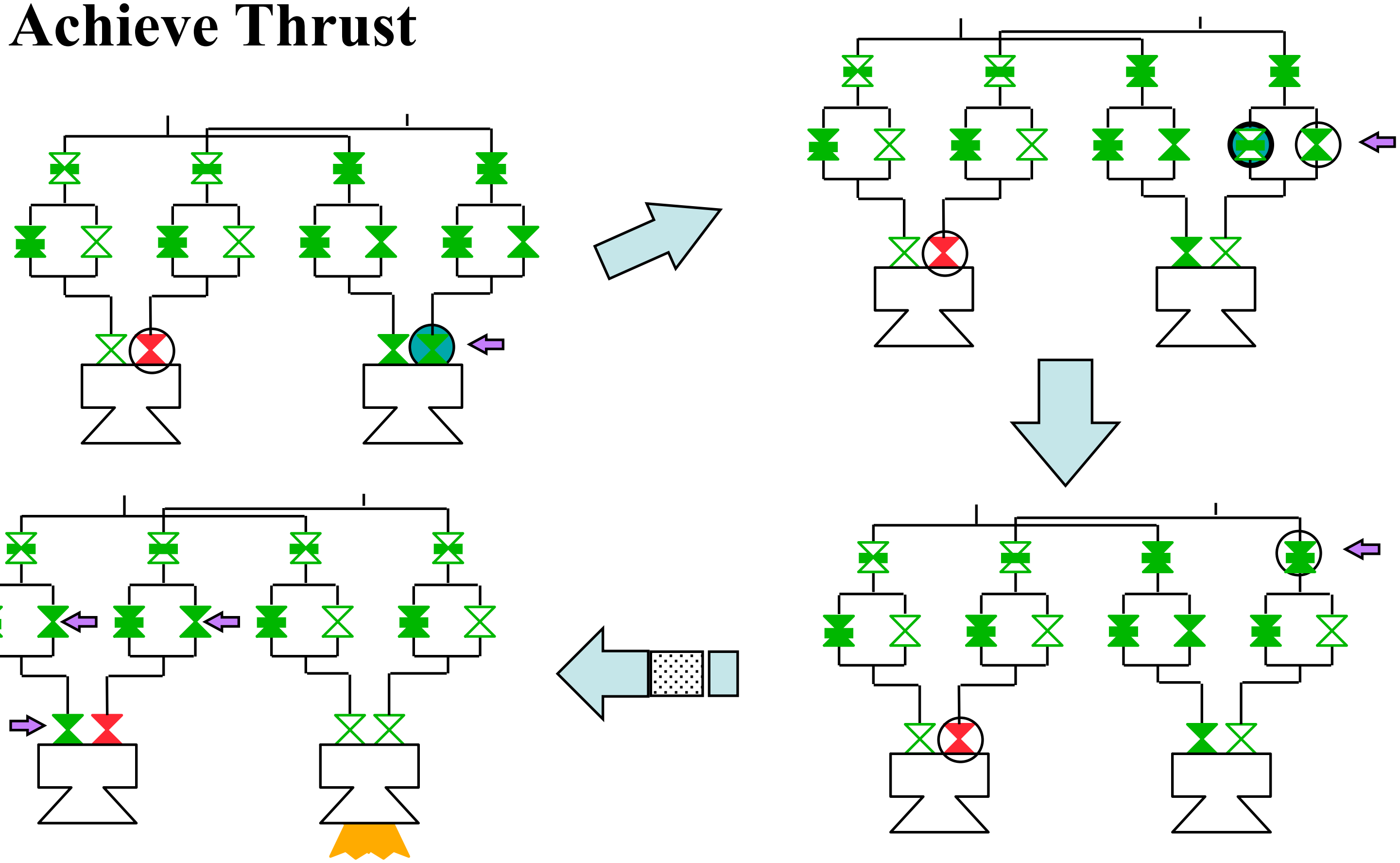


Implementation: Optimal CSPs and Conflict-directed A*.

Reconfiguring Modes using Conflict-directed A*

arg max $R_t(Y)$
s.t. $\Psi(X,Y)$ entails $G(X,Y)$
s.t. $\Psi(X,Y)$ is consistent
Y are reachable modes

Goal: Achieve Thrust

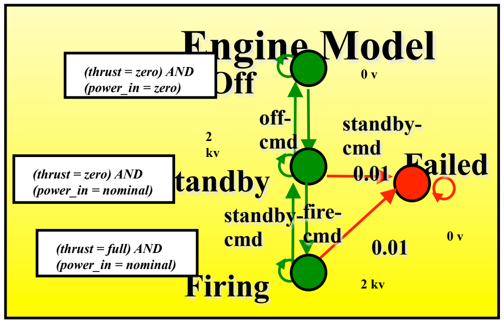


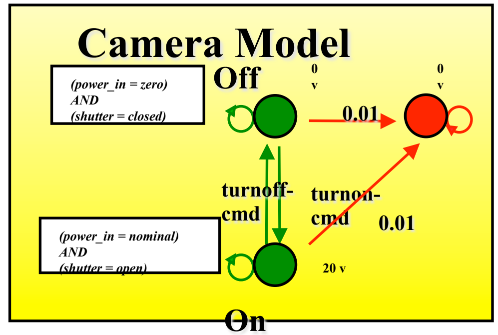
A *conflict* is a partial assignment to mode variables that prevents goal achievement (entails the negation of the goal).

Mode Estimation

- Estimates and kernels.
- By divide and conquer (GDE).
- Likely estimates and Conflict-directed A*.
- Mode reconfiguration.
- Estimating probabilistic (hybrid) constraint automata.

Tracking Mode Changes Over Time



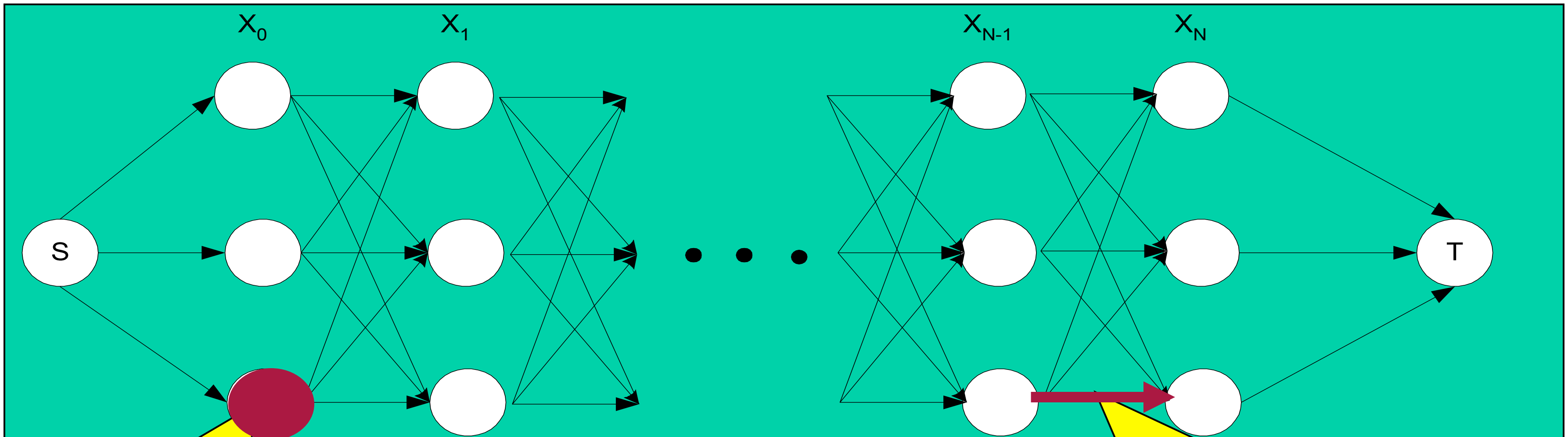


- PCCA encode an HMM compactly through concurrency and constraints.
- Mode estimation abstracts state to modes.

Assumes:

- Transitions only permitted on modes.
- Transitions are conditionally independent.
- For each time t,
  all consistent assignments are equally likely.

Mode Estimation as Belief State Update for Concurrent PCA



- Assigns a value to each variable (e.g.,3,000 vars).
- Consistent with all state constraints (e.g., 12,000).

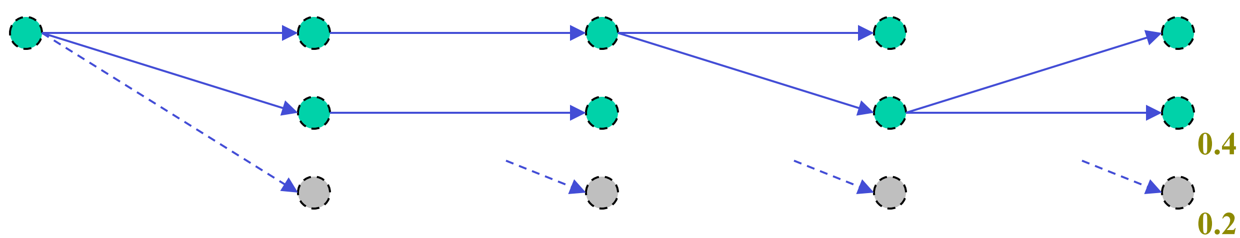
- A set of concurrent transitions, one per automata (e.g., 80).
- Previous & Next states consistent with source & target of transitions

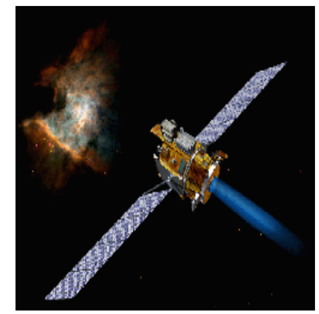
1. Infer most likely mode trajectories.
2. Infer distribution on likely mode assignments.

Approximating The Belief State

Best-first Trajectory Enumeration (BFTE):

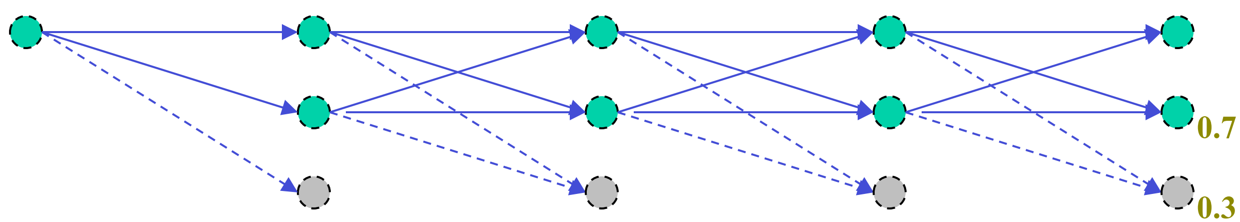
[Williams and Nayak, AAAI-96][Kurien and Nayak, AAAI-00][Williams et al., IEEE '03]

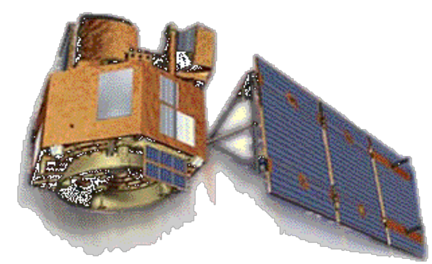




Deep Space One

- Best-first State Enumeration (BFSE):
  [Martin, Williams and Ingham, AAAI-05]



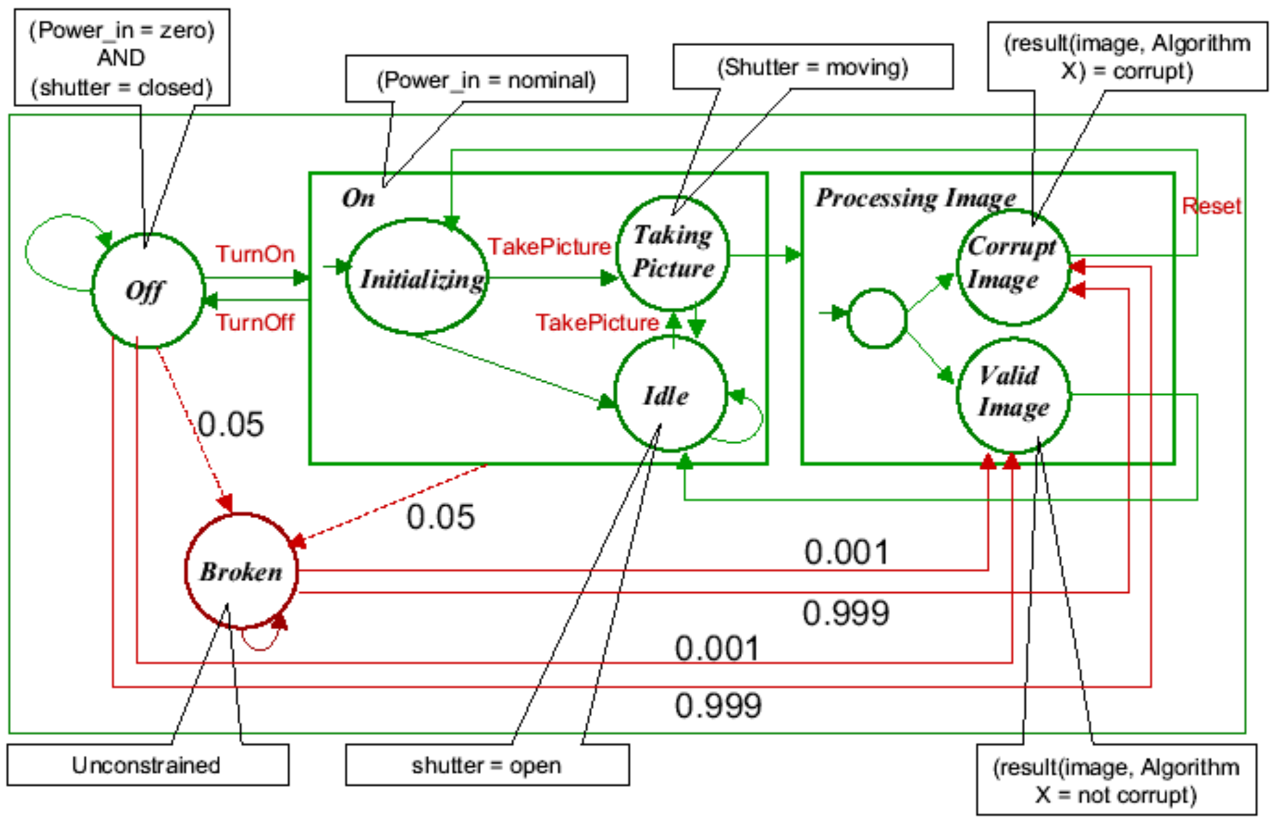


Earth Observing One

  – Improves accuracy through compact encoding.
  – Accuracy improves runtime!

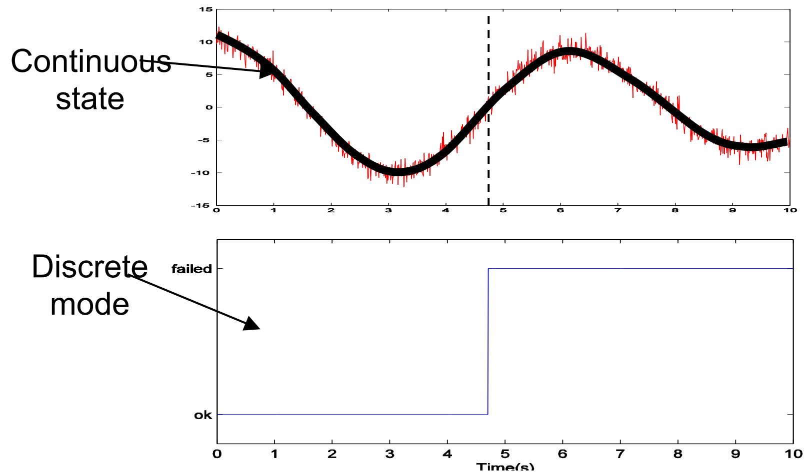
Monitoring Complex Hardware / Software Systems through Hierarchical Probabilistic Constraint Automata

Example:
Rover Image Acquisition



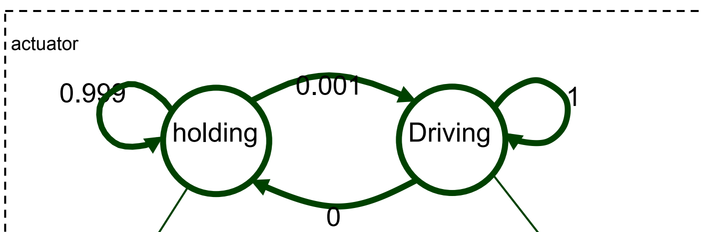
Estimating Hybrid States from Noisy Observations

1. **Free**
2. **Driving**
3. **Holding**
4. **Backdriven**



Hybrid probabilistic constraint automata

– Stochastic transitions between discrete modes



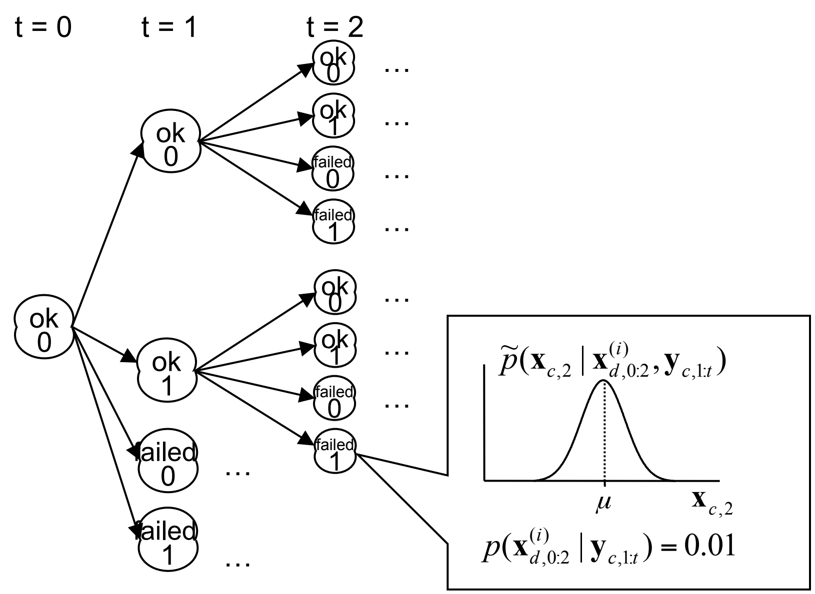
– Different continuous dynamics for each mode

$$\mathbf{x}_{t+1} = f_{\text{nominal}}(\mathbf{x}_t, \mathbf{u}_t, t) + \upsilon_{\text{nominal}}(t)$$
$$\mathbf{y}_{t+1} = g_{\text{nominal}}(\mathbf{x}_{t+1}, \mathbf{u}_t) + \omega_{\text{nominal}}(t)$$

$$\mathbf{x}_{t+1} = f_{\text{failed}}(\mathbf{x}_t, \mathbf{u}_t, t) + \upsilon_{\text{failed}}(t)$$
$$\mathbf{y}_{t+1} = g_{\text{failed}}(\mathbf{x}_{t+1}, \mathbf{u}_t) + \omega_{\text{failed}}(t)$$

[Blackmore, Funiak, Williams AAAI 05]

Kalman Filters Track Subset of Trajectories



Outline

- Robust, Goal-directed Execution
- Plan Dispatching
- Diagnosis and Mode Estimation
- Plan Generation

Planning

Based on slides from David Smith, NASA Ames.

Find:

program of actions that achieves the objective.

Planning

Based on slides from David Smith, NASA Ames.

Find:

program of actions that achieves the objective.

partially-ordered set of actions.

typically unconditional.

no loops.

Goals.

Paradigms

Classical planning,

(STRIPS, operator-based, first-principles)

"generative."

Hierarchical Task Network planning,

"practical" planning.

MDP & POMDP planning,

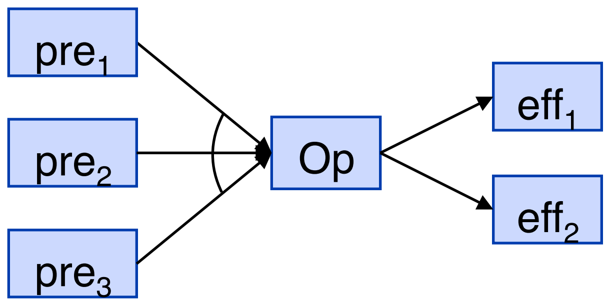
planning under uncertainty.

Classical Problem Statement

Propositions: P_i

Initial Conditions: P_1 P_2 P_3 P_4

Operators:



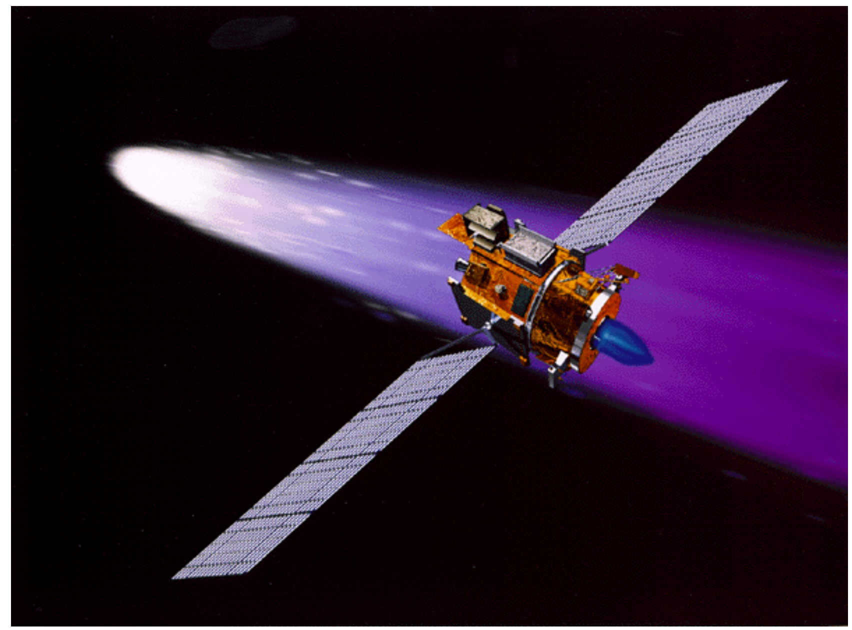
Goals: $Goal_1$ $Goal_2$ $Goal_3$

Simple Spacecraft Problem

Observation-1
target
instruments
Observation-2
Observation-3
Observation-4
...

pointing

calibrated

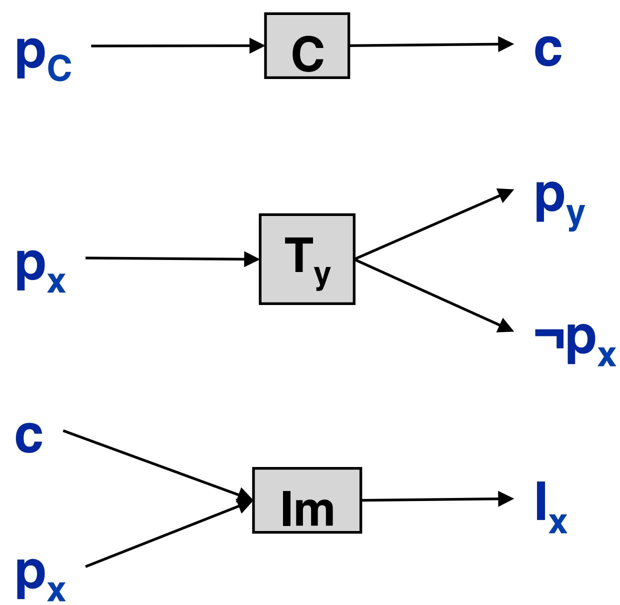


Propositions: Target Pointed To, Camera Calibrated?, Has Image?
Operators: Calibrate, Turn to Y, and Take Image.

Example

Init | **Actions** | **Goal**

p_C



I_A

Propositions: Target Pointed To, Camera Calibrated?, Has Image?

Operators: Calibrate, Turn to Y, and Take Image.

Planning Domain Description Language (PDDL)

```
(:action TakeImage :parameters (?target, ?instr)
     :precondition (and (Status ?instr Calibrated)
                        (Pointing ?target))
     :effect        (Image ?target))

(:action Calibrate :parameters (?instrument)
     :precondition (and (Status ?instr On)
                        (Calibration-Target ?target),
                        (Pointing ?target)
     :effect        (and (not (Status ?inst On))
                        (Status ?instr Calibrated)))

(:action Turn :parameters (?target)
     :precondition (and (Pointing ?direction)
                        ?direction ≠ ?target)
     :effect        (and (not (Pointing ?direction)
                        (Pointing ?target)))
```

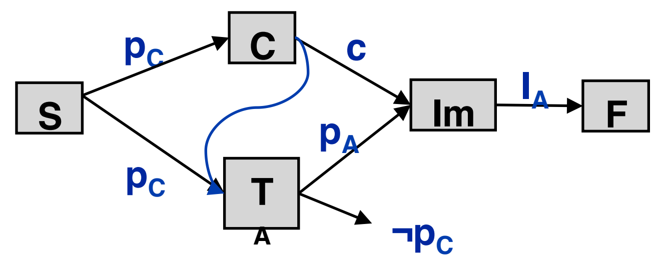
By convention, parameters start with "?", as in ?var.

Based on slides by Dave Smith, NASA Ames

Planning Paradigms

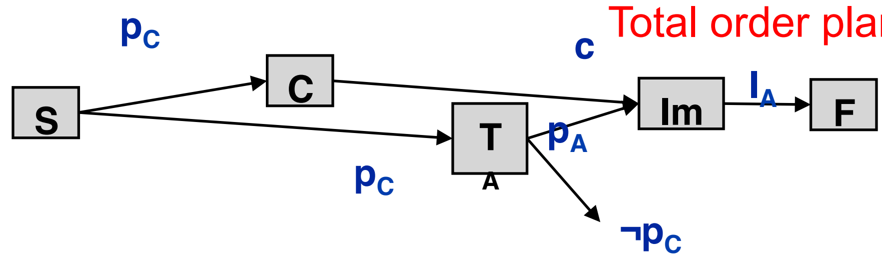
- From Goals
  - Goal Regression
  - (SNLP, UCPOP, Burton, Europa, Aspen, …)

Partial order plan

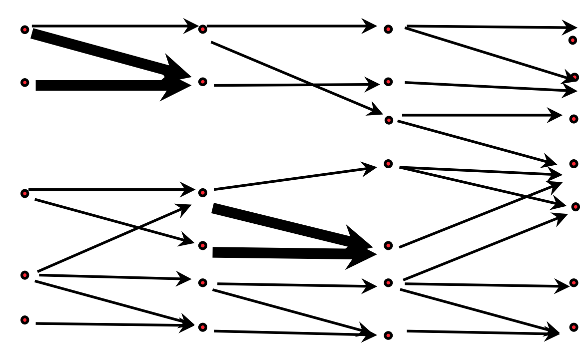


- From Initial State.
  - Heuristic Forward Search
  - (FF, HSP, Colin …)

Total order plan



- By Solving Constraints.
  - Plan Graphs
  - (SatPlan, Blackbox, Kongming …)



Based on slides by Dave Smith, NASA Ames

Continuously replanning as a human helps and hinders.
Planner: heuristic forward search.

Based on slides by Dave Smith, NASA Ames

Assumptions of Classic Planning

TakeImage (?target, ?instr):
    Pre: Status(?instr, Calibrated),
        Pointing(?target)
    Eff: Image(?target)

Calibrate (?instrument):
    Pre: Status(?instr, On),
        Calibration-Target(?target),
        Pointing(?target)
    Eff: ¬Status(?inst, On),
        Status(?instr, Calibrated)

Turn (?target):
    Pre: Pointing(?direction),
        ?direction ≠ ?target
    Eff: ¬Pointing(?direction),
        Pointing(?target)

- Atomic time,
- Agent is omniscient (no sensing necessary),
- Agent is sole cause of change,
- Actions have deterministic effects, and
- No indirect effects.

The Simple Spacecraft Revisited: Complications

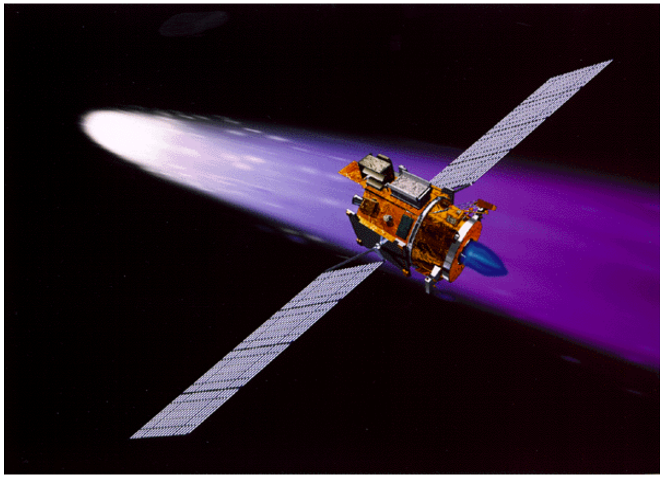
Observation-1
- priority
- time window
- target
- instruments
- duration

Observation-2

Observation-3

Observation-4

…



Objective:
maximize science return.

The Simple Spacecraft Revisited: Complications

linked

Observation-1
- priority
- time window
- target
- instruments
- duration

Observation-2

Observation-3

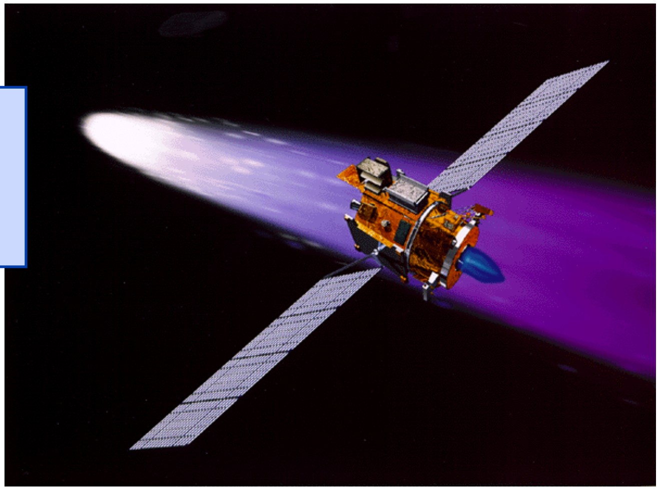
Observation-4

…

angle between targets
⇒ turn duration

calibration
- target1
- target2
- …

consumables:
- fuel
- power
- data storage
- cryogen



Objective:
maximize science return

Based on slides by Dave Smith, NASA Ames

More Expressive Planners Include

Time

Resources

Utility

Uncertainty

Hidden State

Indirect Control

Reasoning methods:

STNs or CSPs,

LPs or CSPs,

MDPs or MILPs,

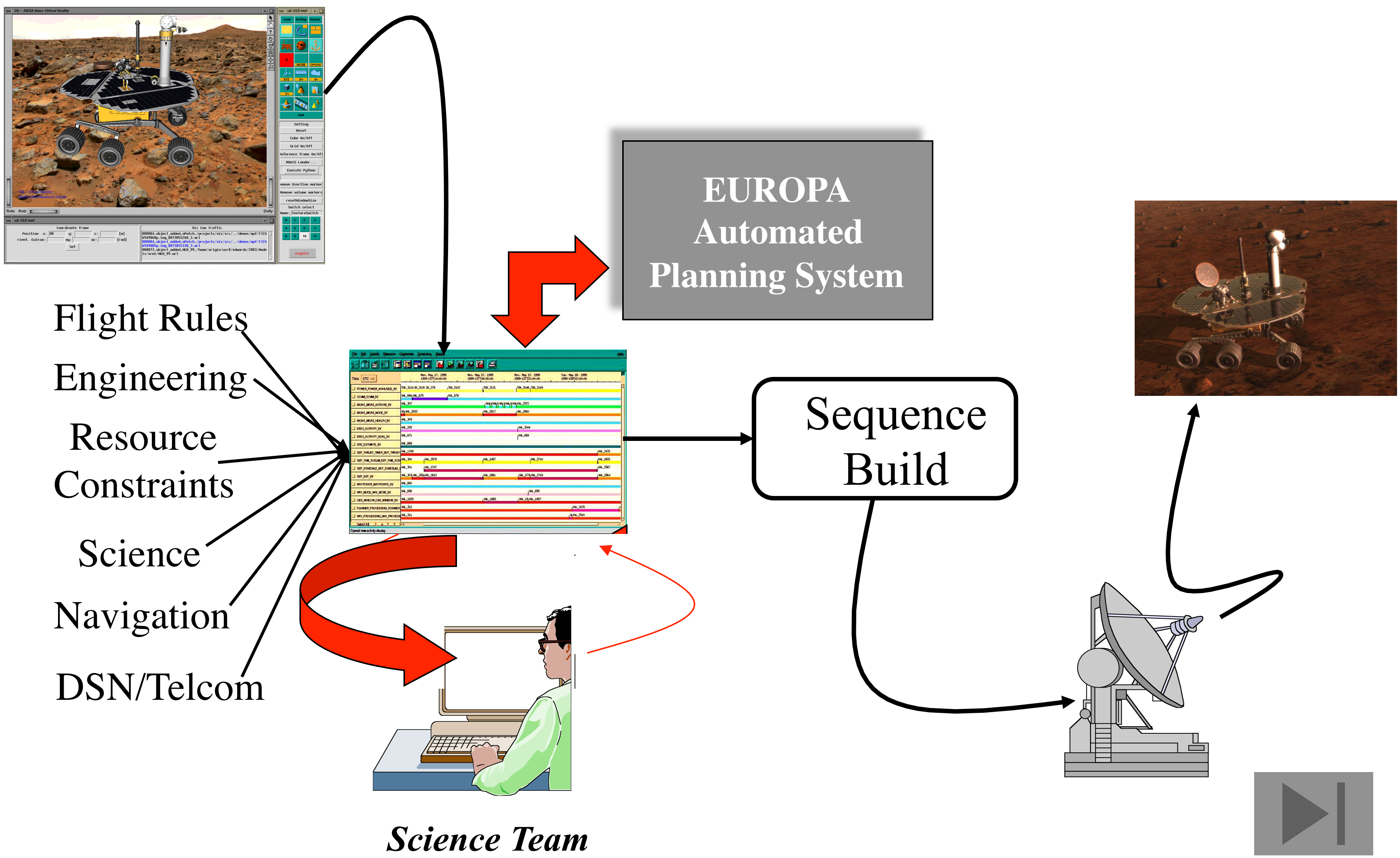
HMMs or BNs,

HMMs or OCSPs,

LPs or RPs.

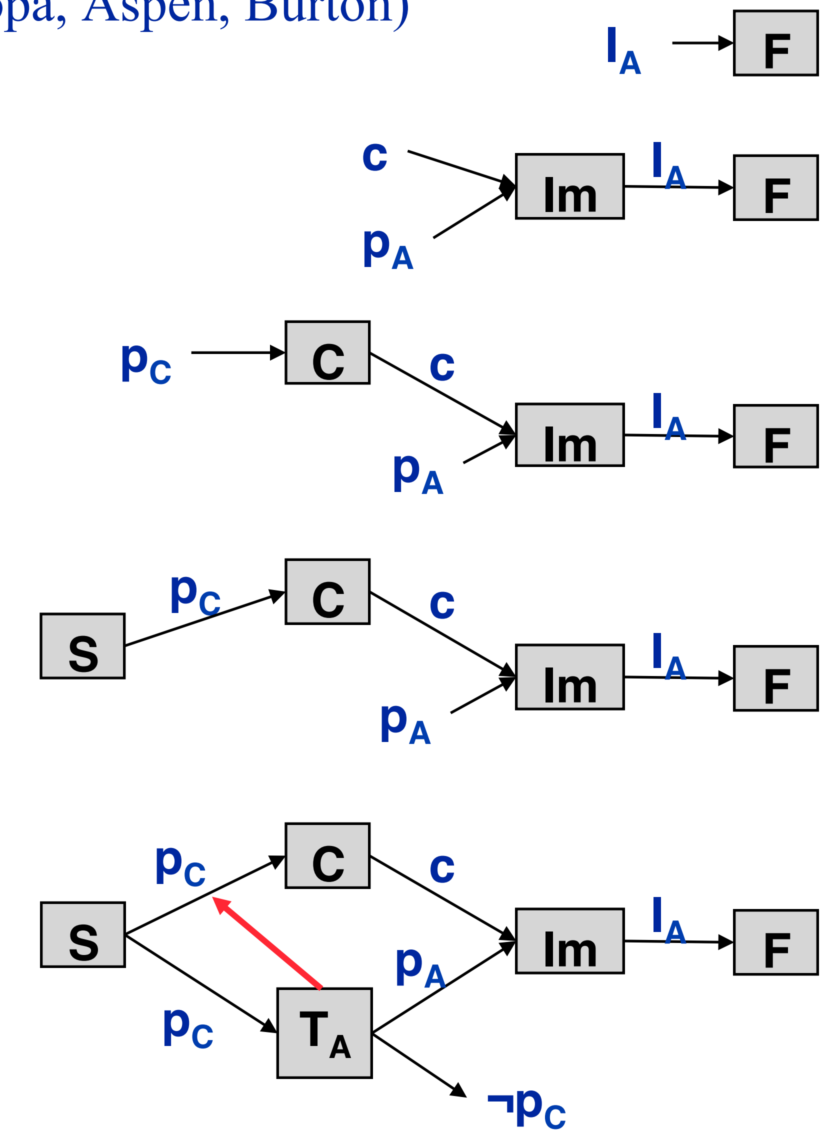
MAPGEN: Automated Science Planning for MER

NASA Ames

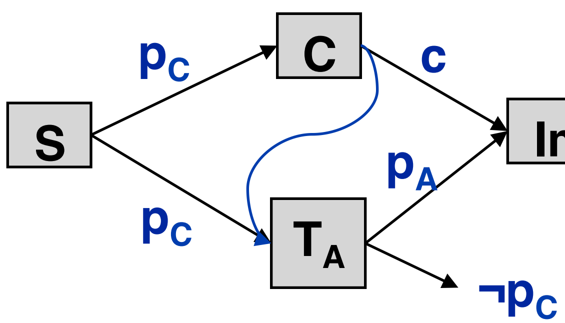


Planning Back from Goals: Partial Order Causal Link Planning

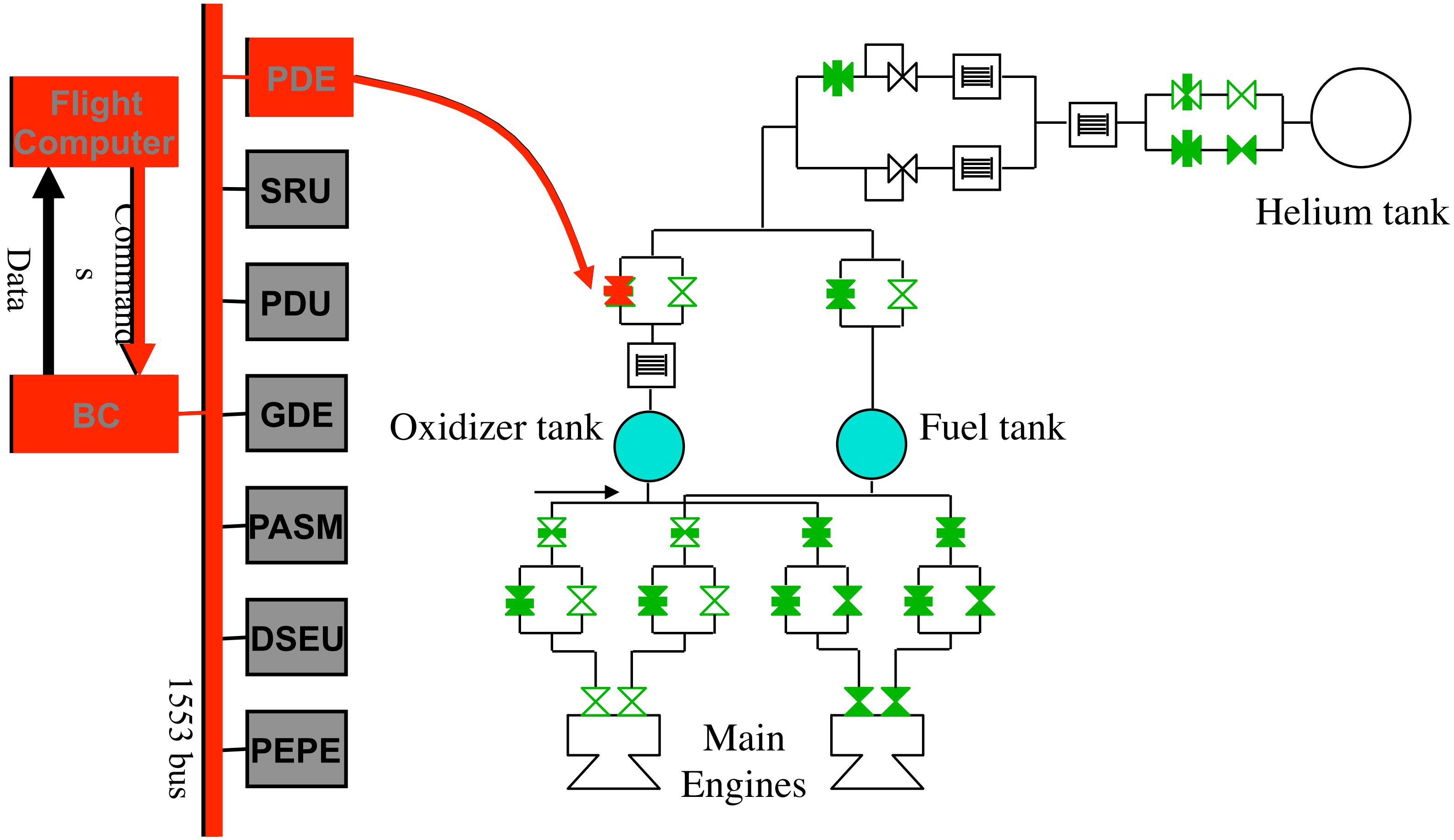
(SNLP, UCPOP, Europa, Aspen, Burton)



1. Select an open condition;
2. Choose an op that can achieve it:
   - Link to an existing instance or
   - Add a new instance;
3. Resolve threats.



Burton: Reactive Planning with Indirect Effects



When causal interactions are acyclic,
and actions are reversible,
The first action in the plan can be generated in ~ constant time.

[Williams & Nayak, IJCAI 97]

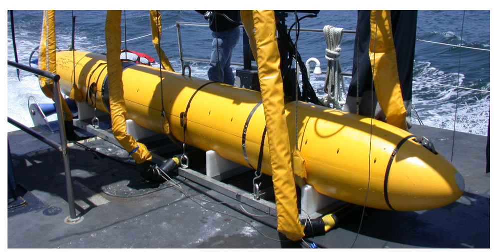
Sulu: Goal-directed Control

00:00 Go to x_1,y_1
00:20 Go to x_2,y_2
00:40 Go to x_3,y_3
…
04:10 Go to x_n,y_n

Command script

Commands

Plant



[Leaute & Williams, AAAI 05]

Sulu: Goal-directed Control

*"Explore **mapping region** for at least **100m**, then explore **bloom region** for at least **50m**, then return to **pickup region**. **Avoid obstacles** at all times."*

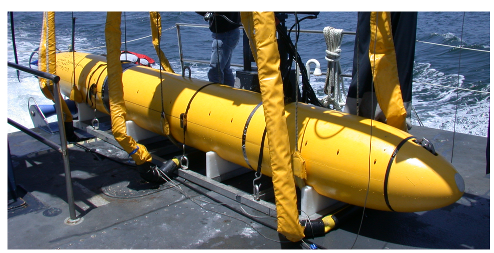
Qualitative State Plan

Model-based Executive

Observations

Commands

Plant

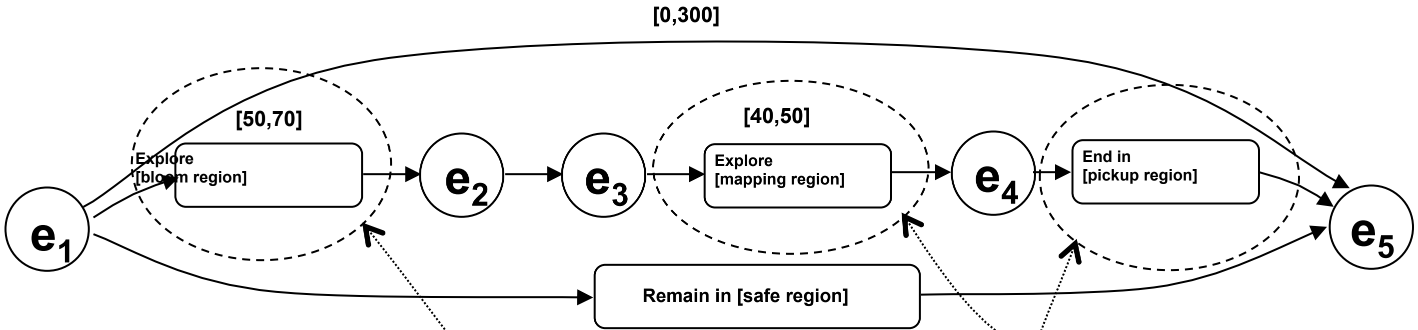


Optimal

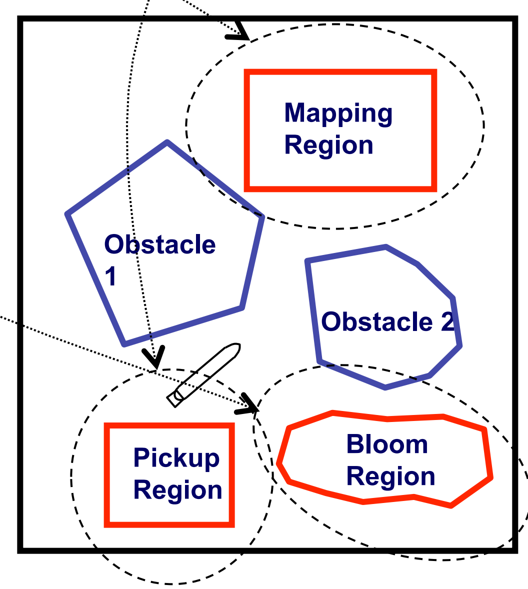
[Leaute & Williams, AAAI 05]

Sulu: Goal-directed Control

A qualitative state plan is a plan of activities that specifies desired states rather than executable actions and provides flexibility in state and time.



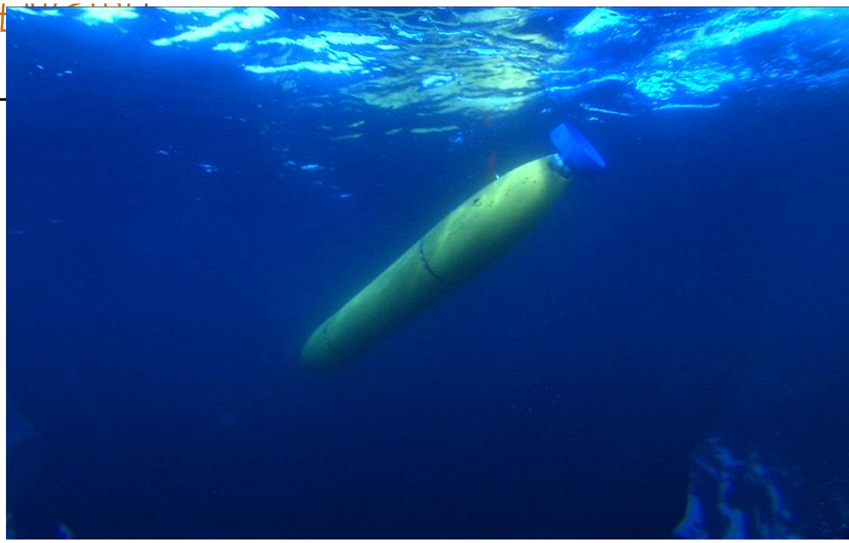
"Explore bloom region for between 50 and 70 seconds. Afterwards, explore mapping region for between 40m and 50m. End in the pickup region. Avoid obstacles at all times. Complete the mission within 300m."

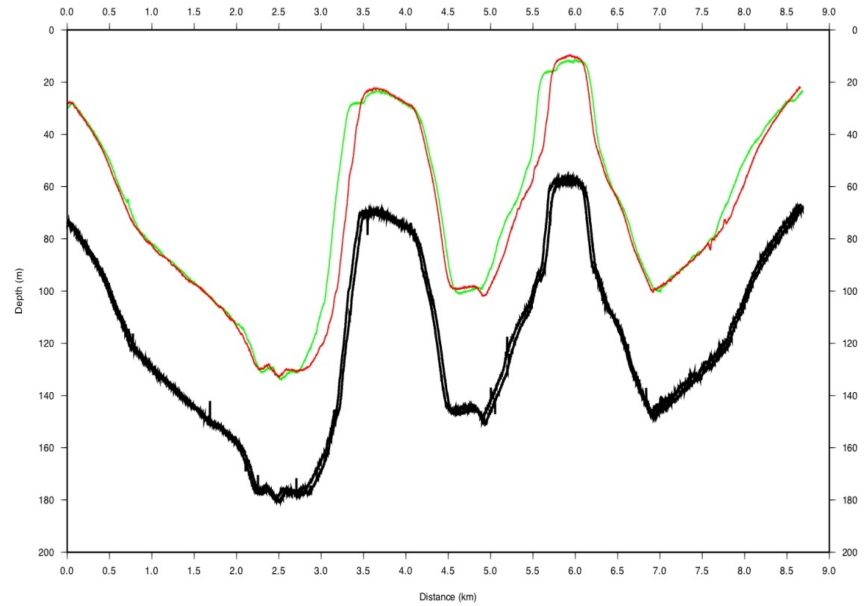


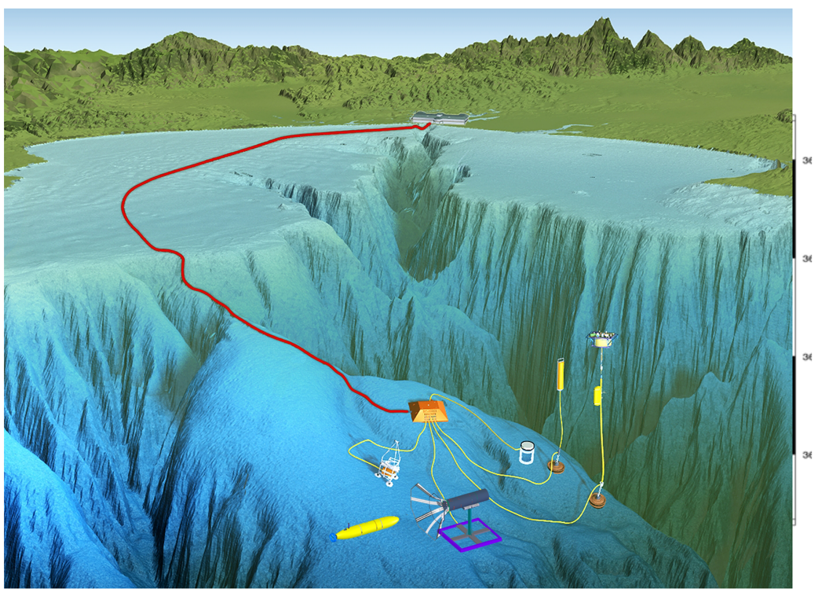
Approach: Frame as Model-Predictive Control using Mixed Logic or Integer / Linear Programming.

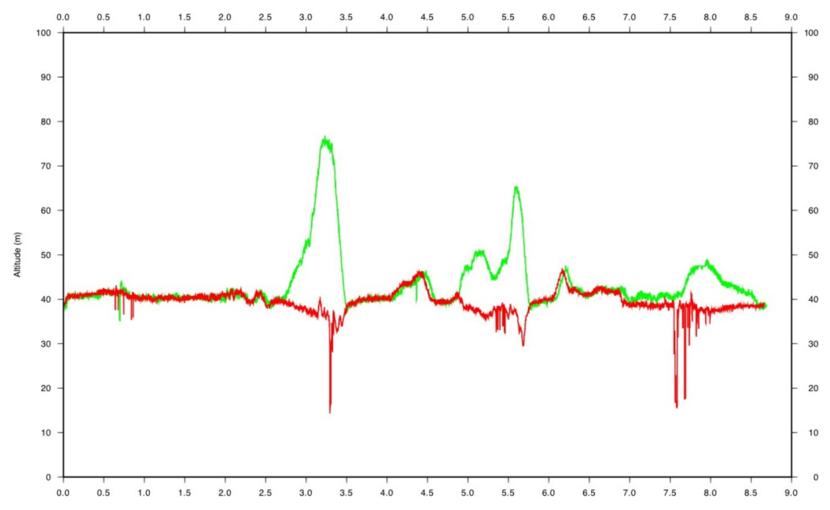
Leaute & Williams, AAAI 05

Sulu Depth Navigation for Bathymetric Mapping – Jan. 23rd, 08



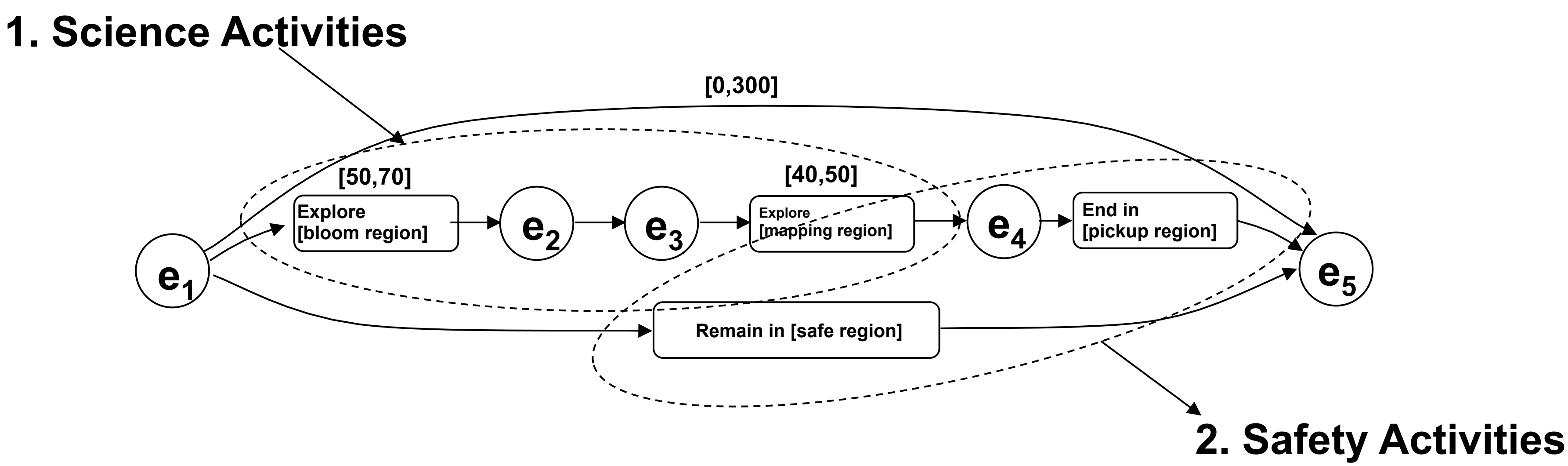






Problem: Managing Risk within Mission-Guidelines

Adding Risk Sensitivity



Chance constraints:

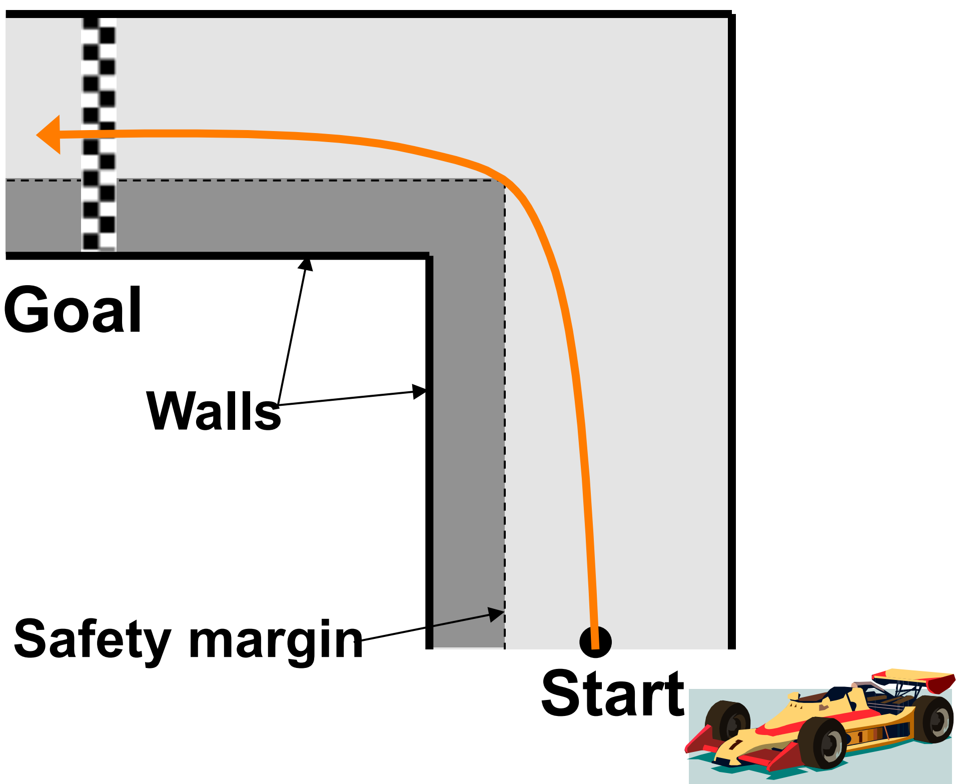
1. *p(* **Remain in [bloom region]** ***fails*** **OR Remain in [mapping region]** ***fails*** **) < 10%.**

2. *p(* **End in [goal region]** ***fails*** **OR Remain in [safe region]** ***fails*** **) < 1%.**

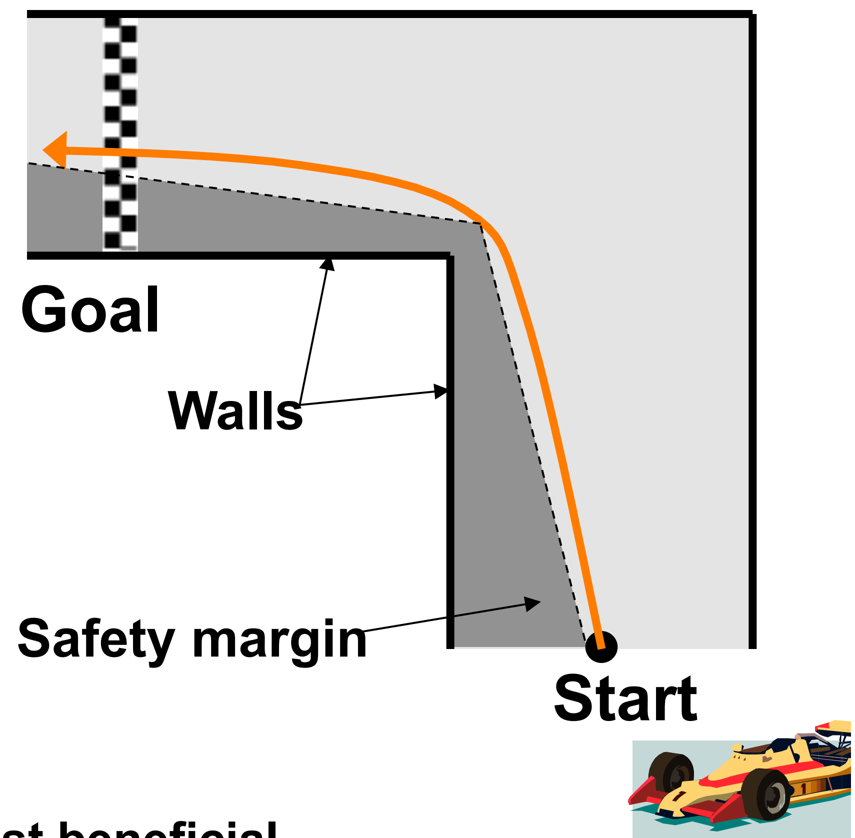
Instance of **Chance-constrained Model-based Programming.**

P Sulu creates safety margin that satisfies risk bounds and maximizes expected utility

(a) Uniform width safety margin



(b) Uneven width safety margin



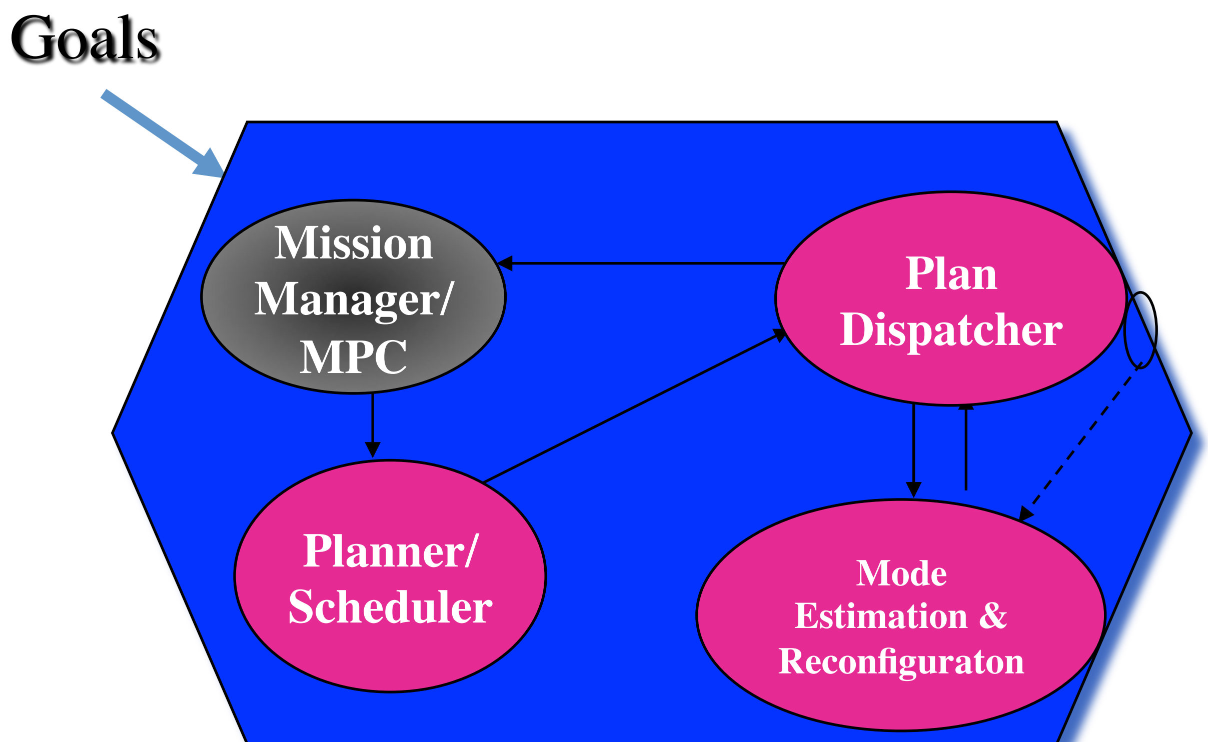
(b) results in better path → takes risk when most beneficial

Problem: How do we find the best safety margin?

[Ono & Williams, AAAI 08]

Model-based Executives

1. Commanded through time evolved goals.
2. Reasons from commonsense models.
3. Closes loop on goals.
4. Model-based programs specify goals and models.



Based on slides by Dave Smith, NASA Ames

For More: Go to MIT Open Course Ware:

- 16.410 Principles of Autonomy and Decision Making
- 16.412 Cognitive Robotics

QUESTIONS?