Model-based Execution

Outline

• Introduction & Overview
• Model-based Programming
• Execution of Model-based Programs
  – Scenario revisited
  – Control Sequencer
  – Deductive Controller (Mode Estimation & Mode Reconfiguration)
  – Model-based Execution Semantics
• Fundamentals of Model-based Reasoning
• Modeling via State Analysis
• Advanced Methods
• Conclusion
**Model-based Executive**

Model-based Executive

- **Model-based Program**
  - Control Program
  - System Model
- **Model-based Executive**
  - Control Sequencer
  - Deductive Controller
  - State estimates
  - Configuration goals

**System Under Control**

**Commands**

**Observations**

**Execution of HCA:**

1. initialize HCA by marking all start locations
2. check maintenance constraints
3. assert states from currently marked locations
4. obtain state update
5. take enabled transitions
6. mark new set of locations
7. return to step 2

**LEGEND:**

- EAS (EngineA = Standby)
- EAF (EngineA = Failed)
- EAR (EngineA = Firing)
- EBS (EngineB = Standby)
- EBF (EngineB = Failed)
- EBR (EngineB = Firing)
- CO (Camera = Off)
Executing HCA

Nominal (i.e. fault-free) orbital insertion scenario

Executing HCA - Step 1

• initialize HCA by marking all start locations
Executing HCA - Step 1

- initialize HCA by marking all start locations
- check maintenance constraints

LEGEND:
- EAS (EngineA = Standby)
- EAF (EngineA = Failed)
- EAR (EngineA = Firing)
- EBS (EngineB = Standby)
- EBF (EngineB = Failed)
- EBR (EngineB = Firing)
- CO (Camera = Off)

MAINTAIN (EAR OR EBR)

MAINTAIN (EAF)

MAINTAIN (EAS AND CO)

MAINTAIN (EAF AND EBS AND CO)
Executing HCA - Step 1

- initialize HCA by marking all start locations
- check maintenance constraints
- assert states from currently marked locations
- obtain state update

• initialize HCA by marking all start locations
• check maintenance constraints
• assert states from currently marked locations
• obtain state update
• take enabled transitions:
  • location's state assignment achieved
  • transition and maintenance conditions currently hold true
Executing HCA - Step 1

- initialize HCA by marking all start locations
- check maintenance constraints
- assert states from currently marked locations
- obtain state update
- take enabled transitions:
  - location’s state assignment achieved
  - transition and maintenance conditions currently hold true
- mark new set of locations

Executing HCA - Step 2

- (EngineA = Standby) & (EngineB = Standby) achieved in this step
Executing HCA - Step 2

- (EngineA = Standby) & (EngineB = Standby) achieved in this step
- two execution threads terminated & two transitions enabled

Executing HCA - Step 3

- (EngineA = Firing) asserted in this step, but not yet achieved
Executing HCA - Step 4

- (EngineA = Firing) achieved in this step
- maintenance condition violated, HCA block exited

Model-based Executive

Model-based Program
- Control Program
- System Model

Model-based Executive
- Control Sequencer
- Deductive Controller
- Stage estimates
- Configuration goals

System Under Control
- Observations
- Commands
Model-based Executive

Model-based Executive

Control Sequencer

Mode Estimation

Mode Reconfig

Deductive Controller

System Under Control

Model-based Program

Control Program

System Model

The System Under Control

Battery

Power Switch

Camera

Fuel

Oxidizer

Valve A1

Valve A2

Valve B1

Valve B2

Engine A

Engine B
Executing HCA - Step 1

MAINTAIN (EAR OR EBR)

EAS
(EngineA = Standby)

EAF
(EngineA = Failed)

EAR
(EngineA = Firing)

EBS
(EngineB = Standby)

EBF
(EngineB = Failed)

EBR
(EngineB = Firing)

CO
(Camera = Off)

LEGEND:

EAS
(EngineA = Standby)

EAF
(EngineA = Failed)

EAR
(EngineA = Firing)

EBS
(EngineB = Standby)

EBF
(EngineB = Failed)

EBR
(EngineB = Firing)

CO
(Camera = Off)

MAINTAIN (EAF)

EAS AND CO

EAR

MAINTAIN (EAF)

EAS AND CO

EAR

MAINTAIN (EAF)

EAS AND CO

EAR

Deductive Controller estimates state and issues commands to achieve goals

Camera

Goal: Off

(power_in = zero) &
(shutter = closed)

Off

Resettable

(power_in = nominal) &
(shutter = open)

On

Obs: Shutter is closed
Deductive Controller estimates state and issues commands to achieve goals

**Engine A**

**Goal:** Standby

- **Off**
  - (eng_cmd = off)
  - (power_in = nominal) & (eng_cmd = standby)
  - (flow_in1 = nominal) & (flow_in2 = nominal)
  - (thrust = zero)
  - (thrust = full) & (flow_in1 = nominal) & (flow_in2 = nominal)

**Standby**

- (eng_cmd = standby)

**Firing**

- (eng_cmd = fire)

---

**Power Switch**

**Goal:** Closed

- **Open**
  - (sw_cmd = open)
  - (flow_in1 = nominal)
  - (flow_in2 = nominal)

- **Stuck open**
  - (sw_cmd = open)
  - (flow_in1 = nominal)
  - (flow_in2 = nominal)

- **Closed**
  - (sw_cmd = close)
  - (flow_in1 = nominal)
  - (flow_in2 = nominal)

- **Stuck closed**
  - (sw_cmd = close)
  - (flow_in1 = nominal)
  - (flow_in2 = nominal)

**Obs:** power_out = nominal

**Goal:** Open

- (sw_cmd = open)

---

**MAINTAIN (EAR OR EBR)**

**EBS**

**CO**

**LEGEND:**

- **EAS** (EngineA = Standby)
- **EAF** (EngineA = Failed)
- **EAR** (EngineA = Firing)
- **EBS** (EngineB = Standby)
- **EBF** (EngineB = Failed)
- **EBR** (EngineB = Firing)
- **CO** (Camera = Off)

**Control Program**

- Deductive Mode Estimator & Reactive Planner

**Configuration**

- Goals
- Observations

**Flight System Control**

- RT Control Layer
- Onboard Sequencer
- State estimates
- System Model
Deductive Controller estimates state and issues commands to achieve goals

Goal: Standby

Engine A

Off

(eng_cmd = off)

0.01

Failed

Standby

(stall = zero) & (eng_cmd = standby)

0.01

Firing

(stall = full) & (eng_cmd = standby)

0.01

Achievement of Standby on Engine B proceeds similarly, in parallel...

Executing HCA - Step 2

MAINTAIN (EAR OR EBR)

Legends:

EAS (EngineA = Standby)
EAF (EngineA = Failed)
EAR (EngineA = Firing)
EBS (EngineB = Standby)
EBF (EngineB = Failed)
EBR (EngineB = Firing)
CO (Camera = Off)
Executing HCA - Step 3

Legend:
- EAS (Engine A = Standby)
- EAF (Engine A = Failed)
- EAR (Engine A = Firing)
- EBS (Engine B = Standby)
- EBF (Engine B = Failed)
- EBR (Engine B = Firing)
- CO (Camera = Off)

Maintain (EAF)

Maintain (EAR OR EBR)

Control Program

Deductive Controller

Engine A = Firing

Goal:
Firing

Standby

Failed

Off

Thrust = full

Thrust = zero

Power_in = nominal

Power_in = nominal

Flow_in1 = nominal

Flow_in2 = nominal

Eng_cmd = off

Eng_cmd = standby

Eng_cmd = fire

Eng_cmd = standby
Deductive Controller estimates state and issues commands to achieve goals

Engine A

Goal: Firing

Off

Failed

MAINTAIN (EAR OR EBR)

LEGEND:

EAS (EngineA = Standby)
EAF (EngineA = Failed)
EAR (EngineA = Firing)
EBS (EngineB = Standby)
EBF (EngineB = Failed)
EBR (EngineB = Firing)
CO (Camera = Off)

Achievement of Open on Valve A2 proceeds similarly, in parallel...

Valve A1

Goal: Open

Obs: outflow = nominal

Open

Stuck open

Closed

Stuck closed

Valve A1 = Open; Valve A2 = Open

Executing HCA - Step 3
Executing HCA - Step 4

Deductive Controller estimates state and issues commands to achieve goals
Executing HCA - Step 4

- MAINTAIN (EAR OR EBR)
- LEGEND:
  - EAS (EngineA = Standby)
  - EAF (EngineA = Failed)
  - EAR (EngineA = Firing)
  - EBS (EngineB = Standby)
  - EBF (EngineB = Failed)
  - EBR (EngineB = Firing)
  - CO (Camera = Off)

- Engine A = Firing

Executing HCA - Step 5

- (EngineA = Firing) achieved in this step
- maintenance condition violated, HCA block exited
**What About Off-nominal Execution?**

Legend:
- **EAS** (Engine A = Standby)
- **EAF** (Engine A = Failed)
- **EAR** (Engine A = Firing)
- **EBS** (Engine B = Standby)
- **EBF** (Engine B = Failed)
- **EBR** (Engine B = Firing)
- **CO** (Camera = Off)

**Maintain (EAS OR EBR)**
- **EAS**
- **EBS**
- **CO**

**Maintain (EAF)**
- **EAS AND CO**

**Maintain (EAR OR EBR)**
- **EAR**

**Camera**
- Goal: Off
- Obs: Shutter is open
- (power_in = nominal) & (shutter = open)
- (cam_cmd = turnoff)
- (cam_cmd = turnon)

**Model-based executive provides in-the-loop robustness**

- **Control Sequencer**
- **Deductive Controller**
- **Observations System Under Control**
- **Commands System Model**
- **State estimates System Model**
- **Flight System Control**
- **Onboard Sequencer**
- **RT Control Layer**
- **Configuration**
- **Goals**
- **State estimates**
- **System Model**
- **Goal: Off**
- (power_in = zero) & (shutter = closed)
Executing HCA - Step 1

Legend:
- EAS (Engine A = Standby)
- EAF (Engine A = Failed)
- EAR (Engine A = Firing)
- EBS (Engine B = Standby)
- EBF (Engine B = Failed)
- EBR (Engine B = Firing)
- CO (Camera = Off)

MAINTAIN (EAF or EBR)
- EAS
- EAF
- EBS
- EBF
- EBR
- CO

MAINTAIN (EAF)
- (EAS AND CO)
- (EAF AND EBS AND CO)
- (EAF AND EBS AND EBR)

Control Sequencer
Deductive Controller

System Under Control
Commands

Observations

Camera = Resettable; Power Switch = Closed

Executing HCA - Step 2

Legend:
- EAS (Engine A = Standby)
- EAF (Engine A = Failed)
- EAR (Engine A = Firing)
- EBS (Engine B = Standby)
- EBF (Engine B = Failed)
- EBR (Engine B = Firing)
- CO (Camera = Off)

MAINTAIN (EAF or EBR)
- EAS
- EAF
- EBS
- EBF
- EBR
- CO

MAINTAIN (EAF)
- (EAS AND CO)
- (EAF AND EBS AND CO)
- (EAF AND EBS AND EBR)

Control Sequencer
Deductive Controller

System Under Control
Commands

Observations

Camera = Off; Engine A = Standby; Engine B = Standby

Camera = Off; Engine A = Standby; Engine B = Standby
Model-based executive provides in-the-loop robustness

**Camera**

Goal: Off

- (power_in = zero) & (shutter = closed)

Resettable:
- (power_in = nominal) & (shutter = open)

Observations:
- Shutter is open

Executing HCA - Step 2

**Deductive Controller**

- Control Sequencer

Configuration

System Under Control

System Model

Observations
Executing HCA - Step 3

Model-based executive provides in-the-loop robustness

Camera
Goal:
Off

Resettable

On

Obs: Shutter is closed

(power_in = zero) & (shutter = closed)

(power_in = nominal) & (shutter = open)
Semantics of Model-based Executives

- Now that we’ve informally described Model-based Execution and illustrated how it works…
- We can now proceed with a (brief) formal semantic specification of Model-based Execution
- This follows from the semantic representations of the Model-based Program that we presented earlier:

\[
CP = \langle L_{cp}, \lambda_{cp}, \tau_{cp}, g_{cp}, \Sigma_s \rangle
\]

\[
SM = \langle \Sigma, T, P_\Theta, P_T, P_O, R \rangle
\]
Semantics of Mode Estimation

• Given latest commands and observations, what is the most likely current state?
• Belief state update to estimate state for POMDPs:

\[ p^{(s_{1})}[s_{j}] = \sum_{k=1}^{n} p_{k}(s_{j} | s_{k}, \mu) \]
\[ p^{(s_{1})}[s_{j}] = p^{(s_{1})}[s_{j}] \frac{P_{O}(o^{(s_{1})} | s_{j})}{\sum_{k=1}^{n} p_{k}(s_{j} | s_{k}, \mu)} \]

Optimal policy \( \pi^{*} \):

Optimal policy \( \pi^{*} \):

Semantics of Mode Reconfiguration

• Given current belief state and configuration goal, what is the first control action from a policy that maximizes expected reward?

Optimal policy \( \pi^{*} \):

Optimal policy \( \pi^{*} \):

Solve Bellman equation to compute optimal policy for POMDPs:

\[ V^{*}(s) = \max_{\pi} E \left[ \sum_{t=1}^{\infty} \gamma^{t} r_{t} \right] \]
\[ \pi^{*}(s) = \arg \max_{\mu} \left[ R_{e}(s) + \gamma \sum_{s' \in S} P_{T}(s' | s, \mu) V^{*}(s') \right] \]
Executive Semantics

- Interleaving model of execution
  \[ \text{cycle} = \text{discrete event} + \text{continuous phase} \]

- Legal execution of MBP:

<table>
<thead>
<tr>
<th>Cycle start time</th>
<th>( t_0 )</th>
<th>( t_1 )</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plant state</td>
<td>( \hat{s}_0 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pgm location</td>
<td>( l_0 )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Such that:
1. initial conditions are valid
2. next state is legal
3. next program location is legal

\[ P_\theta(\hat{s}_0) > 0 \]
\[ l_0 = \lambda_{cp} \]

Executive Semantics

- Interleaving model of execution
  \[ \text{cycle} = \text{discrete event} + \text{continuous phase} \]

- Legal execution of MBP:

<table>
<thead>
<tr>
<th>Cycle start time</th>
<th>( t_0 )</th>
<th>( t_1 )</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plant state</td>
<td>( \hat{s}_0 ), ( \hat{s}_1 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pgm location</td>
<td>( l_0 )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Such that:
1. initial conditions are valid
2. next state is legal
3. next program location is legal

\[ g_0 = g_{cp}(l_0) \]
\[ \mu_0 = MR(SM, \hat{s}_0, g_0) \]
\[ \hat{s}_1 = ME(SM, \hat{s}_0, \mu_0, o_1) \]
Executive Semantics

- Interleaving model of execution
  \[ \text{cycle} = \text{discrete event} + \text{continuous phase} \]
- Legal execution of MBP:

<table>
<thead>
<tr>
<th>Cycle start time</th>
<th>( t_0 )</th>
<th>( t_1 )</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plant state</td>
<td>( \hat{s}_0 \rightarrow \hat{s}_1 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pgm location</td>
<td>( l_1 \rightarrow l_2 )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Such that:
1. initial conditions are valid
2. next state is legal
3. next program location is legal

\[
l_i = \tau_{cp}(l_0, \hat{s}_0)
\]
**Implementation Approximations**

Mode Estimation:
- Full belief state update is computationally infeasible
- Assume probability of a few most-likely states dominates probability of other possible states
- Track a limited set of most-likely states, from one cycle to the next

Mode Reconfiguration:
- Assume probability of nominal behavior dominates off-nominal
- Assume reward of being in goal state dominates reward of getting to goal state
- Perform MR in 2 steps:
  - **Goal Interpretation**: find the max-reward goal state, reachable via nominal transitions, that satisfies the configuration goal
  - **Reactive Planning**: returns series of control actions that achieve the goal state

**Deductive Controller Implementation**

- **Model-based Program**
  - Control Program
  - System Model

- **Model-based Executive**
  - Control Sequencer
    - Mode Estimation
      - State estimates
    - Mode Reconfig
      - Configuration goals
  - Deductive Controller
    - State estimates

- **System Under Control**
- **Commands**
- **Observations**
Model-based reasoning algorithm:
- Tracks a limited set of most-likely states
- Explores state space in best-first order

Formulate Optimal Constraint Satisfaction Problem (OCSP), to identify "k-best" extensions to current trajectories ("shortest path" from set of current possible states to next possible states)
- \( \text{OCSP} < x, f, C > \)
- \( \text{decision vars } x, \text{ such that } \text{dom}(x) = \text{reachable target modes} \)
- \( \text{objective function } f(x) = \text{prior probability of state } x, \text{i.e.:} \)
  \[
  \Pi_j P_{Tj}( x_j | s^{(i)}, \mu^{(i)} )
  \]
- \( \text{constraint } C(x), \text{ such that } x \land C_{Mx} \land o^{(i+1)} \text{ is consistent} \)
Mode Estimation

- Formulate Optimal Constraint Satisfaction Problem (OCSP), to identify “k-best” extensions to current trajectories ("shortest path" from set of current possible states to next possible states)
- Solve using OPSAT engine

Model-based reasoning algorithms to compute a series of commands that progress the system towards a least-cost state that achieves the configuration goal, one command at a time.
**Goal Interpreter**

**INPUT**
- Current State
  - Tank = full
  - Pressure = nominal
  - Driver = off
  - Valve = closed
  - Thruster = off
- Configuration Goal
  - Thrust = on

**OUTPUT**
- Goal State
  - Tank = full
  - Pressure = nominal
  - Driver = off
  - Valve = on
  - Thruster = on

Generate optimal goal state that achieves the **Configuration Goal**

OCSP < $x$, $f$, $C$>:
- decision vars $x$, such that $\text{dom}[x] = \text{reachable target modes}$
- objective function $f(x) = \text{cost of state } x$
- constraint $C(x)$, such that $x \land C_Mx$ is consistent and entails $g^{(i)}$

**Reactive Planner**

**INPUT**
- Current State
  - Tank = full
  - Pressure = nominal
  - Driver = off
  - Valve = closed
  - Thruster = off
- Goal State
  - Tank = full
  - Pressure = nominal
  - Driver = off
  - Valve = open
  - Thruster = on

**OUTPUT**
- Command
  - Turn driver on

Planner guarantees to:
- Ensure progress toward the goal state
- Only generate non-destructive actions
- Never propose actions that lead to dead-end plans
- Operate at reactive time scale

---

**Reconfiguration Order:**
1. Tank = full
2. Pressure = nominal
3. Valve = open
4. Thruster = on
5. Driver = off

<table>
<thead>
<tr>
<th>Current Goal</th>
<th>On</th>
<th>Off</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tank</td>
<td>On</td>
<td>On</td>
</tr>
<tr>
<td>Pressure</td>
<td>On</td>
<td>Off</td>
</tr>
<tr>
<td>Valve</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Thruster</td>
<td>Off</td>
<td>Off</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Current Goal</th>
<th>Open</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turn driver on</td>
<td>Open</td>
<td>idle</td>
<td>driver = on cmd = on</td>
</tr>
<tr>
<td></td>
<td>Closed</td>
<td>driver = on cmd = open</td>
<td>idle</td>
</tr>
</tbody>
</table>

Resettable

<table>
<thead>
<tr>
<th>Command</th>
<th>Current Goal</th>
<th>Open</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Resettable</td>
<td>idle</td>
<td>fail</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Current Goal</th>
<th>Open</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmd = off</td>
<td>Cmd = off</td>
<td>cmd = off</td>
<td>fail</td>
</tr>
<tr>
<td>cmd = on</td>
<td>Cmd = on</td>
<td>cmd = on</td>
<td>idle</td>
</tr>
</tbody>
</table>

Stuck

<table>
<thead>
<tr>
<th>Command</th>
<th>Current Goal</th>
<th>Open</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>fail</td>
<td>Stuck</td>
<td>fail</td>
<td>fail</td>
</tr>
</tbody>
</table>
Summary

- Model-based Executive is made up of 2 components:
  - Control Sequencer
  - Deductive Controller

- The Deductive Controller performs two functions:
  - Mode Estimation
  - Mode Reconfiguration

- Model-based Execution has formal semantics:
  - legal state evolutions of a factored POMDP
  - intent expressed in the form of a deterministic automaton

- The implemented Model-based Executive overcomes the computational complexity of this problem by leveraging a few key assumptions and proven model-based reasoning algorithms

Up Next…

- Fundamentals of Model-based Reasoning
  - A little historical context
  - A glimpse into the details behind the algorithms implemented in the Deductive Controller
  - A discussion of how we can improve the run-time performance of these algorithms by performing their most expensive reasoning steps off-line, at compile time