Fault Aware Systems: Model-based Programming and Diagnosis

Brian C. Williams
16.412J/6.834J
March 8th, 2004

Outline

- Fault Aware Systems Through Model-based Programming
- Diagnosis as Detective Work
- Model-based Diagnosis

Mars Polar Lander Failure

Leading Diagnosis:
- Legs deployed during descent.
- Noise spike on leg sensors latched by software monitors.
- Laser altimeter registers 50ft.
- Begins polling leg monitors to determine touch down.
- Latched noise spike read as touchdown.
- Engine shutdown at ~50ft.

Fault Aware Systems: Create embedded languages That reason and coordinate on the fly from models

Programmers are overwhelmed by the bookkeeping of reasoning about unlikely hidden states

Like Storyboards, Model-based Programs Specify The Evolution of Abstract States

Embedded programs evolve actions by interacting with plant sensors and actuators:
- Read sensors
- Set 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.

Programmer maps between state and sensors/actuators.

Descent Example

Turn camera off and engine on

Descent Example Diagram:

System Model

Titan Model-based Executive

Generates target goal states conditioned on state estimates

Tracks likely plant states

Tracks least cost goal states

Valve

Open

Closed

Open

Closed

0.01

inflow iff outflow
State-based Execution: The model-based program sets the state to thrusting, and the deductive controller . . . .

Deduces that valves on the backup engine will achieve thrust, and plans needed actions.

Deduces that thrust is off, and the engine is healthy.

Model-based Programs

Control program specifies
state trajectories:
- fires one of two engines
- sets both engines to 'standby'
- prior to firing engine, camera must be turned off to avoid plume contamination
- in case of primary engine failure, fire backup engine instead

Plant Model describes behavior of each component:
- Nominal and Off nominal
- qualitative constraints
- likelihoods and costs

Plant Model

Component modes...
- described by finite domain constraints on variables...
  deterministic and probabilistic transitions
  cost/reward

Modeling Complex Behaviors through Probabilistic Constraint Automata

- Complex, discrete behaviors
  modeled through concurrency, hierarchy and timed transitions.
- Anomalies and uncertainty
  modeled by probabilistic transitions
- Physical interactions
  modeled by discrete and continuous constraints

The Plant’s Behavior

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

The Plant's Behavior

Possible Behaviors Visualized by a Trellis Diagram

• 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

Plant Model

component modes...
- described by finite domain constraints on variables...
  deterministic and probabilistic transitions
  cost/reward

Modeling Complex Behaviors through Probabilistic Constraint Automata

- Complex, discrete behaviors
  modeled through concurrency, hierarchy and timed transitions.
- Anomalies and uncertainty
  modeled by probabilistic transitions
- Physical interactions
  modeled by discrete and continuous constraints

The Plant’s Behavior

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

The Plant’s Behavior

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

Plant Model

component modes...
- described by finite domain constraints on variables...
  deterministic and probabilistic transitions
  cost/reward

Modeling Complex Behaviors through Probabilistic Constraint Automata

- Complex, discrete behaviors
  modeled through concurrency, hierarchy and timed transitions.
- Anomalies and uncertainty
  modeled by probabilistic transitions
- Physical interactions
  modeled by discrete and continuous constraints

The Plant’s Behavior

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

Plant Model

component modes...
- described by finite domain constraints on variables...
  deterministic and probabilistic transitions
  cost/reward

Modeling Complex Behaviors through Probabilistic Constraint Automata

- Complex, discrete behaviors
  modeled through concurrency, hierarchy and timed transitions.
- Anomalies and uncertainty
  modeled by probabilistic transitions
- Physical interactions
  modeled by discrete and continuous constraints

The Plant’s Behavior

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
Modeling Complex Behaviors through Probabilistic Constraint Automata

- Complex, discrete behaviors
  - modeled through concurrency, hierarchy and timed transitions.
- Anomalies and uncertainty
  - modeled by probabilistic transitions
- Physical interactions
  - modeled by discrete and continuous constraints

The Plant’s Behavior

- 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

Outline

- Fault Aware Systems Through Model-based Programming
- Diagnosis as Detective Work
- Model-based Diagnosis
Issue 1: Handling Hidden Failures Requires Reasoning from a Model: STS-93

Symptoms:
- Engine temp sensor high
- LOX level low
- GN&C detects low thrust
- H2 level possibly low

Problem: Liquid hydrogen leak

Effect:
- LH2 used to cool engine
- Engines run hot
- Consumes more LOX

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

- Sherlock Holmes. The Sign of the Four.

1. Test Hypothesis
2. If Inconsistent, learn reason for inconsistency (a Conflict).
3. Use conflicts to leap over similarly infeasible options to next best hypothesis.

---

Compare Most Likely Hypothesis to Observations

Isolate Conflicting Information

Leap to the Next Most Likely Hypothesis that Resolves the Conflict

New Hypothesis Exposes Additional Conflicts

The red component modes conflict with the model and observations.

It is most likely that all components are okay.

The next hypothesis must remove the conflict.

Another conflict, try removing both.
Final Hypothesis Resolves all Conflicts

Outline
- Fault Aware Systems Through Model-based Programming
- Diagnosis as Detective Work
- Model-based Diagnosis

Model-based Diagnosis
Given a system with symptomatic behavior and a model of the system, find diagnoses that account for symptoms.

Diagnosis as Hypothesis Testing
1. Generate candidates, given symptoms.
2. Test if candidates account for all symptoms.

Desired Properties:
- Set of diagnoses should be complete.
- Set of diagnoses should consider all available information.

Issue 2: Failures are Often Novel:
Mars Observer: Explosion due to oxidizer/fuel leakage?
Issue 2: How Should Diagnoses Account for Novel Failures?

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

Suspending Constraints: Make no presumptions about faulty component behavior.

Diagnosis identifies consistent modes

Adder(i):
- G(i): Out(i) = In1(i)+In2(i)
- U(i):

Candidate = {A1=G, A2=G, M1=G, M2=G, M3=G}

- Candidate: Assignment to all component modes.

Diagnosis identifies All sets of consistent modes

Adder(i):
- G(i): Out(i) = In1(i)+In2(i)
- U(i):

Diagnosis = {A1=G, A2=U, M1=G, M2=U, M3=G}

- Diagnosis D: Candidate consistent with model Phi and observables OBS.
  - As more constraints are relaxed, candidates are more easily satisfied.
  - Typically an exponential number of candidates.

Issue 3: Multiple Faults Occur

- three shorts, tank-line and pressure jacket burst, panel flies off.

  ➔ Divide & Conquer
  ➔ Diagnose each symptom.
  ➔ Summarize (conflicts)
  ➔ Combine

APOLLO 13
Representing Diagnoses Compactly: Kernel Diagnoses

“Smallest” sets of modes that remove all symptoms

Every candidate that is a subset of a kernel diagnosis is a diagnosis.

Testing Consistency

→ Propositional Logic
- DPLL Sat algorithm
- Unit propagation (incomplete)

- Finite Domain Constraints
  - Backtrack Search w/ Forward Checking, ...
  - AC-3/Waltz constraint propagation (incomplete)

- Algebraic Constraints
  - Sussman/Steele Constraint Propagation:
    - Propagate newly assigned values through equations mentioning variables.
    - To propagate, use assigned values of constraint to deduce unknown value(s) of constraint.

Summary: Consistency-based Diagnosis

- Component Model + Structure:

  X ∈ {1,0}
  \[ X = 1 \lor \neg X = 0 \]
  \[ \neg X = 1 \lor \neg X = 0 \]

Outline

Model-based Diagnosis
- Conflicts and Kernel Diagnoses
- Generating Kernels from Conflicts
- Finding Consistent Modes
- Estimating Likely Modes
- Conflict-directed A*

Diagnosis by Divide and Conquer

Given model Phi and observations OBS
- 1. Find all symptoms
- 2. Diagnose each symptom separately
   (each generates a conflict → candidates)
- 3. Merge diagnoses
   (set covering → kernel diagnoses)

General Diagnostic Engine
[de Kleer & Williams, 87]
Conflicts Explain How to Remove Symptoms

Symptom:
F is observed 10, but should be 12 if A1, M1 & M2 are okay.

Conflict:
A1=G & M1=G & M2=G is inconsistent
A1=U or M1=U or M2=U removes conflict.
I.e., at least one is broken

Find Another Symptom

Symptom:
G is observed 12, but should be 10 ...

... and its Conflict

Symptom:
G is observed 12, but should be 10
Conflict:
A1=G & M2=G & M1=G & M3=G is inconsistent
Conflict not just upstream from symptom
A1=U or A2=U or M1=U or M3=U removes conflict

... and its Conflict

Summary: Conflicts

Conflict:
A set of component modes M that are inconsistent with the model and observations.
Properties:
• Every superset of a conflict is a conflict
• Only need conflicts that are minimal under subset
• Logically, not M is an implicate of Model & Obs

Symptom:
F is observed 10, but should be 12 if A1, M1 & M2 are okay.
Conflict:
A1=G & M1=G & M2=G is inconsistent
A1=U or M1=U or M2=U removes conflict.
I.e., at least one is broken
Summary: Kernel Diagnoses

Kernel Diagnosis
= \{A_2=U \land M_2=U\}

Partial Diagnosis: A set of component modes M all of whose extensions are diagnoses.
- M removes all symptoms
- M entails Model & Obs

Kernel Diagnosis: A minimal partial diagnosis K
- M is a prime implicant of model & obs

Outline

Model-based Diagnosis
- Conflicts and Kernel Diagnoses
- Generating Kernels from Conflicts
- Finding Consistent Modes
- Estimating Likely Modes
- Conflict-directed A*

Diagnoses Found by Mapping Conflicts to Kernels

Conflict: A set of component modes M that are inconsistent with the model and observations.
- not M is an implicate of Model & Obs

Kernel Diagnosis: A minimal set of component modes K that eliminate all symptoms.
- M is a prime implicant of Model & Obs
- Conflicts map to Kernels by minimal set covering

(see “Characterizing Diagnosis,” de Kleer, Reiter, Mackworth)

Generate Kernels From Conflicts

\{A_1=G, M_1=U, M_2=U\} conflict 1.
\{A_1=U, A_2=U, M_1=U, M_3=U\} conflict 2

A_1=U or M_1=U or M_2=U removes conflict 1.
A_1=U or A_2=U or M_1=U or M_3=U removes conflict 2

Kernel Diagnoses = \{A_1=U\}

“Smallest” sets of modes that remove all conflicts
Generate Kernels From Conflicts
{A1=G, M1=U, M2=U} conflict 1
{A1=U, A2=U, M1=U, M3=U} conflict 2
A1=U or M1=U or M2=U removes conflict 1.
A1=U or A2=U or M1=U or M3=U removes conflict 2

"Smallest" sets of modes that remove all conflicts

Outline
Model-based Diagnosis
- Conflicts and Kernel Diagnoses
- Generating Kernels from Conflicts
- Finding Consistent Modes
- Estimating Likely Modes
- Conflict-directed A*

Diagnosis With Only the Unknown

Nominal and Unknown Modes

Exhaustive Fault Modes

Notational Note:
G(i) = [i = G]
Solution: Diagnosis as Estimating Behavior Modes

Inverter(i):
- \( G(i) \):\hspace{1em} \text{Out}(i) = \text{not}(\text{In}(i))
- \( S1(i) \):\hspace{1em} \text{Out}(i) = 1
- \( S0(i) \):\hspace{1em} \text{Out}(i) = 0 \quad \text{• Isolates surprises}
- \( U(i) \):\hspace{1em} \text{Expects}

Nominal, Fault and Unknown Modes

Example Diagnoses

\[
\begin{array}{l}
0 \quad A \quad X \quad B \quad Y \quad C \quad 0 \\
\hline
\end{array}
\]

Diagnosis: \([S1(A), G(B), U(C)]\)

Kernel Diagnosis: \([U(C)]\)

1. Find Symptoms & Conflicts

Conflict:
- \( \text{not } [G(A), G(B) \text{ and } G(C)] \)

More Symptoms & Conflicts

Not \([S1(A), G(B), \text{and } G(C)]\)

Not \([S0(B) \text{ and } G(C)]\)
More Symptoms & Conflicts

2. Constituent Diagnoses from Conflicts

- \(<S1(C)>
  => G(C), S0(C) or U(C)
- \(<S0(B), G(C)>
  => G(B), S1(B), U(B), S1(C), S0(C) or U(C)
- \(<S1(A), G(B), G(C)>
  => G(A), S0(A), U(A), S1(B), S0(B), U(B), S1(C), S0(C) or U(C)
- \(<G(A), G(B), G(C)>
  => S1(A), S0(A), U(A), S1(B), S0(B), U(B), S1(C), S0(C) or U(C)

3. Generate Kernel Diagnoses

- \([U(C)]\)
- \([S0(C)]\)
- \([G(C), S0(C), U(C)]\)
- \([G(B), S1(B), U(B), S1(C), S0(C), U(C)]\)
- \([G(A), S0(A), U(A), S1(B), S0(B), U(B), S1(C), S0(C), U(C)]\)
- \([S1(A), S0(A), U(A), S1(B), S0(B), U(B), S1(C), S0(C), U(C)]\)

3. Generating Kernel Diagnoses

- \([U(C)]\)
- \([S0(C)]\)
- \([G(C), S0(C), U(C)]\)
- \([G(B), S1(B), U(B), S1(C), S0(C), U(C)]\)
- \([G(A), S0(A), U(A), S1(B), S0(B), U(B), S1(C), S0(C), U(C)]\)
- \([S1(A), S0(A), U(A), S1(B), S0(B), U(B), S1(C), S0(C), U(C)]\)
3. Generating Kernel Diagnoses

- [G(C),S0(C),U(C)]
- [G(B),S1(B),U(B),S1(C),S0(C),U(C)]
- [G(A),S0(A),U(A),S1(B),S0(B),U(B),S1(C),S0(C),U(C)]
- [S1(A),S0(A),U(A),S1(B),S0(B),U(B),S1(C),S0(C),U(C)]

- [U(C)]
- [S0(C)]
- [U(B),G(C)]

3. Generate Kernel Diagnoses

- [G(C),S0(C),U(C)]
- [G(B),S1(B),U(B),S1(C),S0(C),U(C)]
- [G(A),S0(A),U(A),S1(B),S0(B),U(B),S1(C),S0(C),U(C)]
- [S1(A),S0(A),U(A),S1(B),S0(B),U(B),S1(C),S0(C),U(C)]

- [U(C)]
- [S0(C)]
- [U(B),G(C)]

- [G(B),S1(B),U(B),S1(C),S0(C),U(C)]
- [G(A),S0(A),U(A),S1(B),S0(B),U(B),S1(C),S0(C),U(C)]
- [S1(A),S0(A),U(A),S1(B),S0(B),U(B),S1(C),S0(C),U(C)]

- [U(C)]
- [S0(C)]
- [U(B),G(C)]

- [S1(B),G(C)]
- [U(A),G(B),G(C)]

3. Generating Kernel Diagnoses

- [G(C),S0(C),U(C)]
- [G(B),S1(B),U(B),S1(C),S0(C),U(C)]
- [G(A),S0(A),U(A),S1(B),S0(B),U(B),S1(C),S0(C),U(C)]
- [S1(A),S0(A),U(A),S1(B),S0(B),U(B),S1(C),S0(C),U(C)]

- [U(C)]
- [S0(C)]
- [U(B),G(C)]

- [S1(B),G(C)]
- [U(A),G(B),G(C)]

Outline

Model-based Diagnosis
- Conflicts and Kernel Diagnoses
- Generating Kernels from Conflicts
- Finding Consistent Modes
- Estimating Likely Modes
- Conflict-directed A*

Due to the unknown mode, there tends to be an exponential number of diagnoses.

But these diagnoses represent a small fraction of the probability density space.

Most of the density space may be represented by enumerating the few most likely diagnoses.
Candidate Initial (prior) Probabilities

\[ p(c) = \prod_{m \in c} p(m) \]

Assume Failure Independence

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>p(G)</td>
<td>.99</td>
<td>.99</td>
<td>.99</td>
</tr>
<tr>
<td>p(S1)</td>
<td>.008</td>
<td>.008</td>
<td>.001</td>
</tr>
<tr>
<td>p(S0)</td>
<td>.001</td>
<td>.001</td>
<td>.008</td>
</tr>
<tr>
<td>p(U)</td>
<td>.001</td>
<td>.001</td>
<td>.001</td>
</tr>
</tbody>
</table>

\[ p(\{G(A),G(B),G(C)\}) = .97 \]
\[ p(\{S1(A),G(B),G(C)\}) = .008 \]
\[ p(\{S1(A),G(B),S0(C)\}) = .00006 \]
\[ p(\{S1(A),S1(B),S0(C)\}) = .0000005 \]

Posterior Probability, after Observation \( x = v \)

\[ p(c | x = v) = \frac{p(x = v | c)p(c)}{p(x = v)} \]

Bayes’ Rule

P(x=v|c) estimated using Model:

- If previous obs, \( c \) and Phi entails \( x = v \)
  - Then \( p(x = v | c) = 1 \)
- If previous obs, \( c \) and Phi entails \( x = v \)
  - Then \( p(x = v | c) = 0 \)
- If Phi consistent with all values for \( x \)
  - Then \( p(x = v | c) \) is based on priors
    - E.g., uniform prior = \( \frac{1}{m} \) for \( m \) possible values of \( x \)

Observe \( \text{out} = 1 \):

\[ C = [G(A),G(B),G(C)] \]
\[ P(C) = .97 \]
\[ P(\text{out} = 1 | C) = ? \]
\[ = 1 \]
\[ P(C | \text{out} = 0 ) = ? \]
\[ = .97/p(x=v) \]

Observe \( \text{out} = 0 \):

\[ C = [G(A),G(B),G(C)] \]
\[ P(C) = .97 \]
\[ P(\text{out} = 0 | C) = ? \]
\[ = 0 \]
\[ P(C | \text{out} = 0 ) = ? \]
\[ = 0 \times .97/p(x=v) = 0 \]

Example: Tracking Single Faults
- which are eliminated?
- which predict observations?
- Which are agnostic?

Priors for Single Fault Diagnoses:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>p(S1)</td>
<td>.008</td>
<td>.008</td>
<td>.002</td>
</tr>
<tr>
<td>p(S0)</td>
<td>.001</td>
<td>.001</td>
<td>.008</td>
</tr>
<tr>
<td>p(U)</td>
<td>.001</td>
<td>.001</td>
<td>.001</td>
</tr>
</tbody>
</table>
Summary: Candidate Probabilities

\[ p(c) = \prod_{m \in c} p(m) \]

Assume Failure Independence

\[ p(c \mid x = v) = \frac{p(x = v \mid c)p(c)}{p(x = v)} \]

Bayes’ Rule

\[ P(x=v \mid c) \] estimated using Model:

- If previous obs, c and Phi entails \( x = v \)
  - Then \( p(x = v \mid c) = 1 \)
- If previous obs, c and Phi entails \( x \neq v \)
  - Then \( p(x = v \mid c) = 0 \)
- If Phi consistent with all values for x
  - Then \( p(x = v \mid c) \) is based on priors
  - E.g., uniform prior = \( \frac{1}{m} \) for m possible values of x

Due to the unknown mode, there tends to be an exponential number of diagnoses.

But these diagnoses represent a small fraction of the probability density space.

Most of the density space may be represented by enumerating the few most likely diagnoses.