Assignment

- Reading
  - Model-based Diagnosis – Lecture notes
  - Propositional Logic AIMA Chapter 6

Diagnostic Agent:
- Monitors & Diagnoses
- Repairs & Avoids
- Probes and Tests

Symptom-directed
Outline

- Model-based diagnosis
- Defining diagnoses
- Searching for diagnoses
- Appendix

Hidden Failures Require 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
- Engine runs hot
- Consumes more LOX

Model-based Diagnosis

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

Output: Diagnoses that account for the symptoms.
Model-based Diagnosis
Input: Observations of a system with symptomatic behavior, and a model of the system,
Output: Diagnoses that account for the symptoms.

Solution: Diagnosis as Hypothesis Testing
1. Generate candidates, given symptoms.
2. Test if candidates account for all symptoms.
   • Set of diagnoses should be complete.
   • Set of diagnoses should exploit all available information.

Outline
- Model-based diagnosis
- Defining diagnoses
  - Explaining failures (appendix)
  - Handling unknown failures
- Searching for diagnoses
- Appendix
How Should Diagnoses Account for Symptoms?

Abductive Diagnosis: Given symptoms, find diagnoses that predict observations.

- Fault Model: O1's output is stuck to 0
- Output shorted to ground

Requires exhaustive fault models.
Input Model: Abductive, Model-based Diagnosis

- Input Model: Abductive, Model-based Diagnosis

Model-based Diagnosis

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

Output: Diagnoses that account for the symptoms.

Input ⇒ Output: Abductive, Model-based Diagnosis

- Input Model: Abductive, Model-based Diagnosis
Abductive Diagnosis by Generate and Test

Given: Exhaustive fault models, structure and observations.
Generate: Consider each mode assignment as a candidate.
Test:
1. Simulate candidate, given inputs.
2. Compare to responses
   - Disagree: Discard
   - Agree: Keep
   - No prediction: Discard
3. Exonerate component if none of its fault models agree

Problem:
- Fault models are often incomplete
- May incorrectly exonerate faulty components

Outline
- Model-based diagnosis
- Defining diagnosis
  - Explaining failures (appendix)
  - Handling unknown failures
- Searching for diagnoses
- Appendix

Issue: Failures are Often Novel
- Mars Observer
- Mars Climate Orbiter
- Mars Polar Lander
- Deep Space 2

courtesy of JPL

10/03/03 copyright Brian Williams, 2003
How Should Diagnoses Account for Novel Symptoms?

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

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

[Diagram of a decision tree or a similar visual representation is shown here.]

Copyright Brian Williams, 2003
How Should Diagnoses Account for Novel Symptoms?

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

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

Consistency-based Diagnosis

And(i):
  • \( G(i) \)
  • \( Out(i) = In1(i) \) AND \( In2(i) \)
  • \( U(i) \)

ALL components have “unknown Mode” \( U \), Whose assignment is never mentioned in \( M \)

* Obs: Assignment to \( O \)
* Candidate \( C \): Assignment of modes to \( X \)
* Diagnosis \( D \): A candidate such that \( D \land Obs \land M(X,Y) \) is satisfiable.

Testing Consistency

→ Propositional Logic
  • DPLL (Titan)
  • Just unit propagation (incomplete) (Livingstone/DS1)

• Finite Domain Constraints
  • Backtrack w forward checking
  • Waltz constraint propagation (incomplete)

• Algebraic Constraints (GDE/Sherlock/GDE+/XDE)
  • Gaussian Elimination
  • Sussman/Steel Constraint Propagation (incomplete)
    • Propagate newly assigned values through equations mentioning variables.
    • To propagate, use assigned values of constraint to deduce unknown value(s) of constraint.
Encoding Models In Propositional Logic

And(i):
- G(i):
  - Out(i) = In1(i) AND In2(i)
  - U(i):
    - ¬(i=G) ∨ ¬(In1(i)=0) ∨ Out(i)=0
    - ¬(i=G) ∨ ¬(In2(i)=0) ∨ Out(i)=0
- ¬(i=G) ∨ ¬(In1(i)=1) ∨ ¬(In2(i)=1) ∨ Out(i)=1

Or(i):
- G(i):
  - Out(i) = In1(i) OR In2(i)
  - U(i):
    - ¬(i=G) ∨ ¬(ln1(i)=1) ∨ Out(i)=1
    - ¬(i=G) ∨ ¬(ln2(i)=1) ∨ Out(i)=1
    - ¬(i=G) ∨ ¬(ln1(i)=0) ∨ ¬(ln2(i)=0) ∨ Out(i)=0

X ∈ {1,0}
- X=1 ∨ X=0
- ¬X=1 ∨ ¬X=0

Outline

- Model-based diagnosis
- Defining diagnosis
  - Explaining failures (appendix)
  - Handling unknown failures
- Searching for diagnoses
  - Conflict learning
  - Single-fault diagnosis
- Appendix
  - Multiple-fault diagnosis

Learning Conflicts From Symptoms

Symptom:
F is observed 0, but should be 1 if O1, O2 and A1 are okay.
Conflict:
¬(At least A1=U or O1=U or O2=U)

10/03/03 copyright Brian Williams, 2003
procedure propagate(C)  // C is a clause
    if all literals in C are false except I, and I is unassigned
    then assign true to I and
    record C as a support for I and
    for each clause C’ mentioning “not I”.
    propagate(C’)
end propagate
Find Symptom Using Unit Propagation

\( O_1 = G \)
\( A = 1 \)
\( O_1 = G \)
\( X = 1 \)
\( -A \lor -X \lor -Y \lor F = 1 \)

\( O_2 = G \)
\( B = 1 \)
\( X = 1 \)
\( Y = 1 \)
\( -F \lor -F \)

\( G \)
\( F = 1 \)

Copyright Brian Williams, 2003
Extract Conflict by Tracing Support

procedure Conflict(C)  // C is an inconsistent clause
  for each literal l in C
    union Support-Conflict(l, support(l))
  end Conflict

procedure Support-Conflict(l,S)
  If unit-clause?(C)
    If mode-assgnment?(l literal (C))
      Then ( literal(C) )
    Else {} 
  Else for each literal l1 in C, other than l
    Union Support-Conflict(l1, support(l1))
  end Support-Conflict

Candidate Test with Conflict Extraction

procedure Test_Candidate(c,M,obs)
  1. Assert candidate assignment c
  2. Propagate obs through model M using unit propagation.
  3. If inconsistent clause return Conflict(c)
  4. Else search for satisfying solution using DPLL
    • If inconsistent return c as a conflict.
    • Else return "consistent"
Outline

- Model-based diagnosis
- Defining diagnosis
- Searching for diagnoses
  - Conflict learning
  - Single-fault diagnosis
- Appendix

Single Fault Diagnosis w Conflicts: Generate Candidates From Symptom

```
Single_Fault_w_Conflicts(M, X, Obs)
\[\text{\textbackslash\textbackslash Model M, Mode variables X, Observation Obs}\]
1. Assume all components okay,
   \[\text{All Good = \{ x=G | x \in X \}} \]
2. Conflict \leftarrow \text{Test Candidate}(All Good, M, Obs)
3. If Conflict = "consistent" return All Good
4. Generate single fault candidates
   \[\text{Cands } \leftarrow \{ (x=U) \cup Z=G | x=G \in \text{Conflict}, Z=X-\{x\} \} \]
5. Test_Candidates(Cands, M, Obs)
```

Generate Candidates From Symptom

![Diagram](image_url)

Symptom: F is observed 0, but should be 1
Conflict: \{O1=G, O3=G, A1=G, A2=G\} is inconsistent
Candidates: \{\{O1=U\...\}, \{O3=U\...\}, \{A1=U\...\}, \{A2=U\...\}\}
Generate Candidates From Symptom

Symptom: F is observed 0, but should be 1
Conflict: \{O1=G, O3=G, A1=G, A2=G\} is inconsistent
Candidates: \{{O1=U…}, {O3=U…}, {A1=U…}, {A2=U…}\}

Single Fault Diagnosis w Conflicts: Test Candidates, Collecting Conflicts

```
Single_Fault_Test_Candidates(C,M,Obs)
\del Candidates C, Model M, Observation Obs
\{ Diagnoses \leftarrow \{\}, Conflicts \leftarrow \{\}
\text{For each } c \text{ in } C
\quad \text{If } c \text{ is a superset of some conflict in Conflicts}
\quad \quad \text{Then inconsistent candidate, ignore.}
\quad \quad \text{Else Conflict = Test_Candidate(c, M, Obs)}
\quad \quad \text{If Conflict = "consistent"}
\quad \quad \quad \text{Then add } c \text{ to Diagnoses}
\quad \quad \quad \text{Else add Conflict to Conflicts}
\text{return Diagnoses}
```

Test Candidates, Collecting Conflicts

Candidates: \{{O1=U…}, {O3=U…}, {A1=U…}, {A2=U…}\}
Diagnoses: \{\}

- First candidate \{O1=U, \ldots\}
Test Candidates, Collecting Conflicts
Candidates: {O1=U…}, {O3=U…}, {A1=U…}, {A2=U…}
Diagnoses: {}

- First candidate {O1=U…}
- Suspend O1’s constraints
- Test consistency
- Consistent: Add to solutions
Test Candidates, Collecting Conflicts

Candidates: \{O3=U\ldots, \{A1=U\ldots, \{A2=U\ldots\}\}

Diagnoses: \{|O1=U\ldots|\}

- Second candidate \{O3=U, \ldots\}
- Suspend O3’s constraints
- Test consistency

Test Candidates, Collecting Conflicts

Candidates: \{O3=U\ldots, \{A1=U\ldots, \{A2=U\ldots\}\}

Diagnoses: \{|O1=U\ldots|\}

- Second candidate \{O3=U, \ldots\}
- Suspend O3’s constraints
- Test consistency → Inconsistent

Test Candidates, Collecting Conflicts

Candidates: \{O3=U\ldots, \{A1=U\ldots, \{A2=U\ldots\}\}

Diagnoses: \{|O1=U\ldots|\}

Conflicts: \{|O1=G, O2=G, A1=G|\}

- Second candidate \{O3=U, \ldots\}
- Extract Conflict: \{O1=G, O2=G, A1=G\}
- Suspend O3’s constraints
- Test → Inconsistent
- Use to prune candidates
Test Candidates, Collecting Conflicts

Candidates: \{A1=U, \ldots\}, \{A2=U, \ldots\}
Diagnoses: \{O1=U, \ldots\}
Conflicts: \{O1=G, O2=G, A1=G\}

• Third candidate \{A1=U, \ldots\}
• Subsumed by conflict? \rightarrow No, since A1 = U, not A1=G
• Suspend A1’s constraints
• Test \rightarrow Consistent

Test Candidates, Collecting Conflicts

Candidates: \{A2=U, \ldots\}
Diagnoses: \{O1=U, \ldots\}, \{A1=U, \ldots\}
Conflicts: \{O1=G, O2=G, A1=G\}

• Fourth candidate \{A2=U, \ldots\}
• Subsumed by conflict? \rightarrow Yes, since O1=G, O2=G and A1=G
• Eliminate candidate
Consistent

Test Candidates, Collecting Conflicts

Candidates: {}  
Diagnoses: \{O1=U, \ldots\}, \{A1=U, \ldots\}
Conflicts: \{O1=G, O2=G, A1=G\}

• Return Solutions \rightarrow O1 or A1 broken
### Single Fault Diagnoses are the Intersection of All Conflicts

\[
\{A1=G, O1=U, O2=U\} \quad \text{conflict 1}
\]
\[
\{A1=U, A2=U, O1=U, O3=U\} \quad \text{conflict 2}
\]

\[
A1=U \text{ or } O1=U \text{ or } O2=U \quad \text{removes conflict 1}
\]
\[
A1=U \text{ or } A2=U \text{ or } O1=U \text{ or } O3=U \quad \text{removes conflict 2}
\]

Single Fault Diagnoses = \{\{A1=U..\}, \{O1=U..\}\}

---

### Outline

- Model-based diagnosis
- Defining diagnosis
  - Explaining failures
  - Handling unknown failures
- Searching for diagnoses
  - Conflict learning
  - Single-fault diagnosis
- Appendix
  - Multiple-fault diagnosis

---

### Summary: Model-based Diagnosis

- A failure is a discrepancy between the model and observations of an artifact.
- Diagnosis is symptom directed.
- Symptoms identify conflicting components as initial candidates.
- Test novel failures by suspending constraints and testing consistency.
- Newly discovered conflicts further prune candidates.
Appendix

- Multiple Fault Diagnosis

Multiple Faults Occur

- Quintuple fault occurs (three shorts, tank-line and pressure jacket burst, panel flies off).
- Power limitations too severe to perform new mission.
- Novel reconfiguration identified, exploiting LEM batteries for power.
- Swaggert & Lovell work on Apollo 13 emergency rig lithium hydroxide unit.

Courtesy of Kanna Rajan, NASA Ames. Used with permission.

Diagnosis identifies consistent modes

\[
\begin{align*}
\text{Adder}(i): \\
& G(i) = \text{In}(i)+\text{In}(j) \\
& U(i) = \text{Out}(i)+\text{Out}(j)
\end{align*}
\]


- Candidate: Assignment to all component modes.
Diagnosis identifies All sets of consistent modes

Adder(i):
- G(i): Out(i) = ln1(i)+ln2(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.

Representing Diagnoses Compactly: Kernel Diagnoses

Kernel Diagnosis = \{A2=U, M2=U\}

“Smallest” sets of modes that remove all symptoms
Every candidate that is a subset of a kernel diagnosis is a diagnosis.

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

Kernel Diagnoses =

“Smallest” sets of modes that remove all conflicts
Generate Kernels From Conflicts

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

Kernel Diagnoses = \{A1=U\}

“Smallest” sets of modes that remove all conflicts

Generate Kernels From Conflicts

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

Kernel Diagnoses = \{M1=U\}
\{A1=U\}

“Smallest” sets of modes that remove all conflicts

Generate Kernels From Conflicts

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

Kernel Diagnoses = \{A2=U, M2=U\}
\{M1=U\}
\{A1=U\}

“Smallest” sets of modes that remove all conflicts
Generate Kernels From Conflicts

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

Kernel Diagnoses = {M2=U,M3=U}
{A2=U,M2=U}
{M1=U}
{A1=U}

“Smallest” sets of modes that remove all conflicts

---

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]
Conflict-Directed A*: Generating The Best Kernel

Insight:
- Kernels found by minimal set covering
- Minimal set covering is an instance of breadth first search.

Insight:
- Kernels found by minimal set covering
- Minimal set covering is an instance of breadth first search.
- To find the best kernel, expand tree in best first order.