

Outline

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

10/03/03

copyright Brian Williams, 2003

4

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

10/03/03

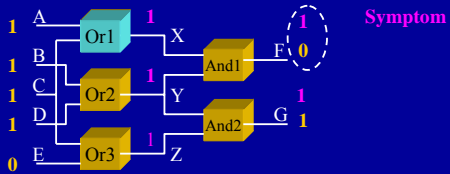
copyright Brian Williams, 2003

5

Model-based Diagnosis

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

Output: Diagnoses that account for the symptoms.



10/03/03

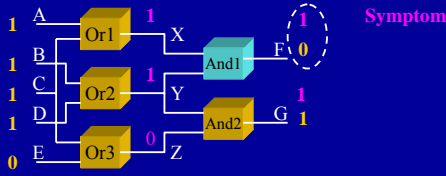
copyright Brian Williams, 2003

6

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

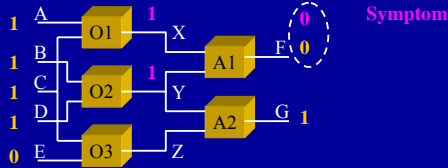
1. **Generate** candidates, given symptoms.
 2. **Test** if candidates account for all symptoms.
- Set of diagnoses should be **complete**.
 - Set of diagnoses should **exploit all available information**.

Outline

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

How Should Diagnoses Account for Symptoms?

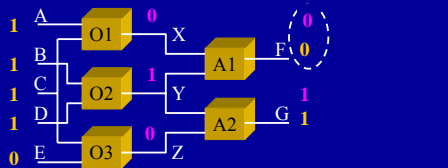
Abductive Diagnosis: Given symptoms, find diagnoses that predict observations.



Requires exhaustive fault models.

How Should Diagnoses Account for Symptoms?

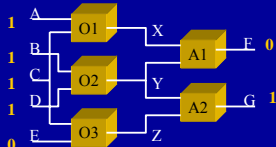
Abductive Diagnosis: Given symptoms, find diagnoses that predict observations.



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

Input: Abductive, Model-based Diagnosis

- Or(i):
 - G(i): $Out(i) = In1(i) \text{ or } In2(i)$
 - Stuck_0(i): $Out(i) = 0$



- Model
 - Structure
 - Model of normal behavior for each component
 - Model for every component failure mode
- Observations
 - Inputs and Response

Input Model: Abductive, Model-based Diagnosis

Or(i):

- G(i): $Out(i) = In1(i) \text{ or } In2(i)$
- Stuck_0(i): $Out(i) = 0$

- X: mode variables, one for each component c
- D_c : modes of component $c = \text{domain of } x_c \in X$
- Y: model variables and their domains
- $M(X, Y)$: model constraints
- O: observable variables $O \subseteq Y$
 - Partitioned into Inputs I and Responses R

10/03/03 copyright Brian Williams, 2003 13

Model-based Diagnosis

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

Output: Diagnoses that account for the symptoms.

10/03/03 copyright Brian Williams, 2003 14

Input \Rightarrow Output: Abductive, Model-based Diagnosis

Or(i):

- G(i): $Out(i) = In1(i) \text{ or } In2(i)$
- Stuck_0(i): $Out(i) = 0$

Candidate = {A1=G, A2=G, O1=G, M2=G, M3=G}

Diagnosis = {A1=G, A2=S0, O1=G, M2=G, M3=G}

- Obs <In, Resp>: Assignment to I and R, respectively
- Candidate C_i : Assignment of modes to X
- Diagnosis D_i : A candidate such that $D_i \wedge In \wedge M(X, Y)$ predicts (entails) Resp

10/03/03 copyright Brian Williams, 2003 15

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

10/03/03 copyright Brian Williams, 2003 16

Outline

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

10/03/03 copyright Brian Williams, 2003 17

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 18

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.

10/03/03 copyright Brian Williams, 2003 20

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.

10/03/03 copyright Brian Williams, 2003 21

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.

10/03/03 copyright Brian Williams, 2003 22

Consistency-based Diagnosis

And(i):

- G(i): $Out(i) = In1(i) \text{ AND } In2(i)$
- U(i):

ALL components have "unknown Mode" U, Whose assignment is never mentioned in M

Diagnosis = {A1=G, A2=U O1=G, O2=U, O3=G}

- Obs: Assignment to O
- Candidate C_i : Assignment of modes to X
- Diagnosis D_i : A candidate such that $D_i \wedge Obs \wedge M(X,Y)$ is satisfiable.

10/03/03 copyright Brian Williams, 2003 23

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/Steele 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.

10/03/03 copyright Brian Williams, 2003 24

Encoding Models In Propositional Logic

And(i):

- G(i): $\neg(i=G) \vee \neg(\text{In1}(i)=0) \vee \text{Out}(i)=0$
- Out(i) = In1(i) AND In2(i) $\neg(i=G) \vee \neg(\text{In2}(i)=0) \vee \text{Out}(i)=0$
- U(i): $\neg(i=G) \vee \neg(\text{In1}(i)=1) \vee \neg(\text{In2}(i)=1) \vee \text{Out}(i)=1$

Or(i):

- G(i): $\neg(i=G) \vee \neg(\text{In1}(i)=1) \vee \text{Out}(i)=1$
- Out(i) = In1(i) OR In2(i) $\neg(i=G) \vee \neg(\text{In2}(i)=1) \vee \text{Out}(i)=1$
- U(i): $\neg(i=G) \vee \neg(\text{In1}(i)=0) \vee \neg(\text{In2}(i)=0) \vee \text{Out}(i)=0$

$X \in \{1,0\}$ $X=1 \vee X=0$
 $\neg X=1 \vee \neg X=0$

10/03/03

copyright Brian Williams, 2003

25

Outline

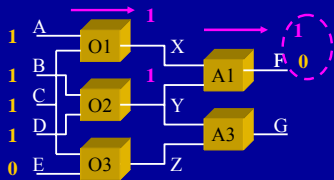
- Model-based diagnosis
- Defining diagnosis
 - Explaining failures (appendix)
 - Handling unknown failures
- Searching for diagnoses
 - Conflict learning
 - Single-fault diagnosis
- Appendix
 - Multiple-fault diagnosis

10/03/03

copyright Brian Williams, 2003

26

Learning Conflicts From Symptoms



Symptom:
 F is observed 0, but should be 1 if O1, O2 and A1 are okay.

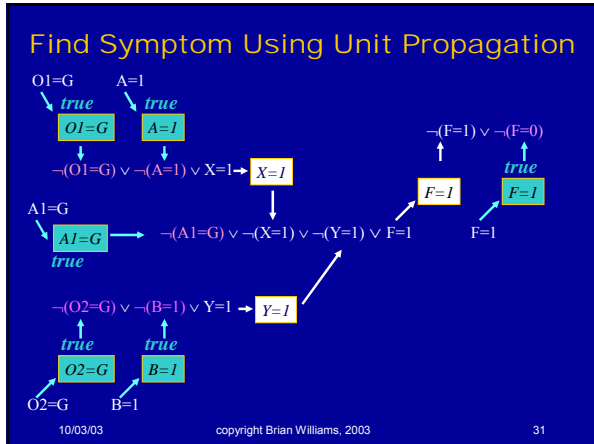
Conflict: {A1=G, O1=G, O2=G} is inconsistent

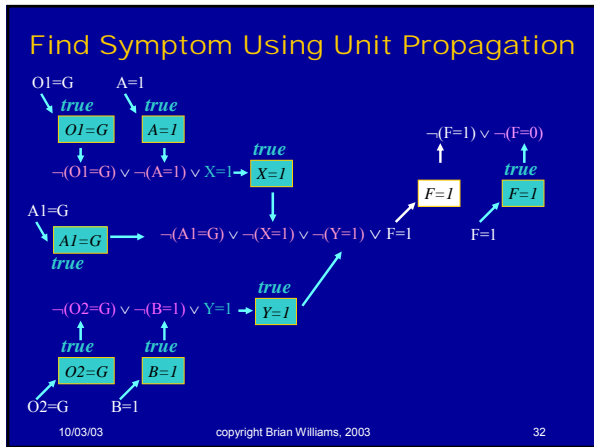
→ At least A1=U or O1=U or O2=U

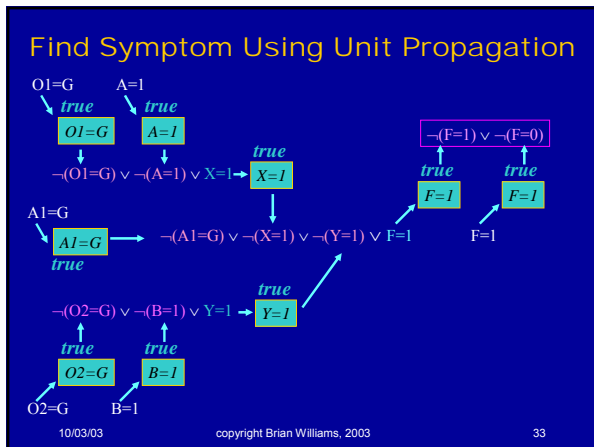
10/03/03

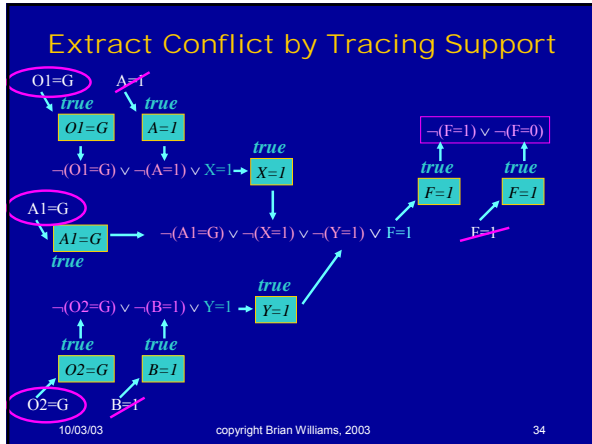
copyright Brian Williams, 2003

27









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-assignment?(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
  
```

10/03/03 copyright Brian Williams, 2003 35

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"
  
```

10/03/03 copyright Brian Williams, 2003 36

Outline

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

10/03/03

copyright Brian Williams, 2003

37

Single Fault Diagnosis w Conflicts: Generate Candidates From Symptom

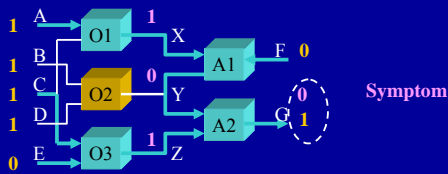
- Single_Fault_w_Conflicts(M, X, Obs)*
 \ Model M, Mode variables X, Observation Obs
1. Assume all components okay, All_Good = { x=G | x ∈ X }
 2. Conflict ← Test_Candidate(All_Good, M, Obs)
 3. If Conflict = "consistent" return All_Good
 4. Generate single fault candidates
 Cands ← { {x=U} ∪ Z=G | x=G ∈ Conflict, Z=X-{x} }
 5. Test_Candidates(Cands, M, Obs)

10/03/03

copyright Brian Williams, 2003

38

Generate Candidates From Symptom



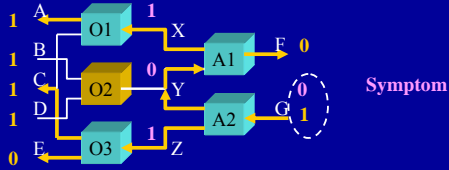
Symptom: F is observed 0, but should be 1
 Conflict: {O1=G, O3=G, A1=G, A2=G} is inconsistent
 Candidates: {{O1=U...}, {O3=U...}, {A1=U...}, {A2=U...}}

10/03/03

copyright Brian Williams, 2003

39

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)
 \ll Candidates C, Model M, Observation Obs

Diagnoses \leftarrow {}, Conflicts \leftarrow {}

For each c in C

If c is a superset of some conflict in Conflicts

Then inconsistent candidate, ignore.

Else Conflict = Test_Candidate(c, M, Obs)

If Conflict = "consistent"

Then add c to Diagnoses

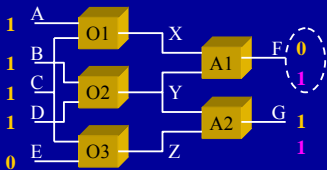
Else add Conflict to Conflicts

return Diagnoses

Test Candidates, Collecting Conflicts

Candidates: {{O1=U...}, {O3=U...}, {A1=U...}, {A2=U...}}

Diagnoses: {}



• First candidate {O1=U, ...}

Test Candidates, Collecting Conflicts

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

- First candidate $\{O1=U, \dots\}$
- Suspend $O1$'s constraints

10/03/03 copyright Brian Williams, 2003 43

Test Candidates, Collecting Conflicts

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

- First candidate $\{O1=U, \dots\}$
- Suspend $O1$'s constraints
- Test consistency

10/03/03 copyright Brian Williams, 2003 44

Test Candidates, Collecting Conflicts

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

- First candidate $\{O1=U, \dots\}$
- Suspend $O1$'s constraints
- Test consistency \rightarrow Consistent: Add to solutions

10/03/03 copyright Brian Williams, 2003 45

Test Candidates, Collecting Conflicts

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

- Second candidate $\{O3=U, \dots\}$
- Suspend $O3$'s constraints
- Test consistency

10/03/03 copyright Brian Williams, 2003 46

Test Candidates, Collecting Conflicts

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

- Second candidate $\{O3=U, \dots\}$
- Suspend $O3$'s constraints
- Test consistency \rightarrow **Inconsistent**

Testing Consistency \neq
Forward Prediction

10/03/03 copyright Brian Williams, 2003 47

Test Candidates, Collecting Conflicts

Candidates: $\{\{\cancel{O3=U\dots}\}, \{A1=U\dots\}, \{A2=U\dots\}\}$
 Diagnoses: $\{\{O1=U\dots\}\}$
 Conflicts: $\{\{O1=G, O2=G, A1=G\}\}$

- Second candidate $\{O3=U, \dots\}$
- Suspend $O3$'s constraints
- Test \rightarrow Inconsistent
- Extract Conflict: $\{O1=G, O2=G, A1=G\}$
- Use to prune candidates

10/03/03 copyright Brian Williams, 2003 48

Test Candidates, Collecting Conflicts

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

- Third candidate $\{A1=U, \dots\}$
- Subsumed by conflict? → No, since $A1 = U$, not $A1=G$
- Suspend $A1$'s constraints
- Test → **Consistent**

10/03/03 copyright Brian Williams, 2003 49

Test Candidates, Collecting Conflicts

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

- Fourth candidate $\{A2=U, \dots\}$
- Subsumed by conflict? → Yes, since $O1=G, O2=G$ and $A1=G$
- Eliminate candidate
- **Consistent**

10/03/03 copyright Brian Williams, 2003 50

Test Candidates, Collecting Conflicts

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

- Return Solutions → $O1$ or $A1$ broken

10/03/03 copyright Brian Williams, 2003 51

Single Fault Diagnoses are the Intersection of All Conflicts

{A1=G, O1=U, O2=U} conflict 1
 {A1=U, A2=U, O1=U, O3=U} conflict 2

A1=U or O1=U or O2=U removes conflict 1
 A1=U or A2=U or O1=U or O3=U removes conflict 2

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

10/03/03 copyright Brian Williams, 2003 52

Outline

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

10/03/03 copyright Brian Williams, 2003 53

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.


10/03/03 copyright Brian Williams, 2003 54

Apendix

- Multiple Fault Diagnosis

10/03/03 copyright Brian Williams, 2003 55

Multiple Faults Occur



courtesy of NASA

- 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.
- Swaggett & Lovell work on Apollo 13 emergency rig lithium hydroxide unit.

APOLLO 13

10/03/03 copyright Brian Williams, 2003 56

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

Diagnosis identifies consistent modes

Adder(i):

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

```

    graph LR
      A[3] --> M1[M1]
      B[2] --> M1
      C[2] --> M2
      D[3] --> M2
      E[3] --> M3
      M1 -- X --> A1[A1]
      M2 -- Y --> A1
      M2 -- Z --> A2[A2]
      M3 -- Z --> A2
      A1 -- 10 --> F
      A2 -- 12 --> G
  
```

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

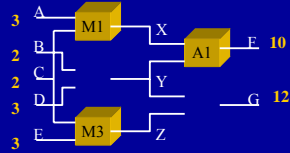
- Candidate: Assignment to all component modes.

10/03/03 copyright Brian Williams, 2003 57

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}

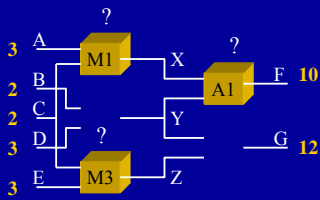
- Diagnosis D: Candidate consistent with model Phi and observables OBS.
- As more constraints are relaxed, candidates are more easily satisfied.
- Typically an exponential number of candidates.

10/03/03

copyright Brian Williams, 2003

58

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.

10/03/03

copyright Brian Williams, 2003

59

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

10/03/03

copyright Brian Williams, 2003

60

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

10/03/03 copyright Brian Williams, 2003 61

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

10/03/03 copyright Brian Williams, 2003 62

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

10/03/03 copyright Brian Williams, 2003 63

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]

Exploring the Improbable

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

- Sherlock Holmes.
The Sign of the Four.

Conflict-Directed A*: Generating The Best Kernel

Conflicts

$A1=U, M1=U, M2=U$

$A1=U, A2=U, M1=U, M3=U$

$A1=U$ $M1=U$ $M1=U \wedge A2=U$ $M2=U \wedge M3=U$

Insight:

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

10/03/03 copyright Brian Williams, 2003 67

Conflict-Directed A*: Generating The Best Kernel

Conflicts

$A1=U, M1=U, M2=U$

$A1=U, A2=U, M1=U, M3=U$

$A1=U$ $M1=U$ $M2=U$ $M1=U$

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.

10/03/03 copyright Brian Williams, 2003 68
