Fault Aware Systems: Model-based Programming and Diagnosis

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

courtesy of JPL

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.

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

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

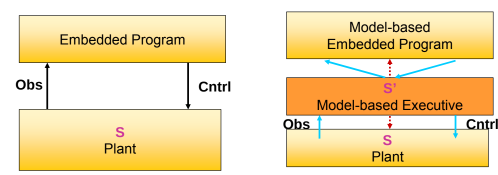
Like Storyboards, Model-based Programs Specify The Evolution of Abstract States

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

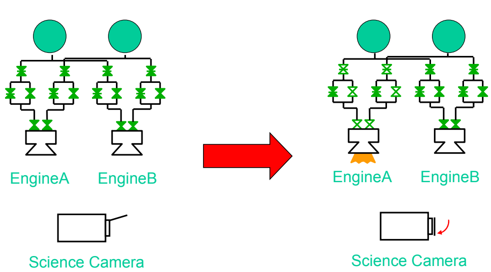


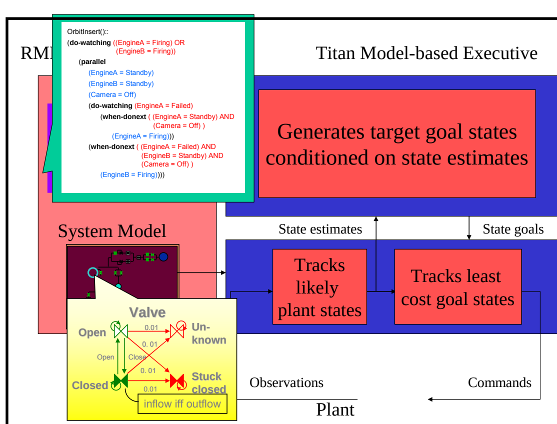
Programmer maps between state and sensors/actuators.

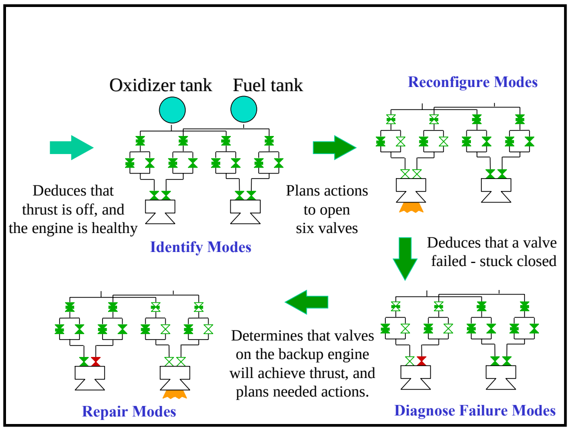
Model-based executive maps between state and sensors/actuators.

Descent Example

Turn camera off and engine on







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

```
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))))
```

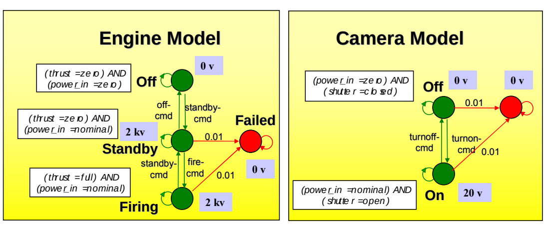
Plant Model

component modes…

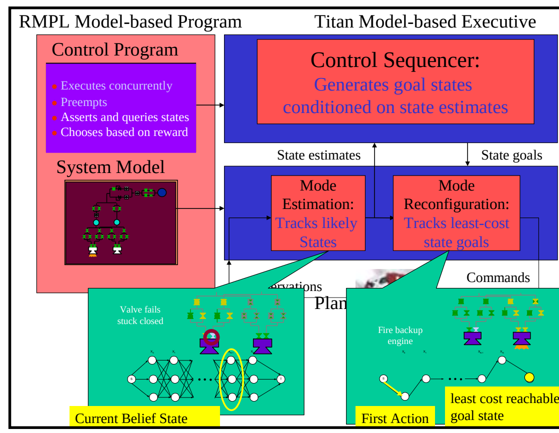
described by finite domain constraints on variables…

deterministic and probabilistic transitions

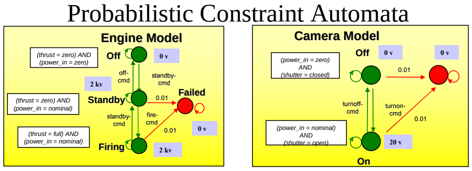
cost/reward



one per component … operating concurrently

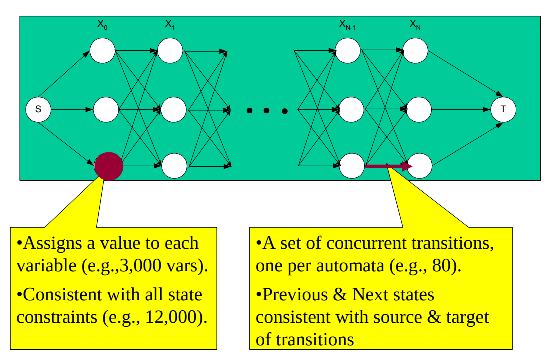


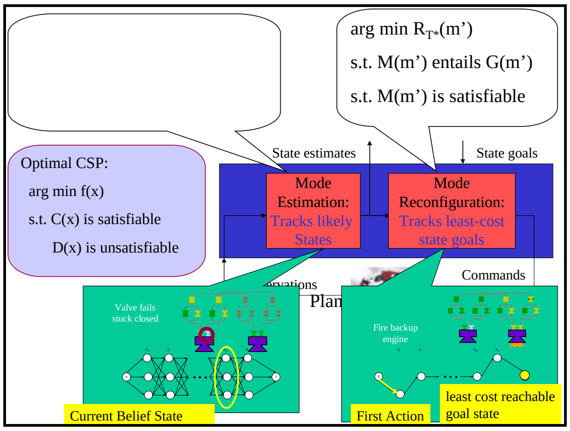
Modeling Complex Behaviors through Probabilistic Constraint Automata

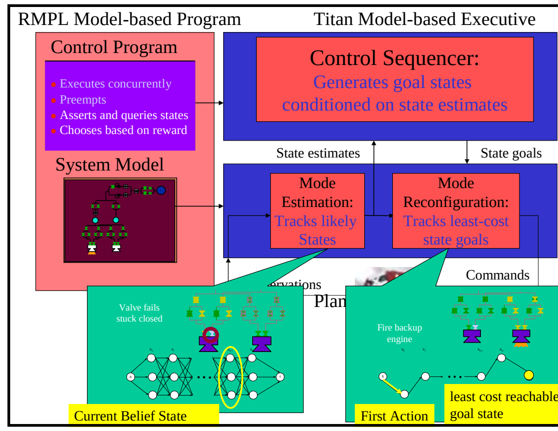


- Complex, discrete behaviors
  - modeled through concurrency, hierarchy and timed transitions.
- Anomalies and uncertainty
  - modeled by probabilistic transitions
- Physical interactions
  - modeled by discrete and continuous constraints

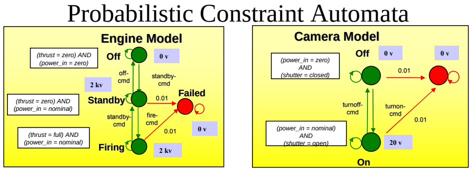
The Plant's Behavior





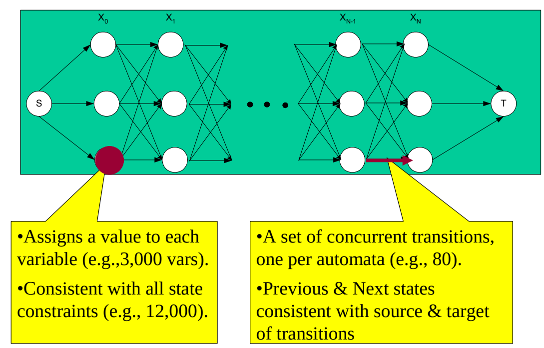


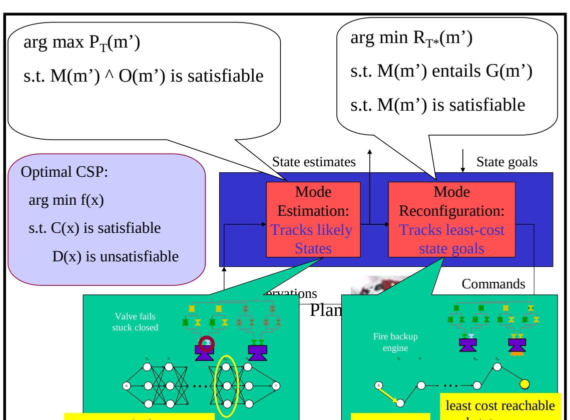
Modeling Complex Behaviors through Probabilistic Constraint Automata



- 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

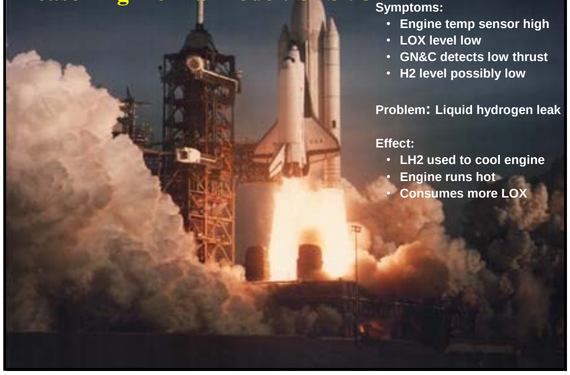




Outline

- 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



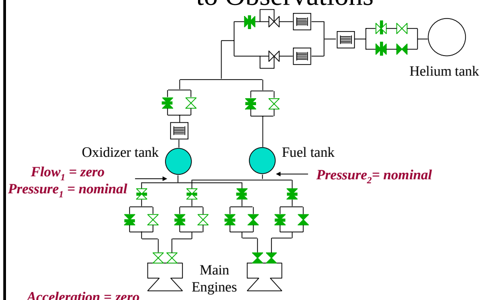
Model- ased Diagnosis as Conflict-directed Best First Search

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

- Sherlock Holmes. The Sign of the Four.

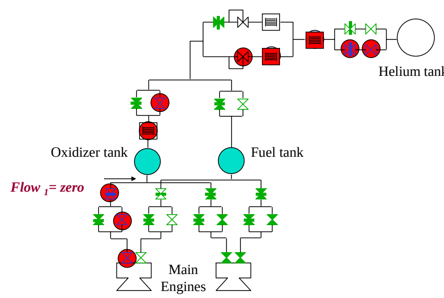
1. Test Hypothesis
2. If Inconsistent, learn reason for inconsistency (a Conflict).
3. Use conflicts to leap over similarly infeasible options to next best hypothesis.

Compare Most Likely Hypothesis to Observations



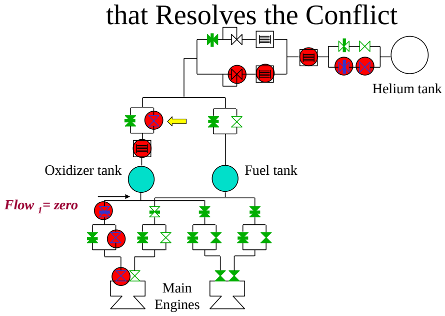
It is most likely that all components are okay.

Isolate Conflicting Information



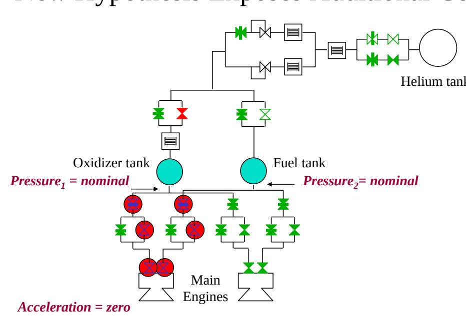
The red component modes *conflict* with the model and observations.

Leap to the Next Most Likely Hypothesis that Resolves the Conflict

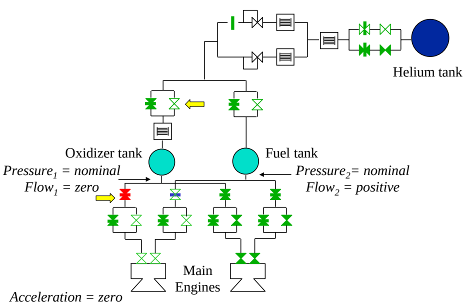


The next hypothesis must remove the conflict

New Hypothesis Exposes Additional Conflicts



Another conflict, try removing both



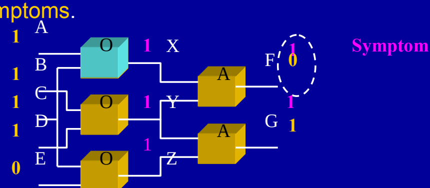
Implementation: Conflict-directed A* search.

Outline

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

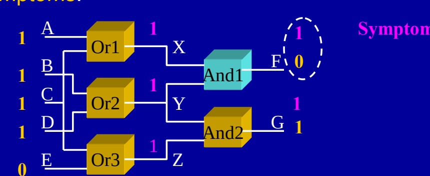
Model-based Diagnosis

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



Model-based Diagnosis

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



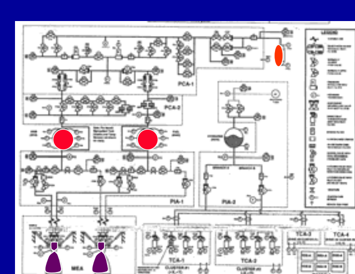
Diagnosis as Hypothesis Testing

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:

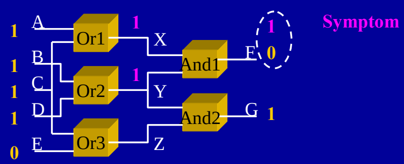


Mars Observer: Explosion due to oxidizer/fuel leakage?

Issue 2: How Should Diagnoses Account for Novel Failures?

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

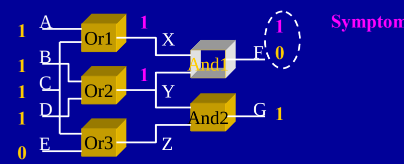
Suspending Constraints: Make no presumptions about faulty component behavior.



Issue 2: How Should Diagnoses Account for Novel Failures?

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

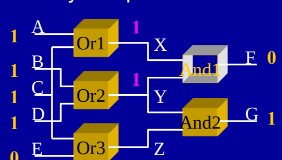
Suspending Constraints: Make no presumptions about faulty component behavior.



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.



Issue 3: Multiple Faults Occur

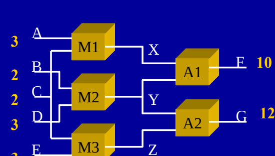
APOLLO 13

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

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

Diagnosis identifies consistent modes

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

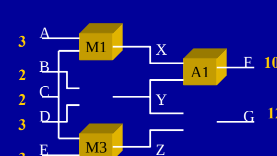


Candidate = {A1=G, A2=G, M1=G, M2=G, M3=G}

- Candidate: Assignment to all component modes.

Diagnosis identifies All sets of consistent modes

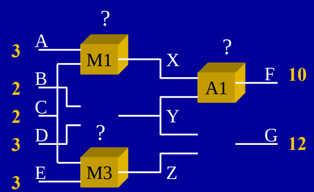
Adder(i):
- G(i):
  Out(i) = In1(i)+In2(i)
- U(i):



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.

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.

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.

Encoding Models In Propositional Logic

And(i):

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

Or(i):

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

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

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

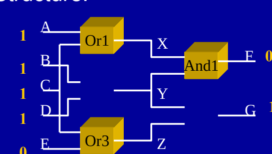
Summary: Consistency-based Diagnosis

- Component Model + Structure:

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 C



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 $D_i \wedge Obs \wedge C(X,Y)$ is satisfiable.

Outline

Model-based Diagnosis

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

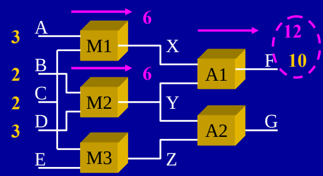
Diagnosis by Divide and Conquer

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)

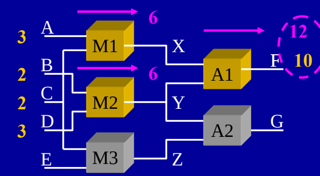
General Diagnostic Engine
[de Kleer & Williams, 87]

Conflicts Explain How to Remove Symptoms



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

Conflicts Explain How to Remove Symptoms



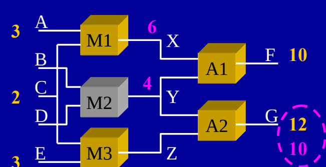
Symptom:
F is observed 10, but should be 12 if A1, M1 & M2 are okay.

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

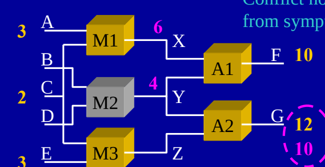
Find Another Symptom



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

... and its Conflict

Conflict not just upstream from symptom

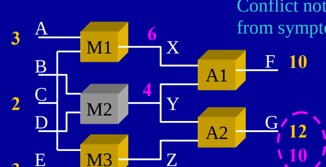


Symptom:
G is observed 12, but should be 10

Conflict: A1=G & M2=G & M1=G & M3=G is inconsistent

... and its Conflict

Conflict not just upstream from symptom

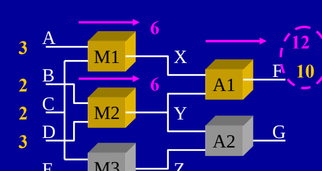


Symptom:
G is observed 12, but should be 10

Conflict: A1=G & M2=G & M1=G & M3=G is inconsistent

A1=U or A2=U or M1=U or M3=U removes conflict

Summary: Conflicts



Conflict:

A set of component modes M that are inconsistent with the model and observations.

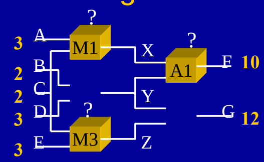
Properties:

- 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

Summary: Kernel Diagnoses

Kernel Diagnosis

= {A2=U & M2=U}



Partial Diagnosis: A set of component modes M all of whose extensions are diagnoses.

- M removes all symptoms
- M entails Model & Obs *(implicant)*

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

{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=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 = {A1=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

Kernel Diagnoses = {M1=U}
{A1=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

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

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

"Smallest" sets of modes that remove all conflicts

Single Fault Diagnoses are the Intersection of All 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.

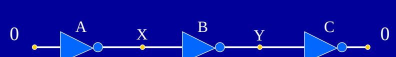
A1=U or A2=U or M1=U or M3=U removes conflict 2

Single Fault Diagnoses = {A1=U, M1=U}

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



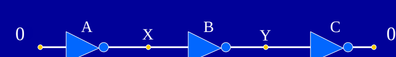
Inverter(i):
- G(i): Out(i) = not(In(i))
- U(i):

- Isolates surprises
- Doesn't explain

Nominal and Unknown Modes

Notational Note:
G(i) ≡ [i = G]

Diagnosis With Only the Known



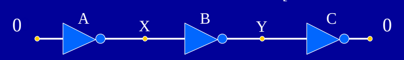
Inverter(i):
- G(i): Out(i) = not(In(i))
- S1(i): Out(i) = 1
- S0(i): Out(i) = 0

- No surprises
- Explains

Exhaustive Fault Modes

Solution: Diagnosis as Estimating Behavior Modes

Sherlock
[de Kleer & Williams, IJCAI 89]



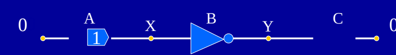
Inverter(i):
- G(i): Out(i) = not(In(i))
- S1(i): Out(i) = 1
- S0(i): Out(i) = 0
- U(i):

• Isolates surprises
• Explains

Nominal, Fault and Unknown Modes

Example Diagnoses

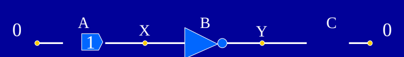
Sherlock
[de Kleer & Williams, 89]



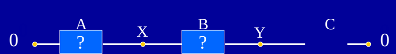
Diagnosis: [S1(A),G(B),U(C)]

Example Diagnoses

Sherlock
[de Kleer & Williams, 89]

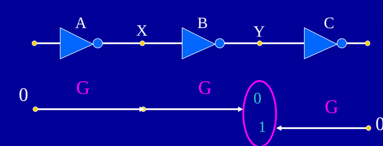


Diagnosis: [S1(A),G(B),U(C)]



Kernel Diagnosis: [U(C)]

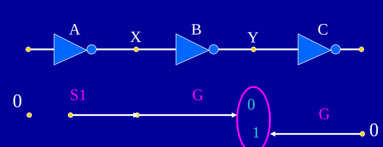
1. Find Symptoms & Conflicts



Conflict:

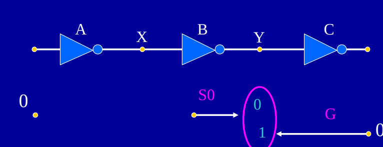
not [G(A), G(B) and G(C)]

More Symptoms & Conflicts



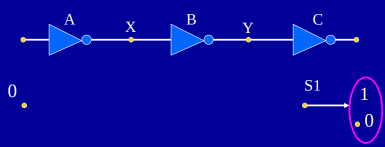
Not [S1(A), G(B), and G(C)]

More Symptoms & Conflicts



not [S0(B) and G(C)]

More Symptoms & Conflicts



not S1(C)

All Conflicts

- < S1(C) >
- < S0(B), G(C) >
- < S1(A), G(B), G(C) >
- < G(A), G(B), G(C) >

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

- [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)]

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)]

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. 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)]

3. Generating Kernel Diagnoses

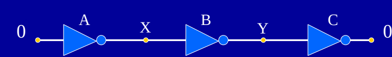
- [G(C),S0(C),U(C)]
- [G(B),S1(B),U(B),S1(C),S0(C),U(C)]
- [G(A),S0(A),U(A),S1(B),S0(B),U(B),S1(C),S0(C),U(C)]
- [S1(A),S0(A),U(A),S1(B),S0(B),U(B),S1(C),S0(C),U(C)]

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

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



Diagnoses: (42 of 64 candidates)

Fully Explained Failures
- [G(A),G(B),S0(C)]
- [G(A),S1(B),S0(C)]
- [S0(A),G(B),G(C)]

. . .

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

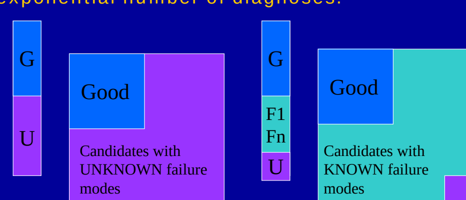
. . .

Fault Isolated, But Unexplained
- [G(A),G(B),U(C)]
- [G(A),U(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

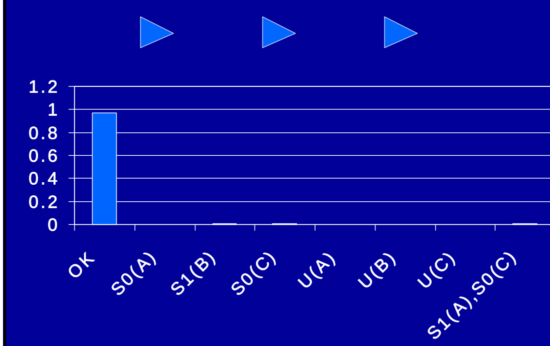
	A	B	C
p(G)	.99	.99	.99
p(S1)	.008	.008	.001
p(S0)	.001	.001	.008
p(U)	.001	.001	.001

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

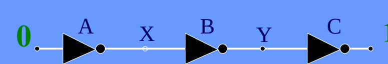
$$p(c \mid x = v) = \frac{p(x = v \mid c)\,p(c)}{p(x = v)}$$

Bayes' Rule

Normalization Term

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

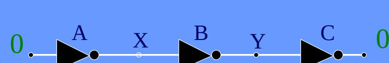
- If previous obs, c and Phi entails x = v
  Then p(x = v | c) = 1
- If previous obs, c and Phi entails x <> v
  Then p(x = v | c) = 0
- If Phi consistent with all values for x
  Then p(x = v | c) is based on priors
  - E.g., uniform prior = 1/m for m possible values of x



Observe out = 1:

$$p(c \mid x = v) = \frac{p(x = v \mid c)\,p(c)}{p(x = v)}$$

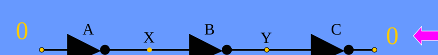
- C = [G(A),G(B),G(C)]
- Prior: P(C) = .97
- P(out = 1 | C) = ?
- = 1
- P(C | out = 0) = ?
- = .97/p(x=v)



Observe out = 0:

$$p(c \mid x = v) = \frac{p(x = v \mid c)\,p(c)}{p(x = v)}$$

- C = [G(A),G(B),G(C)]
- P(C) = .97
- P(out = 0 | C) = ?
- = 0
- P(C | out = 0) = ?
- = 0 x .97/p(x=v) = 0

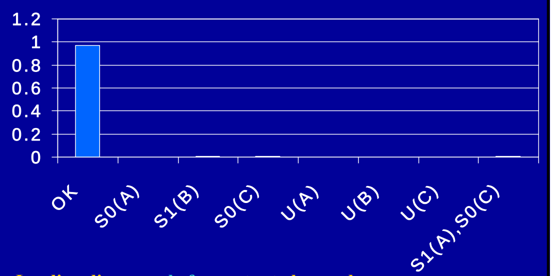


Example: Tracking Single Faults
- which are eliminated?
- which predict observations?
- Which are agnostic?

Priors for Single Fault Diagnoses:

	A	B	C
p(S1)	~~.008~~	$\boxed{.008}$	~~.001~~
p(S0)	$\boxed{.001}$	~~.001~~	$\boxed{.008}$
p(U)	.001	.001	.001

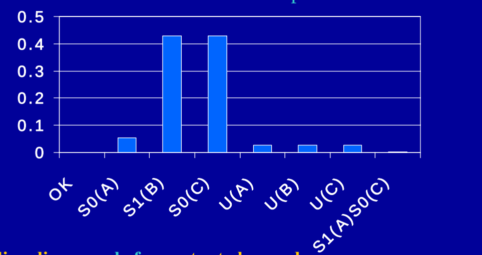
0



Leading diagnoses before output observed

0 0

Top 6 of 64 = 98.6% of P



Leading diagnoses before output observed

Summary: Candidate Probabilities

$$p(c) = \prod_{m \in c} p(m)$$

Assume Failure Independence

$$p(c \mid x = v) = \frac{p(x = v \mid c)\,p(c)}{p(x = v)}$$

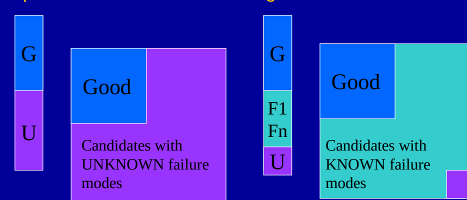
Bayes' Rule

Normalization Term

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 = 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