BASICS OF DEFAULT LOGIC (NON-MONOTONIC REASONING)

What is non-monotonic reasoning

Some approaches to non-monotonic reasoning

Some problems with non-monotonic reasoning

Consistency-based diagnosis

NON-MONOTONIC REASONING

Motivations

- **Decisions with incomplete knowledge**
- **Rules with exceptions**
- **Handling inconsistent information**

Monotonicity

$$A \vdash q \Rightarrow A \cup \{p\} \vdash q$$

Types of non-monotonic reasoning

- **Default reasoning – rules with exceptions**
- **Autoepistemic reasoning – knowledge about knowledge and non-knowledge**
- **Reasoning on the basis of communication conventions**

SOME BASIC PROBLEMS (1)

Closed world assumption (CWA)

p is derivable from T under CWA means that

T ∪ {¬q | q atomic and not T |- q} |- p

Problem with inconsistent CWAs, e.g., T = {a ∨ b}

Frame problem

Specifying what does not change when an event occurs

Using persistence defaults

SOME BASIC PROBLEMS (2)

Qualification problem

Defaults for describing normal effects of actions

Negation as failure

flies(_x) :- bird(_x), not abnormal(_x).
abnormal(_x) :- penguin(_x).
bird(tweety).
flies(tweety)? ⇒ SUCCESS

If we add penguin(tweety).

flies(tweety)? ⇒ FAIL

PRINCIPLED APPROACHES

	Representation of defaults	*Derivation techniques*
Reiter's default logic	**Non-classical inference rules**	**Fixpoint construction to define theorems**
Modal approach (McDermott, Doyle, Moore)	**Modal operator for beliefs**	
Circumscription (McCarthy, Lifschitz)	**Validity in minimal models**	**Elimination of uninteresting models by axiom schema**
Inconsistency-tolerant reasoning	**Inconsistent sets of premises**	**Derivation w.r.t. preferred sets of premises**

REITER'S DEFAULT LOGIC

Components

Defaults theorys is a pair (D,W)

- **W: set of facts (formulas in 1. order pred. logic)**
- **D: set of "inference rules" (defaults)**

$$\mathbf{A:B_1,\ldots B_n/C}$$

If A derivable, and $\neg B_i$ not, then infer C

Example

W = {penguin(x) ⇒ abnormal(x), bird(tweety)}

D = {bird(x): abnormal(x)/flies(x)}

Default theories generate *extensions*

EXTENSIONS IN REITER'S DEFAULT LOGIC

Desirable properties of extensions

1 contain W

2 deductively closed (classically)

3 uses as many defaults as possible

4 contain no unjustified facts

Fixpoint construction

$\mathbf{E_0 = W}$

$\mathbf{E_{i+1} = Th(E) \cup \{c \mid a{:}b_1,\ldots b_n/c \text{ in } D, a \text{ in } E_i, \neg b_i \text{ not in } E\}}$

$\mathbf{E \text{ is an extension of } (D,W) \text{ iff } E = \cup E_i}$

PROPERTIES OF EXTENSIONS

Building extensions

Theories may have several extensions

Example: $W = \{p \vee q\}$ $D = \{\frac{:\neg p}{\neg p}, \frac{:\neg q}{\neg q}\}$ $E_1=\{p,\neg q\}$, $E_2=\{\neg p,q\}$

Automated construction process in the general case NP hard

Position towards extensions

***Credulous* reasoner – Valid in at least one extension**

***Sceptical* reasoner – Valid in at all extensions**

A PROBLEM WITH DEFAULT LOGIC - ALTERNATIVES

Too weak: Case distinctions

Emu:runs	**Ostrich:runs**	**Emu ∨ Ostrich**
runs	**runs**	

runs derivable?

A PROBLEM WITH DEFAULT LOGIC - ALTERNATIVES

Too weak: Case distinctions

<u>Emu:runs</u> **<u>Ostrich:runs</u>** **Emu ∨ Ostrich**
runs **runs**

runs derivable? – No.

A PROBLEM WITH DEFAULT LOGIC - ALTERNATIVES

Too weak: Case distinctions

Emu:runs **Ostrich:runs** **Emu ∨ Ostrich**
runs **runs**

runs derivable – No.

Too strong– No global consistency

:usable(x)&¬broken(x) **broken(l-arm) ∨ broken(r-arm)**
usable(x)

usable(l-arm) & usable(r-arm) derivable?

A PROBLEM WITH DEFAULT LOGIC - ALTERNATIVES

Too weak: Case distinctions

<u>Emu:runs</u> **<u>Ostrich:runs</u>** **Emu ∨ Ostrich**
runs **runs**

runs derivable? – No.

Too strong– No global consistency

<u>:usable(x)&¬broken(x)</u> **broken(l-arm) ∨ broken(r-arm)**
usable(x)

usable(l-arm) & usable(r-arm) derivable? – Yes.

FLOATING CONCLUSIONS (Horty)

- **Witness John says: the suspect *shot* the victim to death**
- **If a witness says P then usually P is the case**
- **So, the suspect *shot* the victim to death**
- **So, the suspect *killed* the victim**

- **Witness Bob says: the suspect *stabbed* the victim to death**
- **If a witness says P then usually P is the case**
- **So, the suspect *stabbed* the victim to death**
- **So, the suspect *killed* the victim**

Is the conclusion warranted?

FLOATING CONCLUSIONS (2)

D1: People live where they work.

D2: People live where their partners live.

W1: Jan works in Amsterdam.

W2: Jan's wife Mary works in Arnheim.

Where do Jan and Mary live?

ANOTHER EXAMPLE

- **American civil law:**
  evidence has to prove claim "on the balance of probabilities
- **(Imaginary) statistics:**
  51% of American husbands commit adultey within 10 years
- **Mary has been married to John for 10 years:**
  Can she sue John for divorce?

  Is the conclusion warranted?

A PROBLEM WITH DEFAULT LOGIC - PRIORITIES

Priorities among defaults required

hurts(x,y) ⇒ guilty(x)

hurts(x,y) & notwehr(x) ⇒ ¬guilty(x)

attacks(x,y) ⇒ notwehr(x)

hurts(Peter,Hans)

attacks(Hans,Peter)

derivable: guilty(Peter), ¬guilty(Peter), ¬notwehr(Peter)

MODELING ACTIONS

Situation calculus (McCarthy, Hayes) – components

- ***Situations* – real world snapshots, causes of actions**
- ***Fluents* – time-dependent properties**
- ***Actions* – lead from situations to successor situations**

Holds(f,s)* – atomic formula: fluent *f* true in situation *s

Result(a,s)* – term: situation after executing action *a* in *s

Example – Blocksworld

1) Holds(On(C,Table),S0)
2) Holds(On(B,C),S0)
3) Holds(On(A,B),S0)
4) Holds(On(D,Table),S0)
5) Holds(Clear(A),S0)
6) Holds(Clear(D),S0)
7) Holds(Clear(Table),S0)

move(x,y) move x on top of y

8) Holds(Clear(x),s) & Holds(Clear(y),s) & x≠ y & x ≠ Table ⇒ Holds(On(x,y),Result(Move(x,y),s)

9) Holds(Clear(x),s) & Holds(Clear(y),s) & Holds(On(x,z),s) & x≠ y & y ≠ z ⇒ Holds(Clear(z),Result(Move(x,y),s)

YALE SHOOTING PROBLEM

Example – Definitions

1) S1 = Result(LOAD,S0)

2) S2 = Result(WAIT,S1)

3) S3 = Result(SHOOT,S2)

4) Holds(Alive,S0)

5) ∀s Holds(LOADED,RESULT(LOAD,s))

6) ∀s Holds(LOADED,s) ⇒
¬Hold2s(ALIVE,Result(SHOOT,s))

7) ∀s ¬Holds(LOADED,Result(SHOOT,s))

8) [¬]Holds(f,s): [¬]Holds(f, Result(e,s)) /
[¬]Holds(f, Result(e,s))

YALE SHOOTING PROBLEM

Example – Definitions

1) S1 = Result(LOAD,S0)

2) S2 = Result(WAIT,S1)

3) S3 = Result(SHOOT,S2)

4) Holds(Alive,S0)

5) ∀s Holds(LOADED,RESULT(LOAD,s))

6) ∀s Holds(LOADED,s) ⇒
¬Hold2s(ALIVE,Result(SHOOT,s))

7) ∀s ¬Holds(LOADED,Result(SHOOT,s))

8) [¬]Holds(f,s): [¬]Holds(f, Result(e,s)) /
[¬]Holds(f, Result(e,s))

	S0	S1	S2	S3
1.	**ALIVE**	**ALIVE**	**ALIVE**	**¬ALIVE**
		LOADED	**LOADED**	**¬LOADED**

Apply persistence default chronologically

YALE SHOOTING PROBLEM

Example – Definitions

1) S1 = Result(LOAD,S0)
2) S2 = Result(WAIT,S1)
3) S3 = Result(SHOOT,S2)
4) Holds(Alive,S0)

5) ∀s Holds(LOADED,RESULT(LOAD,s))
6) ∀s Holds(LOADED,s) ⇒
¬Hold2s(ALIVE,Result(SHOOT,s))
7) ∀s ¬Holds(LOADED,Result(SHOOT,s))
8) [¬]Holds(f,s): [¬]Holds(f, Result(e,s)) /
[¬]Holds(f, Result(e,s))

	S0	S1	S2	S3
1.	ALIVE	ALIVE	ALIVE	¬ALIVE
		LOADED	LOADED	¬LOADED

Apply persistence default chronologically

	S0	S1	S2	S3
2.	ALIVE	ALIVE	ALIVE	ALIVE
		LOADED	¬LOADED	¬LOADED

Apply persistence default for ALIVE up to S3, then derive ¬LOADED in S2 (6, 8)
Unintended – gun unloads mysteriously

YALE SHOOTING PROBLEM – ASSESSMENT

Early simple (naive?) formalization

- **Based on minimizing the changes**
- **Changes in the fluents over time are as minimal as possible**

A severe obstacle to the use of logic for formalizing dynamical scenarios

Better formalizations provide solutions

- **Predicate completion in the specification of actions: according to this solution, the fact that shooting causes Fred to die is formalized by the preconditions: alive and loaded, and the effect is that alive changes value**
- **Erik Sandewall includes a new condition of occlusion, which formalizes the "permission to change" for a fluent. What is minimized is not the set of changes, but the set of occlusions being true.**

Several authors obtain prices and publish in AI journal

PROBLEM WITH GLOBAL INCONSISTENCY

Properties of extensions

- **Consistency of applied default in single extension**
- **Global consistency of defaults not guaranteed**
- **E.g., lottery paradox: a lot does'nt win |~ no lot wins**
- **Undesired and unintuitive results may occur**

Important property – cumulativity

- **Adding a theorem does'nt change derivable formulas**
- **If X |~ a then X {a} |~ b if and only if X |~ b**
- **Essential if inferencing meant as 'making explicit'**

Counterexample

1) true:p/p

2) p ∨ q:¬p/¬p

Extension contains p, hence p ∨ q

Adding p ∨ q to the premises yields an extension ¬p

OVERCOMING GLOBAL INCONSISTENCY

Extensions – Brewka's cumulative default logic

- **Logic based on assertions (p,Q), with p being a formula and Q a set of consistency constraints**

Also possible – restrictions on feasible defaults (disallowing disjunctions)

CONSISTENCY-BASED DIAGNOSIS

Information

- **Set of components (K)**
- **Effect of components if error-free (model M)**
- **Observations (N)**

Explanation for O, i.e., minimal set of components K',

such correctness of K \ K' is consistent with M ∪ O

Default theory

K' is diagnosis iff

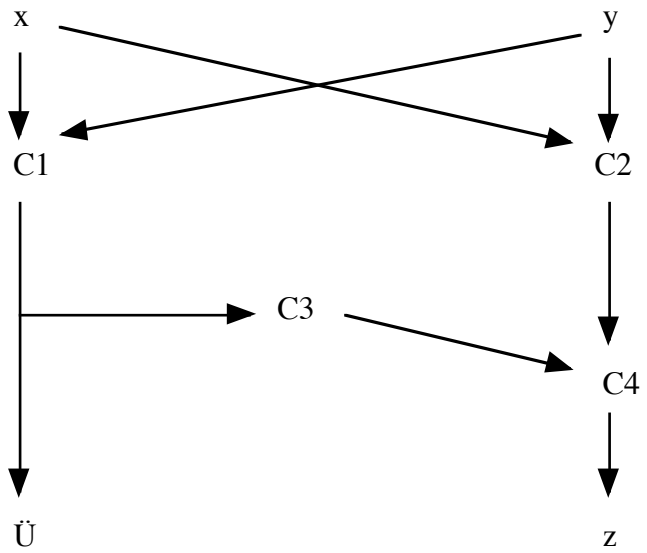
there is an extension E of default theory (D,W) with

- **W = M ∪ O**
- **D = {true: OK(x)/OK(x) | x in K}**
- **¬OK(x) for all k in K'**

Well suited for error diagnosis (e.g., electronic circuits)

CONSISTENCY-BASED DIAGNOSIS

Example – semiadder



Types of components

ADD(C1), OR(C2), NOT(C3), AND(C4)

Topology

x = input1(C1) = input1(C2)
y = input2(C1) = input2(C2)
Ü = output(C1) = input(C3)
output(C3) = input1(C4)
output(C2) = input2(C4)
z = output(C4)

CONSISTENCY-BASED DIAGNOSIS (2)

Normal behavior

AND(x) ∧ OK(x) ⇒ [output(x) = 1 ⇔ (input1(x) = 1 ∧ input2(x) = 1)]

OR(x) ∧ OK(x) ⇒ [output(x) = 1 ⇔ (input1(x) = 1 ∨ input2(x) = 1)]

NOT(x) ∧ OK(x) ⇒ [output(x) = 1 ⇔ ¬input(x) = 1]

Defaults for correct behavior

true: OK(x)/OK(x) for all x = C1, … C4

Observations (example)

x =1, y = 1, z = 1

Extensions (only 1 component faulty)

OK(C1), OK(C2), OK(C3) ⇒ ¬OK(C4)

OK(C1), OK(C2), OK(C4) ⇒ ¬OK(C3)

OK(C1), OK(C3), OK(C4) ⇒ ¬OK(C2)

OK(C2), OK(C3), OK(C4) ⇒ ¬OK(C1)

CONSISTENCY-BASED DIAGNOSIS (3)

Addidional observation (example)

output(C3) = 1
OK(C1), OK(C2), OK(C4) ⇒ ¬OK(C3)
OK(C2), OK(C3), OK(C4) ⇒ ¬OK(C1)

Alternative observation (example)

output(C3) = 0
OK(C1), OK(C2), OK(C3) ⇒ ¬OK(C4)

Additional domain knowledge – Some components are more reliable than others

Iterative building of extensions (D_1, …, D_{n-1}, W)
E extension of (D_1, …, D_n, W) iff
n = 1 and E is extension of (D_1, W)
n > 1 and there is extension E' of (D_1, …, D_{n-1}, W) such that E is extension of (D_n, E')

***Example* – NOT component more reliable than others**

Default theory (D_1, D_2, W)
D_1 = {:OK(C3)/OK(C3)}
D_3 = {:OK(Ci)/OK(Ci)} i = 1, 2, or 4
W = M ∪ O
Yields only extensions with OK(C3)

EXTENSIONS FOR INCOMPLETE OBSERVATIONS (Havel 2010)

Assumptions

Models may be complex
Not all components (easily) observable
Hierarchical models of systems are meaningful

Motivation

Assessing the competence of diagnosis with limited resources
Simplification of the diagnostic model

Ideas

Assessing components as observable or not (given set of observable points)
Abstracting away components that cannot be diagnosed into compound items
Future idea – segmenting the model into parts, solving by "divide-and-conquer"

THE NOTION OF DIAGNOSIBILITY

A definition which is close to the idea of component diagnosability

We say that a system is diagnosable with a given set of sensors iff

(i) for any relevant combinations of sensor readings
there is only one minimal candidate diagnosis and

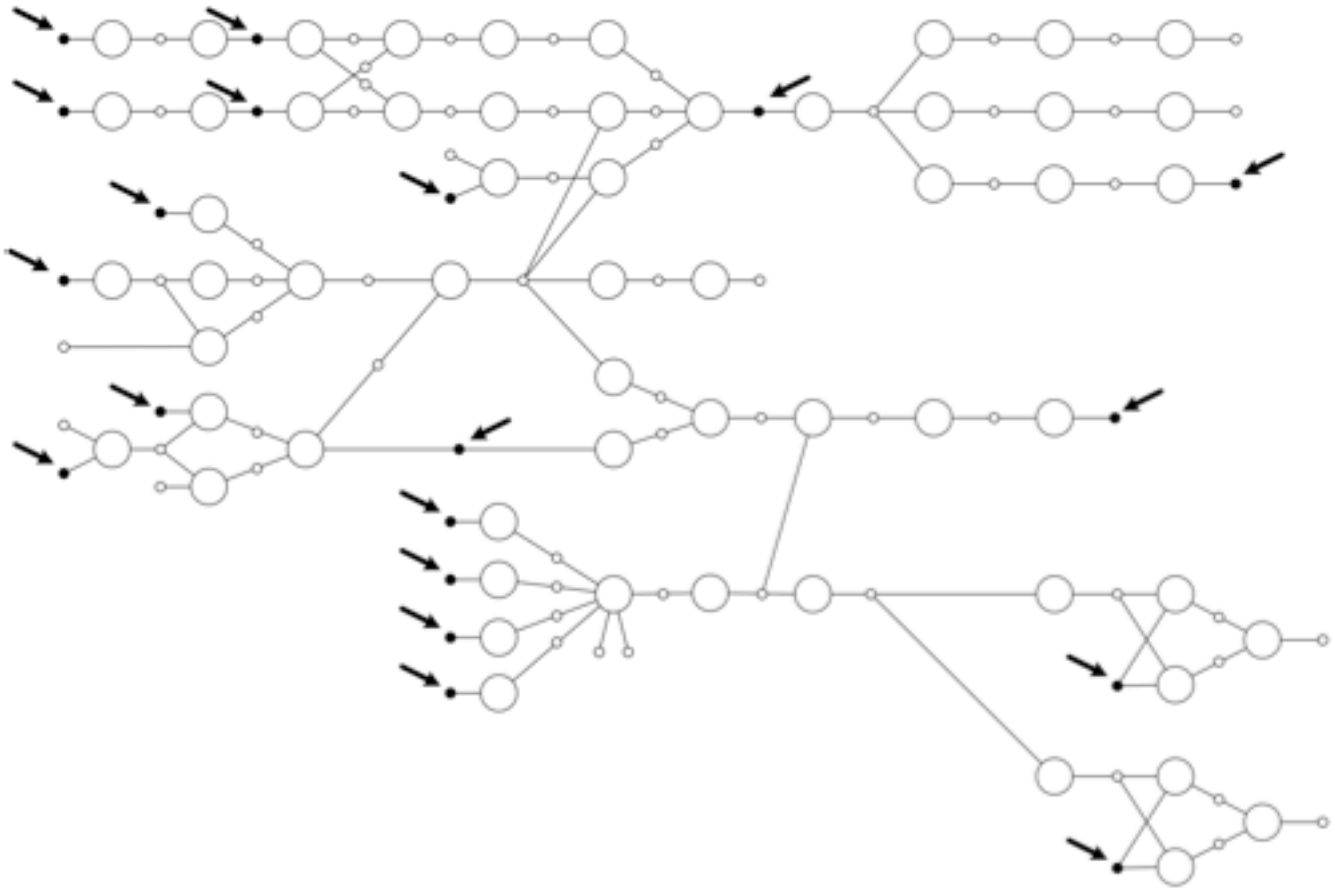
(ii) all faults of the system correspond to
a candidate diagnosis for some sensor reading

e.g., an AND-gate is observable on the basis of its output and one of its inputs

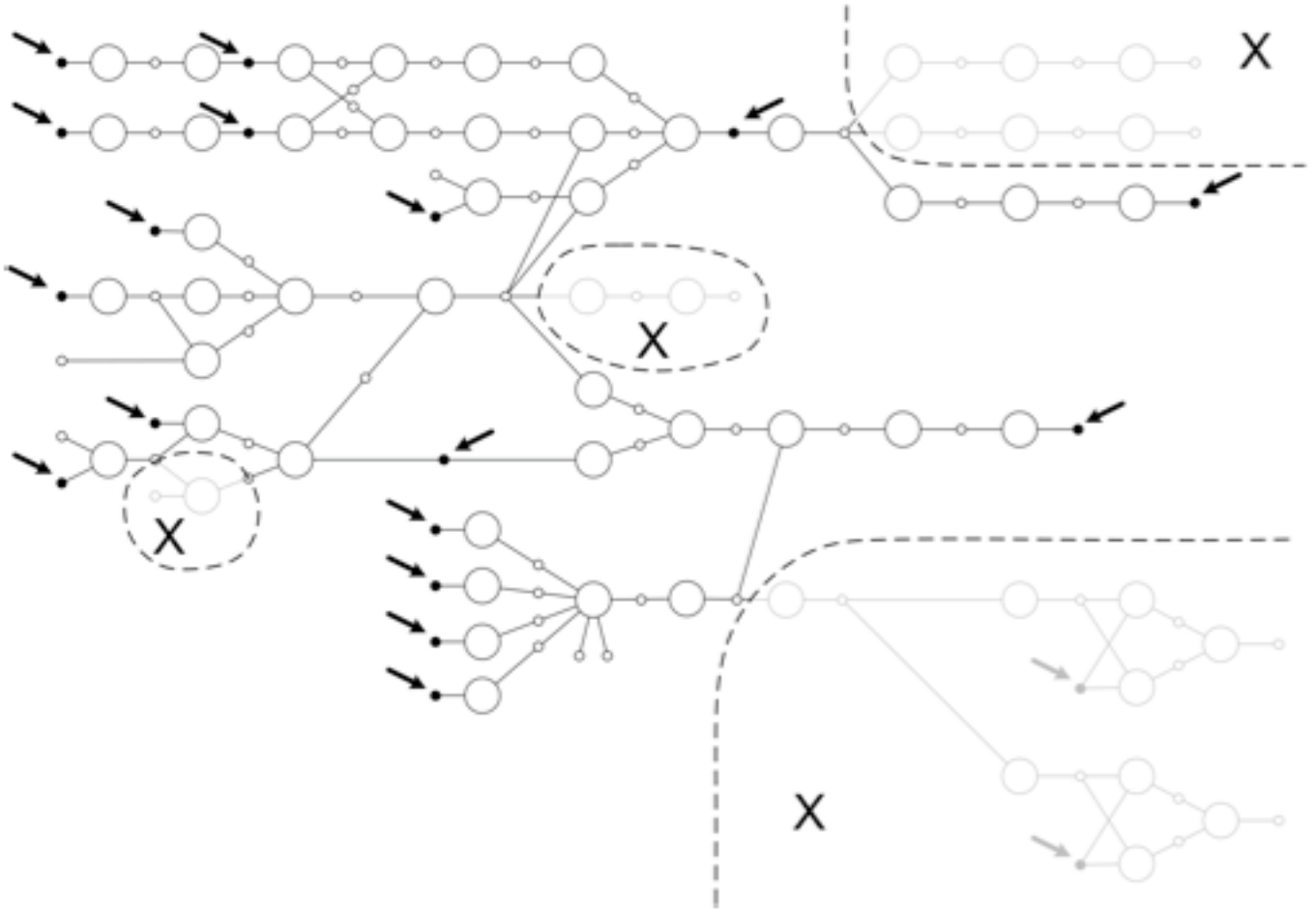
Suppose we plan to examine a system
that is expected to be observed only partially

The proposed model simplification procedure attempts to
reduce the size of the diagnostic model of a system
while preserving the diagnostic power –
with respect to the expected observation conditions

A DIAGNOSTIG MODEL BEFORE SIMPLIFICATION



A DIAGNOSTIG MODEL AFTER SIMPLIFICATION



COMPONENT DIAGNOSABILITY ALGORITHM

Input: a component description Δ in CNF, a set of health variables $\mathbf{A}$, a set of observed ports $\mathbf{P}_o$, and a set of unobserved ports $\mathbf{P}_u$.
Output: decision whether the component is diagnosable.

1. **For** each assumable A from $\mathbf{A}$ do: $\Delta' = \Delta \cup \{A\}$.
2. Create ordering $o = (O_1, \ldots, O_k, U_1, \ldots, U_l, A_1, \ldots, A_m)$, where $O_1, \ldots, O_l$ are members of $\mathbf{P}_o$, $U_1, \ldots, U_l$ are members of $\mathbf{P}_u$ and $A_1, \ldots, A_m$ members of $\mathbf{A}$ in arbitrary order.
3. Perform **directional resolution** on theory Δ', ordering o, and only over variables in $\mathbf{A} \cup \mathbf{P}_u$.
4. **If** the result is 'Δ is unsatisfiable',
   return 'component behavior is faulty by nature'
else
   store the *buckets*.
5. **If** $\forall i, i \in (1 \ldots Card(\mathbf{P}_o)) \quad bucket_i = \emptyset$,
   return 'component is not diagnosable'
else
   return 'component is diagnosable'.

MODEL PRUNING ALGORITHM (1)

Input: a set of components $\mathbf{C}$, a component description Δ^c and a set of assumables $\mathbf{A}^c$ for each $c \in \mathbf{C}$, a set of nodes $\mathbf{N}$, a set of observed nodes $\mathbf{N_o} \subseteq \mathbf{N}$, and projections $\varepsilon^c : \mathbf{P} \rightarrow \mathbf{N}$ of the component ports onto nodes.
Output: simplified model of the system.

1. **Initialize:**
   Assign an initial state to every node, *'bound'* if the node is physically observed, *'unknown'* otherwise.
   For each node n in the model **do**:
   if $state(n) \neq$ *'bound'* and $c - card(n) = 1$ **then**
   set $state(n) =$ *'free'* and insert n into **DEL**.

MODEL PRUNING ALGORITHM (2)

2. **If DEL** is not empty **then**
   remove the first node n from **DEL** and go to step 3
   else
   stop and **return** the current model.

3. Find component c_n connected to node n.
   If two or more ports of c_n share a node **then**
   replace all with one substitutive port.
   For each port p of component c_n **do**:
   if $state(\varepsilon(p)) =$ *'free'* **then**
   insert p into the set of unobserved ports $\mathbf{P}_{\mathbf{u}}^{\mathbf{c_n}}$
   else
   insert p into the set of observed ports $\mathbf{P}_{\mathbf{o}}^{\mathbf{c_n}}$.

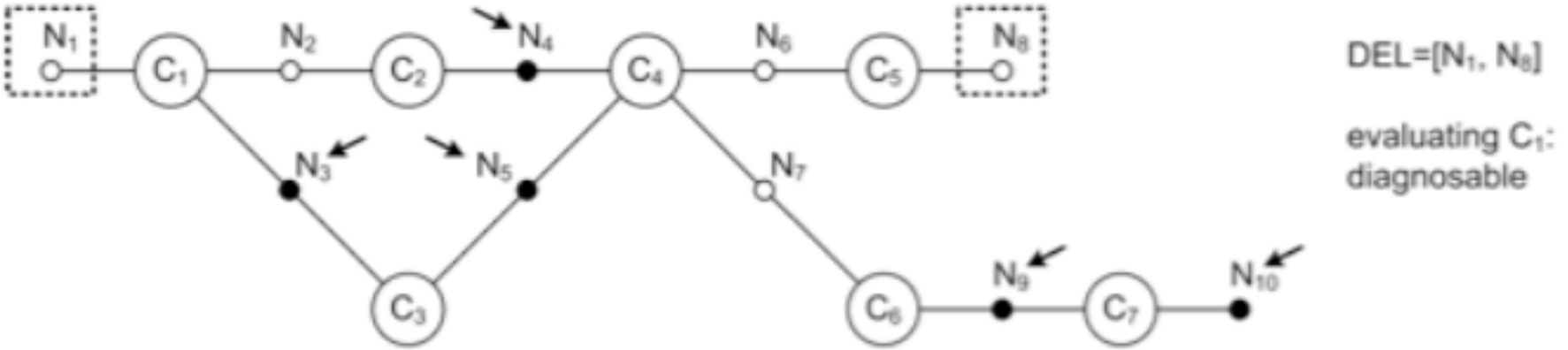
4. Decide **component diagnosability** of the component c_n, using component description Δ^{c_n}, set of health variables $\mathbf{H}^{\mathbf{c_n}}$, and sets of ports $\mathbf{P}_{\mathbf{o}}^{\mathbf{c_n}}$ and $\mathbf{P}_{\mathbf{u}}^{\mathbf{c_n}}$.

MODEL PRUNING ALGORITHM (3)

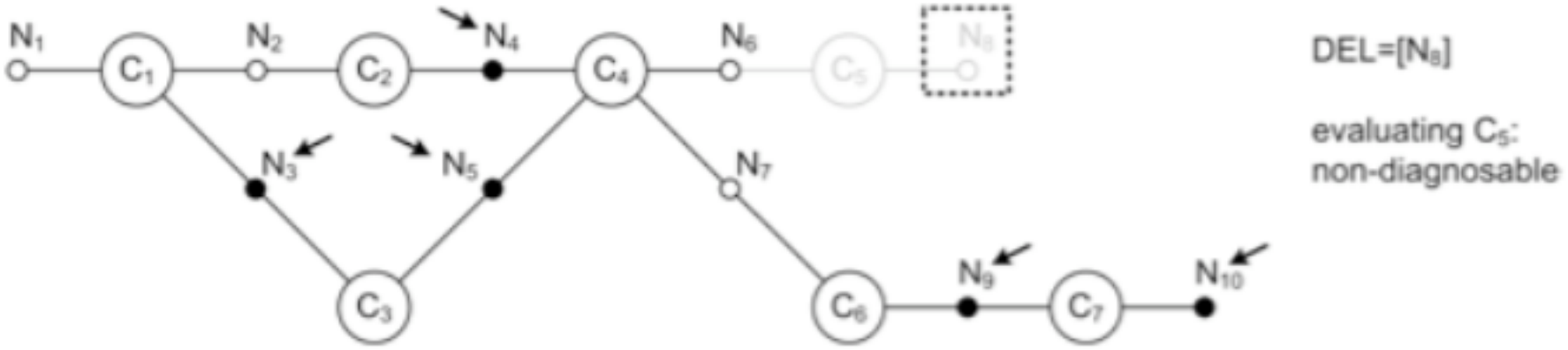
5. **If** the result is 'component is diagnosable' **then**
   set $state(n) =$ *'bound'* and go back to step 2.

6. Delete the component c_n and the node n from the model.
   For each other node n' previously connected to the component c_n **do**:
   if n' is in **DEL then**
   remove n' from **DEL** and delete it from the model
   else
   if $c - card(n') = 0$ **then** delete n' from the model
   if $c - card(n') = 1$ and $state(n') \neq$ *'bound'* **then**
   set $state(n') =$ *'free'* and insert n' into **DEL**.
   Go back to step 2.

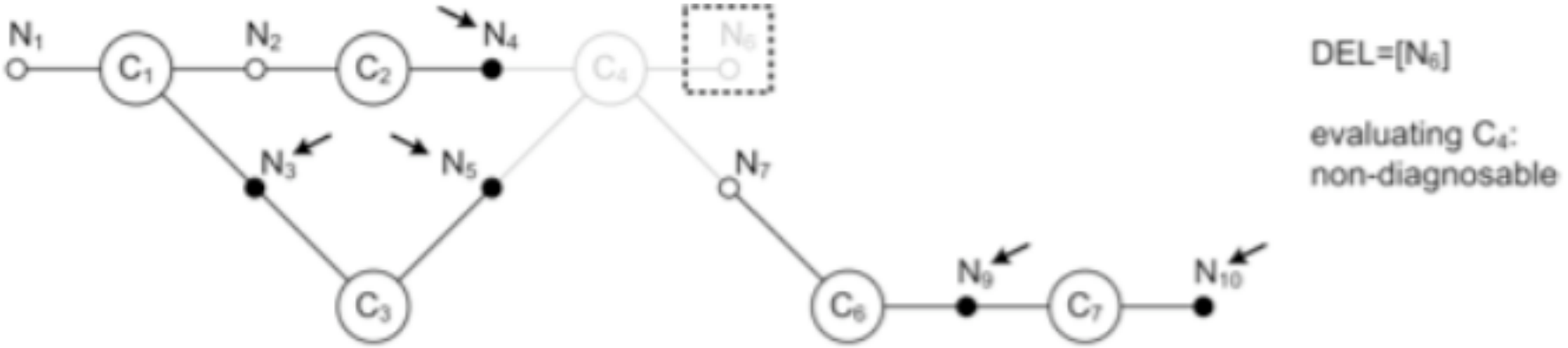
EXAMPLE MODEL PRUNING SEQUENCE (1)



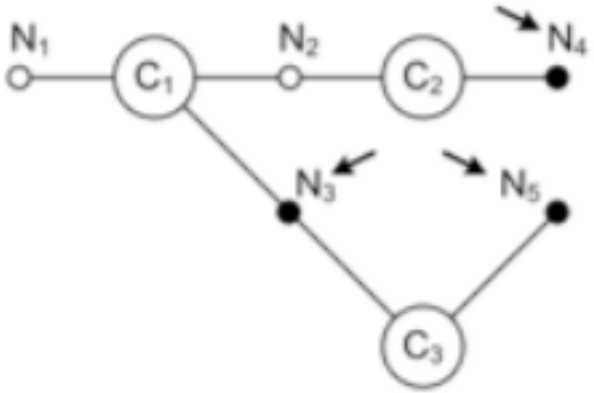
EXAMPLE MODEL PRUNING SEQUENCE (2)



EXAMPLE MODEL PRUNING SEQUENCE (3)



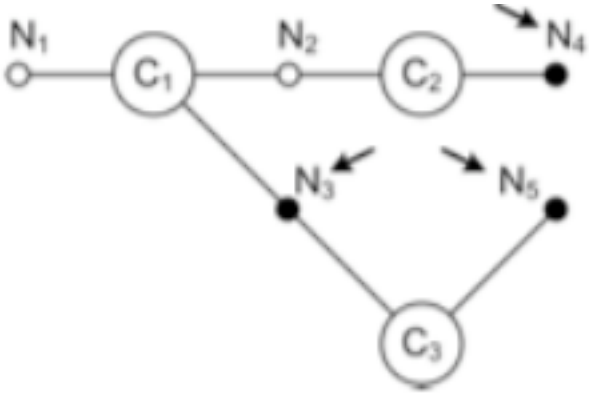
EXAMPLE MODEL PRUNING SEQUENCE (4)



DEL=[N_7]

evaluating C_6:
non-diagnosable

EXAMPLE MODEL PRUNING SEQUENCE (5)



DEL=[]

procedure stops

LIMITATIONS OF THE MODEL

The pruning algorithm examines only diagnosability of individual components

the only chance to remove any undiagnosable component is only

when at least one of its ports is a free dead-end

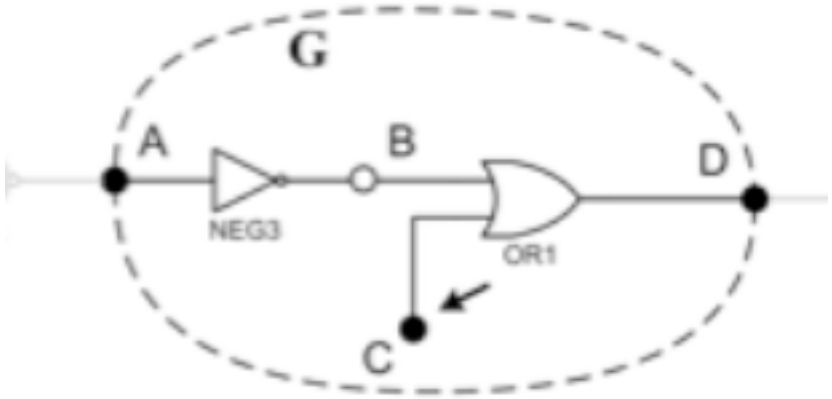
To cut the model not only from dead-ends but also from inside,

one must examine not only individual components but groups of components

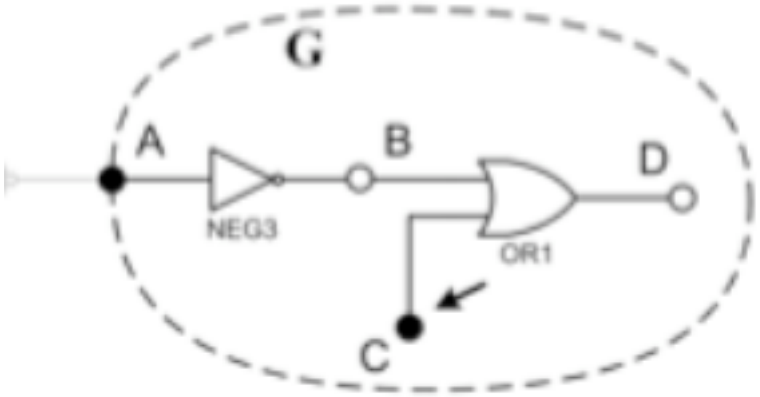
Nodes of a component group are classified into

- **inner nodes of the group,**
- **boundary nodes connected to both components in the group and outside the group, and**
- **outer nodes not incident to the group**

DIAGNOSABILITY OF COMPONENT GROUPS



diagnosable



non-diagnosable

ALGORITHM MODIFICATION FOR DIAGNOSABILITY OF COMPONENT GROUPS

The adaptation of the original component diagnosability algorithm to the component-group diagnosability algorithm comprises:

- **substitution of individual component port variables for corresponding system node variables,**
- **replacement of single component descriptions (behavior equations) with the group's joint description,**
- **consideration of the physically observed nodes and boundary nodes as observed for this evaluation,**
- **consideration of the unobserved inner nodes as unobserved for this evaluation, and finally**
- **performing directional resolution as it is done in the original algorithm.**

EXTENDED MODEL PRUNING ALGORITHM

Input: a model $\mathbf{M} = (\mathbf{C}, \mathbf{N}, \mathbf{\Delta}, \mathbf{A}, \mathbf{P}, \mathbf{E})$ and a set of observed nodes $\mathbf{N}_o \subseteq \mathbf{N}$.
Output: simplified model of the system.

1. Let $\mathbf{R} = \emptyset$ is the set of *removed* components.
2. **For** every component group $\mathbf{G}, \mathbf{G} \subseteq \mathbf{C}$, **do**
3. Let $\mathbf{G} = \mathbf{G} \setminus \mathbf{R}$ (ignore components already removed from the model)
   If $\mathbf{G} = \emptyset$ **then**
   continue with the next group.
4. **Decide component-group diagnosability** of $\mathbf{G}$, using component descriptions $\Delta^c; \forall c \in \mathbf{G}$, sets of assumables $\mathbf{A}^c; \forall c \in \mathbf{G}$, set of group's inner nodes $\mathbf{N}_i^{\mathbf{G}}$, set of group's boundary nodes $\mathbf{N}_b^{\mathbf{G}}$, set of group's observed nodes $\mathbf{N}_o^{\mathbf{G}} = \mathbf{N}_o \cap (\mathbf{N}_i^{\mathbf{G}} \cup \mathbf{N}_b^{\mathbf{G}})$, and projections $\varepsilon^c; \forall c \in \mathbf{G}$.
5. **If** the result is 'component-group is not diagnosable' **then**
   remove all components in $\mathbf{G}$ from the model and
   let $\mathbf{R} = \mathbf{R} \cup \mathbf{G}$.