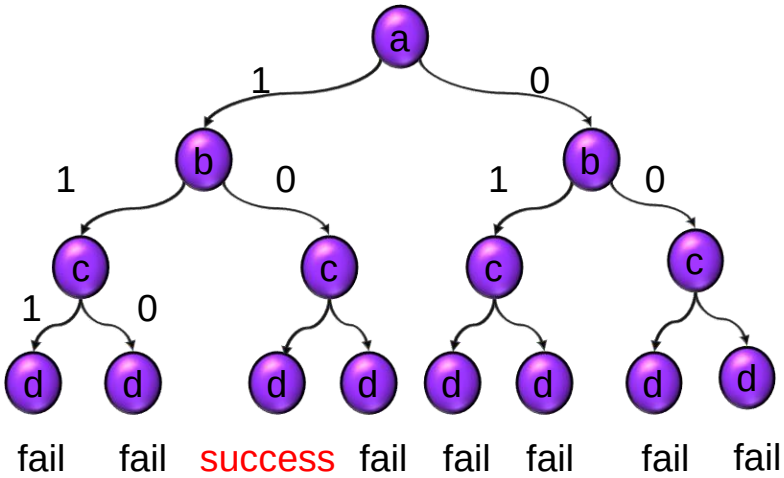
Combinational ATPG

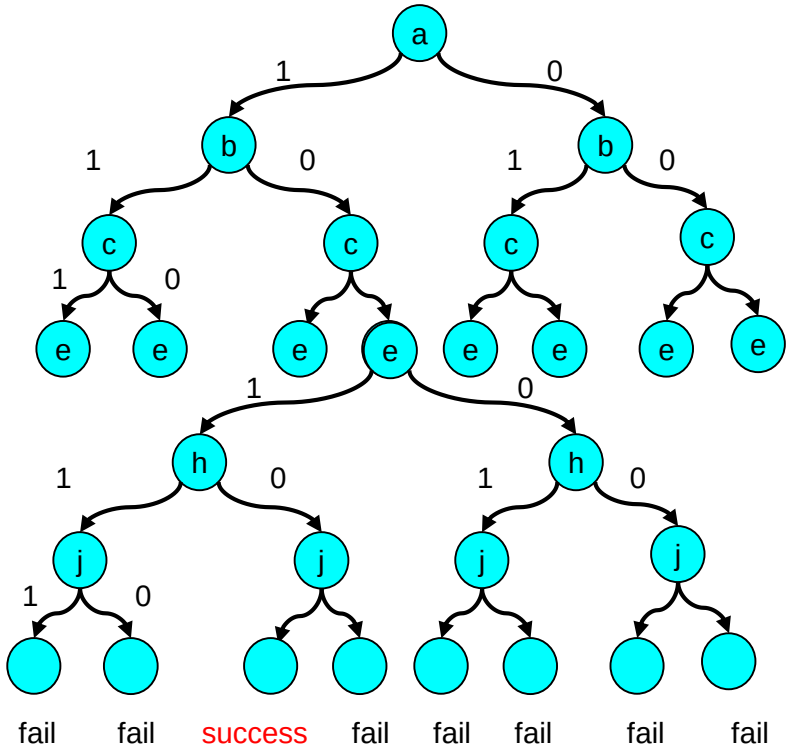
- **Deterministic Test Pattern Generation**
  - Boolean difference approach*
  - Path sensitization method**
  - D-Algorithm [Roth 1966] **
  - **PODEM [Goel 1981]****
    - **Idea**
    - **Heuristics**
    - **Algorithms**
    - **Summary**
  - FAN [Fujiwara 1983]**
  - SAT-based [Larrabee 1992]*

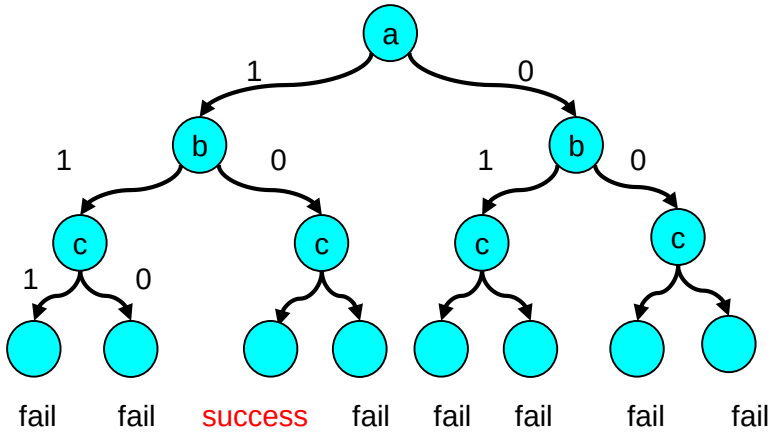


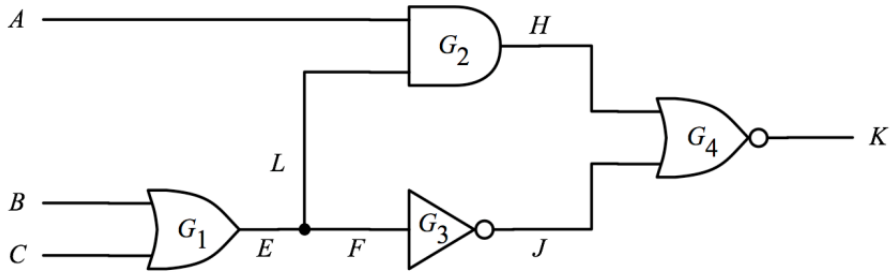
*Boolean-based methods
**path-based methods

$2^6 \rightarrow 2^3$

- D-alg. makes decision at intern nodes
  - 6 internal nodes
  - 2^6=64 decisions! large
- PODEM makes decision at PI
  - 3 PI
  - 2^3=8 decisions, smaller





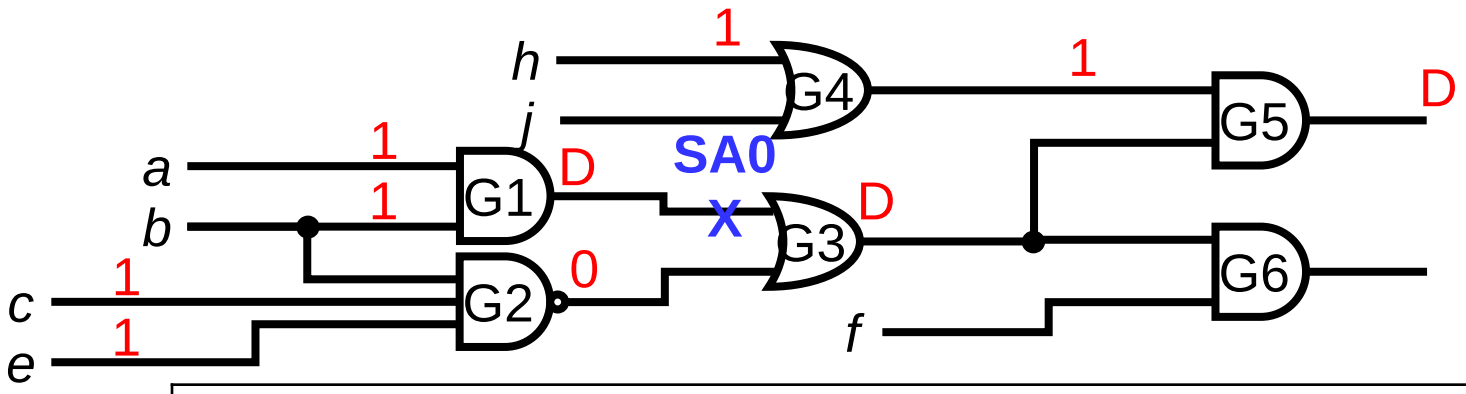


PODEM [Goel 1981]

- *Path Oriented Decision Making* (PODEM)

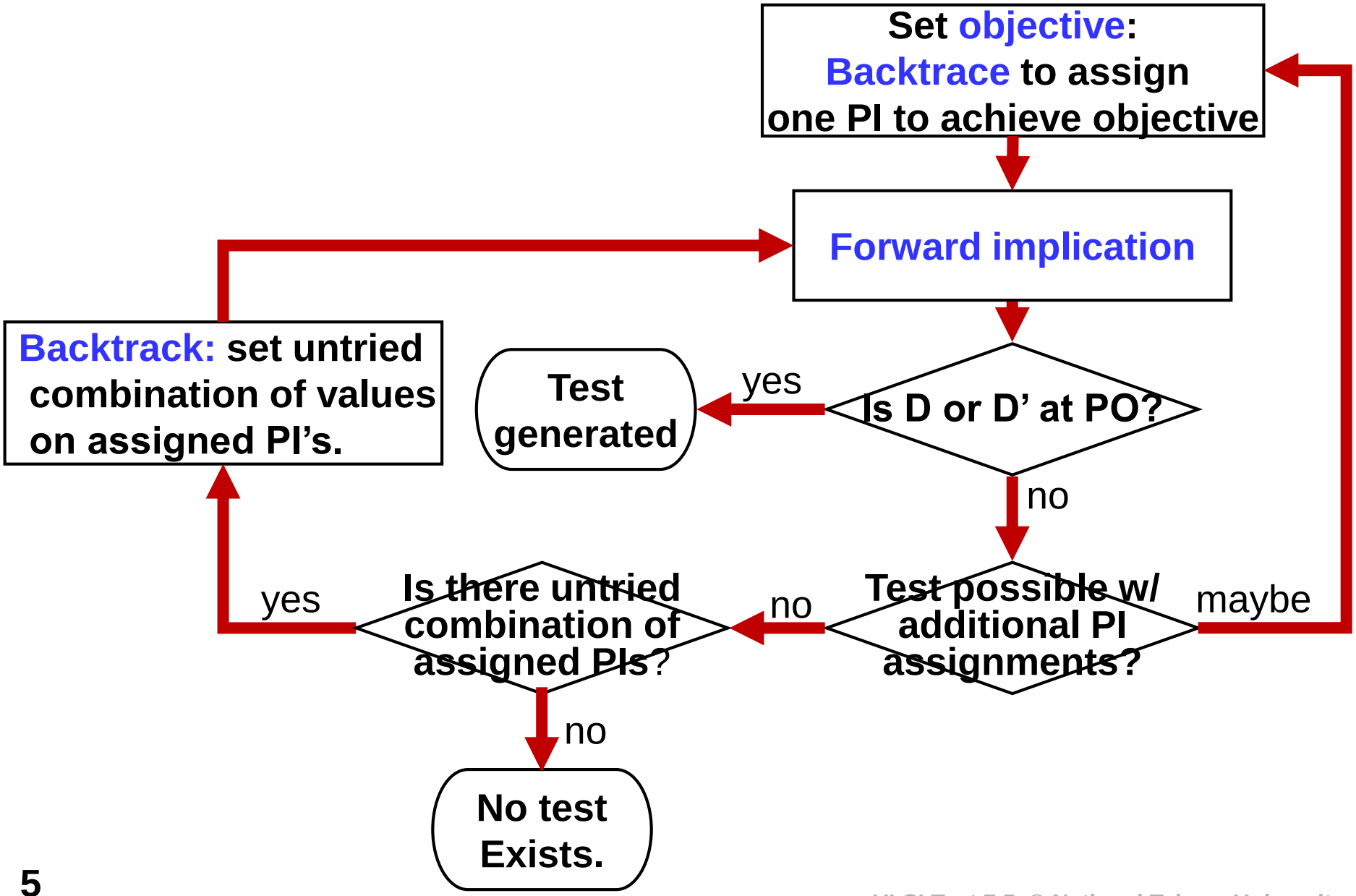
- IDEAS:
- 1. Only allow assignments to *PI only*
  - Doesn't assign internal nodes
  - Greatly reduces search tree
- 2. Assigned PI are then forward implication
  - No justification needed (Why ? FFT)
- 3. Flip last PI assignment when two conditions:
  - A. Fault not activated
  - B. No propagation path to any output

PODEM Example



Initial objective: G_1 =1
Backtrace to PI: *b* = 1. simulation, objective not achieved
Backtrace to PI: *a* = 1. simulation, objective achieved
Objective: G2 = 0 (propagate through G3)
Backtrace to PI: *C* = 1. simulation, objective not achieved
Backtrace to PI: *e* = 1. simulation, objective achieved
Objective: G4 = 1 (we choose to propagate through G5)
Backtrace to PI: h= 1. objective achieved.
Test Generated: abcehjf=11111XX

Flowchart of PODEM



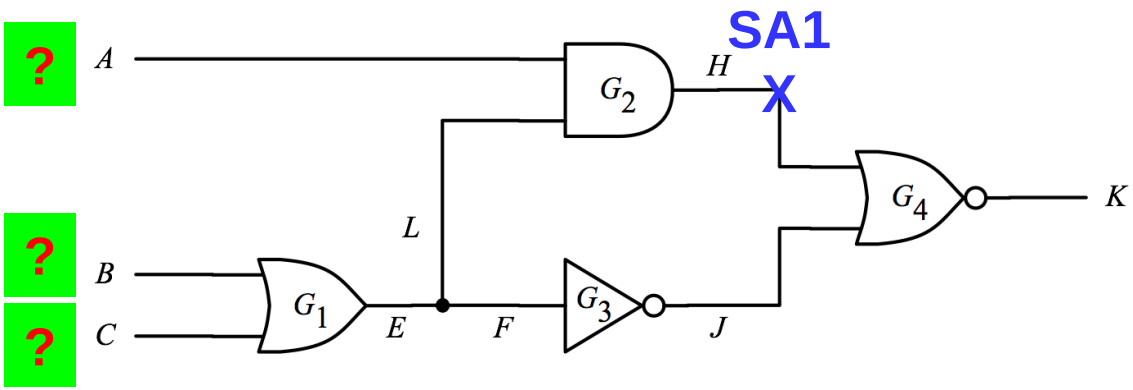
Components of PODEM

- Determine an *objective*
  - If D/D' has not appeared at the fault site,
    - set objective to activate fault
  - If D/D' has appeared
    - set objective to propagate fault effect
- Given objective, determine PI value
  - *Backtrace* to PI
  - *Backtrack* if conflict occurs

- NOTE: *Backtrace* is different from *backtrack*
  - Backtrace goes back to primary inputs of a certain signal
    - In netlist
  - Backtrack goes back to last decision
    - in decision tree

Quiz

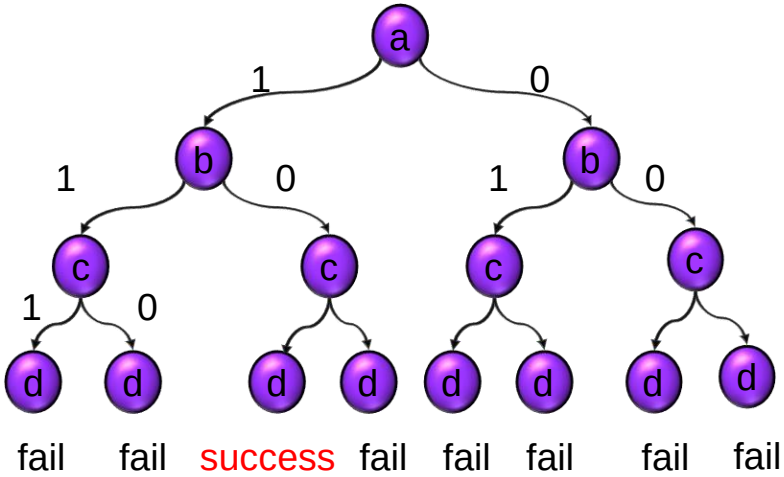
Q: Use PODEM to generate a test for H SA 1 fault please mark your
1. objective,
2. backtrace



Totally 2 objectives, 2 backtraces

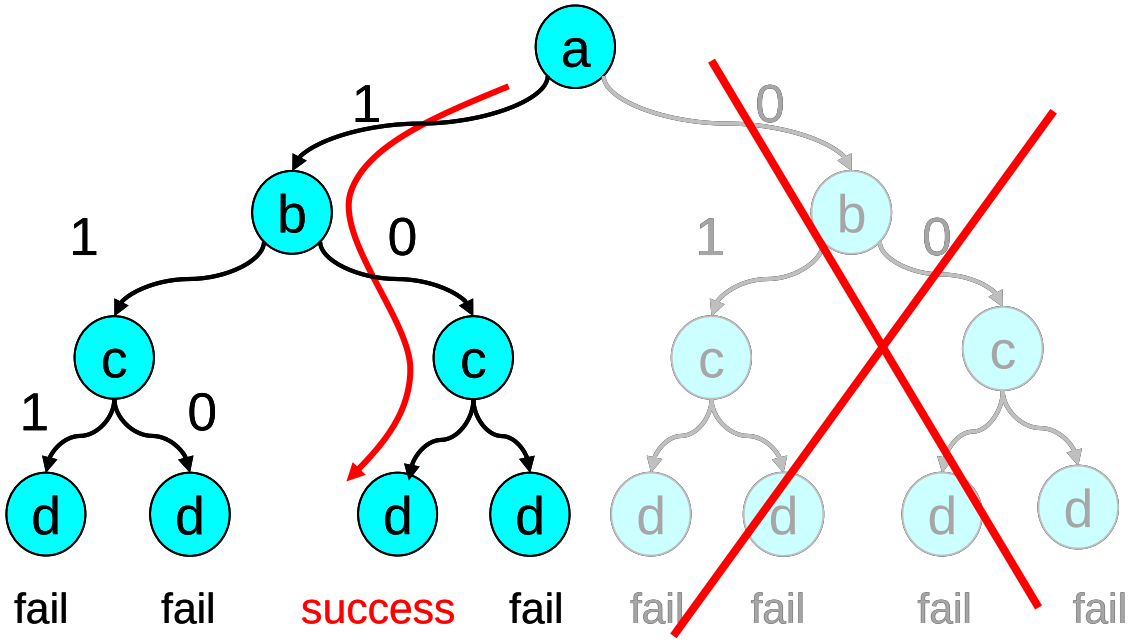
PODEM

- **PODEM [Goel 1981]**
  - **Idea**
  - **Heuristics**
  - **Algorithms**
  - **Summary**



ATPG Decision Tree

- Need smart heuristics to speed up
  - 1. Prune impossible sub-trees ASAP
  - 2. Find good assignments ASAP
- Heuristics are experience-based rules that help correct decision
  - Note: Heuristics Do NOT guaranteed to be correct all the time



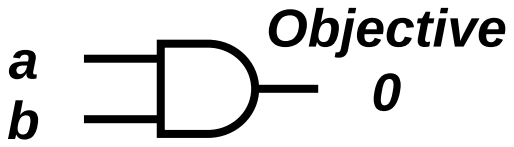
Questions to Be Answered

- **PODEM proposed three heuristics to answer three questions:**
  - **Q1: What path to *backtrace*?**
  - **Q2: What input value to assign?**
  - **Q3: What path to propagate D (D') to PO?**

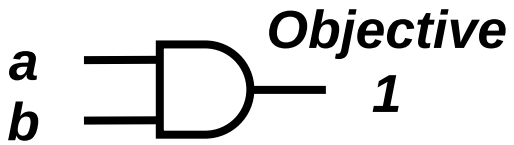
Good Heuristic = Simple and Effective (most of the time)

Q1: What Path to Backtrace?

- ***Decision Gate***:
  - Only one input can control gate output to objective value
    - OR/NAND with output objective =1
    - AND/NOR with output objective =0
  - choose ***easiest*** gate input



- ***Imply Gate***:
  - One input can't control gate output to objective value
    - OR/NAND with output objective =0
    - AND/NOR with output objective =1
  - Choose ***hardest*** gate input

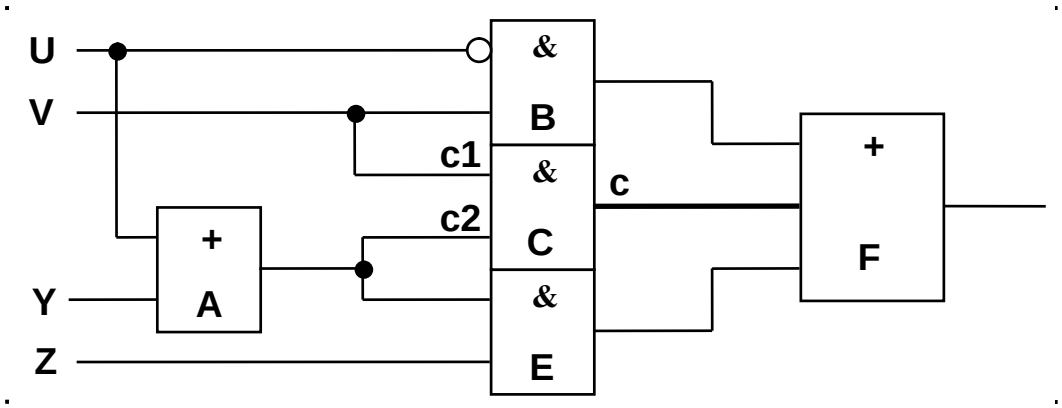


- Why? find out if test exists or not as soon as possible

Heuristic#1: Make Correct Backtrace ASAP

Example

- If objective is c=0; backtrace c1, then V=0 (decision gate)
- If objective is c=1; backtrace c2, then U=1 or Y=1 (imply gate)

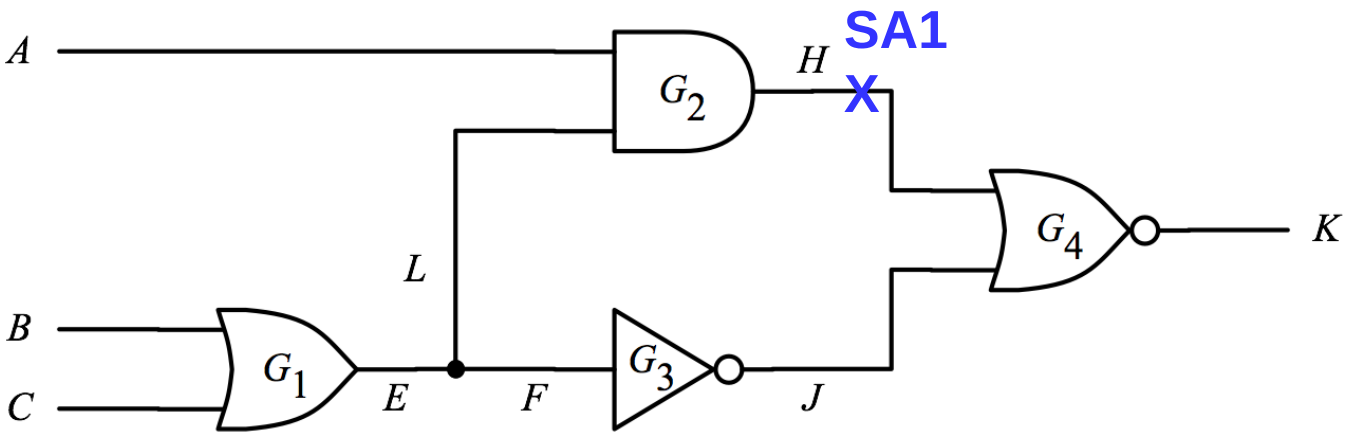


- Q: how do you know c1 is easy and c2 is hard?
  - Level of c1 smaller than level of c2
  - (Use other controllability measure, like SCOAP, also fine)

Quiz

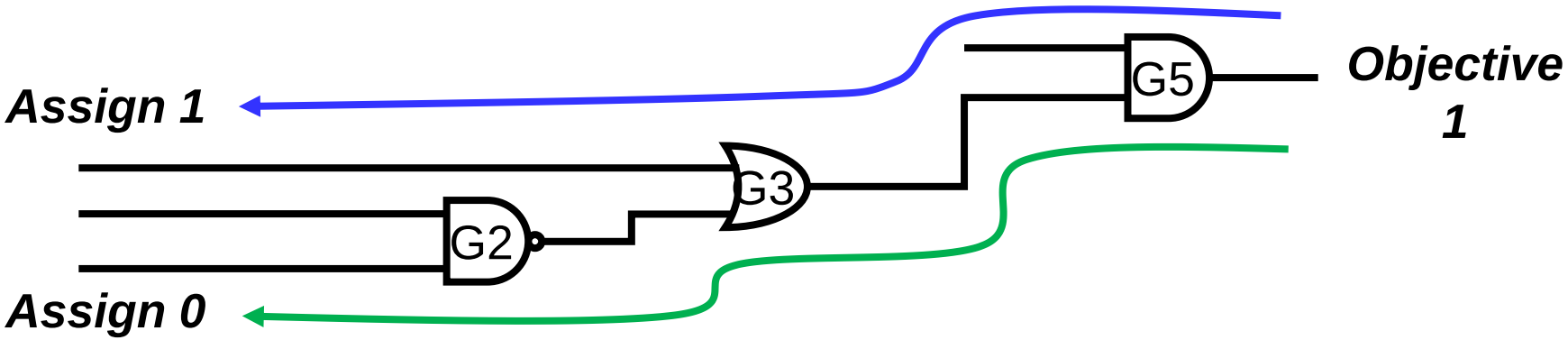
Q: Given SCOAP, generate a test for H SA1 fault.
A1: Follow heuristic
A2: Do not follow heuristic

	A	B	C	E	F	L	H	J	K
CC^0	1	1	1	3	3	3	2	3	5
CC^1	1	1	1	2	2	2	4	4	6
CO	7	6	6	4	4	6	4	3	0



Q2: What Input Values to Assign?

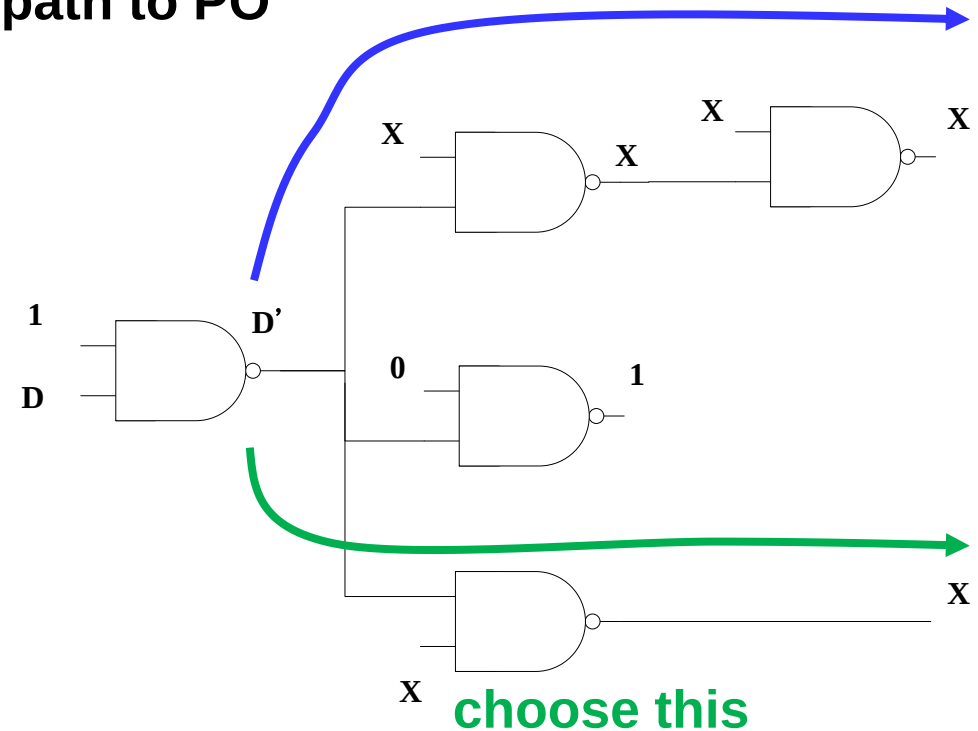
- If even inversion parity path
  - Assign same value as objective
- If odd inversion parity path
  - Assign opposite value to objective
- Why? This assignment is most likely to be correct



Heuristic#2: Inversion Parity for Assignment

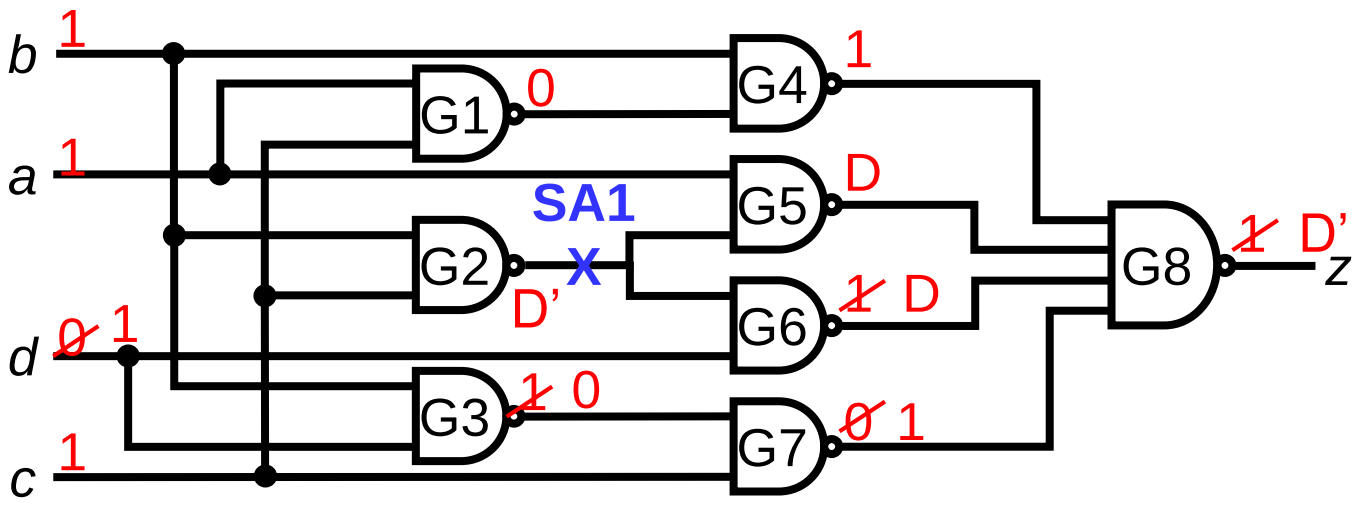
Q3: What Path to Propagate ?

- All gate output of the chosen path must have X values
  - Called *X-PATH*
- If more than one X-path to choose,
  - chose shortest X-path to PO
- If X-path disappear,
  - *backtrack*



Heuristic#3: X-path

PODEM Example w/ Backtrack

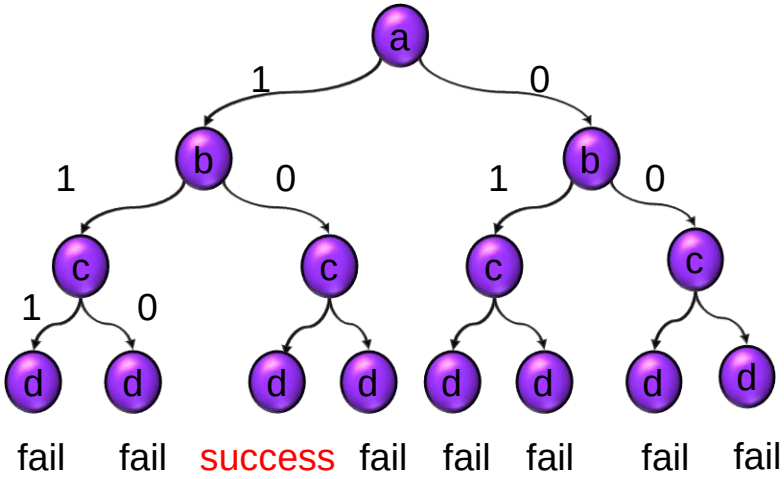


Initial objective: (G2, 0)
Backtrace to PI: *b* = 1
Initial objective: (G2, 0)
Backtrace to PI: *c* = 1
Implication:G2 = D'
Choose shortest X-path {G5} Objective a = 1 Assign a=1
Implication: G1 = 0,G4 = 1,G5 = D

Try propagate through G8.
objective: (G6,1)
Backtrace to PI: *d* = 0
Implication:G3 = 1,G7 = 0,G8 = 1
X-path disappear! Backtrack to most recent PI assignment: *d* = 0 → *d* = 1
Implication:G3 = 0,G6 = D,G8 = D'
Test generated!

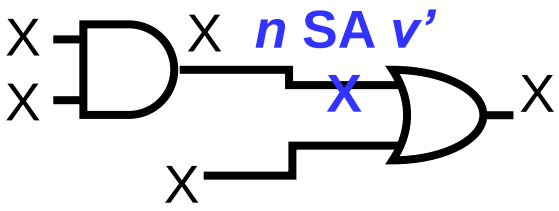
PODEM

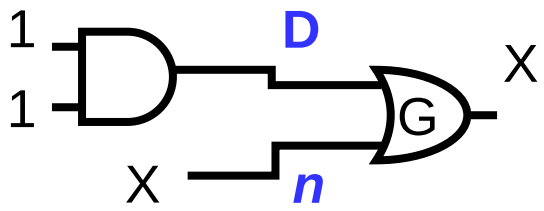
- **PODEM [Goel 1981]**
  - **Idea**
  - **Heuristics**
  - **Algorithms**
  - **Summary**



Objective

- **Pick an objective: (1) fault activation (2) propagate fault effect**






```
Objective (n, v)  // target fault: net n stuck-at v'
     // if fault has not been activated
1.    if (net n is unknown)
2.       return (n, v);
     // else, propagate fault effect
3.    select a gate G from D-frontier on shortest X-path
4.    select an unassigned input n of G
5.    if (gate G has non-controlling value)
6.       v = non-controlling value of G; // AND v=1; OR, v=0
7.    return (n, v );  // n is objective net; v is objective value
```

Backtrace

- Translates objective to PI assignment
- *Depth-first search*: recursively calls itself until hits PI

```
Backtrace (n, v_s)  /* n is objective net; v_s is objective y; */
1.  v = v_s;
2.  while (n is gate output)
3.      if (n is NAND or INVERTER or NOR) v = v';  // inversion parity
4.      if (objective requires setting all inputs)  // imply gate
5.          a = hardest gate input that is still X;
6.      else // decision gate
7.          a = easiest gate input that is still X;
8.      n = a;
9.      (n, v) = Backtrace (n, v);  // recursive call
       // out of while loop, n is now PI
10.  return (n, v)  // assign PI n to value v
```

PODEM

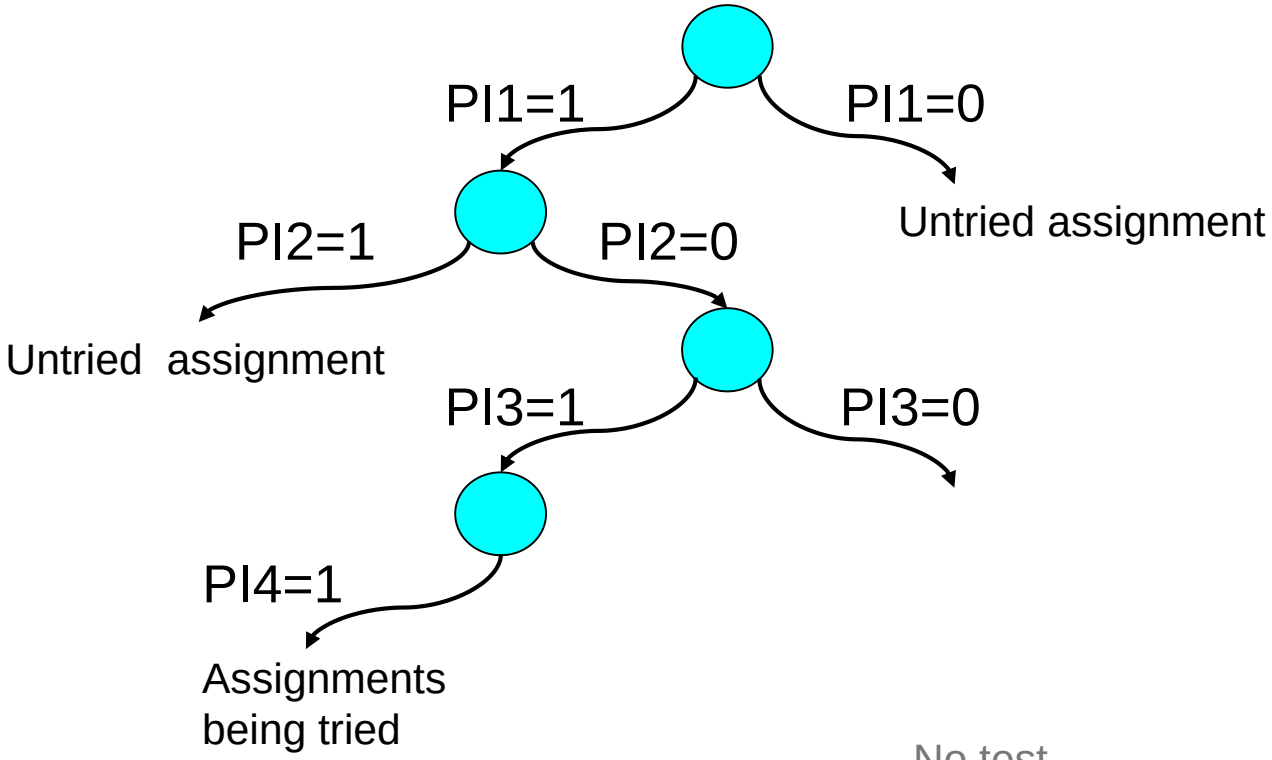
- ***Branch and bound*** **search algorithm**

```
PODEM (fault , v_fault)
1.   if (D or D' at PO) return (SUCCESS)
2.   if (test impossible) return (FAILURE)
3.   else  (n, v_s) = Objective (fault, v_fault);
4.         (pi, v) = Backtrace (n, v_s);
5.         Imply (pi, v); // assign pi, forward implication
6.         if (PODEM (fault, v_fault) == SUCCESS) return (SUCCESS);
           // backtrack
7.         Imply (pi, v');
8.         if (PODEM (fault, v_fault) ==  SUCCESS) return (SUCCESS);
9.         Imply (pi, "X");  // release PI as unknown
10.        return (FAILURE); // this node is pruned
11. end;
```

test impossible for 2 reasons:
(1) target fault cannot be activated
(2) X-path disappear

Decision Tree of PODEM (1)

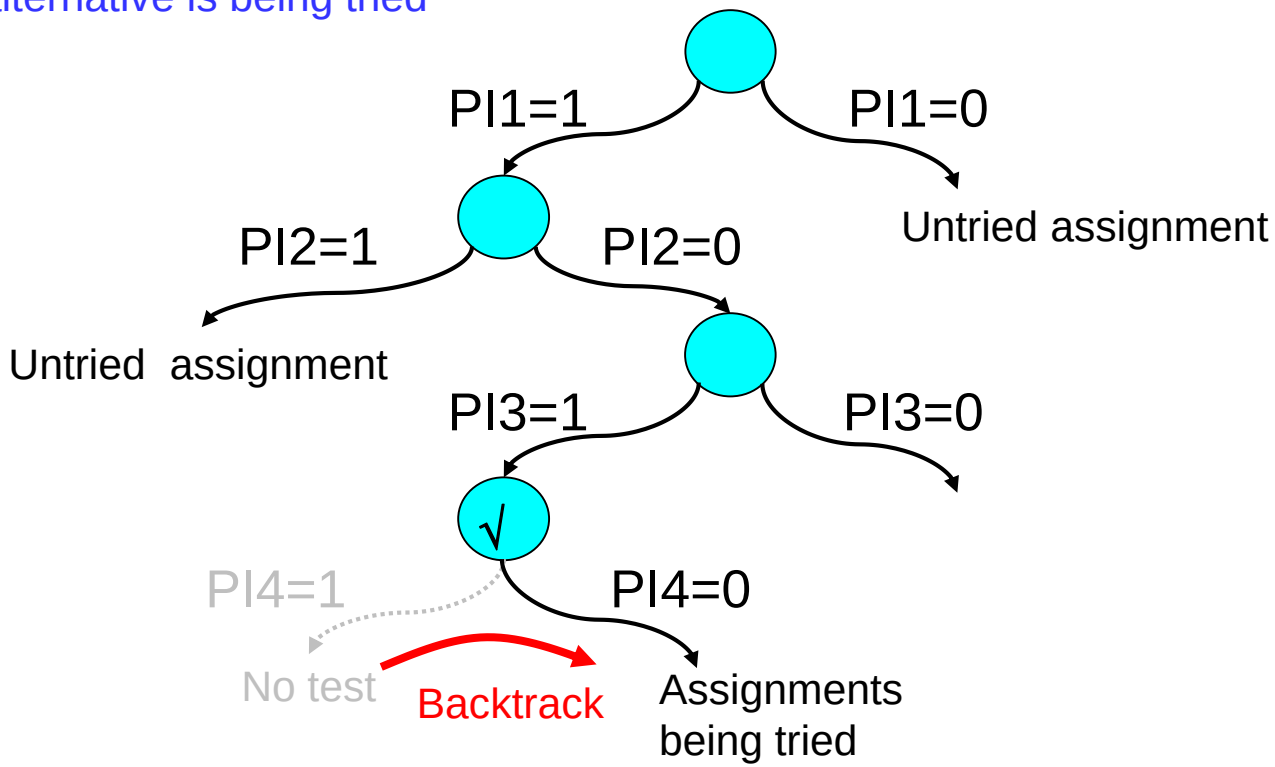
- ***Branch and bound*** **search algorithm**
  - **Each decision is a PI**



Decision Tree of PODEM (2)

- ***Branch and bound*** **search algorithm**
- ***Chronological backtrak:*** **[DPLL satisfiability 1962]**
  - **Flip *last* PI that has not been tried**

√ **:** initial assignment rejected
alternative is being tried

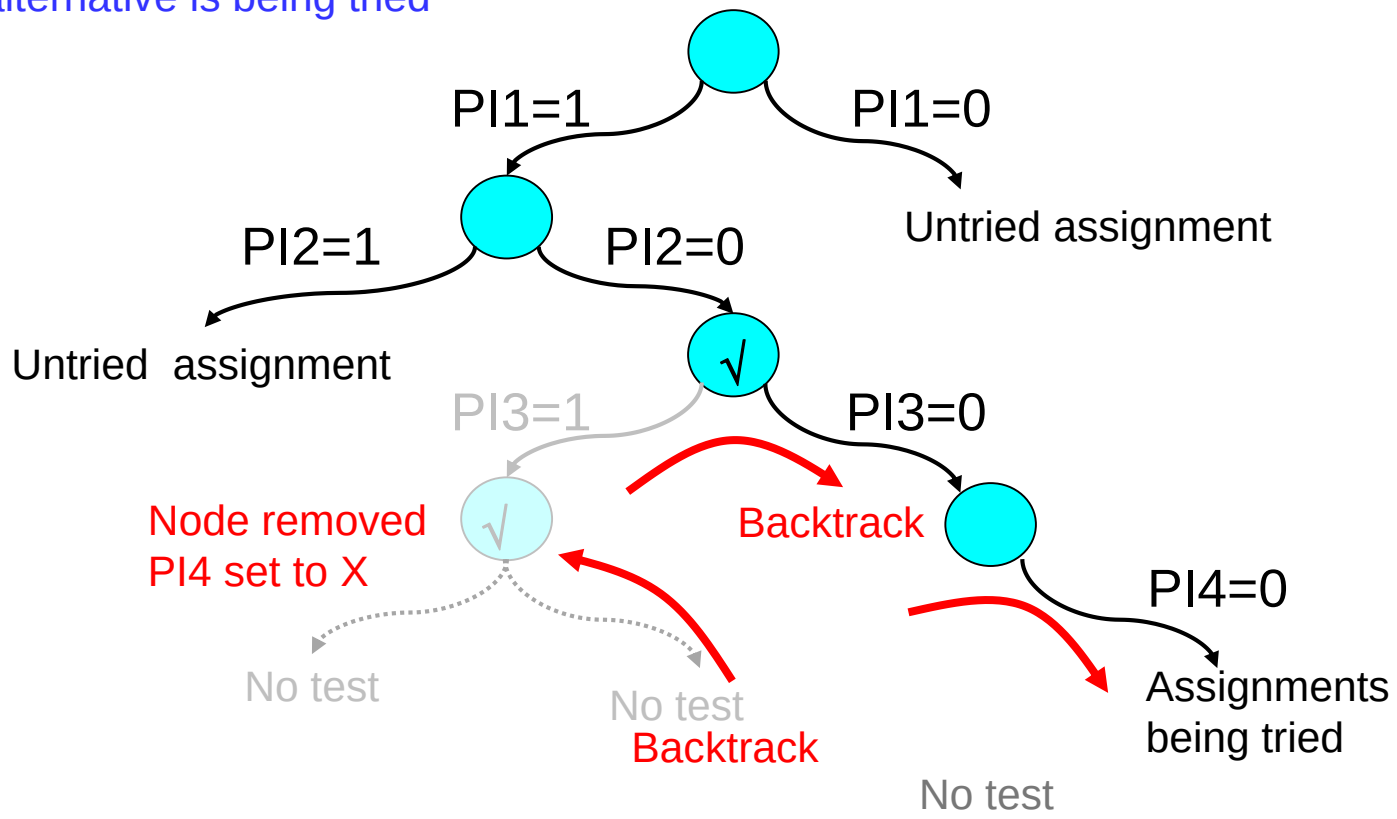


No test

Decision Tree of PODEM (3)

- ***Branch and bound*** **search algorithm**
  - **A node is removed when both alternatives have been tried**
  - **Prune PI3=1 subtree**

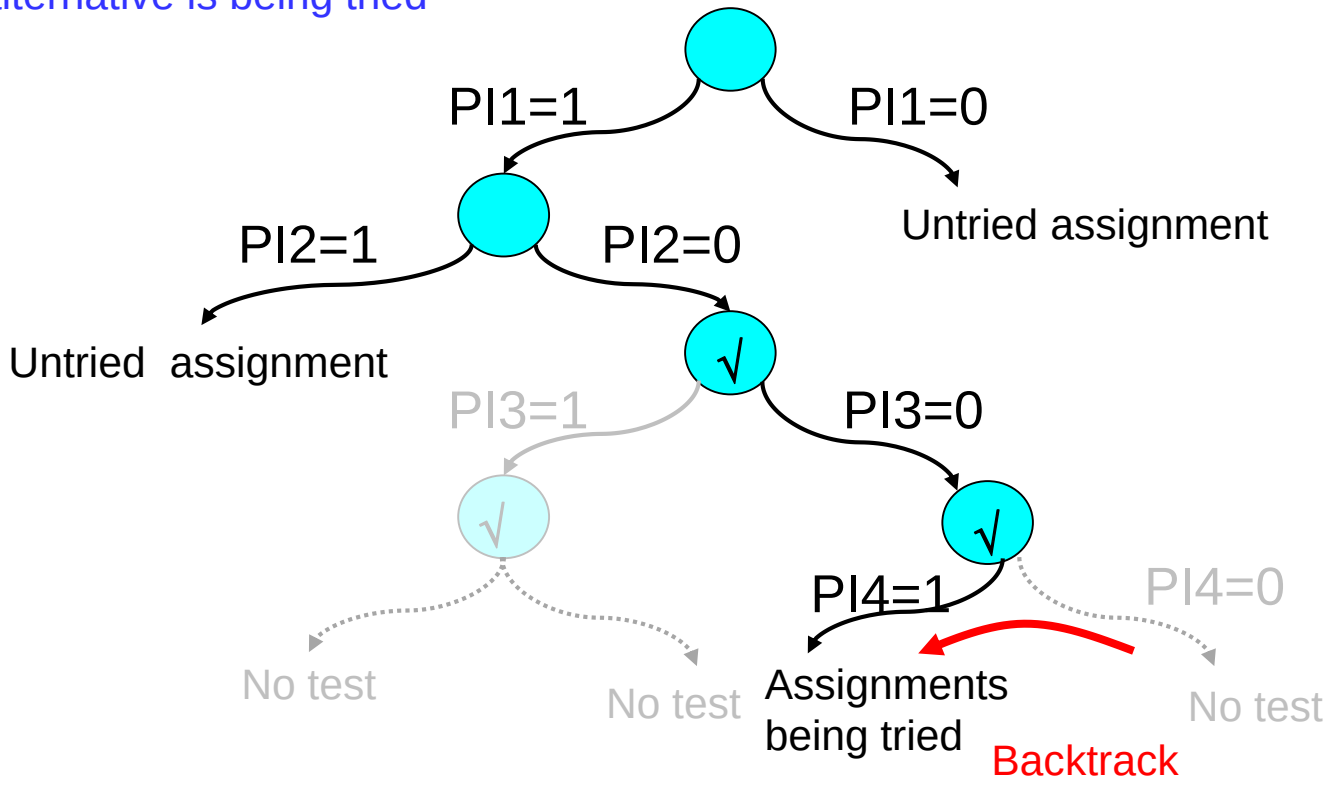
√ **:** initial assignment rejected
alternative is being tried



Decision Tree of PODEM (4)

- ***Branch and bound*** **search algorithm**

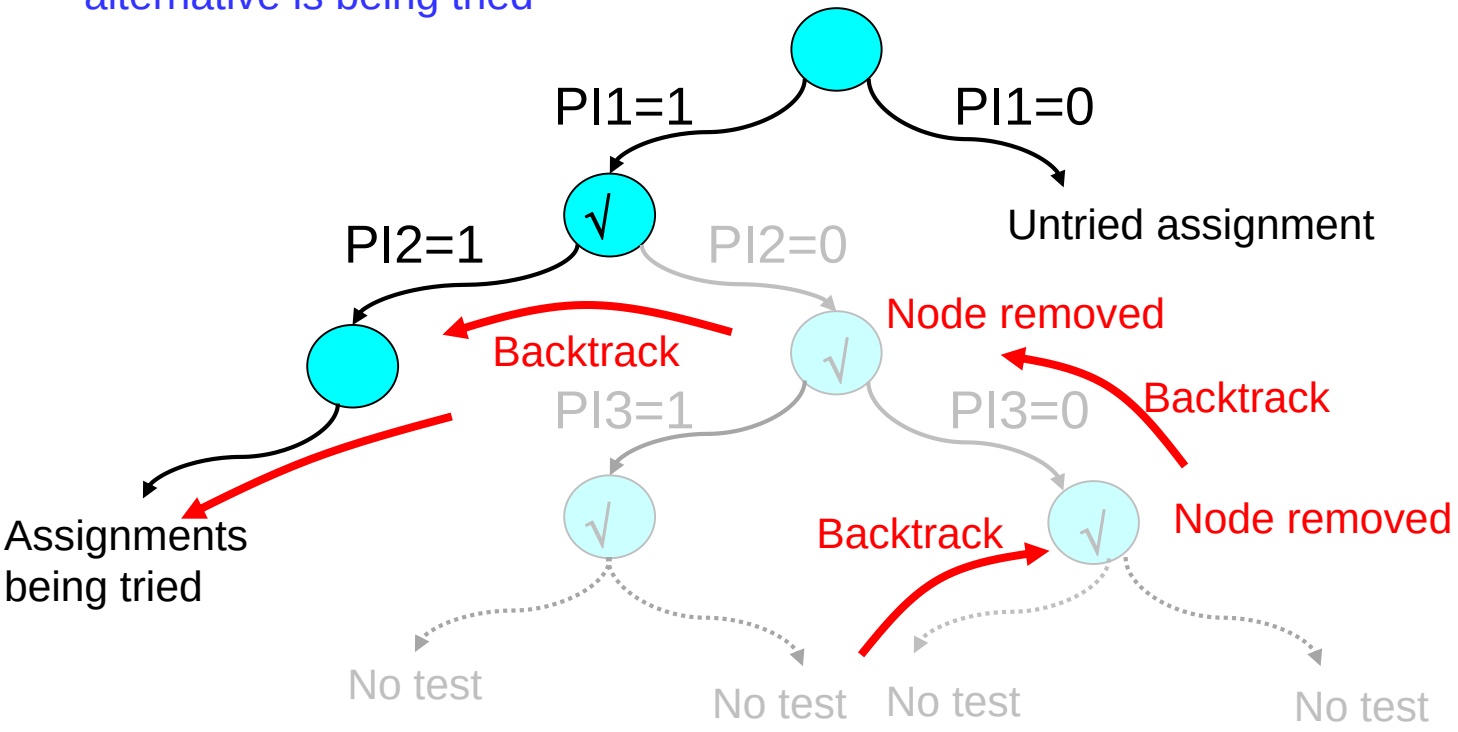
√ **:** initial assignment rejected
alternative is being tried



Decision Tree of PODEM (5)

- ***Branch and bound*** **search algorithm**
  - **Prune PI3=0 subtree**
  - **Prune PI2=0 subtree**

√ **:** initial assignment rejected
alternative is being tried



PODEM: Summary

- **PODEM [Goel 1981]**
  - **Idea:**
    - **assign PI, not internal nodes**
    - **forward implication only, no justification**
  - **Heuristics**
    - **(1) backtrace easy/hard input for decision/imply gate**
    - **(2) keep path inversion parity**
    - **(3) propagate along X-path**
  - **Algorithms**
    - **branch and bound search**

FFT

- **Q1: Is PODEM a complete ATPG algorithm?**
- **Q2: Why does PODEM have no justification?**
- **Q3: Please give examples where heuristic #1/#2 gives wrong guess**