

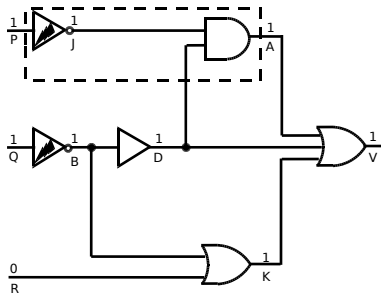
# New Advances in Sequential Diagnosis

Jinbo Huang

NICTA and Australian National University  
*Joint work with Sajjad Siddiqi*

30 April 2010

# An Abnormal System



- Model: circuit description + expected abnormal behavior
  - $Pr(broken) = 0.1$ ; if broken,  $Pr(output = 1) = 0.5$
- Diagnosis: set of faulty gates
- Can't always be determined

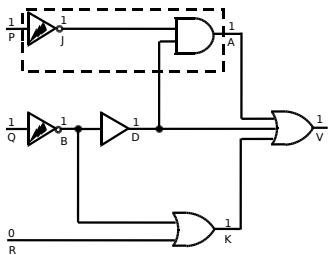
# Sequential Diagnosis

- Measure system variables until faults are located
- Assume equal measurement costs—minimize # of measurements
- Optimal policy (tree) exists, intractable
- Greedily maximize utility of each measurement

# Measurement Point Selection

- Reasonably close to optimal
  - combine component failure probability and wire entropy
- Efficient to compute
  - Treat system as Bayesian network
  - Harness compilation for efficient (repeated) computation of probabilities
- For further scalability
  - Abstraction
  - Component cloning

# System Modeling



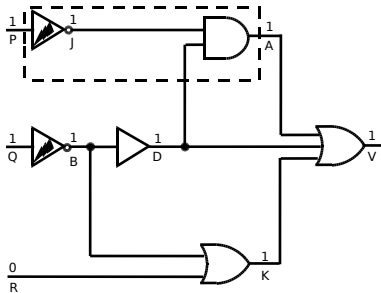
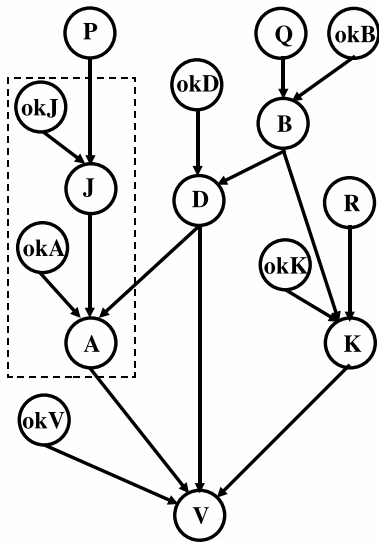
$okA \rightarrow (A \leftrightarrow (J \wedge D)), okJ \rightarrow (J \leftrightarrow \neg P)$

- $okX \rightarrow \text{NORMALBEHAVIOR}(X)$
- $Pr(okA) = Pr(okJ) = 0.9$

$\neg okA \rightarrow (A \leftrightarrow \theta_A), \neg okJ \rightarrow (J \leftrightarrow \theta_J)$

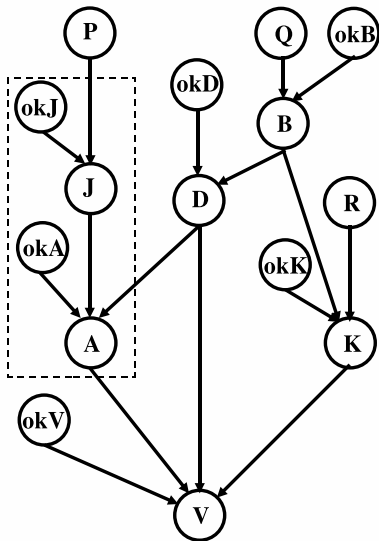
- $Pr(\theta_A) = Pr(\theta_J) = 0.5$

# System as Bayesian Network



*Markov Property:*  
 $Pr(\text{node})$  independent of  
*nondescendants* given  
parents

# System as Bayesian Network

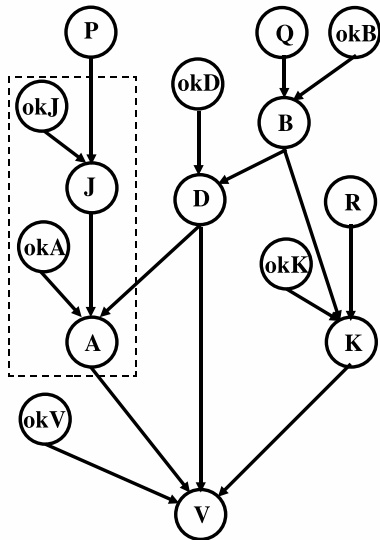


$P$	$\theta_P$
1	0.5
0	0.5

$okJ$	$\theta_{okJ}$
1	0.9
0	0.1

$P$	$okJ$	$J$	$\theta_{J P,okJ}$
1	1	1	0
1	1	0	1
1	0	1	0.5
1	0	0	0.5
0	1	1	1
0	1	0	0
0	0	1	0.5
0	0	0	0.5

# Bayesian Network

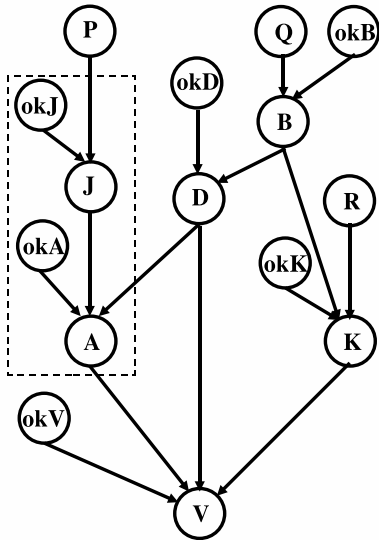


$\exists$  unique  $Pr$  satisfying independences asserted by graph, and CPTs

$Pr(X_1, X_2, \dots, X_n)$  is  $\prod$  of numbers, one from each CPT according to instantiation

Other  $Pr$  can be obtained by summing

# Bayesian Network



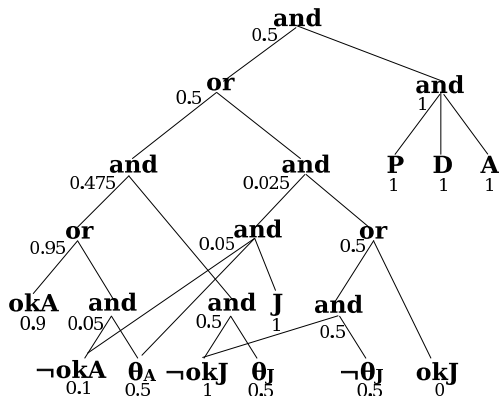
$\exists$  unique  $Pr$  satisfying independences asserted by graph, and CPTs

$Pr(X_1, X_2, \dots, X_n)$  is  $\prod$  of numbers, one from each CPT according to instantiation

Other  $Pr$  can be obtained by summing

Computing  $Pr$  this way inefficient, though

# Compilation into Arithmetic Circuit



$$okA \rightarrow (A \leftrightarrow (J \wedge D))$$

$$okJ \rightarrow (J \leftrightarrow \neg P)$$

$$Pr(okA) = Pr(okJ) = 0.9$$

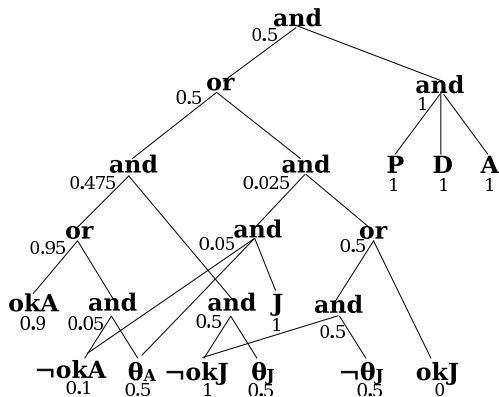
$$\neg okA \rightarrow (A \leftrightarrow \theta_A)$$

$$\neg okJ \rightarrow (J \leftrightarrow \theta_J)$$

$$Pr(\theta_A) = Pr(\theta_J) = 0.5$$

$$\text{observation: } A \wedge P \wedge D$$

# Compilation into Arithmetic Circuit



$$okA \rightarrow (A \leftrightarrow (J \wedge D))$$

$$okJ \rightarrow (J \leftrightarrow \neg P)$$

$$Pr(okA) = Pr(okJ) = 0.9$$

$$\neg okA \rightarrow (A \leftrightarrow \theta_A)$$

$$\neg okJ \rightarrow (J \leftrightarrow \theta_J)$$

$$Pr(\theta_A) = Pr(\theta_J) = 0.5$$

$$\text{observation: } A \wedge P \wedge D$$

$$Pr(\neg okJ \mid \text{obs})?$$



# Measurement Selection: Previous Method

Entropy over set of diagnoses

- $\xi(D) = -\sum(p_d \log p_d)$
- reflects uncertainty over true diagnosis
- greedily minimize  $\xi(D)$
- need to maintain  $\{D\}$  and  $Pr(D)$
- $|\{D\}|$  can be large (exponential in worst case)
- updating  $Pr(D)$  inefficient
- can approximate  $\{D\}$  with “preferred diagnoses,” sacrifices accuracy

# Measurement Selection: New Method

Entropy over each system variable

- $\xi(X) = -(p_x \log p_x + p_{\bar{x}} \log p_{\bar{x}})$
- reflects expected info gain by measurement
- measure variable with highest  $\xi(X)$
- $Pr(X)$  obtainable in linear time post compilation

# Measurement Selection: New Method

Entropy over each system variable

- $\xi(X) = -(p_x \log p_x + p_{\bar{x}} \log p_{\bar{x}})$
- reflects expected info gain by measurement
- measure variable with highest  $\xi(X)$
- $Pr(X)$  obtainable in linear time post compilation
- is it as good?

# Measurement Selection: New Method

Entropy over each system variable

- $\xi(X) = -(p_x \log p_x + p_{\bar{x}} \log p_{\bar{x}})$
- reflects expected info gain by measurement
- measure variable with highest  $\xi(X)$
- $Pr(X)$  obtainable in linear time post compilation
- is it as good?
- not yet

# Measurement Selection: New Method

Measuring variable with highest entropy

- alone didn't work well (higher diagnostic cost)
- large # of unlikely diagnoses reduces usefulness of variable entropy
- confirmed empirically by pruning diagnoses with  $> k$  faults: after pruning, works as well as previous method

# Measurement Selection: New Method

Measuring variable with highest entropy

- alone didn't work well (higher diagnostic cost)
- large # of unlikely diagnoses reduces usefulness of variable entropy
- confirmed empirically by pruning diagnoses with  $> k$  faults: after pruning, works as well as previous method
- can't use "entropy + pruning" as  $k$  not known

# Measurement Selection: New Method

Measuring variable with highest entropy

- alone didn't work well (higher diagnostic cost)
- large # of unlikely diagnoses reduces usefulness of variable entropy
- confirmed empirically by pruning diagnoses with  $> k$  faults: after pruning, works as well as previous method
- can't use "entropy + pruning" as  $k$  not known
- pick component with highest  $Pr(\neg okX)$ , pick its variable with highest entropy
- automatically achieves similar effect to pruning

# Measurement Selection: New Method

- As good as previous state of the art in diagnostic cost
  - observed on problems solvable by both
- More efficient & scalable

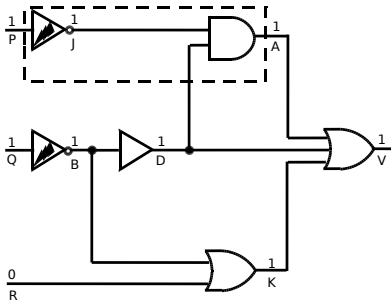
# Measurement Selection: New Method

- As good as previous state of the art in diagnostic cost
  - observed on problems solvable by both
- More efficient & scalable
- What if compilation unsuccessful or too large?

# Measurement Selection: New Method

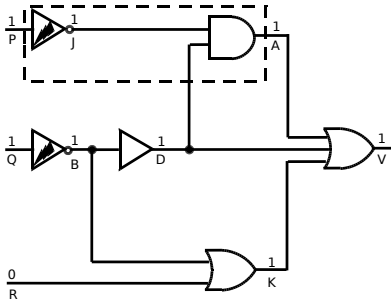
- As good as previous state of the art in diagnostic cost
  - observed on problems solvable by both
- More efficient & scalable
- What if compilation unsuccessful or too large?
- Structure-based techniques for further scalability: abstraction & component cloning

# Abstraction



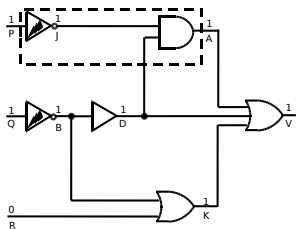
- Identify **cones**, treat as **abstract component**
- *Cone*: subsystem where all components are *dominated* by some component  $X$
- $X$  *dominates*  $Y$  if any path from  $Y$  to output contains  $X$
- Identification automatic, efficient

# Abstraction



- # of components, # of health variables reduced
- Compilation scales to larger systems
- Look inside cone **only if** cone as a whole identified as faulty in abstract level—then compile & diagnose recursively

# Abstraction: Encoding of Cone



Previous full encoding

- $okA \rightarrow (A \leftrightarrow (J \wedge D))$   
 $okJ \rightarrow (J \leftrightarrow \neg P)$
- need  $okX$  for every component

Abstract encoding

- $okA \rightarrow (A \leftrightarrow (J \wedge D))$   
 $\neg okA \rightarrow (A \not\leftrightarrow (J \wedge D)), J \leftrightarrow \neg P$
- single  $okA$  for root
- explicitly force wrong output under  $\neg okA$
- need to compute  $Pr(okA)$

# Abstraction: Failure Probability for Cone

- XOR healthy cone & actual cone
- Compute  $Pr(output = 1)$  by compilation
- Done once (recursively) for all cones as preprocessing

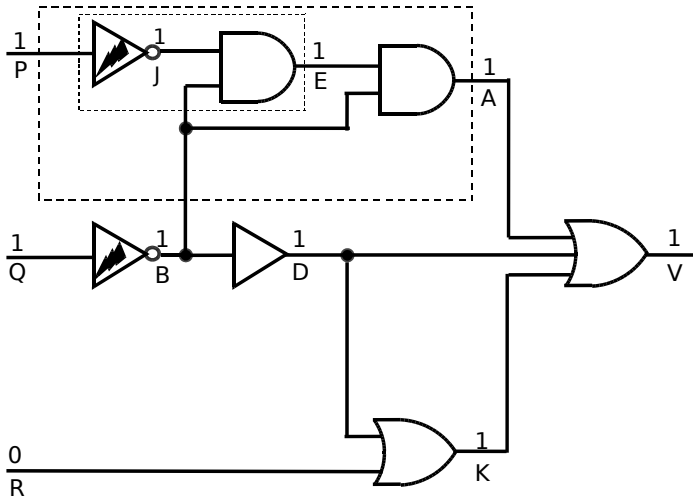
## Abstraction: Summary

- System size reduced by abstraction
- Initially, only measure variables outside cones
- Look inside cone only if root identified as faulty
- Diagnose cone recursively
- Extended solvable benchmarks from c1355 (546 gates) to c2670 (1193 gates, 160 abstract gates)
- Diagnostic cost similar to baseline (on problems solvable by both)

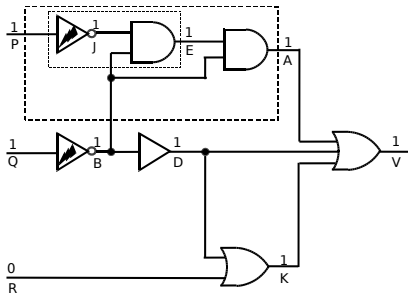
# For Even Larger Systems

- Intractable **even after** abstraction
- Idea: reduce abstraction size by creating more cones

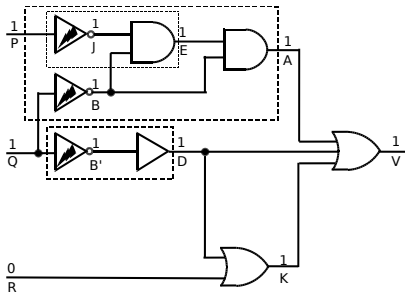
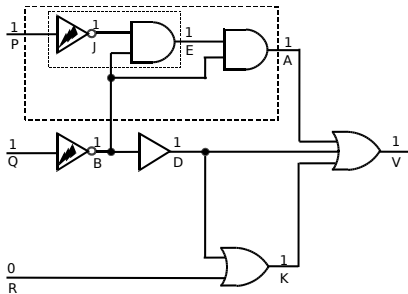
# Any More Cones?



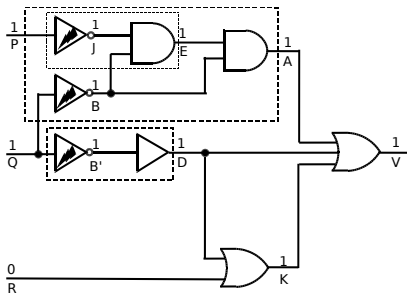
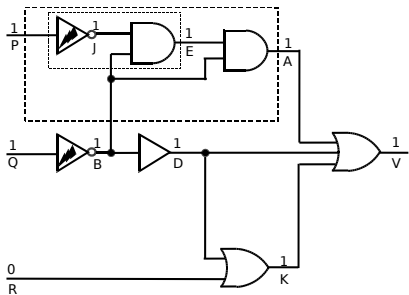
# Component Cloning



# Component Cloning



# Component Cloning

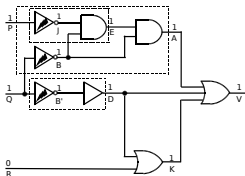


- Pick component
- Create one or more clones
- Distribute parents among clones

# Component Cloning: Choices

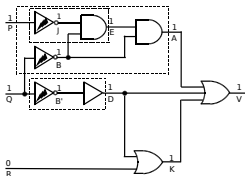
- Pick component
  - components in abstraction that are not roots of cones
- Create one or more clones
  - partition parents into  $\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_q$  such that each  $\mathbf{P}_i$  lies entirely in a cone
  - create  $q - 1$  clones
- Distribute parents among clones
  - give each clone one  $\mathbf{P}_i$

# Component Cloning



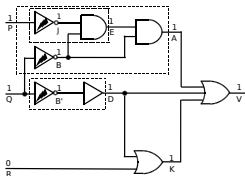
- Smaller abstraction, easier to compile & diagnose
- What's the catch?

# Component Cloning



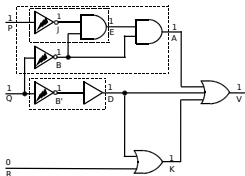
- Smaller abstraction, easier to compile & diagnose
  - What's the catch?
- 
- New system is a **relaxation**
    - two copies can fail independently

# Component Cloning



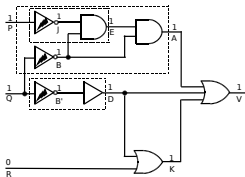
- Smaller abstraction, easier to compile & diagnose
  - What's the catch?
- 
- New system is a **relaxation**
    - two copies can fail independently
  - Solution: filter spurious diagnoses (insist on same health state for all copies)

# Component Cloning



- Smaller abstraction, easier to compile & diagnose
  - What's the catch?
- 
- Probability space different, skewing measurement selection heuristic

# Component Cloning



- Smaller abstraction, easier to compile & diagnose
  - What's the catch?
- 
- Probability space different, skewing measurement selection heuristic
  - Solution: none needed, diagnostic cost only slightly affected

# Component Cloning: Summary

- Abstraction size substantially reduced
- Extended solvable benchmarks from c2670 (1193 gates, 160 abstract gates) to c7552 (3512 gates, 545 abstract gates, 378 after cloning)

# Summary

- New measurement point selection heuristic
  - more efficient to compute, equally effective
- Abstraction by cone identification
  - size of largest solvable benchmark doubled
- Component cloning to reduce abstraction size
  - size of largest solvable benchmark further tripled