

# **Relax, Compensate, then Recover:**

**A Theory of Anytime, Approximate Inference**

**Adnan Darwiche**

**UCLA**

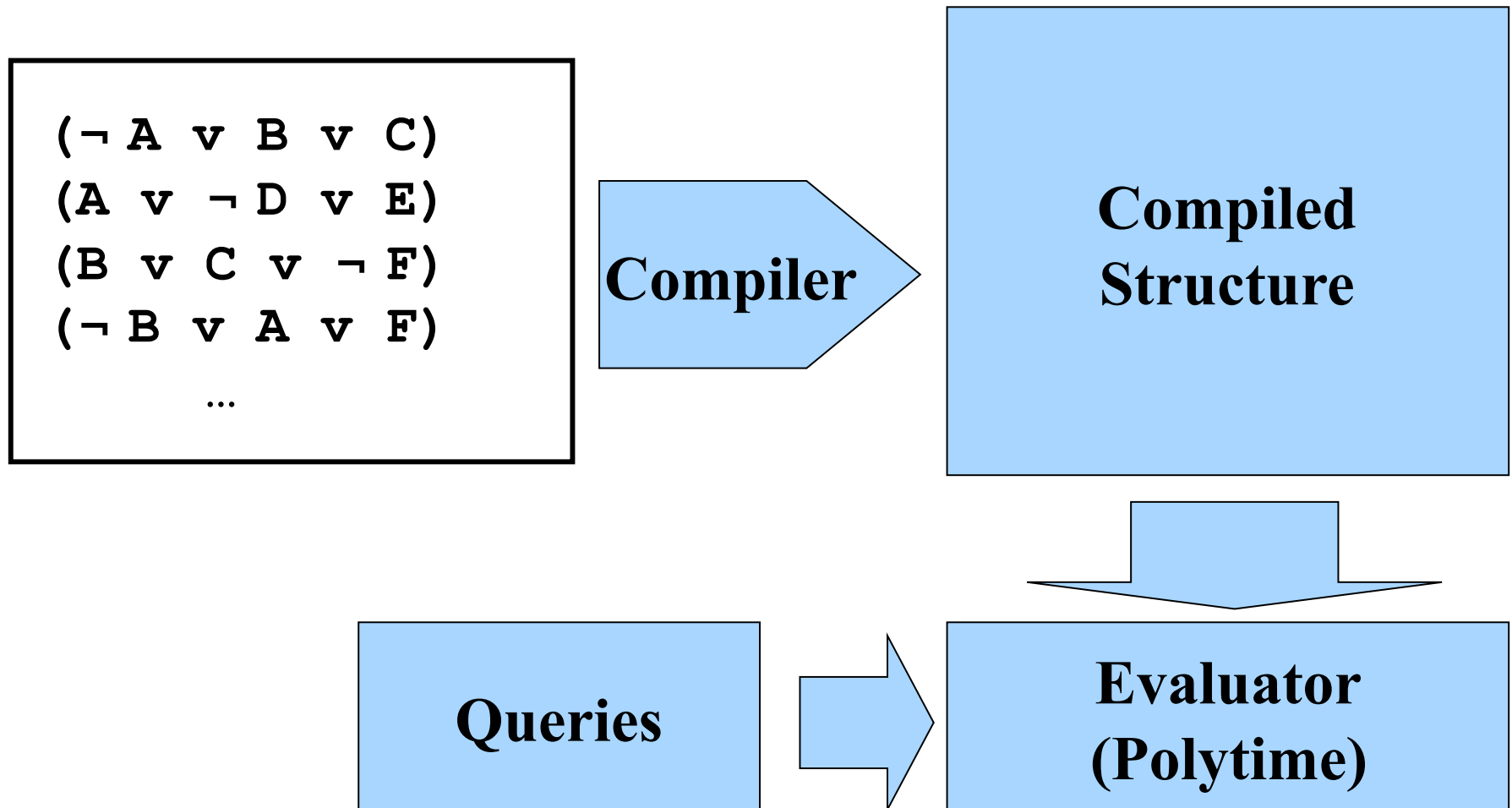
# Inference

- **Probabilistic Graphical Models:**  
Marginal and conditional probabilities  
Most likely instantiations...
- **Propositional Knowledge Bases:**  
Logical entailment  
Existential quantification  
Model counting...

# Two Main Themes

- **Exact inference as:**  
Enforcing decomposability and determinism on propositional knowledge bases
- **Approximate inference as:**  
Relaxing, compensating for, and recovering equivalence constraints (equalities)

# Knowledge Compilation

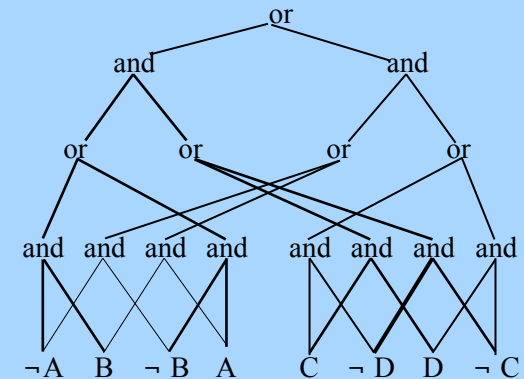


# Knowledge Compilation

$(\neg A \vee B \vee C)$   
 $(A \vee \neg D \vee E)$   
 $(B \vee C \vee \neg F)$   
 $(\neg B \vee A \vee F)$   
...

**Compiler**

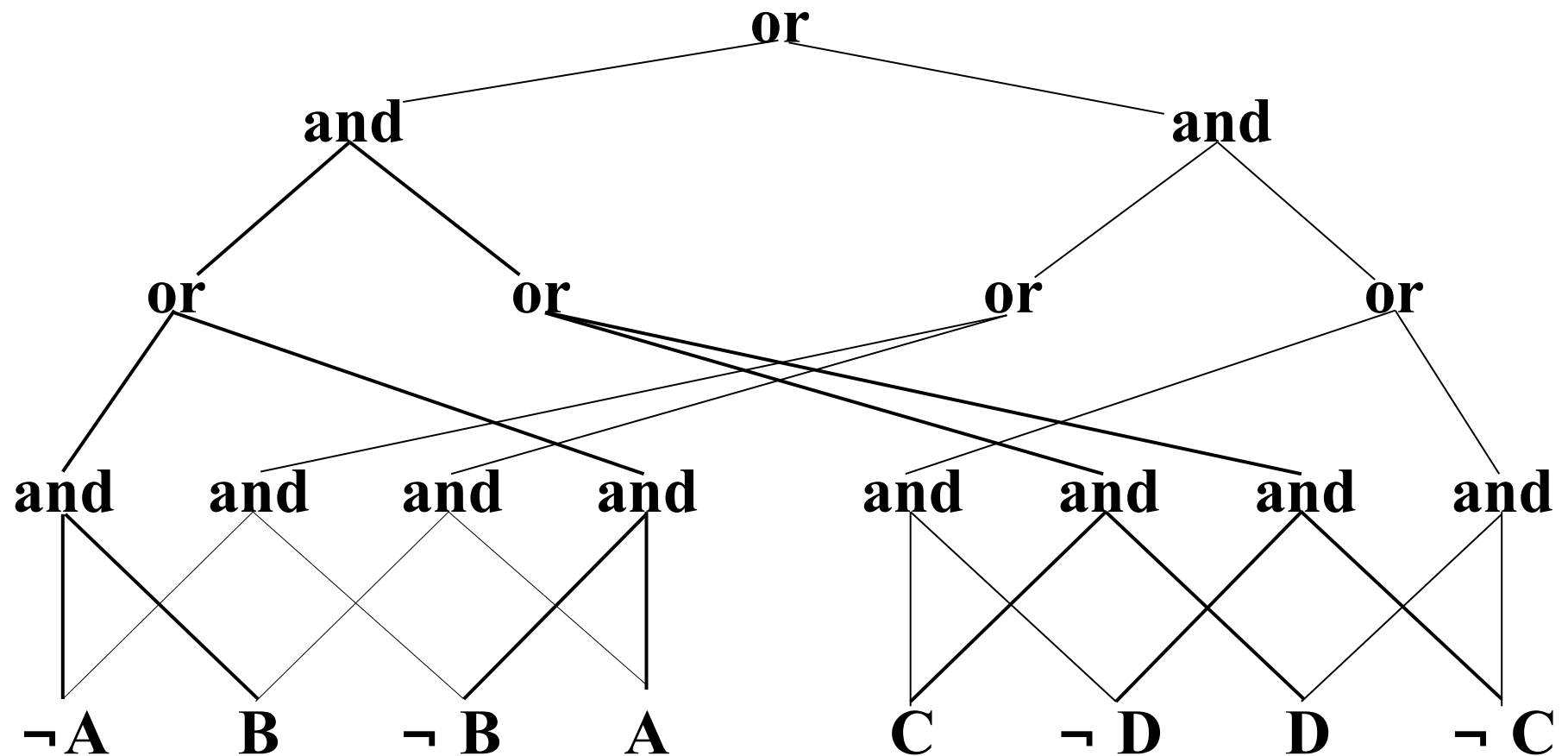
Subsets of NNF



**Queries**

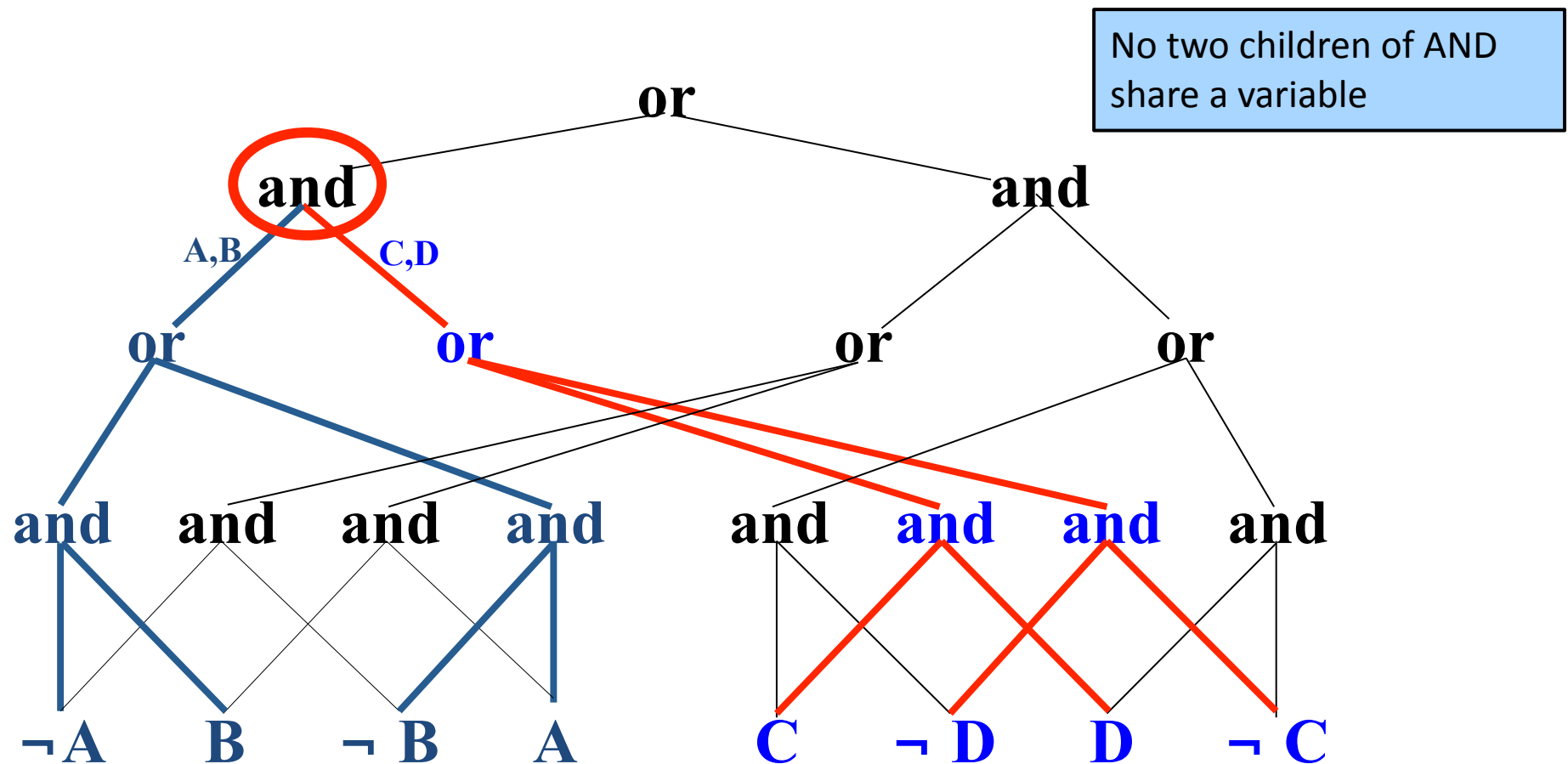
**Evaluator  
(Polytime)**

# Negation Normal Form



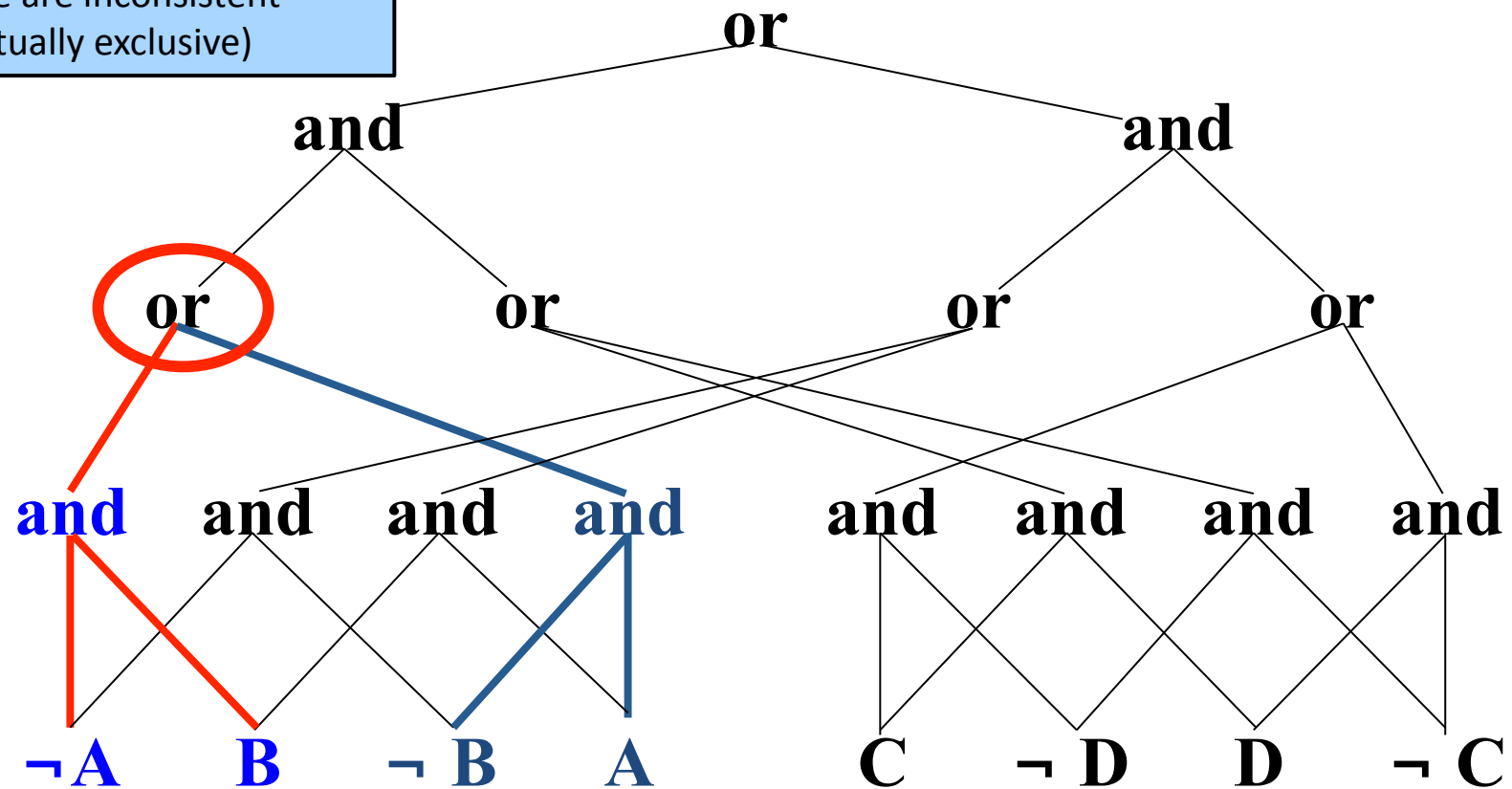
rooted DAG (Circuit)

# Decomposability (DNNF)



# Determinism (d-DNNF)

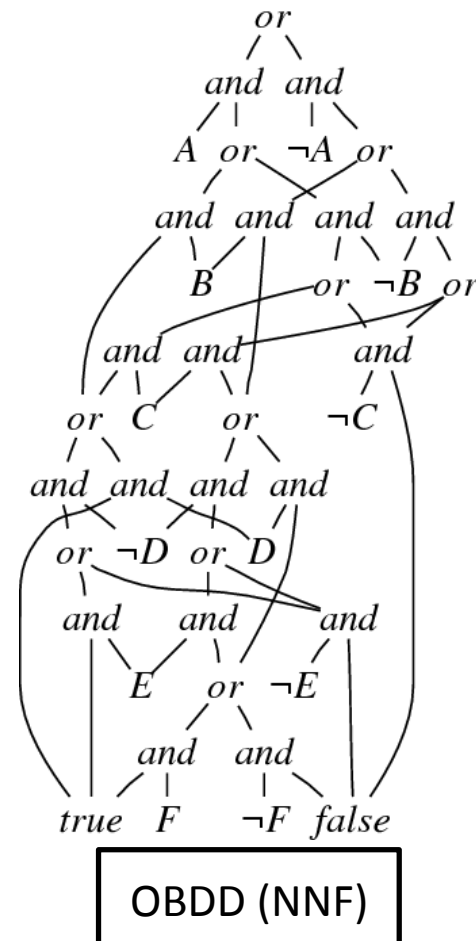
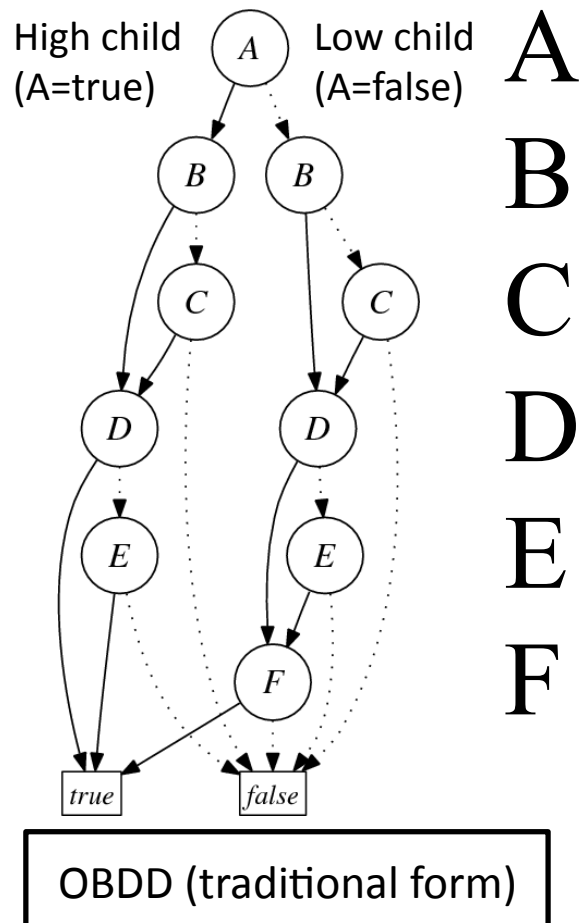
Every pair of children of or-node are inconsistent (mutually exclusive)





# OBDD:

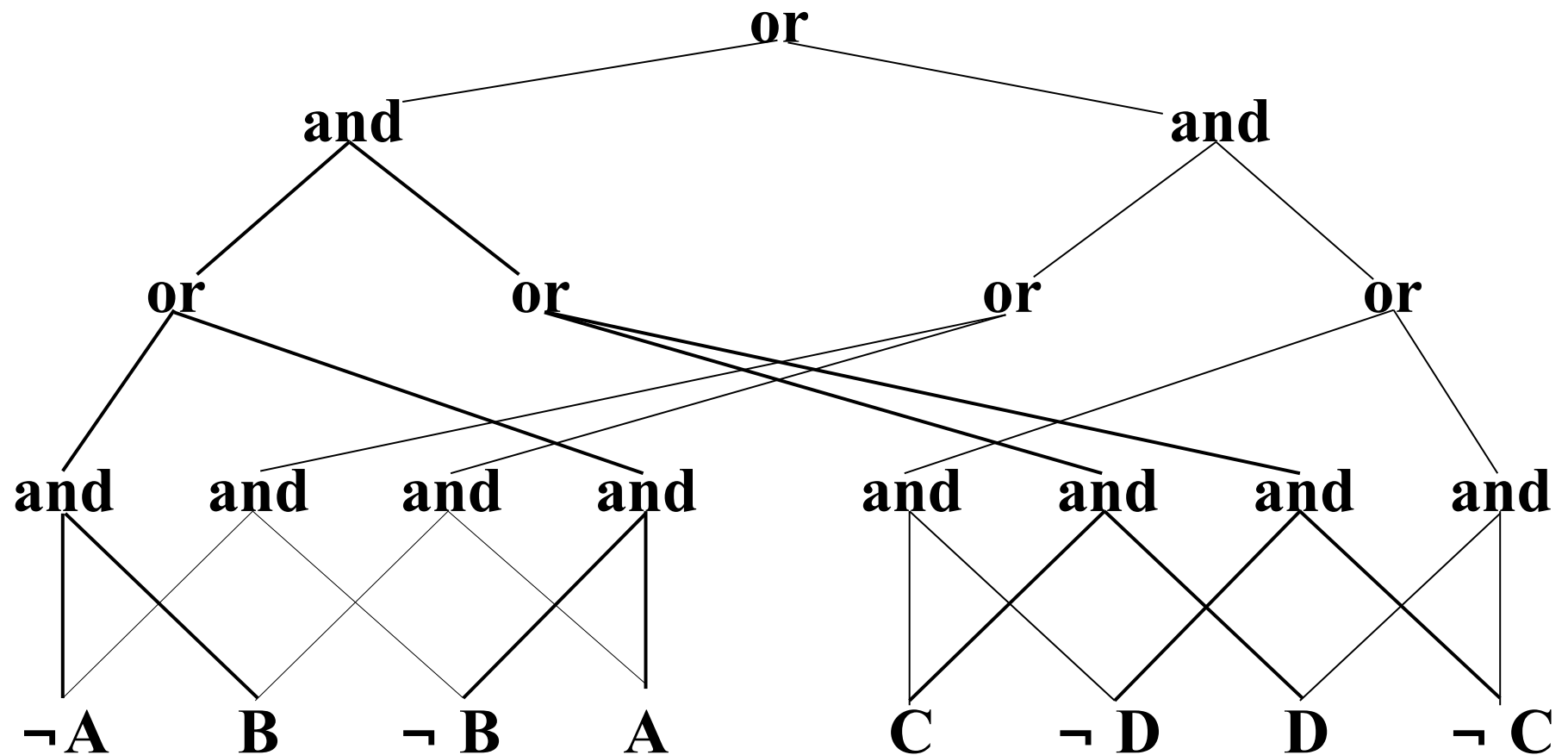
## d-DNNF + Additional Properties



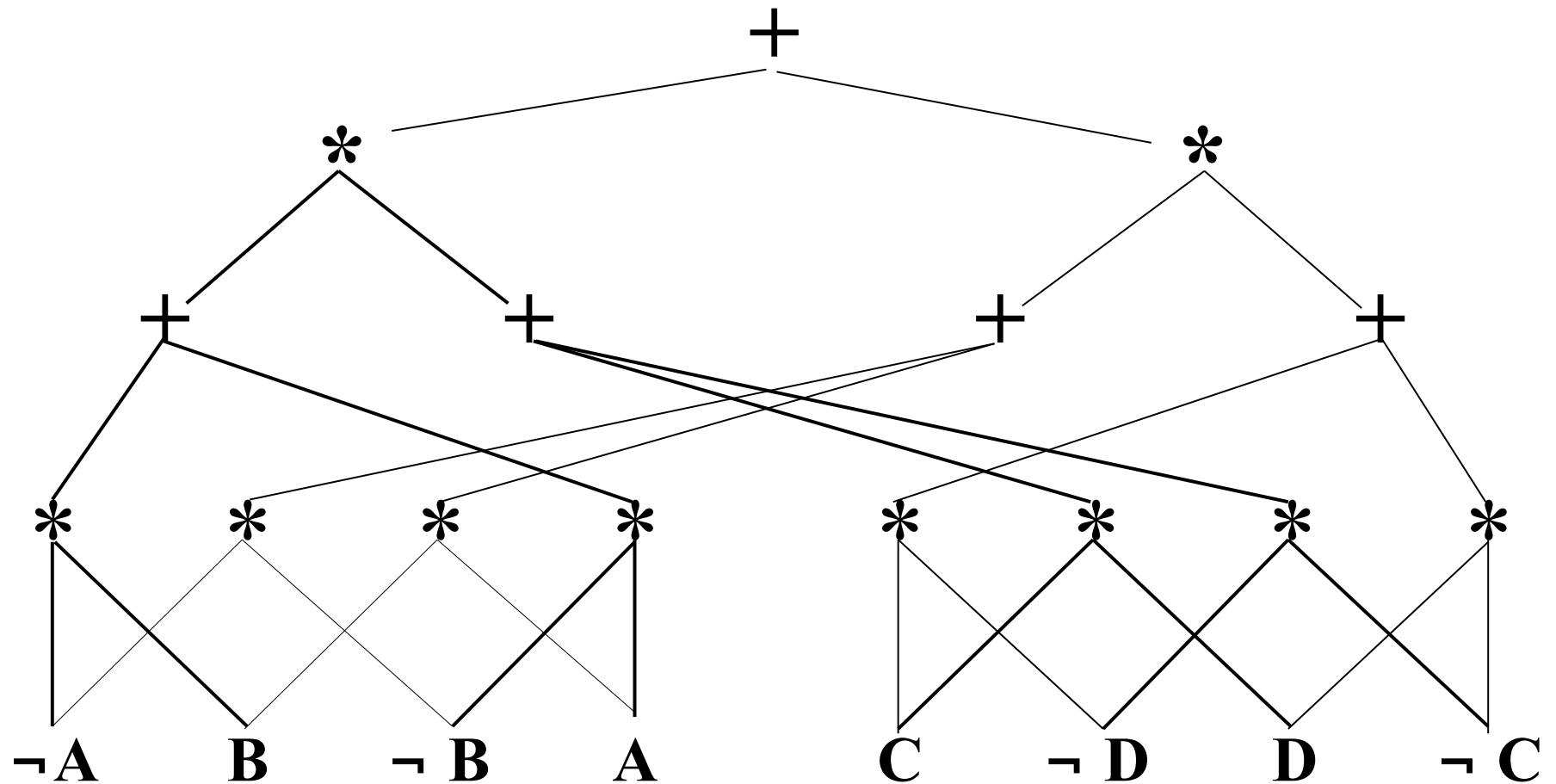
# Queries and Transformations

- **Queries**  
SAT, MAXSAT, logical entailment, equivalence testing, model counting,...
- **Transformations:**  
Existential quantification, conjunction, disjunction, negation...
- More properties imply more polytime queries and transformations, but less succinctness

# Counting Models (d-DNNF)

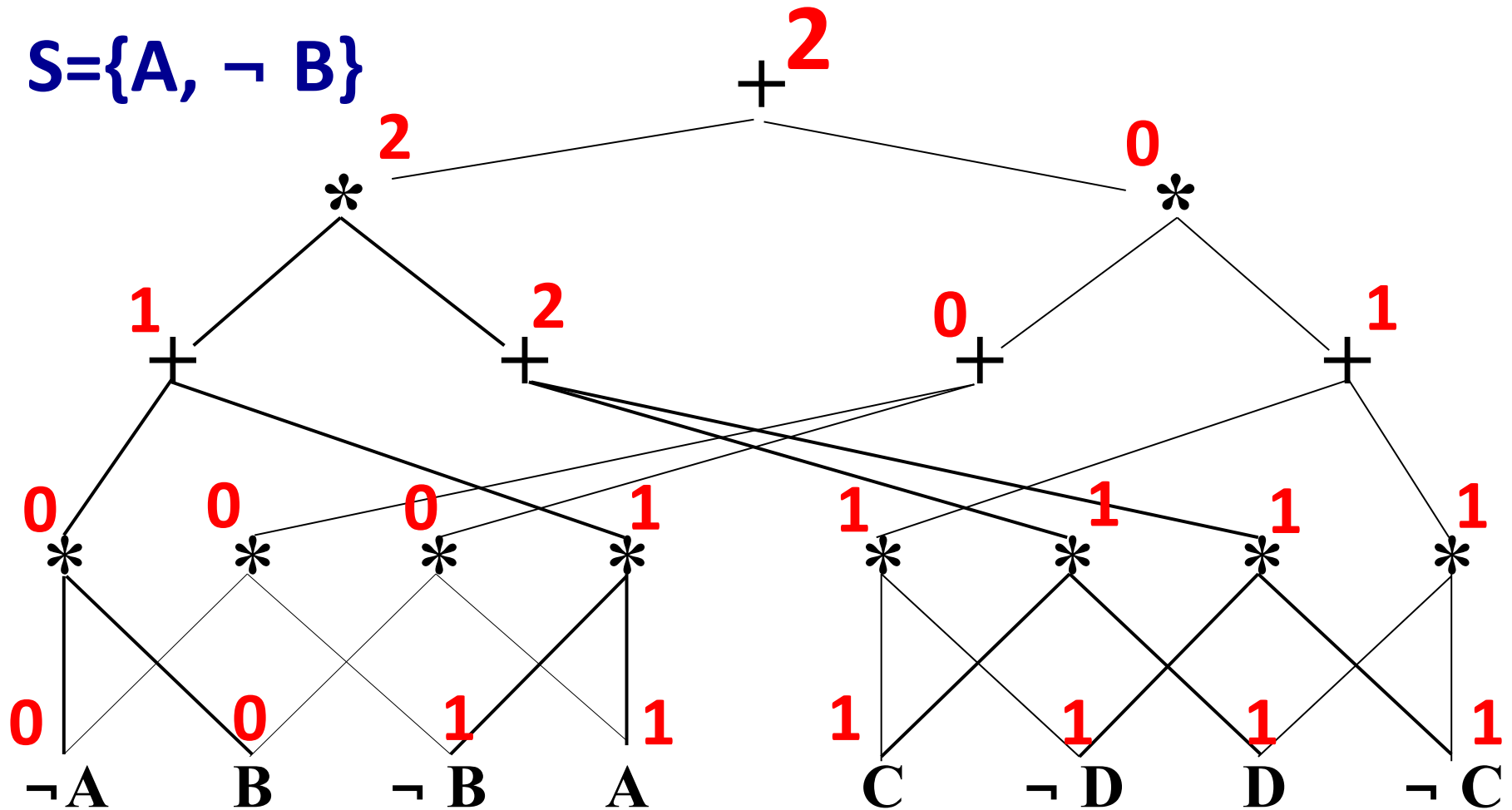


# Counting Graph

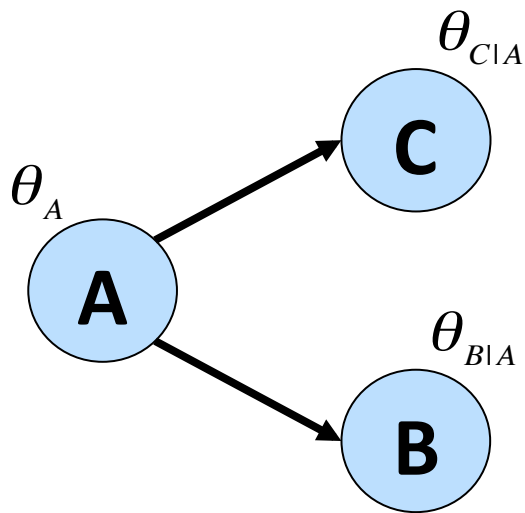


# Counting Graph

$S = \{A, \neg B\}$



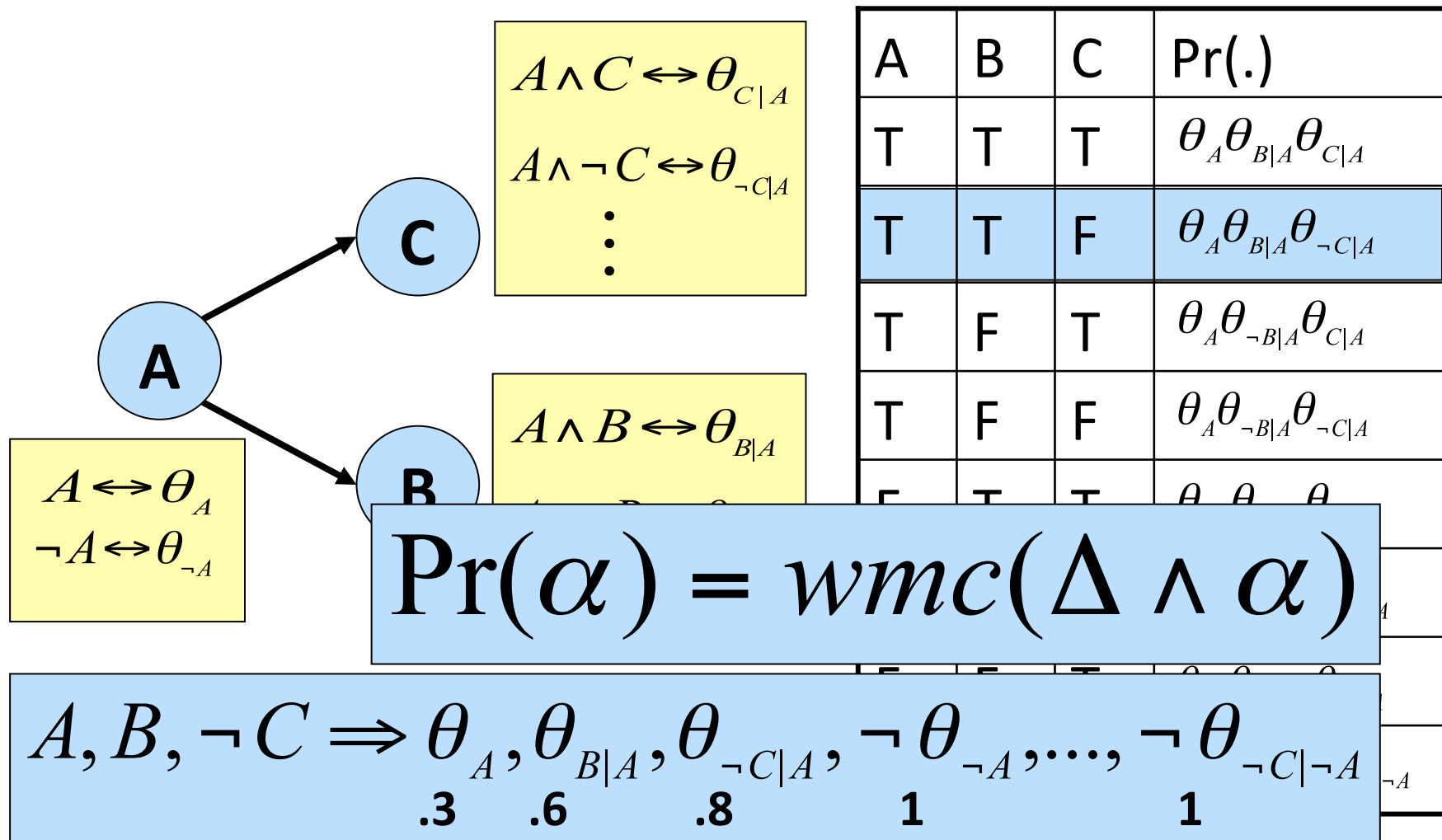
# Probabilistic Inference by Weighted Model Counting



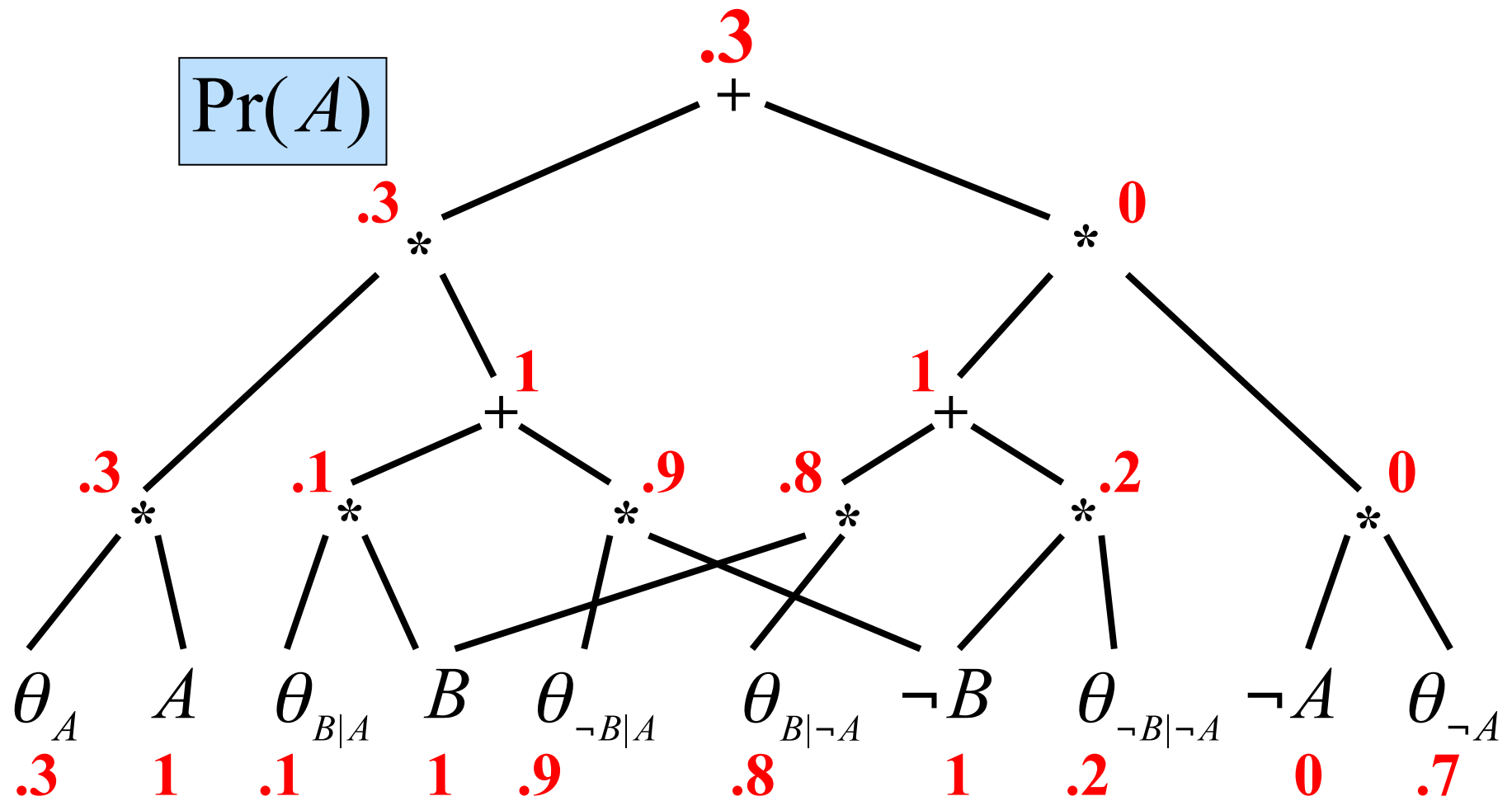
$\Pr(\neg B)$

A	B	C	Pr(.)
T	T	T	$\theta_A \theta_{B A} \theta_{C A}$
T	T	F	$\theta_A \theta_{B A} \theta_{\neg C A}$
T	F	T	$\theta_A \theta_{\neg B A} \theta_{C A}$
T	F	F	$\theta_A \theta_{\neg B A} \theta_{\neg C A}$
F	T	T	$\theta_{\neg A} \theta_{B \neg A} \theta_{C \neg A}$
F	T	F	$\theta_{\neg A} \theta_{B \neg A} \theta_{\neg C \neg A}$
F	F	T	$\theta_{\neg A} \theta_{\neg B \neg A} \theta_{C \neg A}$
F	F	F	$\theta_{\neg A} \theta_{\neg B \neg A} \theta_{\neg C \neg A}$

# Probabilistic Inference by Weighted Model Counting



# Weighted Model Counting (Arithmetic Circuits)





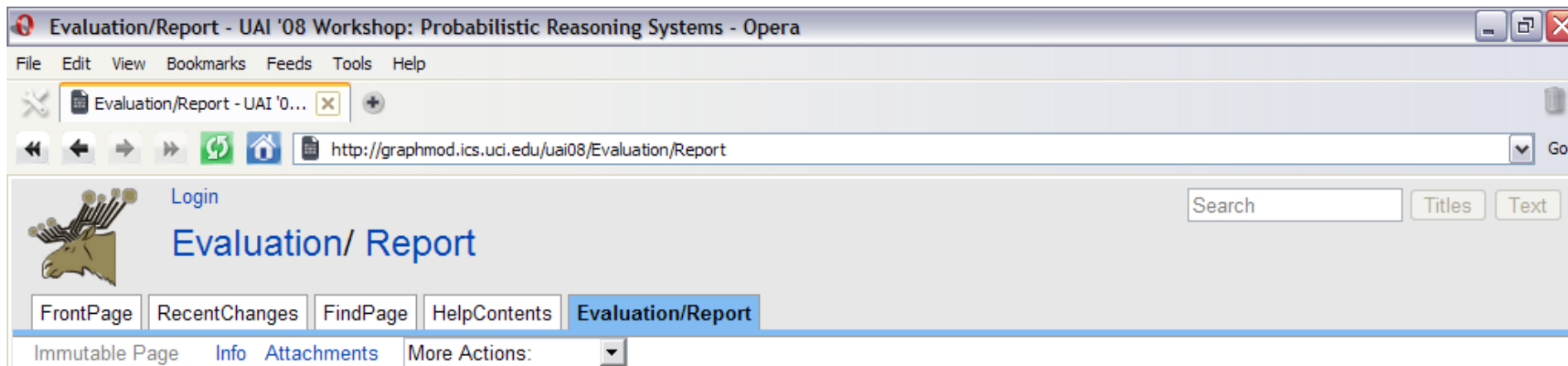
# Why Logic?

- Encoding local structure is easy:
  - Zero-parameters encoded by adding clauses:

$$\theta_{C|A} = 0 \quad \neg A \vee \neg C$$

- Context-specific independence encoded by collapsing variables:

$$\theta_{C|AB} = \theta_{C|A\neg B}$$



## Results from the Probablistic Inference Evaluation of UAI'08

A. Darwiche, R. Dechter, A. Choi, V. Gogate, and L. Otten

This page describes the Probabilistic Inference Evaluation that was held at UAI 2008. The original motivation, goals, and setup are described in the [workshop CFP](#) and the [evaluation CFP](#). Below we give the details of the evaluation setup and the results obtained.

The [organizers](#) wish to thank all the teams and their members for participating in the evaluation. All teams are to be commended as the success of this evaluation was crucially dependent on each and every one of the teams. All the teams did very well (as the results below show). The co-chairs also wish to thank the significant help provided by the graduate student organizers.

The presentation given at UAI'08 is also available here as [ppt slides](#). The information in the presentation is also organized and elaborated on in the pages below.

### The Evaluation Setup

- [Evaluation description](#): This is the original call for participation, which includes the setup of the evaluation and the task definitions.
- [Evaluation environment](#): This describes the system on which the evaluation was performed.
- [Benchmarks and benchmark descriptions](#): This page contains the Bayesian and Markov networks used in the evaluation, available for download, with summary descriptions.

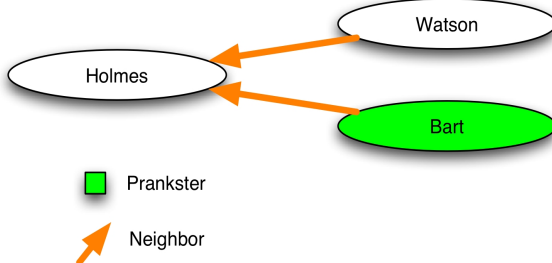
### Participation

- [Solver descriptions](#): a list of participants and solvers evaluated

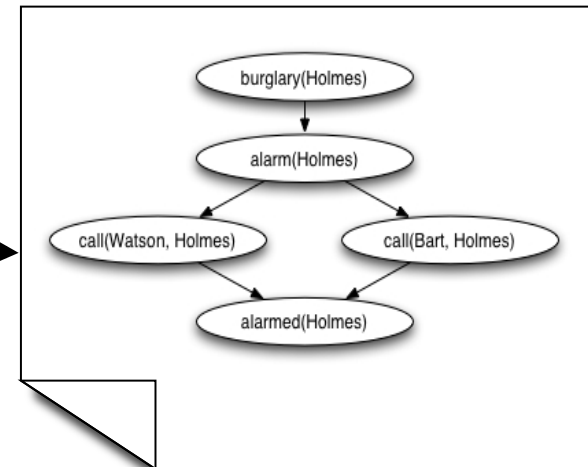
**Workshop page: <http://graphmod.ics.uci.edu/uai08/>**

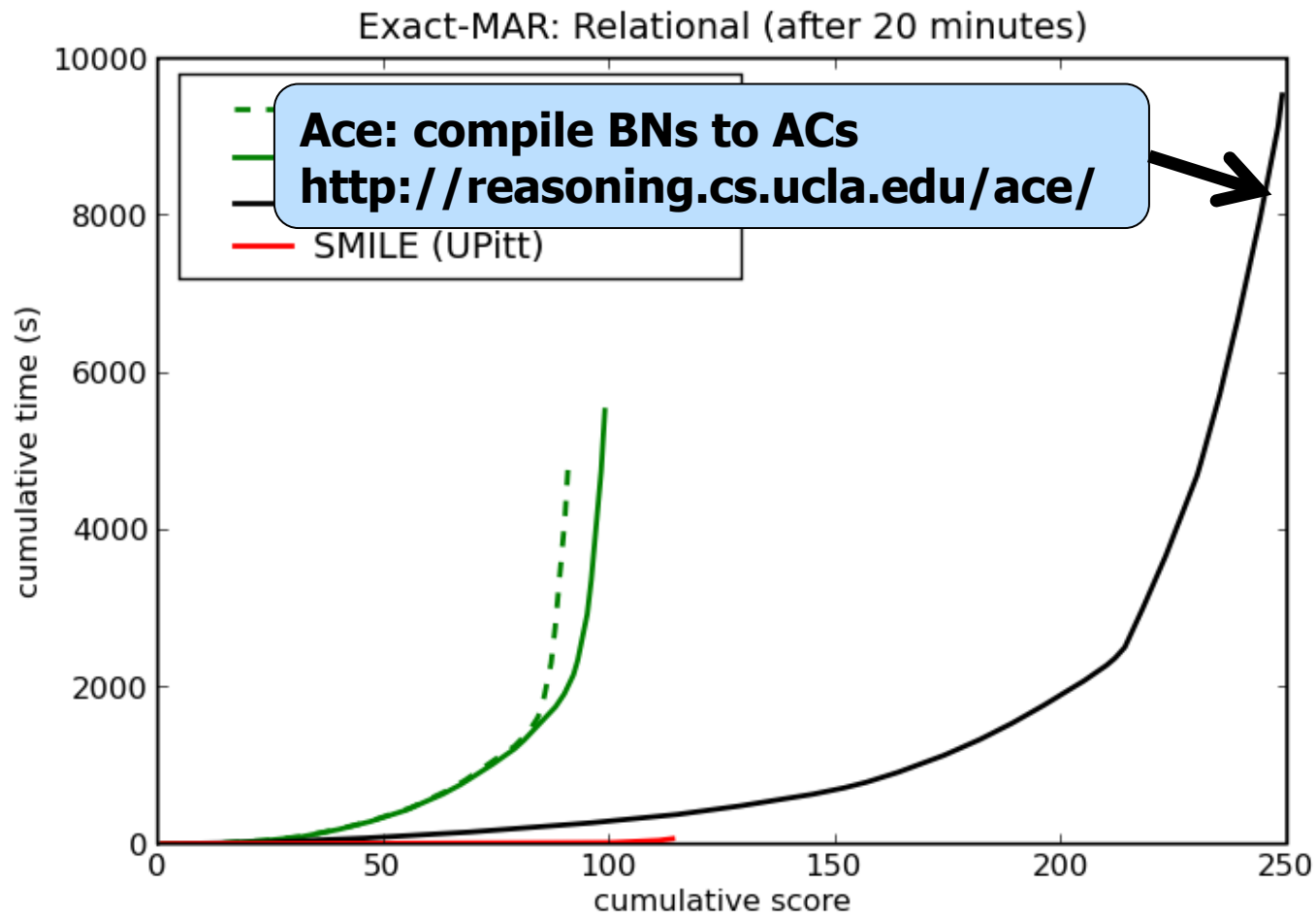
# High Order Specifications: Relational Models...

```
burglary(v)=0.005;  
alarm(v)=(burglary(v):0.95,0.01);  
calls(v,w)=  
  (neighbor(v,w):  
    (prankster(v):  
      (alarm(w):0.9,0.05),  
      (alarm(w):0.9,0)),0);  
alarmed(v)=  
  n-or{calls(w,v) | w:neighbor(w,v)}
```



Primula





- Relational networks (251 networks)
  - Average cluster size is 50

# Alchemy - Open Source AI

[Home](#)

[Documentation](#)

[Tutorial](#)

[User's Manual](#)

[Developer's Manual](#)

[APIs](#)

[Change Log](#)

[FAQs](#)

Welcome to the Alchemy system! Alchemy is a software package providing a series of algorithms for statistical relational learning and probabilistic logic inference, based on the Markov logic representation. Alchemy allows you to easily develop a wide range of AI applications, including:

- Collective classification
- Link prediction
- Entity resolution
- Social network modeling
- Information extraction

If you are not already familiar with Markov logic, we recommend that you first read the paper [Unifying Logical and Statistical AI](#).

[Run](#)

[Requ](#)

[Dow](#)

**Alchemy is a software package providing a series of algorithms for statistical relational learning and probabilistic logic inference, based on Markov logic representations.**

[Mailing Lists](#)

[Alchemy](#)

[Alchemy-announce](#)

[Alchemy-update](#)

[Alchemy-discuss](#)

[Repositories](#)

[Code](#)

[Datasets](#)

[MLNs](#)

[Publications](#)

- Generative weight learning
- Structure learning
- MAP/MPE inference (including memory efficient)
- Probabilistic inference: MC-SAT, Gibbs Sampling, Simulated Tempering, Belief Propagation (including lifted)
- Support for native and linked-in functions
- Block inference and learning over variables with mutually exclusive and exhaustive values
- EM (to handle ground atoms with unknown truth values during learning)
- Specification of indivisible formulas (i.e. formulas that should not be broken up into separate clauses)
- Support of continuous features and domains
- Online inference
- Decision Theory

In the next release we plan to include:

- Online learning
- Exact inference for small domains

[Contributors](#)

# Current Challenges

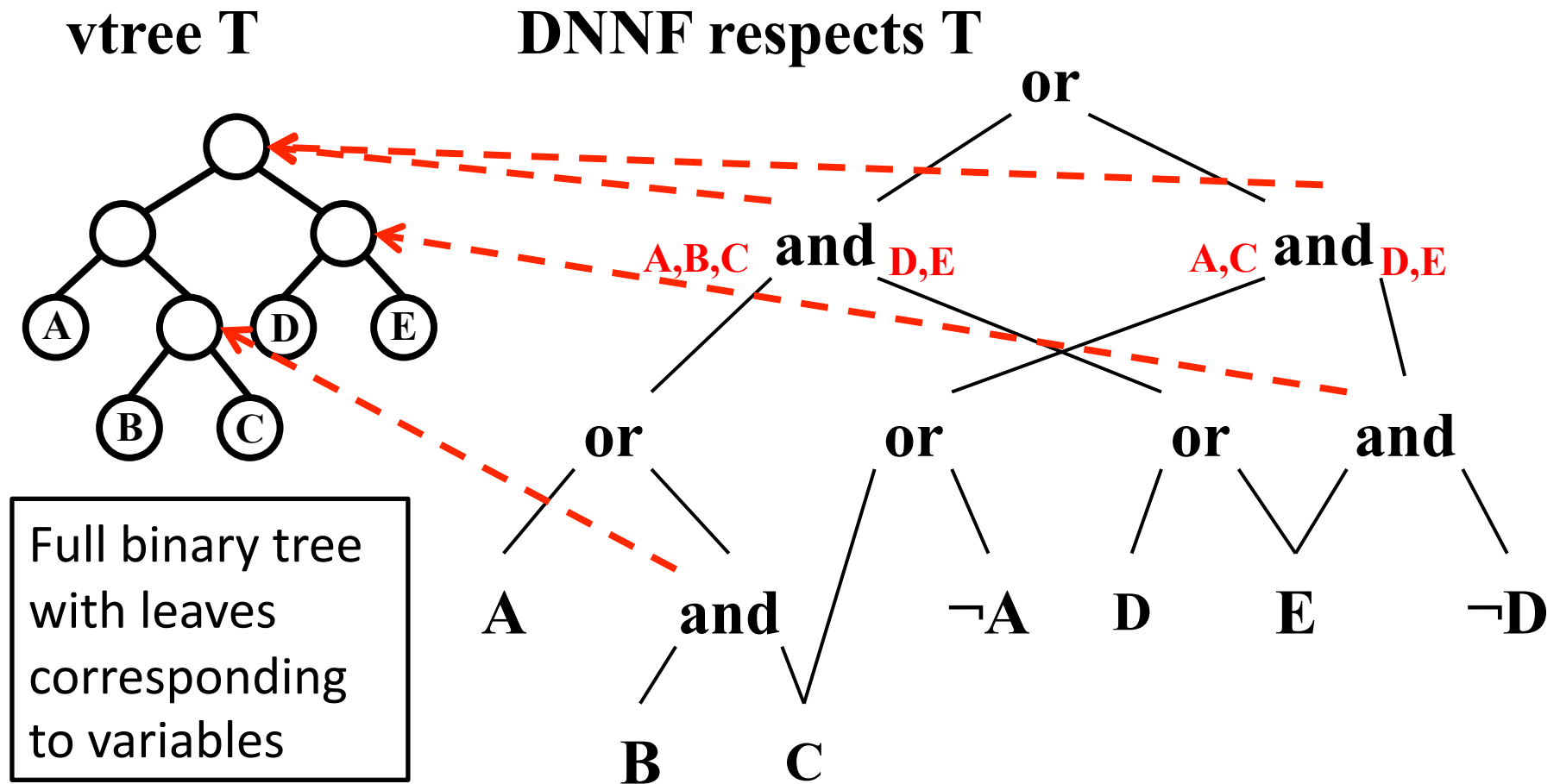
- **Incremental compilation:**

- What? Current compilers monolithic: c2d (UCLA) and DSharp (Toronto)
- Need:
  - Logic: planning and verification applications
  - Probability: approximate inference
- Main insight:
  - Structured decomposability & vtrees (AAAI-08, AAAI-10)

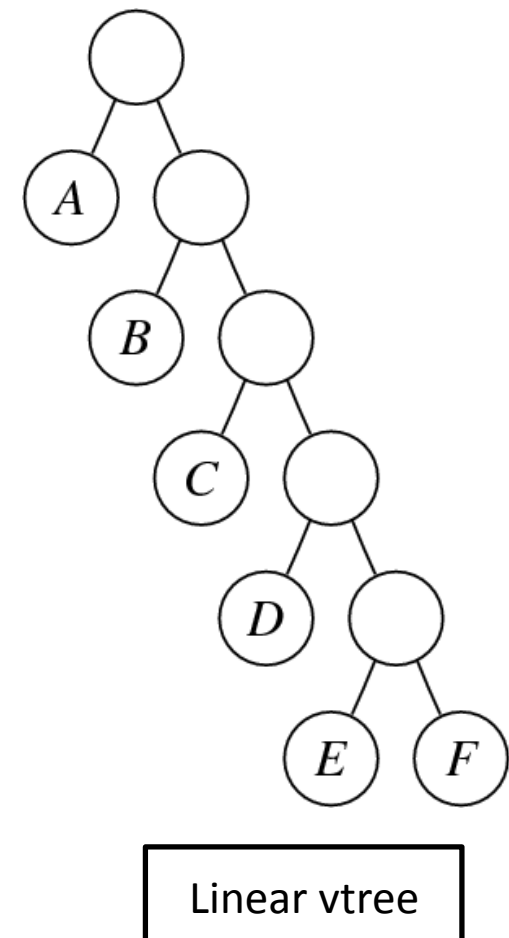
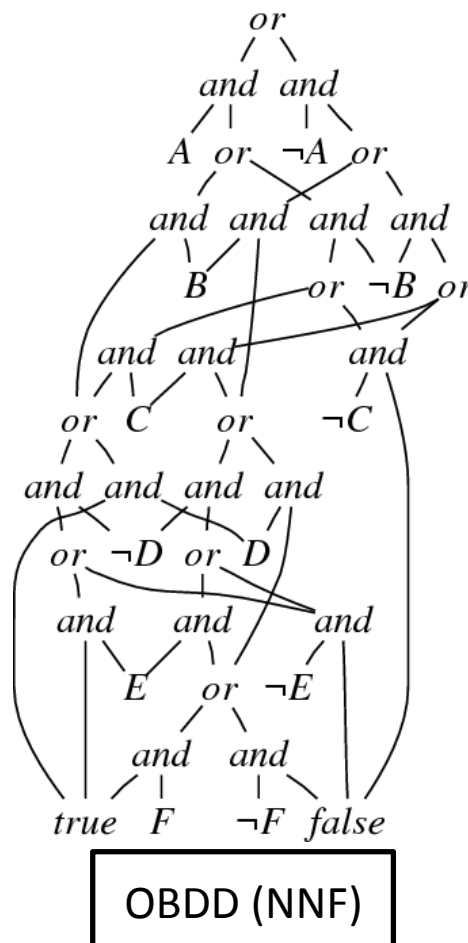
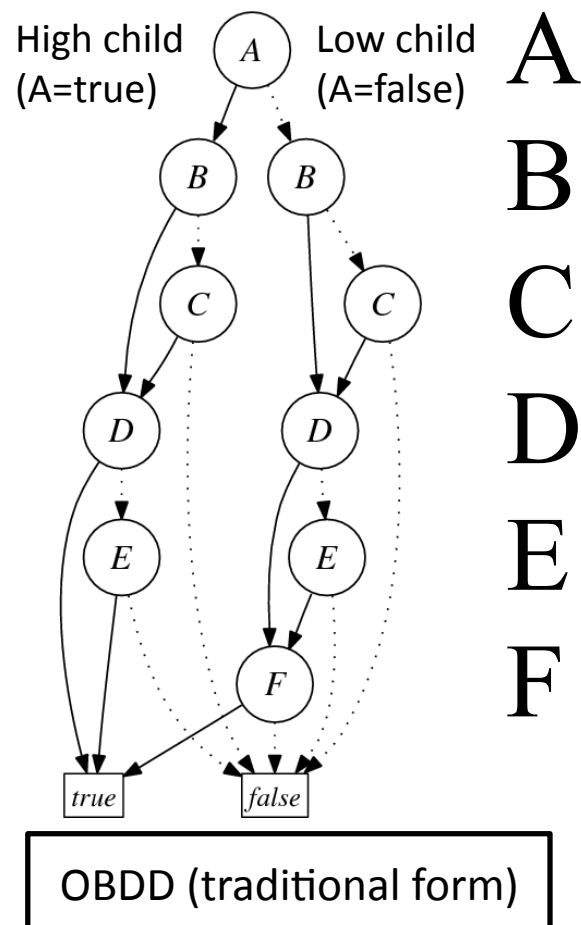
- **Guarantees and Complexity results:**

- Upper & lower bounds on size of compilation (AAAI-10, ECAI-10)
- Main insights:
  - The notion of a decomposition (AAAI-10)
  - The notion of an interaction function (ECAI-10)

# Structured Decomposability



# OBDD: DNNF that Respects Linear vtree





## Decomposition of Boolean Functions (AAAI-10)

- Examples:  $f = (X_1 \vee X_2) \wedge (Y_1 \vee X_2) \wedge (X_1 \vee Y_2) \wedge (Y_1 \vee Y_2) \vee (X_2 \wedge Y_3)$   
 –  $\mathbf{X} = \{X_1, X_2\}$ ,  $\mathbf{Y} = \{Y_1, Y_2, Y_3\}$ :

~~$$f(\mathbf{X}, \mathbf{Y}) = g(\mathbf{X}) \wedge h(\mathbf{Y})$$~~

$$f(\mathbf{X}, \mathbf{Y}) = f_1 \vee f_2 \vee f_3 \vee \dots \vee f_m$$

$\swarrow \quad \searrow$   
 $g_1(\mathbf{X}) \wedge h_1(\mathbf{Y})$

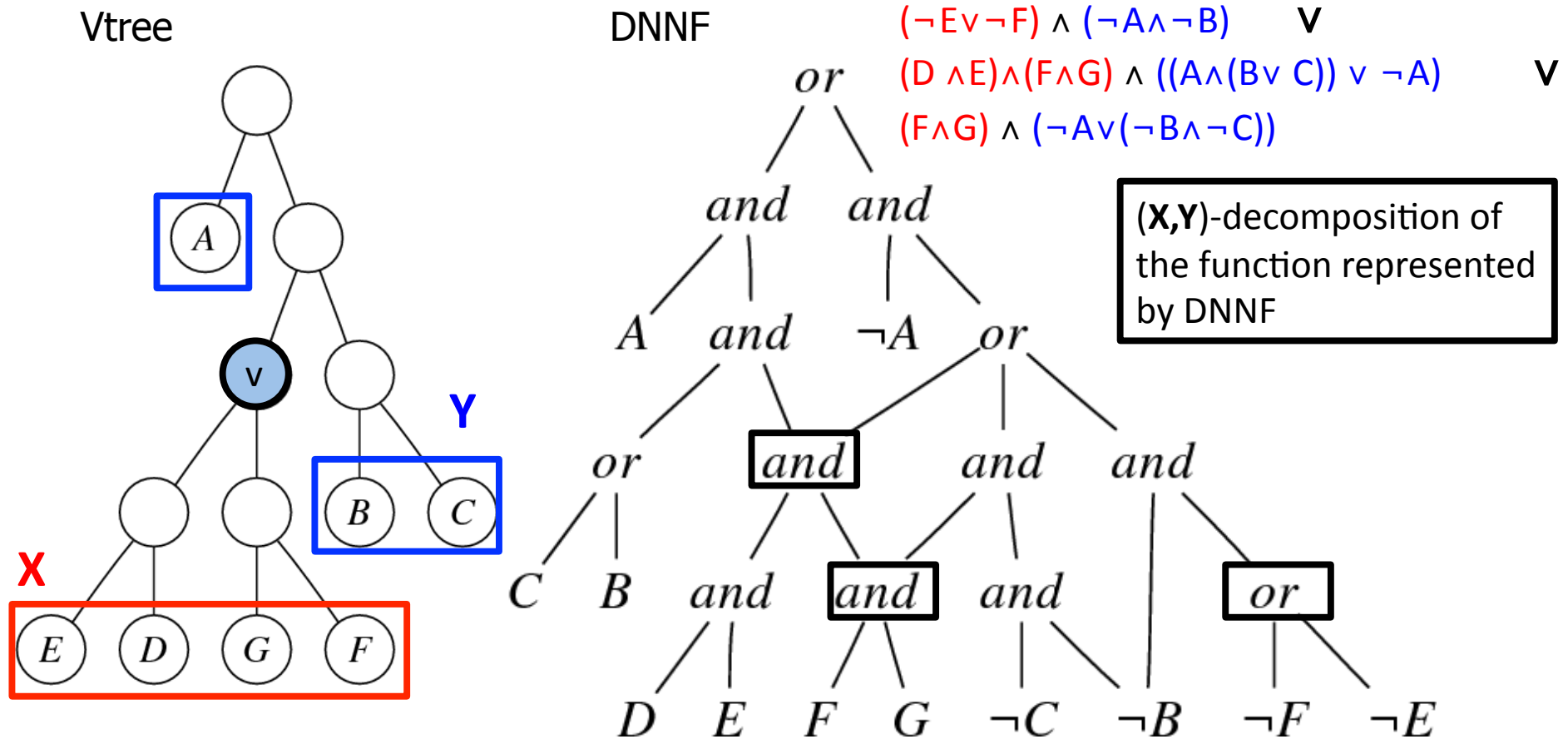
$\swarrow \quad \searrow$   
 $g_2(\mathbf{X}) \wedge h_2(\mathbf{Y})$

$\swarrow \quad \searrow$   
 $g_3(\mathbf{X}) \wedge h_3(\mathbf{Y})$

$\swarrow \quad \searrow$   
 $g_m(\mathbf{X}) \wedge h_m(\mathbf{Y})$

$(\mathbf{X}, \mathbf{Y})$ -decomposition of  $f$

# Lower Bounds (AAAI-10)



# The Interaction Function (ECAI-10)

$$f(\mathbf{X}, \mathbf{Y}) = g(\mathbf{X}) \wedge h(\mathbf{Y}) \wedge l(\mathbf{X}, \mathbf{Y})$$

Captures precisely  
knowledge about  
variables in  $\mathbf{X}$

$\exists \mathbf{Y} f$

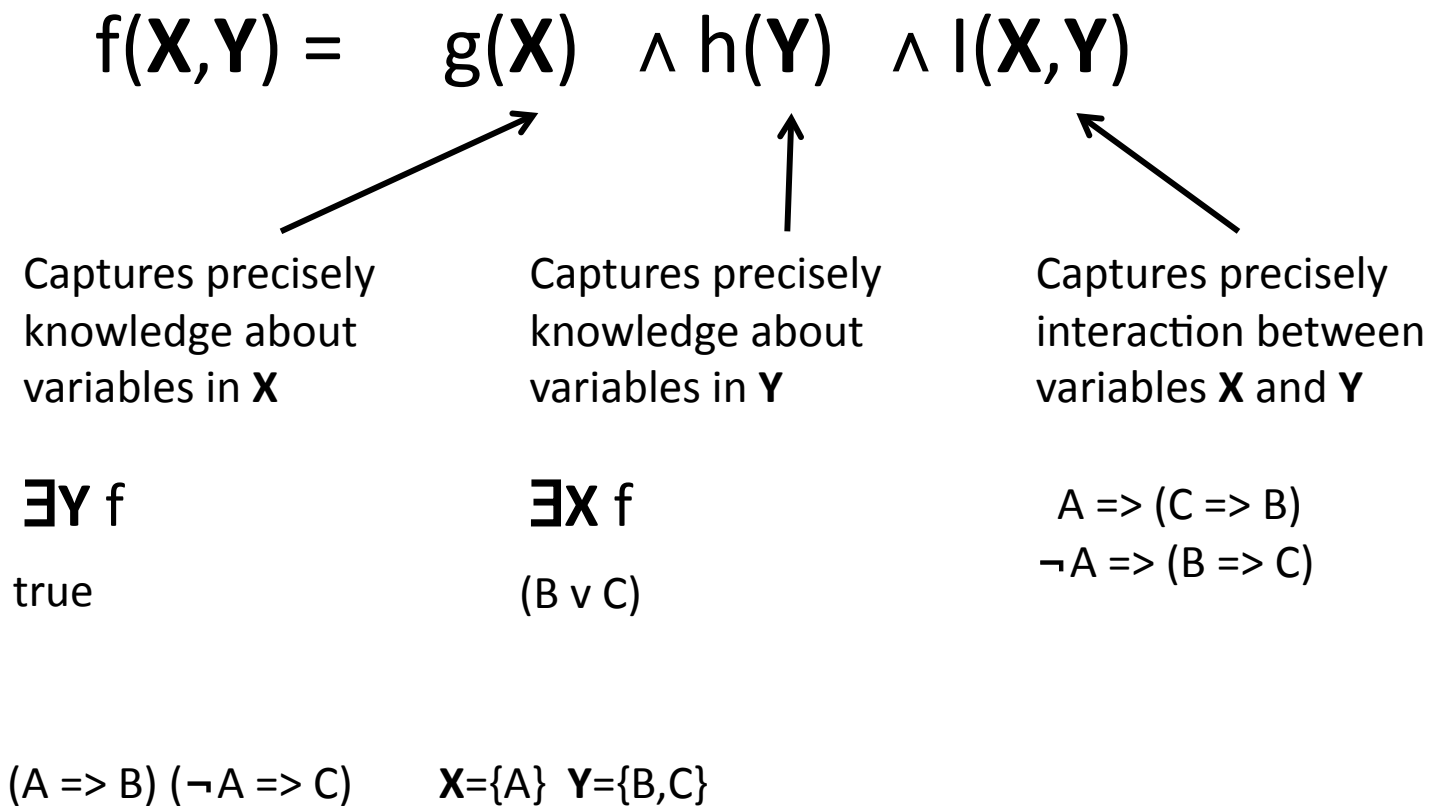
Captures precisely  
knowledge about  
variables in  $\mathbf{Y}$

$\exists \mathbf{X} f$

Captures precisely  
interaction between  
variables  $\mathbf{X}$  and  $\mathbf{Y}$

$$f_{\mathbf{XY}} = f \vee \neg(\exists \mathbf{X} f) \vee \neg(\exists \mathbf{Y} f)$$

# The Interaction Function (ECAI-10)



# Current Research

- Searching for good vtrees (on-going)
- Characterizing and searching for optimal decompositions
- Upper and lower bounds on size of DNNF
- Key objective: incremental compiler for DNNF and d-DNNF

# Two Main Themes

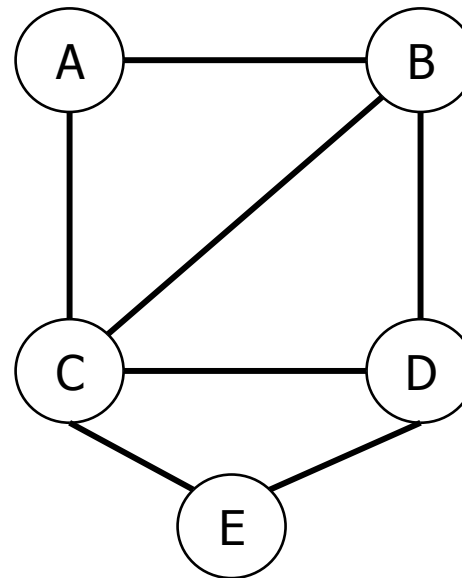
- **Exact inference as:**  
Enforcing decomposability and determinism on propositional knowledge bases
- **Approximate inference as:**  
Relaxing, compensating for, and recovering equivalence constraints (equalities)

# Treewidth

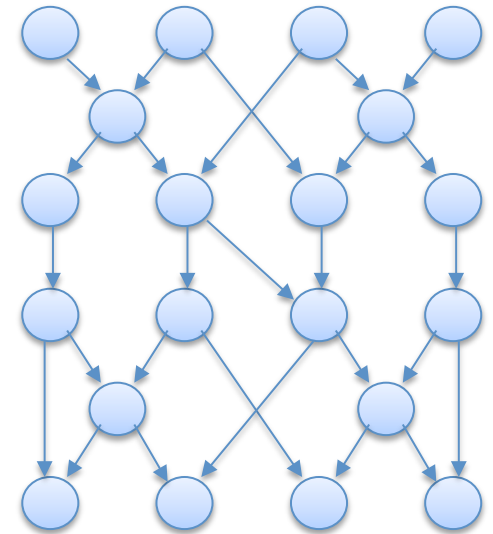
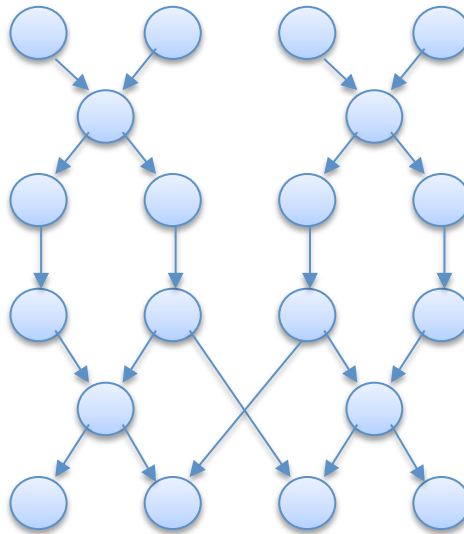
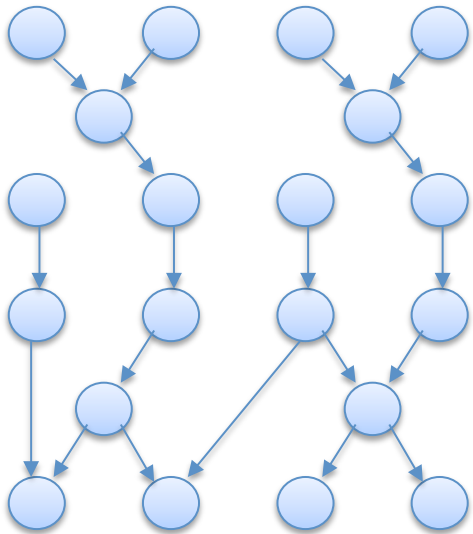
CNF

$(A \vee B \vee \neg C) \wedge$   
 $(A \vee \neg B \vee C) \wedge$   
 $(C \vee \neg D \vee E) \wedge$   
 $(B \vee D) \wedge$   
 $(D \vee E)$

Constraint Graph



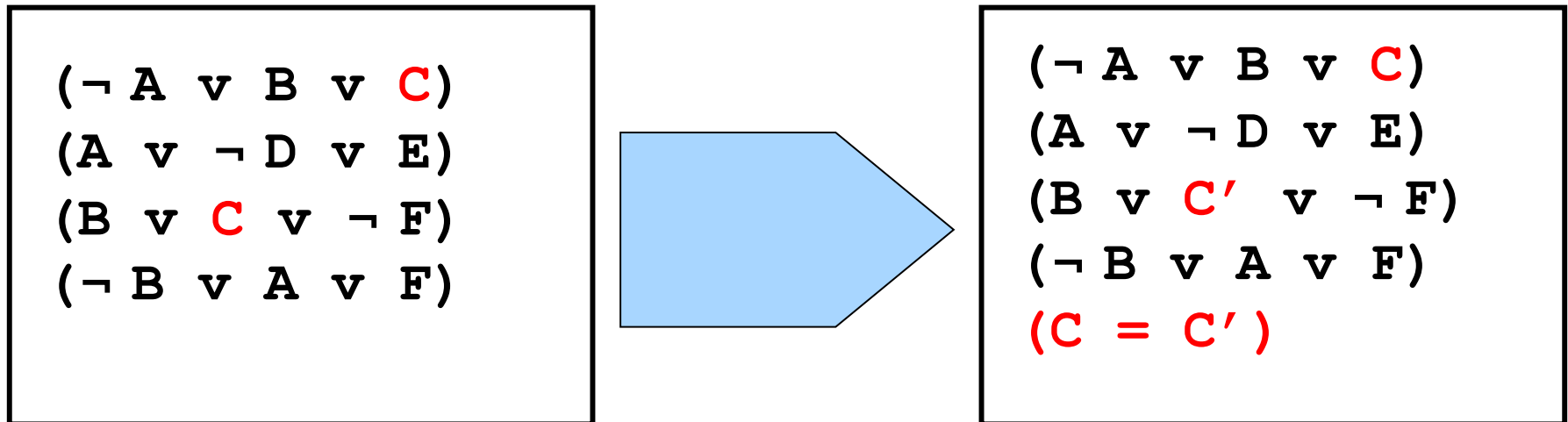
# Treewidth



$$O(n \exp \{w\})$$



# Equivalence Constraints



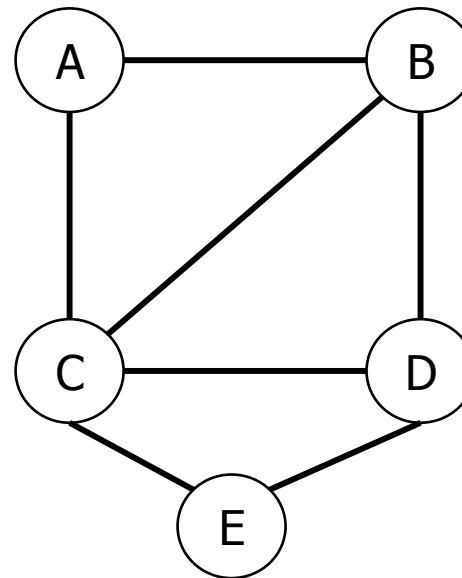
Relaxing equivalence constraints reduces treewidth

# Treewidth

CNF

$(A \vee B \vee \neg C) \wedge$   
 $(A \vee \neg B \vee C) \wedge$   
 $(C \vee \neg D \vee E) \wedge$   
 $(B \vee D) \wedge$   
 $(D \vee E)$

Constraint Graph

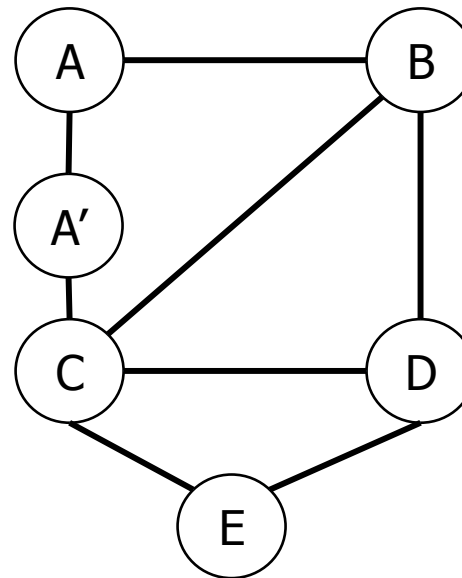


# Treewidth

CNF

$(\mathbf{A} \vee B \vee \neg C) \wedge$   
 $(\mathbf{A}' \vee \neg B \vee C) \wedge$   
 $(C \vee \neg D \vee E) \wedge$   
 $(B \vee D) \wedge$   
 $(D \vee E) \wedge$   
 $(\mathbf{A} = \mathbf{A}')$

Constraint Graph

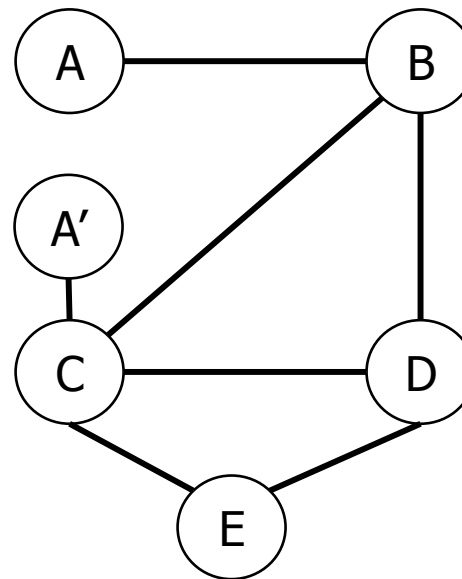


# Treewidth

CNF

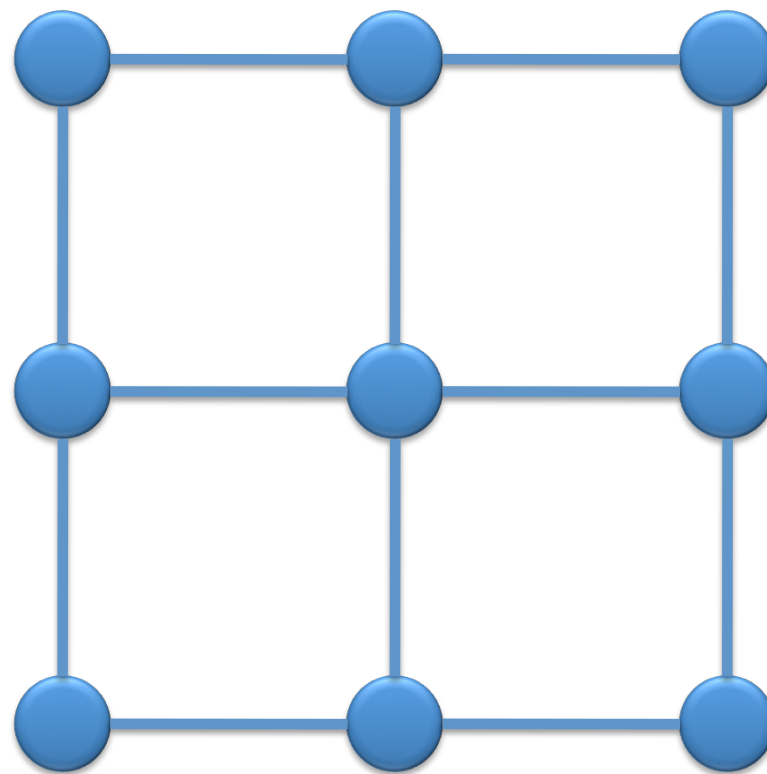
$(\mathbf{A} \vee B \vee \neg C) \wedge$   
 $(\mathbf{A}' \vee \neg B \vee C) \wedge$   
 $(C \vee \neg D \vee E) \wedge$   
 $(B \vee D) \wedge$   
 $(D \vee E)$

Constraint Graph



# Equivalence Constraints

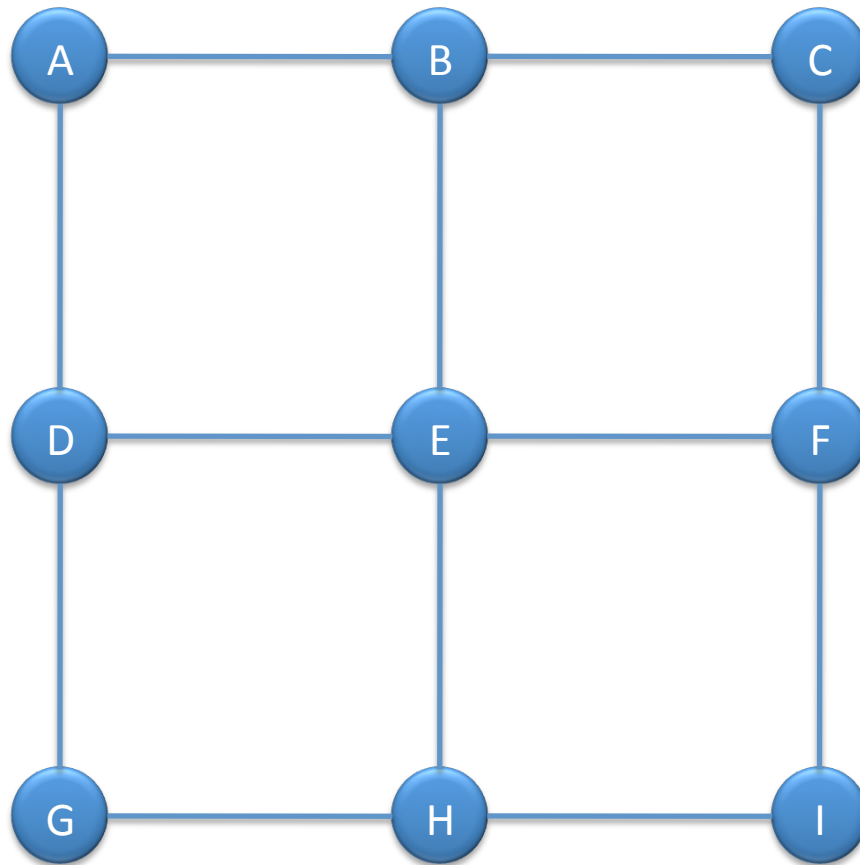
$$\psi_{eq}(X_i=x_i, X_j=x_j) = \begin{cases} 1 & \text{if } x_i = x_j \\ 0 & \text{otherwise} \end{cases}$$



$$\Pr(x_1, \dots, x_n) = \frac{1}{Z} \psi(x_1, x_5) \dots \psi(x_2, x_4) \dots \theta(x_1) \dots \theta(x_n)$$

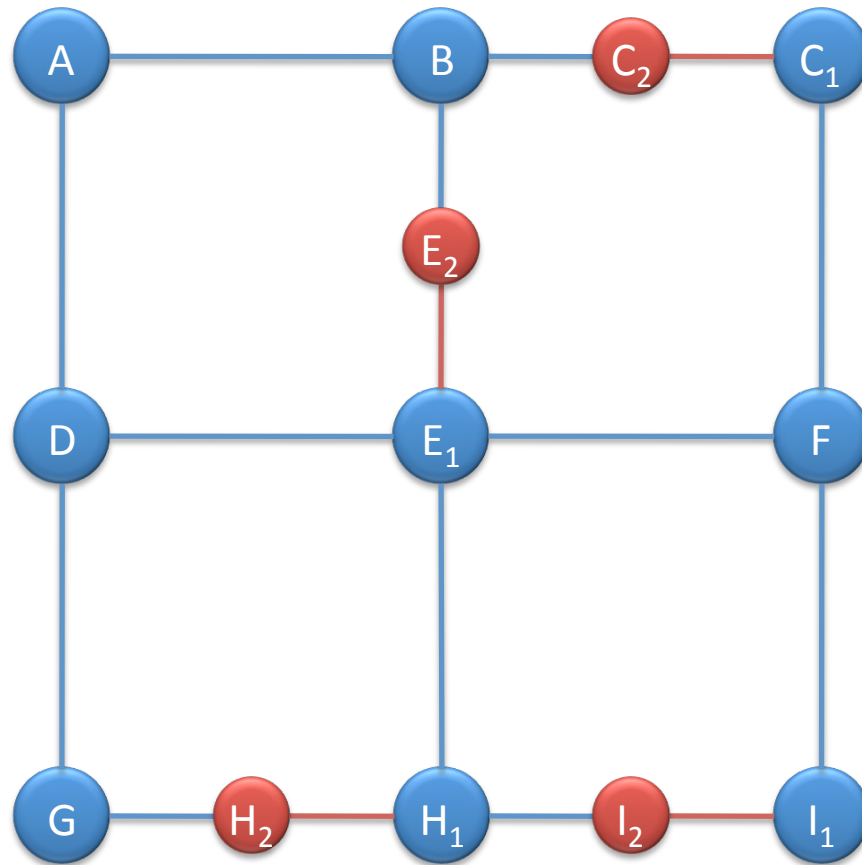
# Relaxing Equivalence Constraints

- Model  $\mathcal{M}$



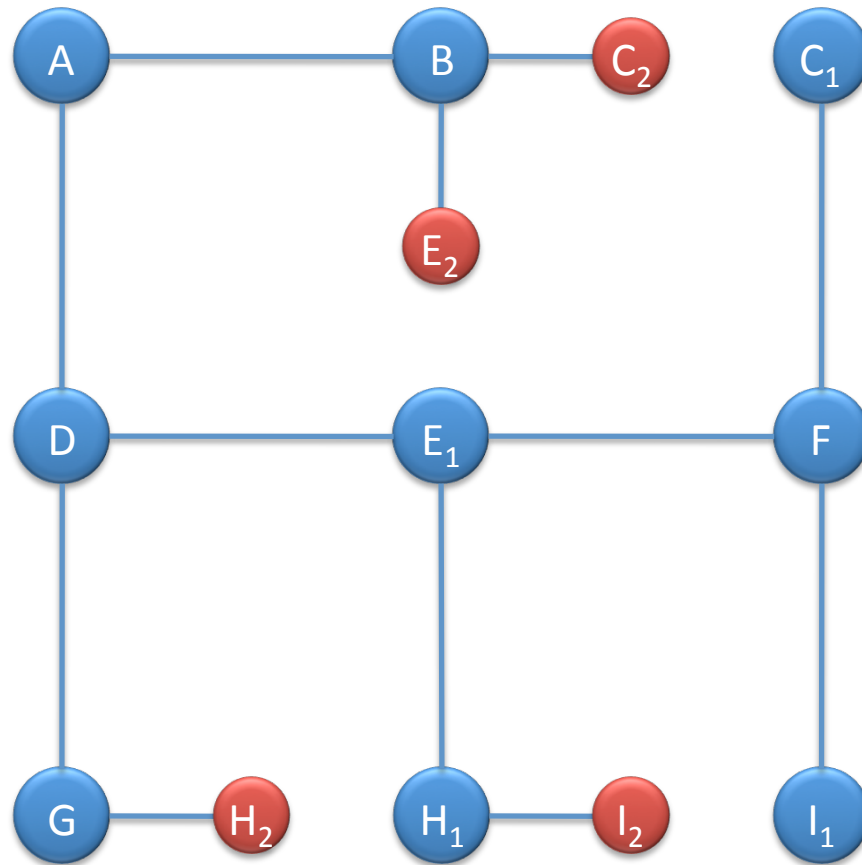
# Relaxing Equivalence Constraints

- Model + Eq.



# Relaxing Equivalence Constraints

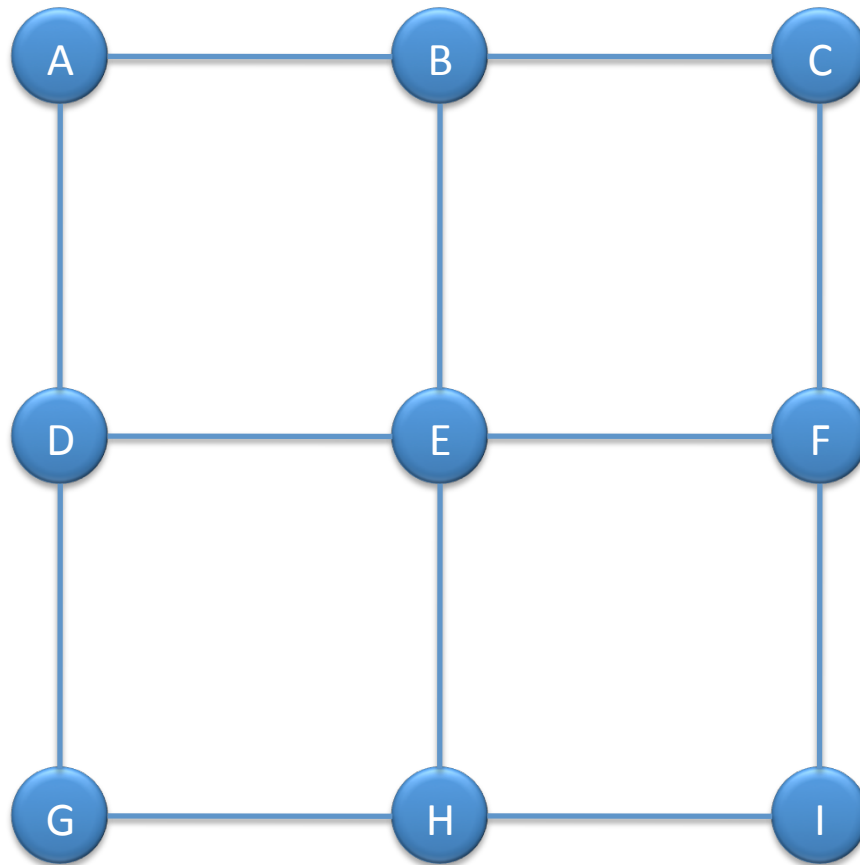
- Relaxed
- Treewidth 1





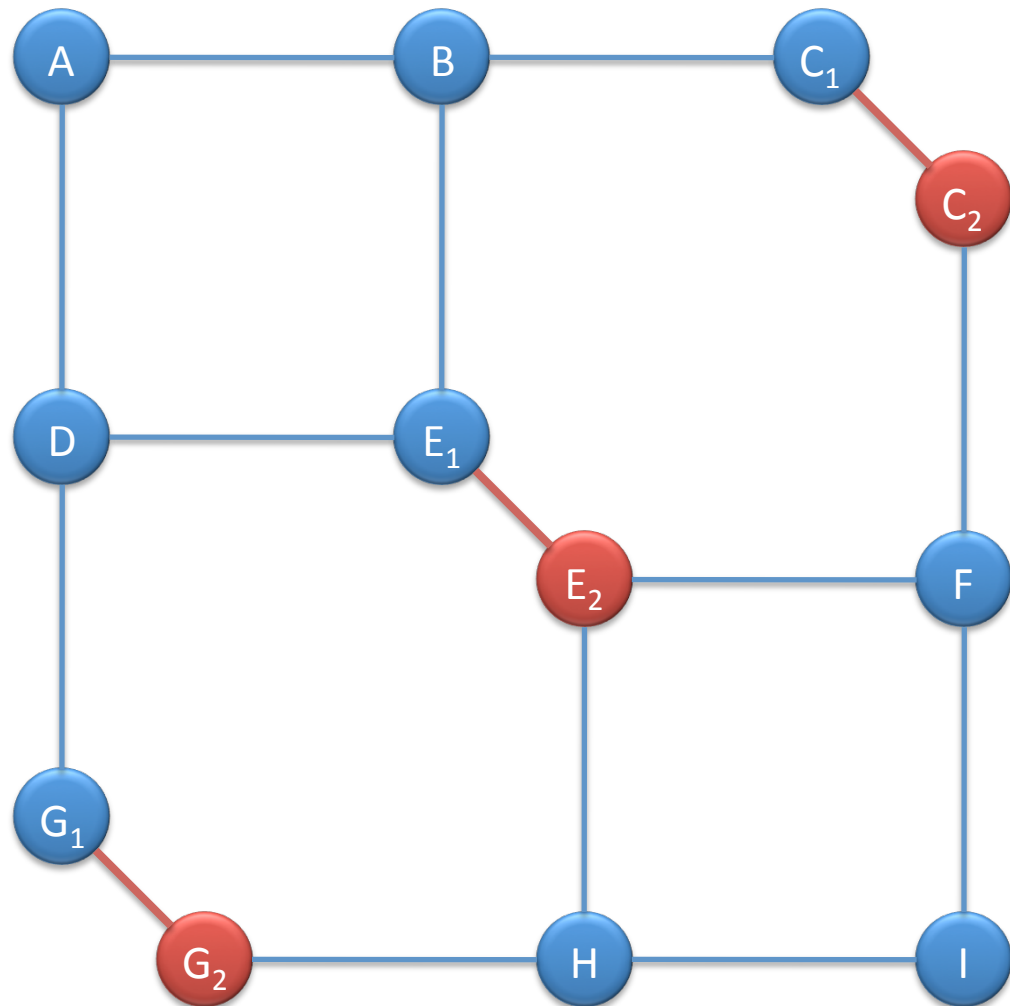
# Relaxing Equivalence Constraints

- Model  $\mathcal{M}$



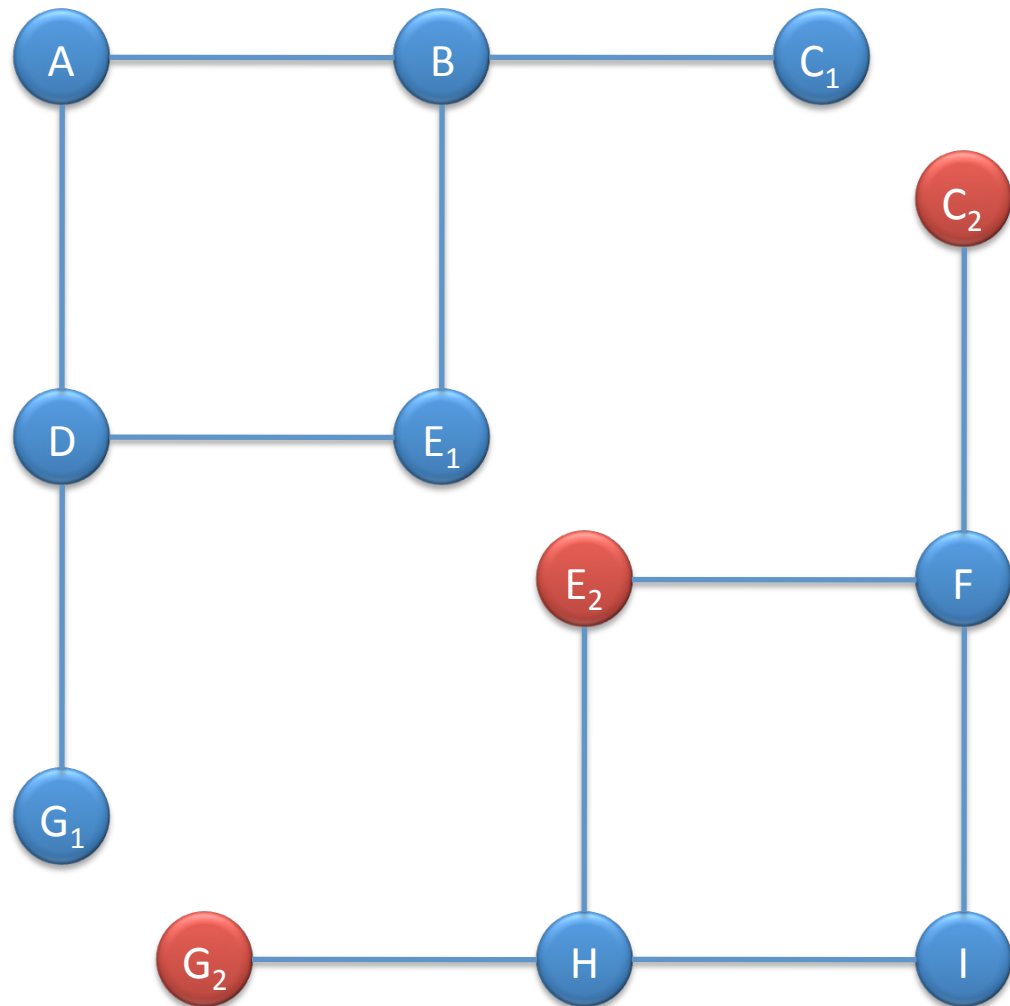
# Relaxing Equivalence Constraints

- Model + Eq.

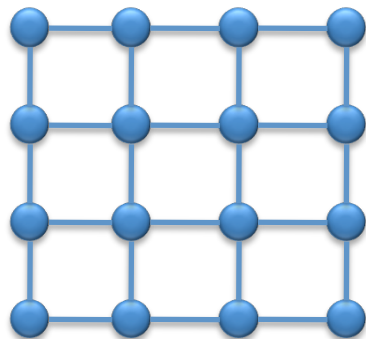


# Relaxing Equivalence Constraints

- Relaxed
- Decomposed

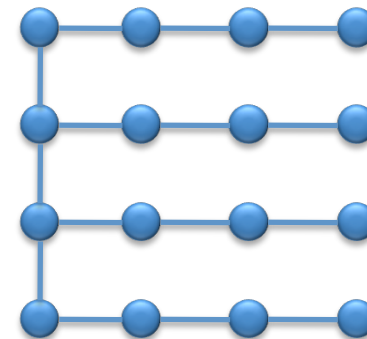


## Model + Eq



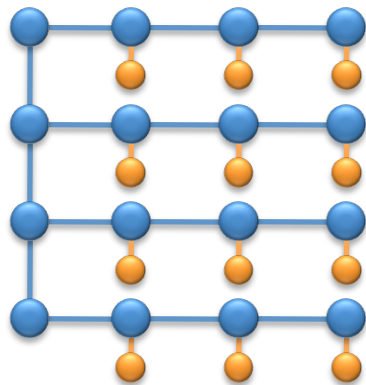
Intractable model, augmented with equivalence constraints

## Relax



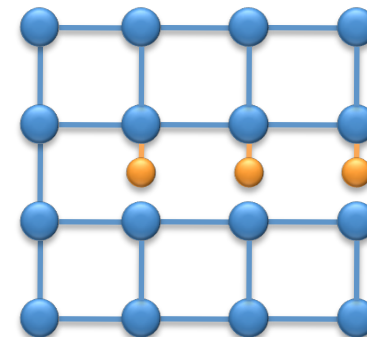
Simplify network structure:  
Relax equivalence constraints

## Compensate



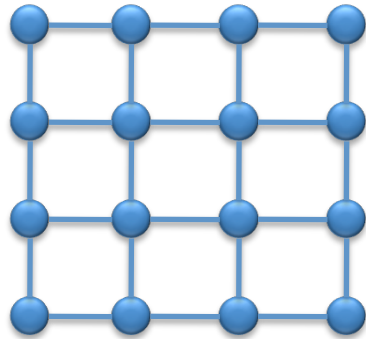
Compensate for relaxation:  
Restore a weaker equivalence

## Recover



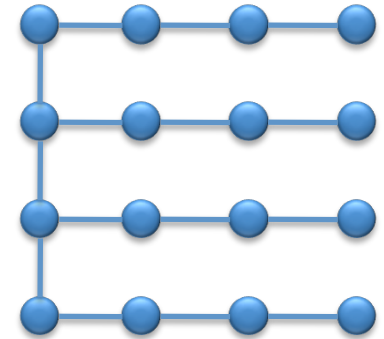
Recover structure, identify  
an improved approximation

## Model + Eq



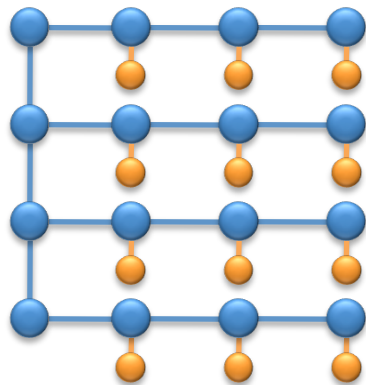
Intractable model, augmented with equivalence constraints

## Relax



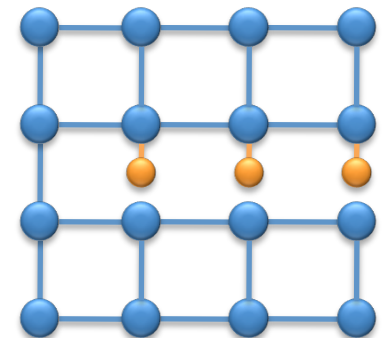
Usually gives upper/lower bounds: mini-buckets, MAXSAT

## Compensate



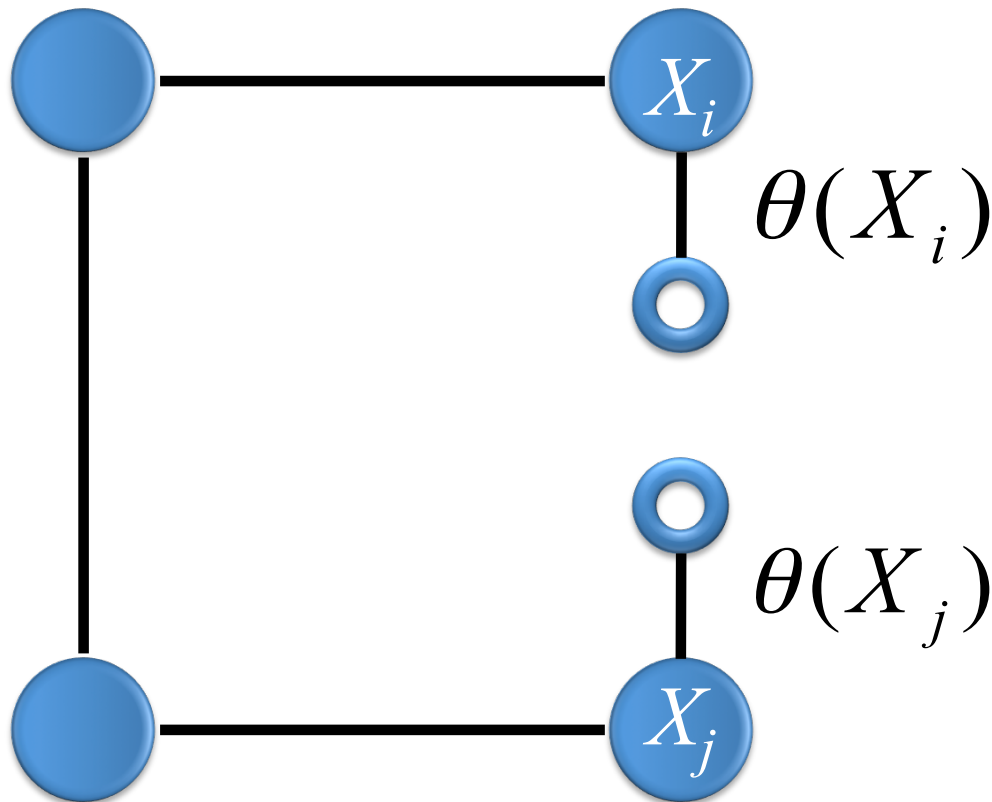
Compensate for relaxation:  
Restore a weaker equivalence

## Recover



Recover structure, identify an improved approximation

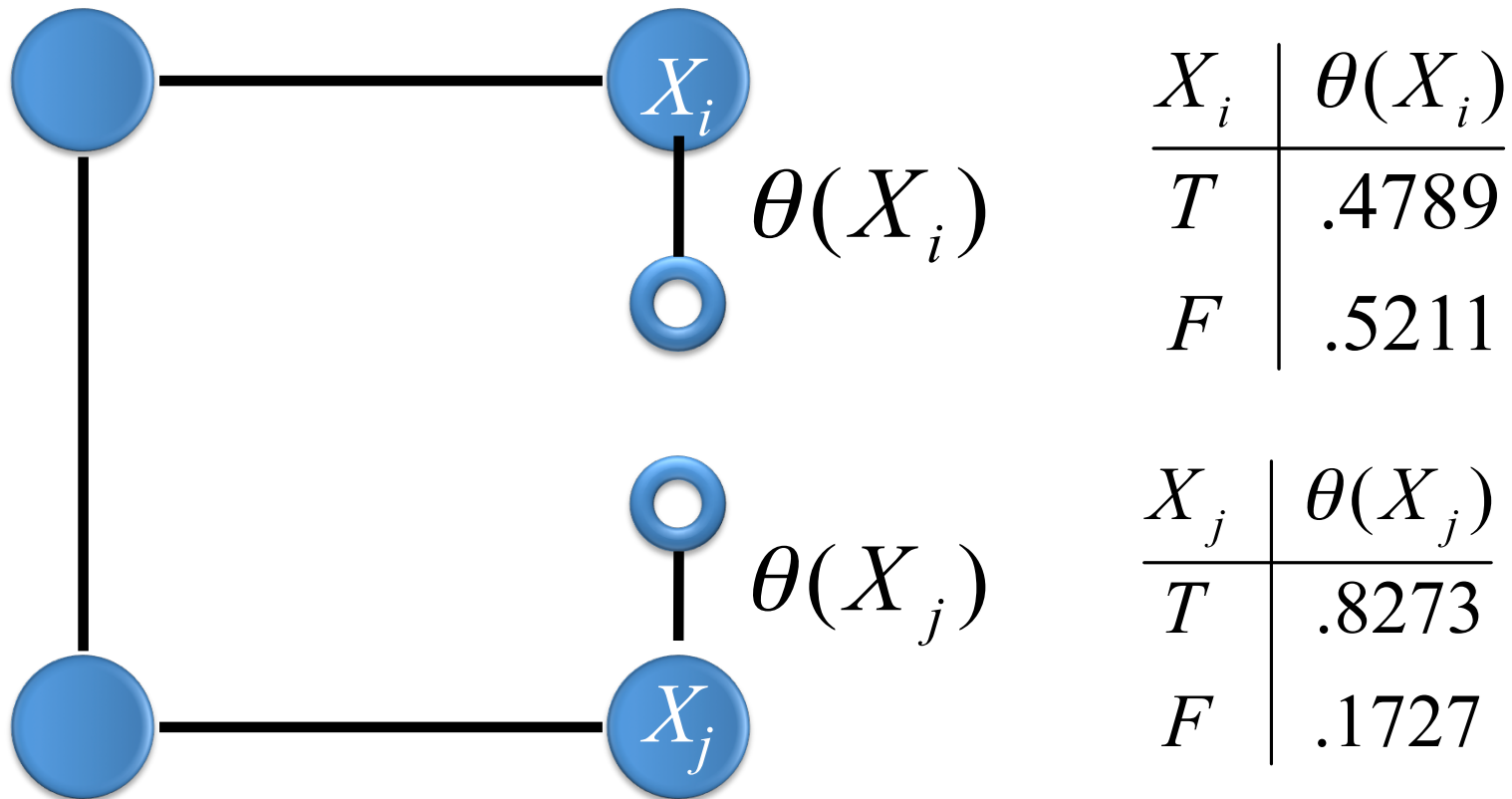
# Compensating for an Equivalence



$X_i$	$\theta(X_i)$
$T$	.4789
$F$	.5211

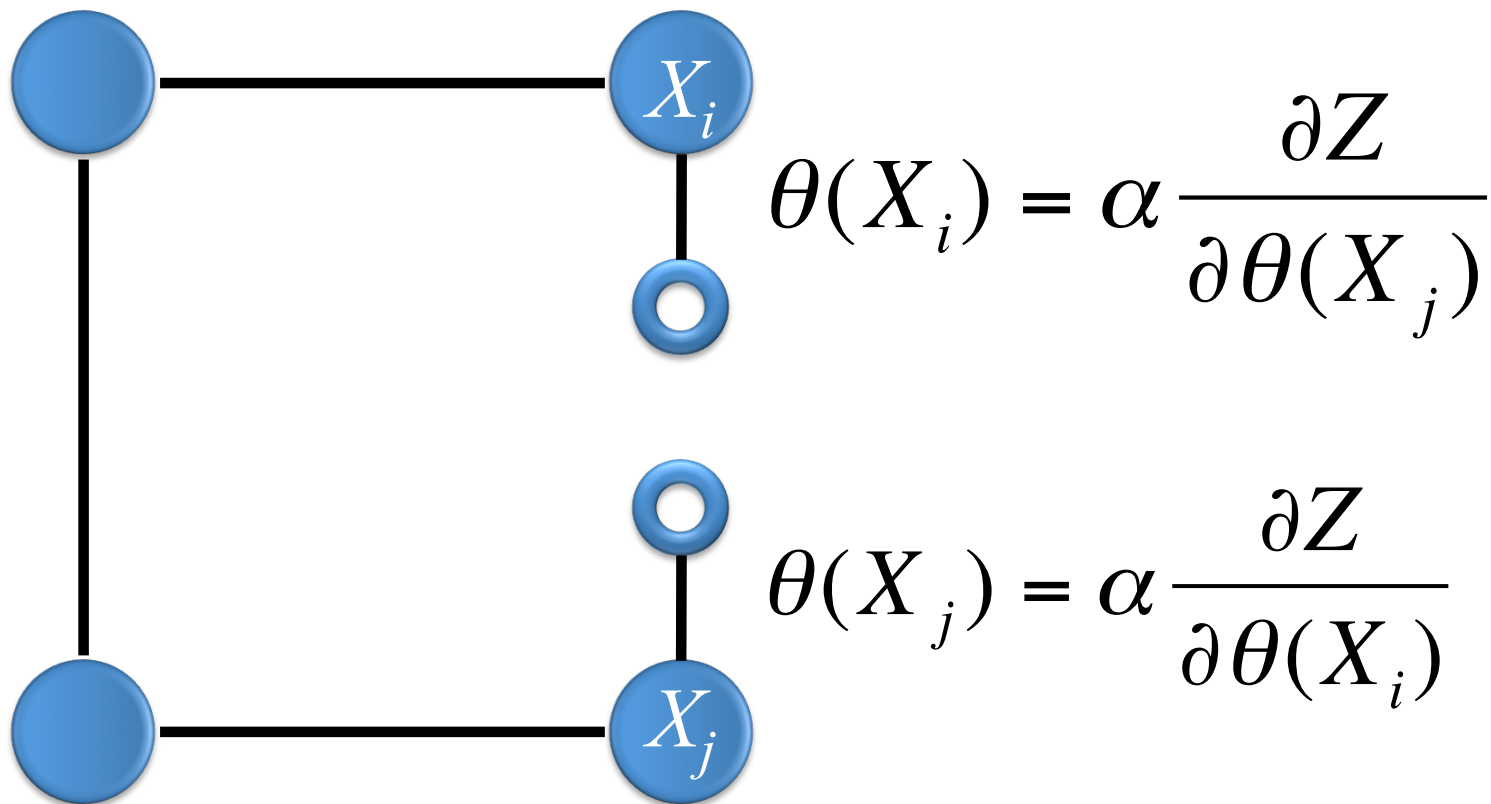
$X_j$	$\theta(X_j)$
$T$	.8273
$F$	.1727

# Compensating for an Equivalence



$$\Pr(X_i = x) = \Pr(X_j = x)$$

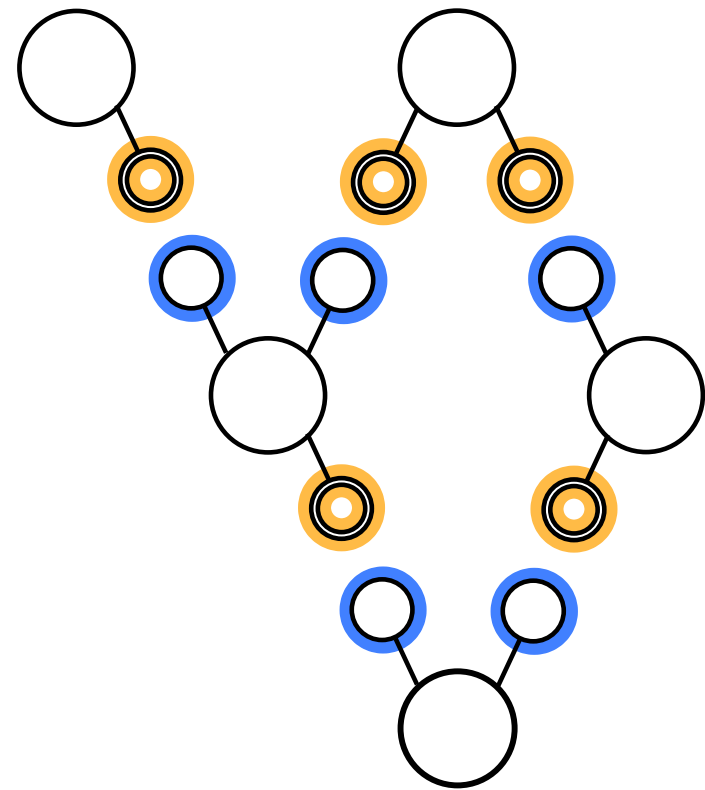
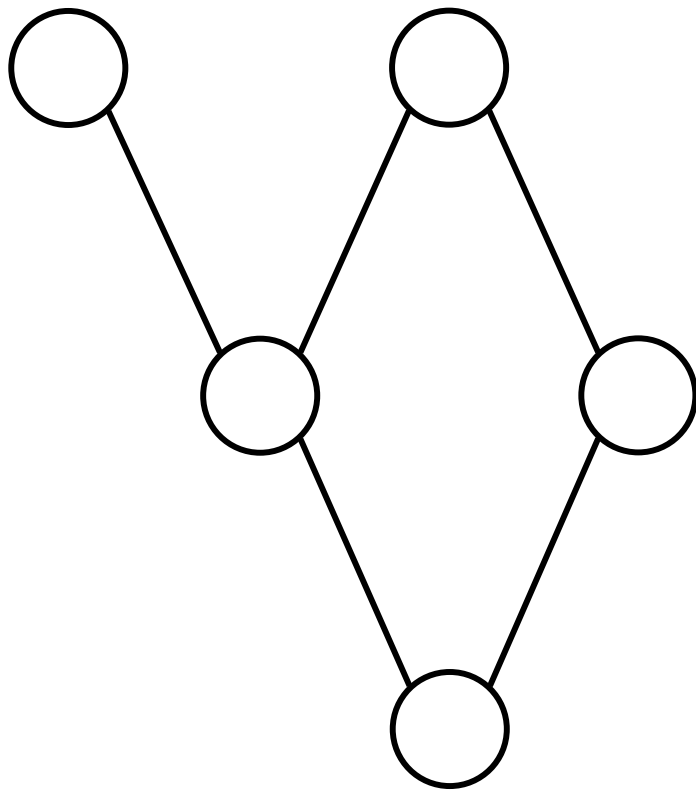
## Compensating for an Equivalence



$$\Pr(X_i = x) = \Pr(X_j = x)$$



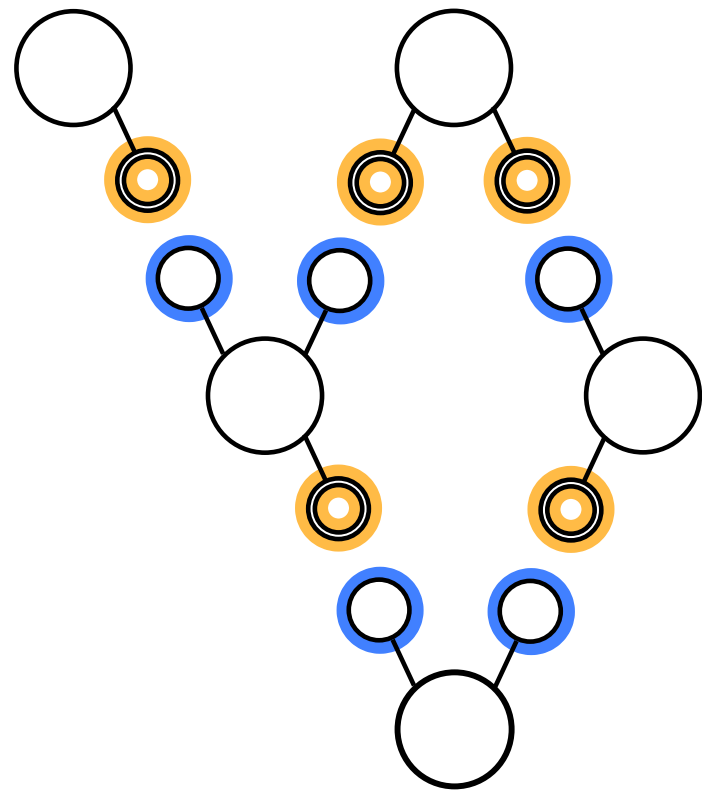
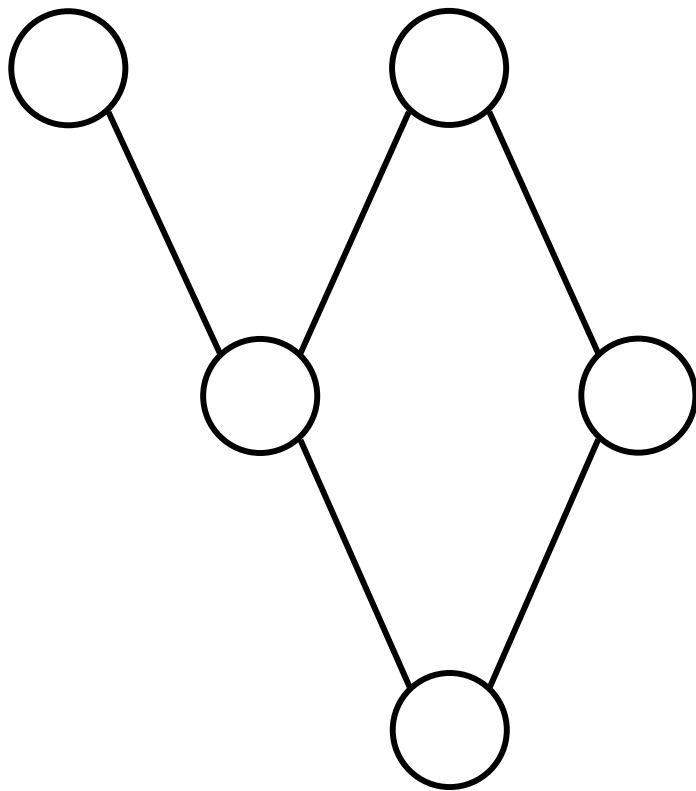
# Parametrizing Edges Iteratively



Iteration  $t = 0$

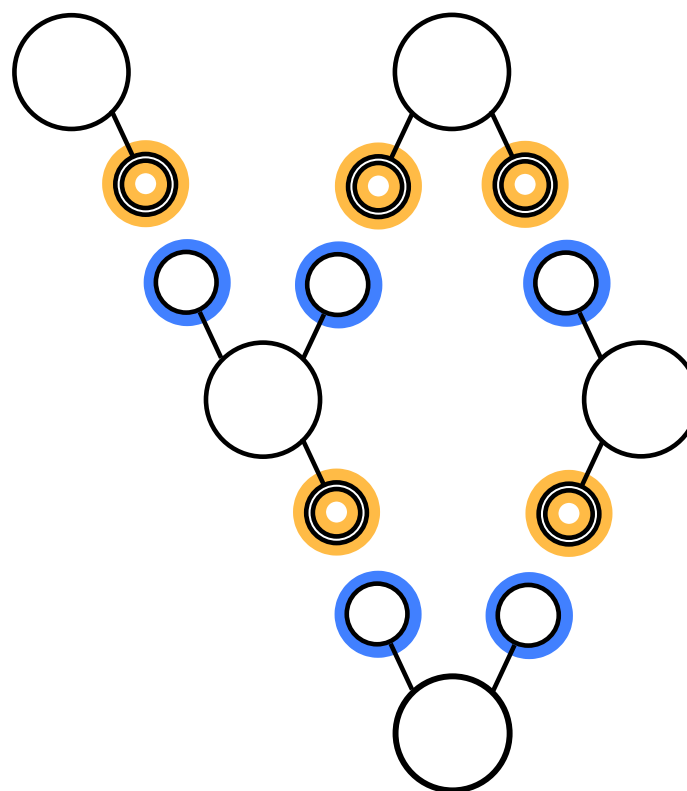
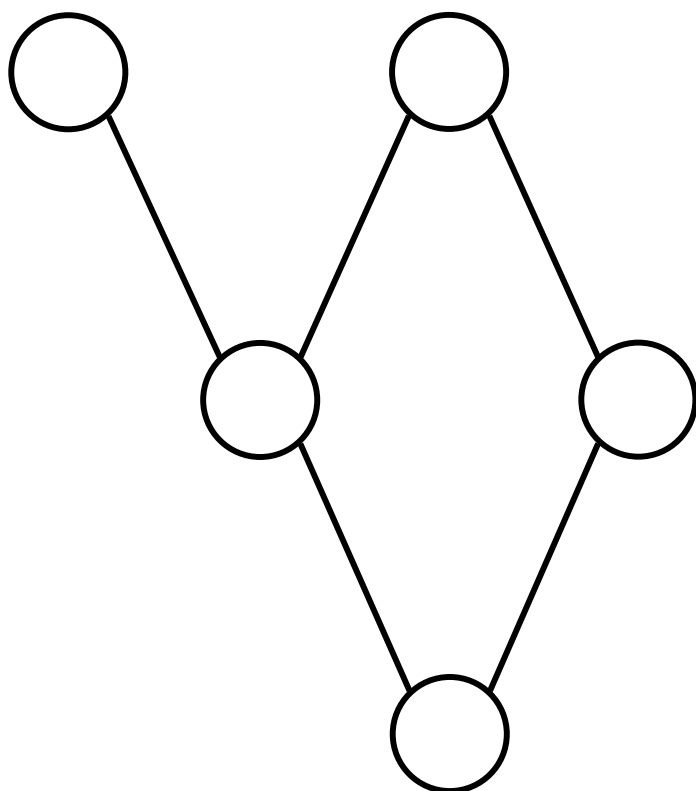
# Initialization

# Parametrizing Edges Iteratively



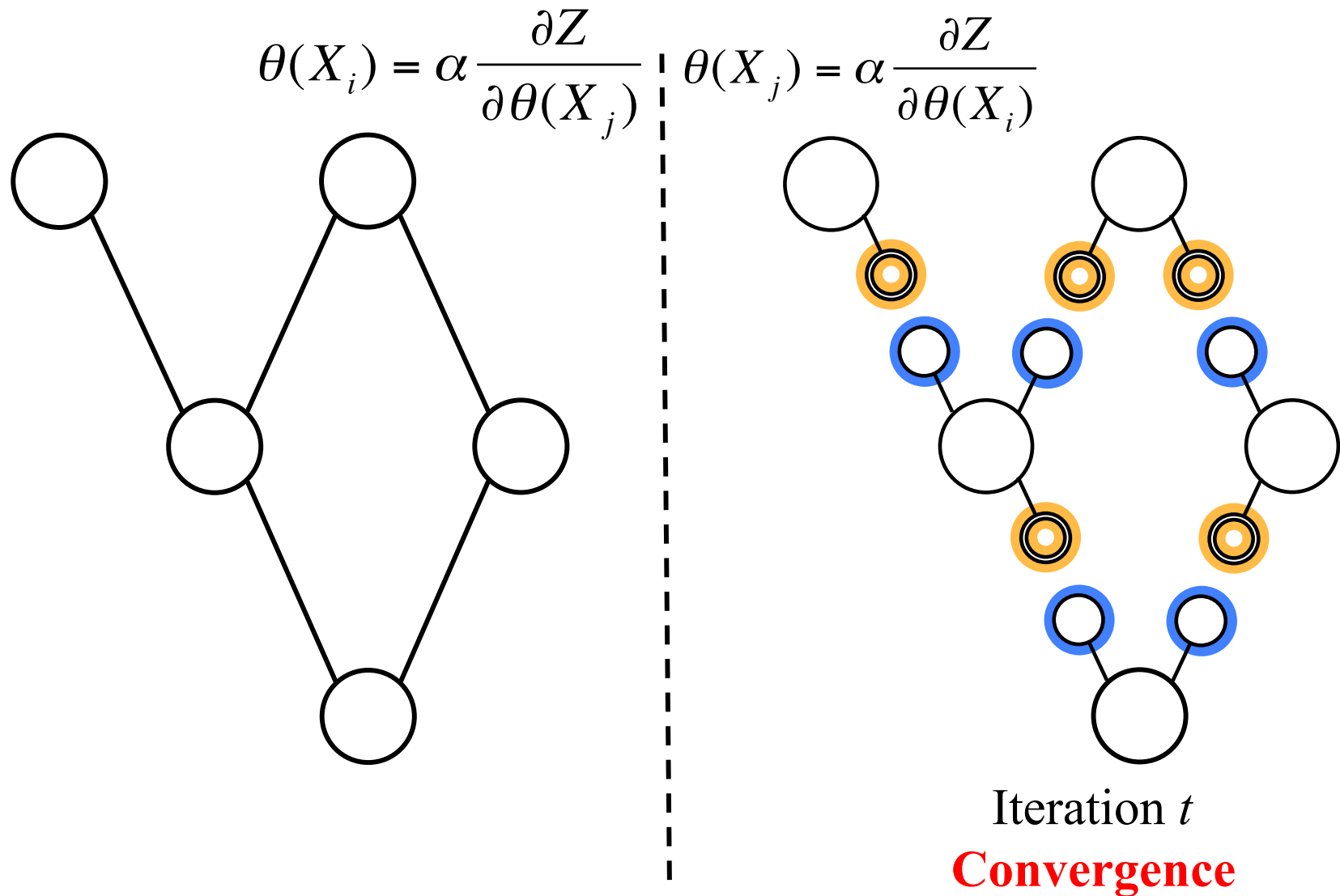
Iteration  $t = 1$

# Parametrizing Edges Iteratively

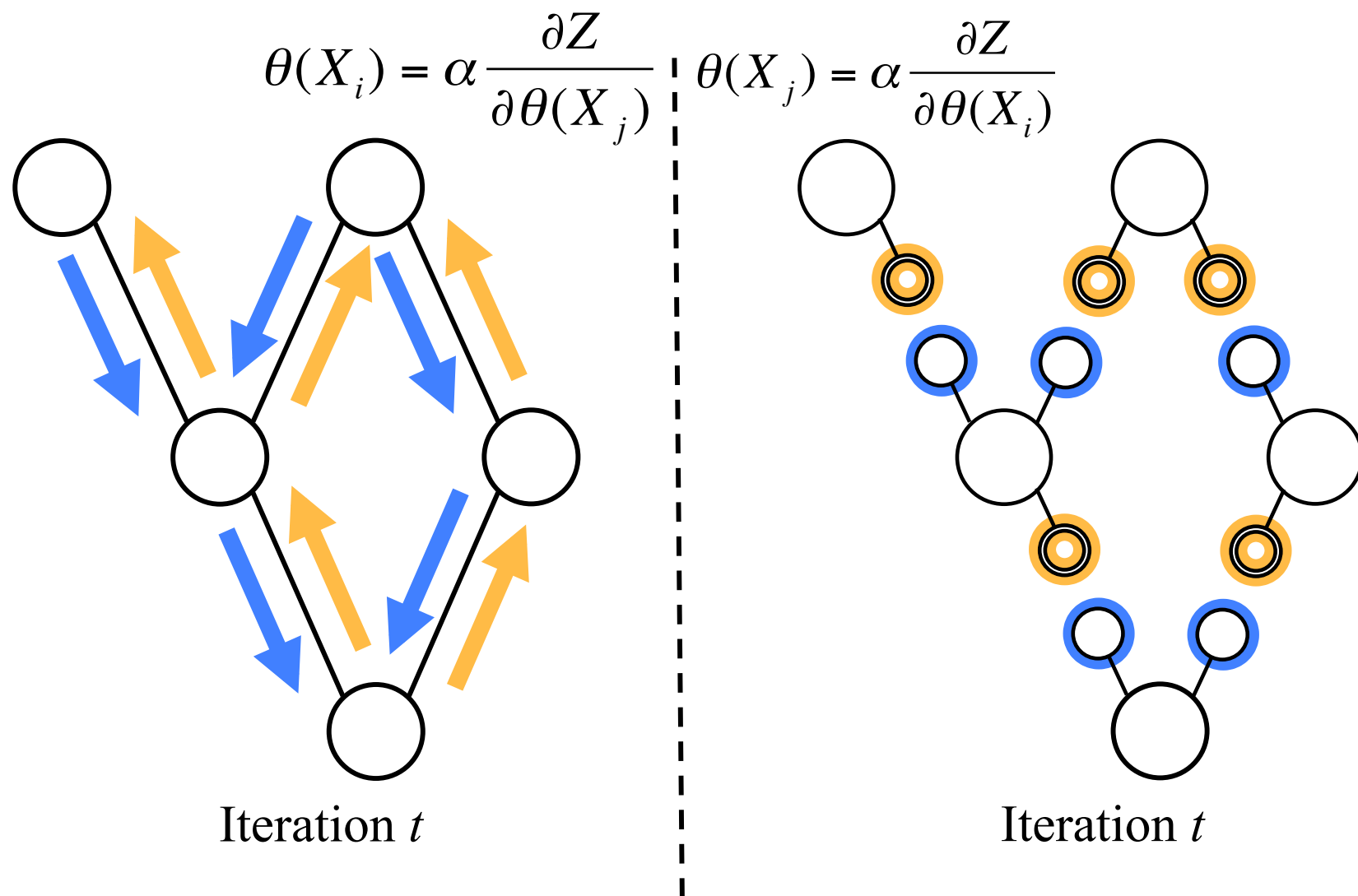


Iteration  $t = 2$

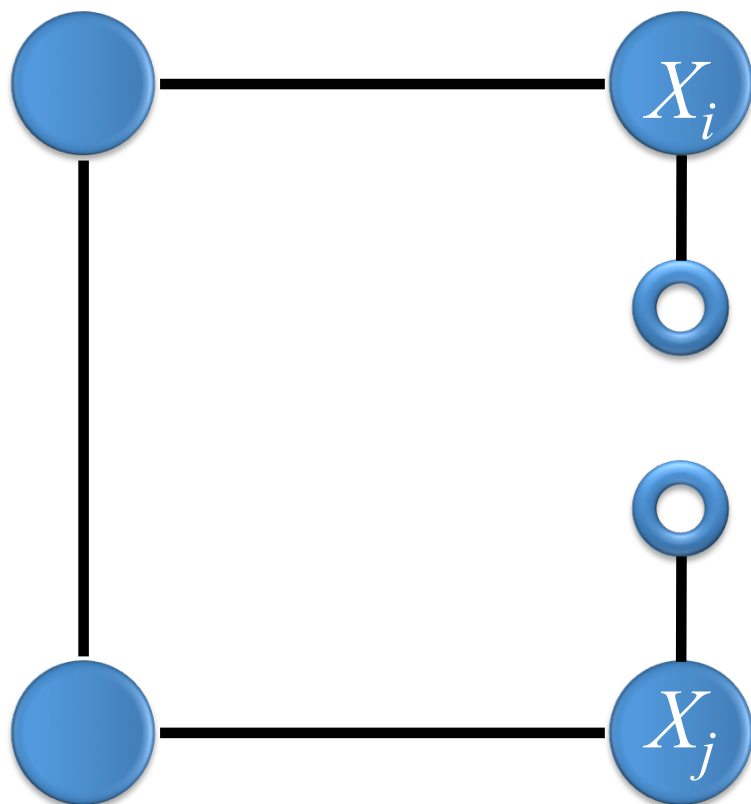
# Parametrizing Edges Iteratively



# Characterizing Loopy Belief Propagation



# Characterizing Loopy Belief Propagation

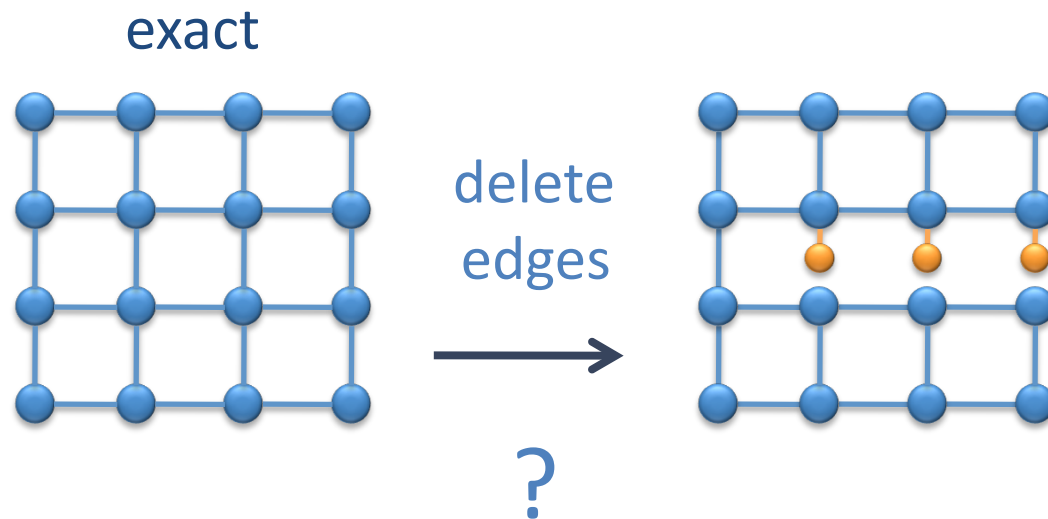


$$\theta(X_i) = \alpha \frac{\partial Z'}{\partial \theta(X_j)}$$

$$\theta(X_j) = \alpha \frac{\partial Z'}{\partial \theta(X_i)}$$

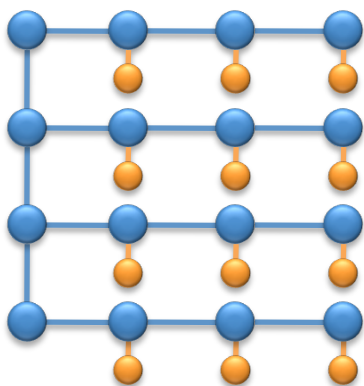
- in a tree:
  - MAR: BP marginals
  - PR: Bethe
  - MPE: BP max-marginals

# Which Edges to Delete?

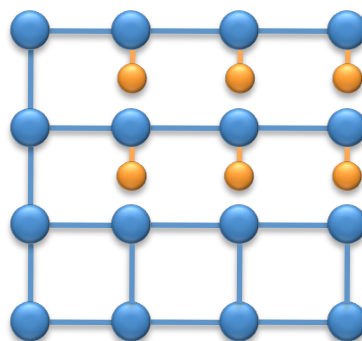


# Edge Recovery

loopy BP



recover  
edges



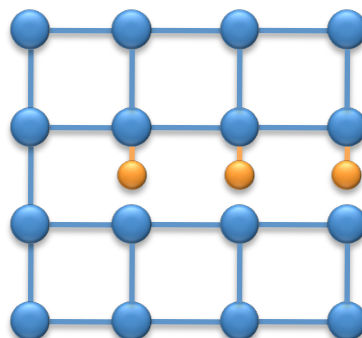
recover  
edges



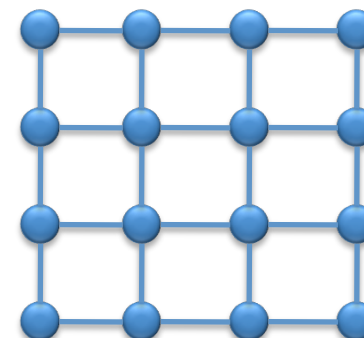
...

...

recover  
edges



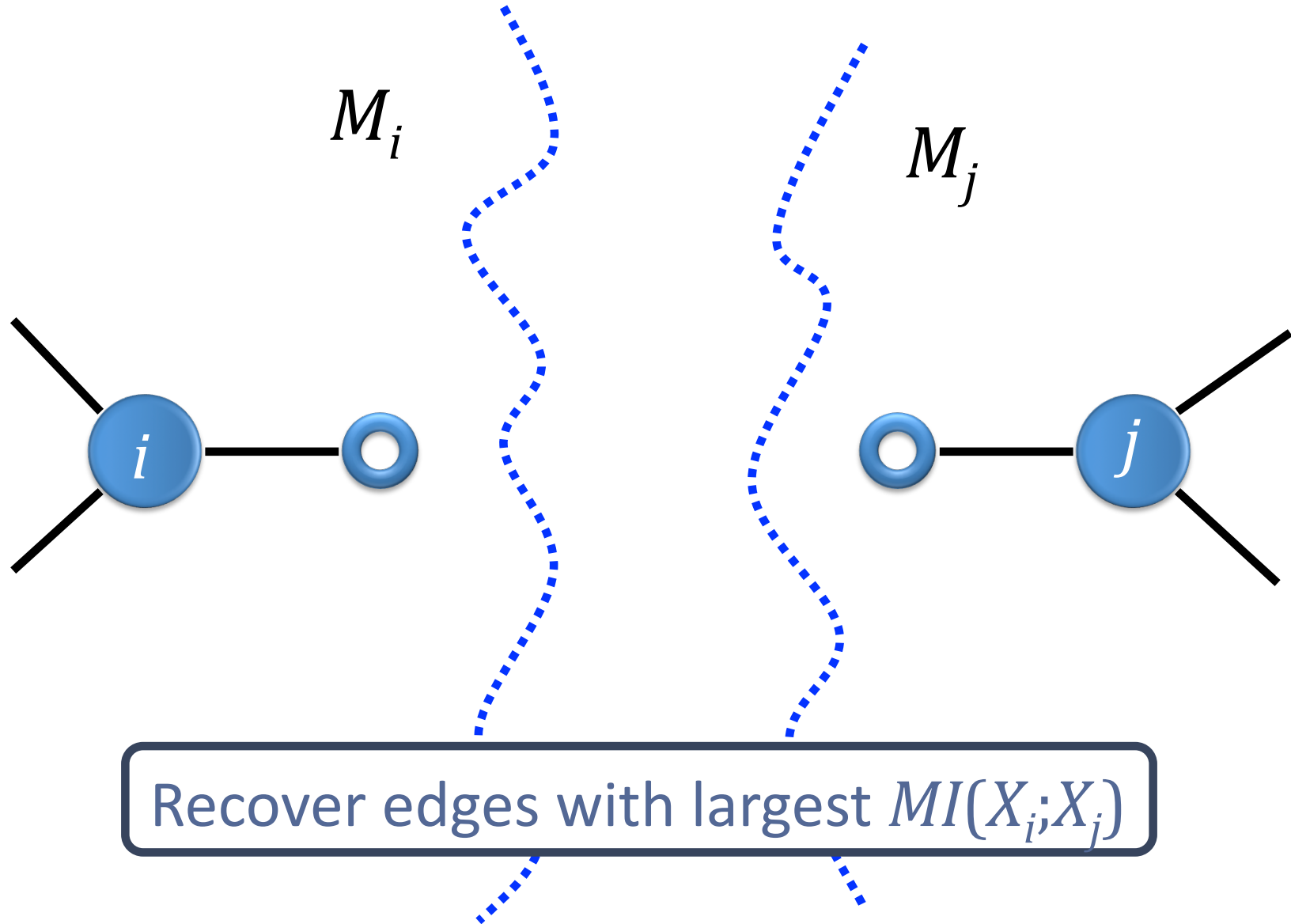
recover  
edges



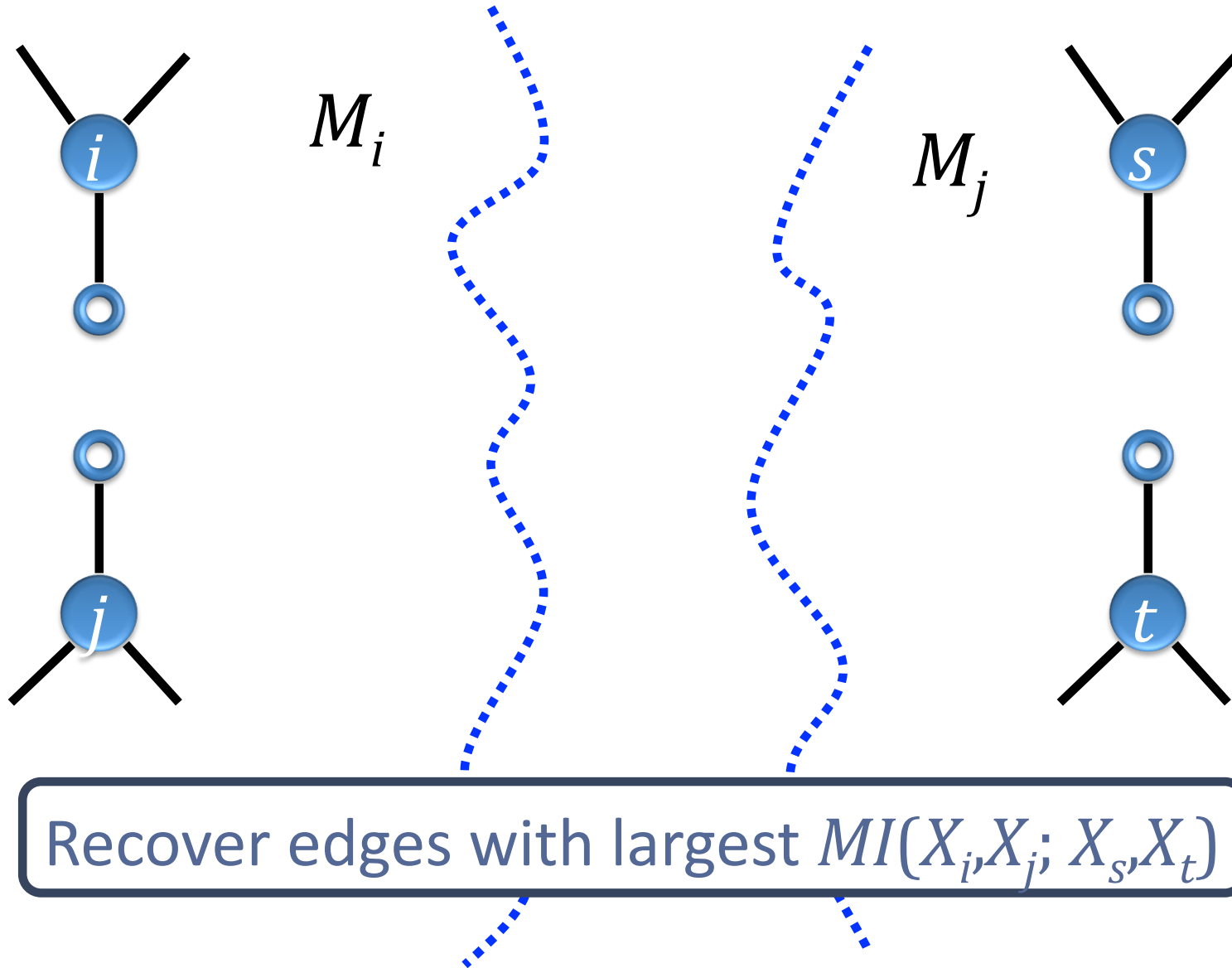
exact



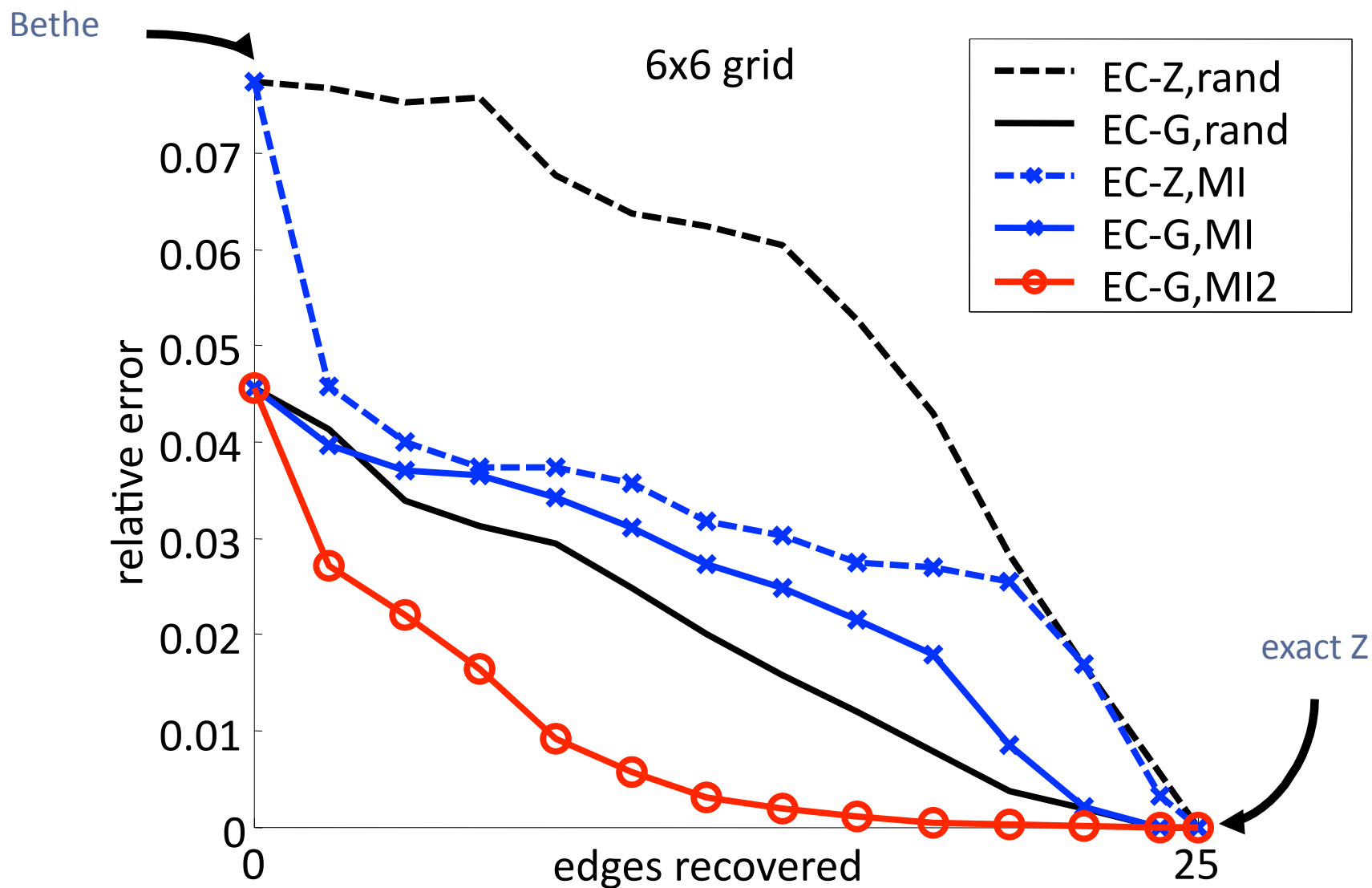
# Edge Recovery: ZERO-EC



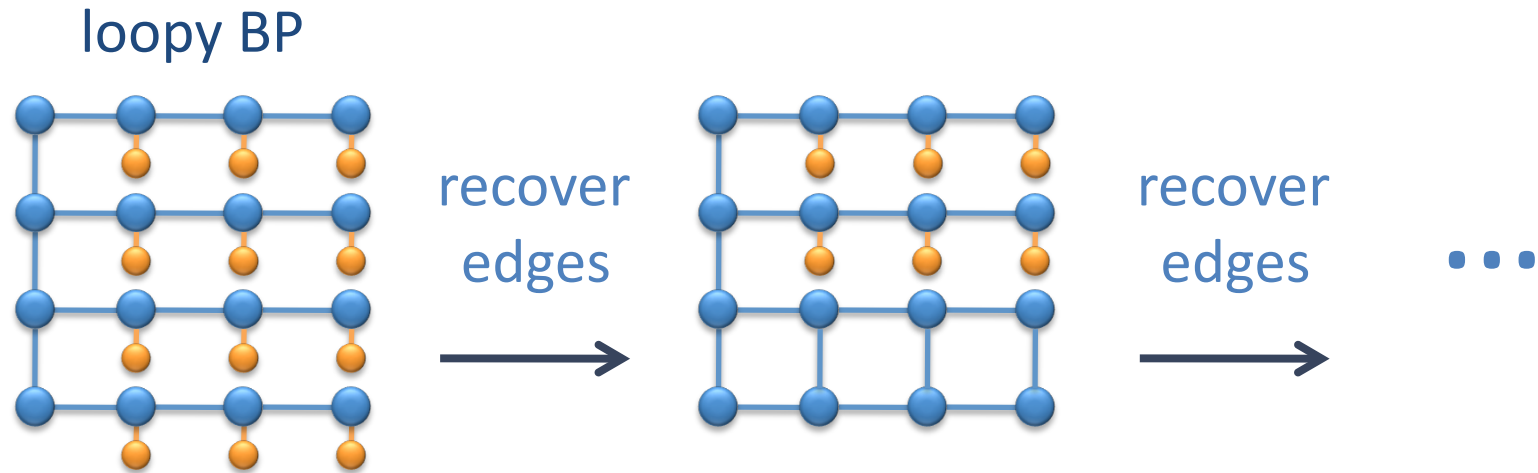
# Edge Recovery: GENERAL-EC



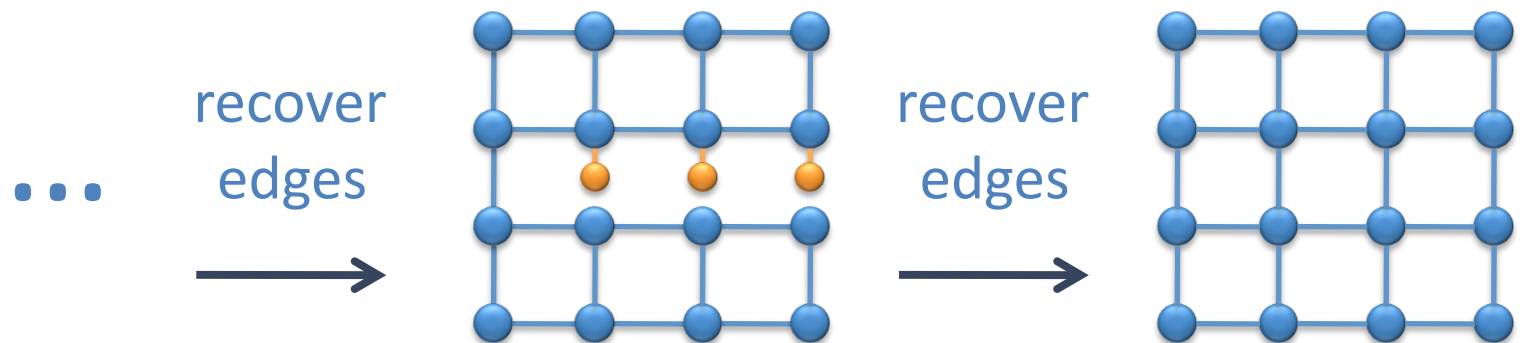
# Edge Recovery



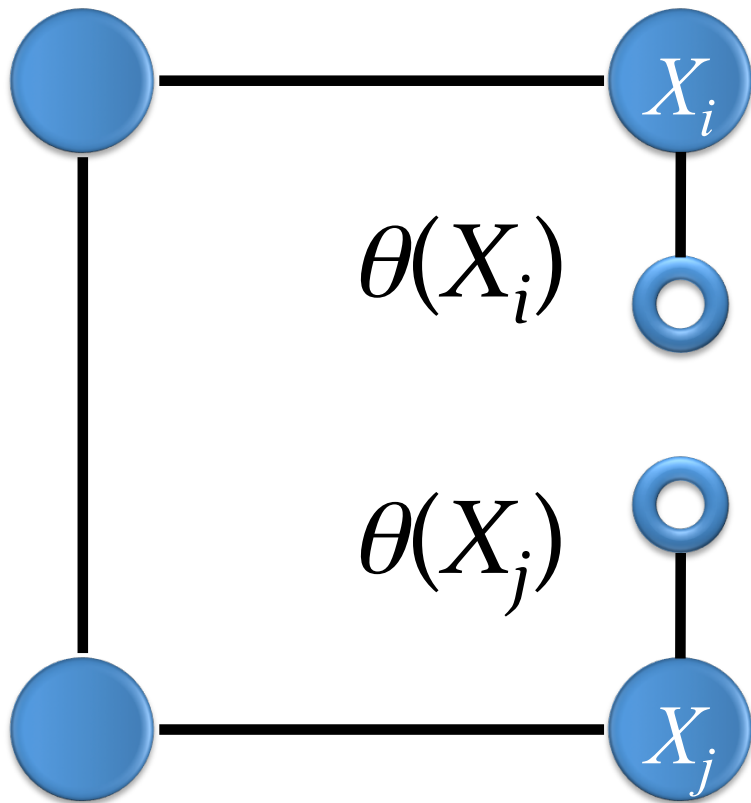
# Edge Recovery: New Idea



Challenge UAI-10: encourage convergence, *residual recovery*



# Residual Recovery



- Recover edges based on how close they are to convergence

$$\theta(X_i) = \alpha \frac{\partial Z'}{\partial \theta(X_j)}$$

$$\theta(X_j) = \alpha \frac{\partial Z'}{\partial \theta(X_i)}$$

# Evaluation Benchmarks

Benchmark	PR	MAR	MPE
CSP	8	8	55
Grids	20	20	40
Image Alignment			10
Medical Diagnosis	26	26	
Object Detection	96	96	92
Pedigree	4	4	
Protein Folding			21
Protein-Protein Interaction			8
Segmentation	50	50	50
TOTAL	204	204	287

# Overall Results

## PR Task: 20 Seconds

Solver	Score
edbr	1.7146
vgogate	2.1620
libDai	2.2775

## MAR Task: 20 Seconds

Solver	Score
edbq	0.2390
libDai2	0.3064
vgogate	0.4409

# Overall Results

## PR Task: 20 Minutes

Solver	Score
vgogate	1.2610
edbp	1.3063
libDai	2.0707

## MAR Task: 20 Minutes

Solver	Score
ijgp	0.1722
edbq	0.1742
libDai3	0.2810



# Overall Results

## PR Task: 1 Hour

Solver	Score
vgogate	1.2609
edbr	1.2699
libDai	2.0707

## MAR Task: 1 Hour

Solver	Score
ijgp	0.1703
edbr	0.1753
libDai3	0.2639

# Overall Results

## PR Task: 1 Hour

Solver	Score
vgogate	1.2609
edbr	1.2699
libDai	2.0707

## MAR Task: 1 Hour

Solver	Score
ijgp	0.1703
edbr	0.1753
libDai3	0.2639

# Exact Solvers

- Can use any exact solver
- Used Hugin and Shenoy-Shafer jointree algorithms. Did not use Ace.
- Did not expect instances to be solvable exactly within the given time limits!

# Ideally...

- Exact inference based on compiling CNFs
- Edge recovery using incremental compilation:
  - conjoin recovered equivalence constraint with current compilation
- Not there yet: more engineering needed!

# Key Ideas

- **Approximate inference:** formulated as exact inference in an approximate model
- **Approximate models:** obtained by relaxing and compensating for equivalence constraints
- **Anytime inference:** selective recovery of equivalence constraints
- **Exact inference:** formulated in terms of enforcing decomposability and determinism of propositional knowledge bases



# AUTOMATED REASONING GROUP

UNIVERSITY OF CALIFORNIA LOS ANGELES

HOME :: PUBLICATIONS :: TALKS :: MEMBERS :: **SOFTWARE** :: RELATED COURSES



## SamIam

[Reasoning Group](#) [Download](#) [Online Help](#) [Report a Bug](#) [Credits](#) [Contact](#)

**S**ENSITIVITY **A**NALYSIS, **M**ODELING, **I**NFERENCE **A**ND **M**ORE

[BatchTool](#)  
[Code Bandit](#)  
[Editing Models](#)  
[EM Learning](#)  
[File Formats](#)  
[Inference](#)  
[MAP](#)  
[MPE](#)  
[Relational Models](#)  
[Sensitivity Analysis](#)  
[Time-Space Tradeoffs](#)  
[Timing MAP](#)



### Try ACE - a companion system for networks exhibiting local structure: determinism and CSI

SamIam is a comprehensive tool for modeling and reasoning with Bayesian networks, developed in Java by the Automated Reasoning Group of Professor Adnan Darwiche at UCLA.



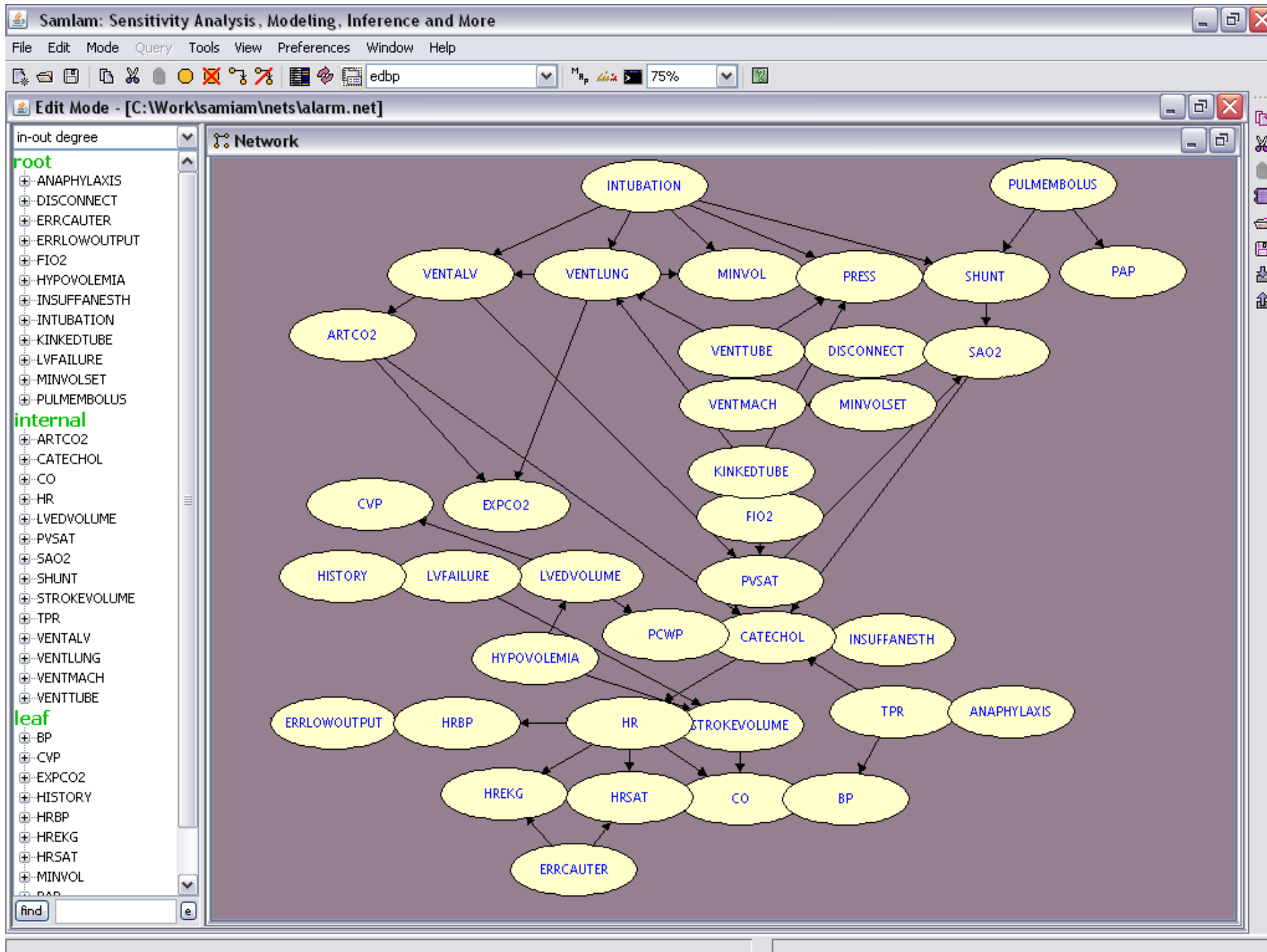
SamIam includes two main components: a graphical user interface and a reasoning engine. The graphical interface allows users to develop Bayesian network models and to save them in a variety of formats. The reasoning engine supports many tasks including: classical inference; parameter estimation; time-space tradeoffs; sensitivity analysis; and explanation-generation based on MAP and MPE.

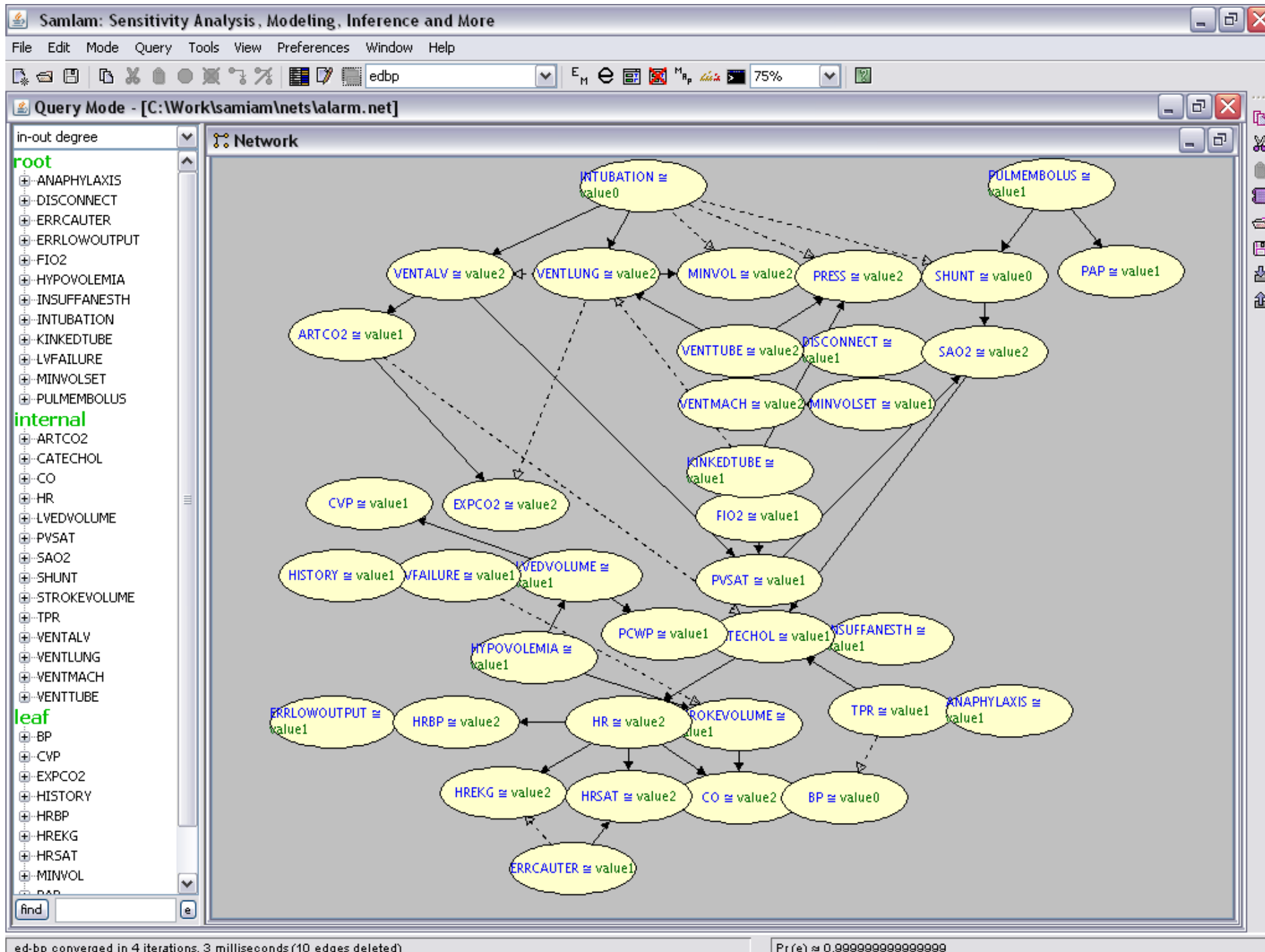
AR Group, UCLA

[Home](#) | [Screenshots](#) | [Sponsors](#) | [Suggestions](#) | [Videos](#) | [FAQ](#) | [Links](#)

Copyright © 2004-2005, Automated Reasoning Group, University Of California, Los Angeles, All Rights Reserved.

<http://reasoning.cs.ucla.edu/samiam/>







Samlam: Sensitivity Analysis, Modeling, Inference and More

File Edit Mode Query Tools View Preferences Window Help

edbp 75%

Edit Mode - [C:\Work\samiam\nets\alarm.net]

in-out degree

- root
  - ANAPHYLAXIS
  - DISCONNECT
  - ERRCAUTER
  - ERRLOWOUTPUT
  - FIO2
  - HYPOVOLEMIA
  - INSUFFANESTH
  - INTUBATION
  - KINKEDTUBE
  - LVFAILURE
  - MINVOLSET
  - PULMEMBOLUS
- internal
  - ARTCO2
  - CATECHOL
  - CO
  - HR
  - LVEDVOLUME
  - PVSAT
  - SAO2
  - SHUNT
  - STROKEVOLUME
  - TPR
  - VENTALV
  - VENTLUNG
  - VENTMACH
  - VENTTUBE
- leaf
  - BP
  - CVP
  - EXPCO2
  - HISTORY
  - HRBP
  - HREKG
  - HRSAT
  - MINVOL
  - PAP

Network

Compile Settings - edbp

automatic edge-deletion belief-propagation sub-algorithm: **zc-hugin**

Bound, maximum iterations: 100 (0 = unbounded)

Time out (milliseconds): 10000 (0 = no time out)

Termination tolerance: 0.0001 (0 <= tolerance)

Edge ranking heuristic: residual recovery

Edge recovery count: [slider] tree ... full network

Random seed: 1

Report exact values: ☒ Ctrl+E approx over/exact under

Sub-algorithm: zc-hugin partial derivatives

INTUBATION  $\cong$  value0

PULMEMBOLUS  $\cong$  value1

PAP  $\cong$  value1

ERRCAUTER  $\cong$  value1

HRBP  $\cong$  value2

HR  $\cong$  value2

STROKEVOLUME  $\cong$  value1

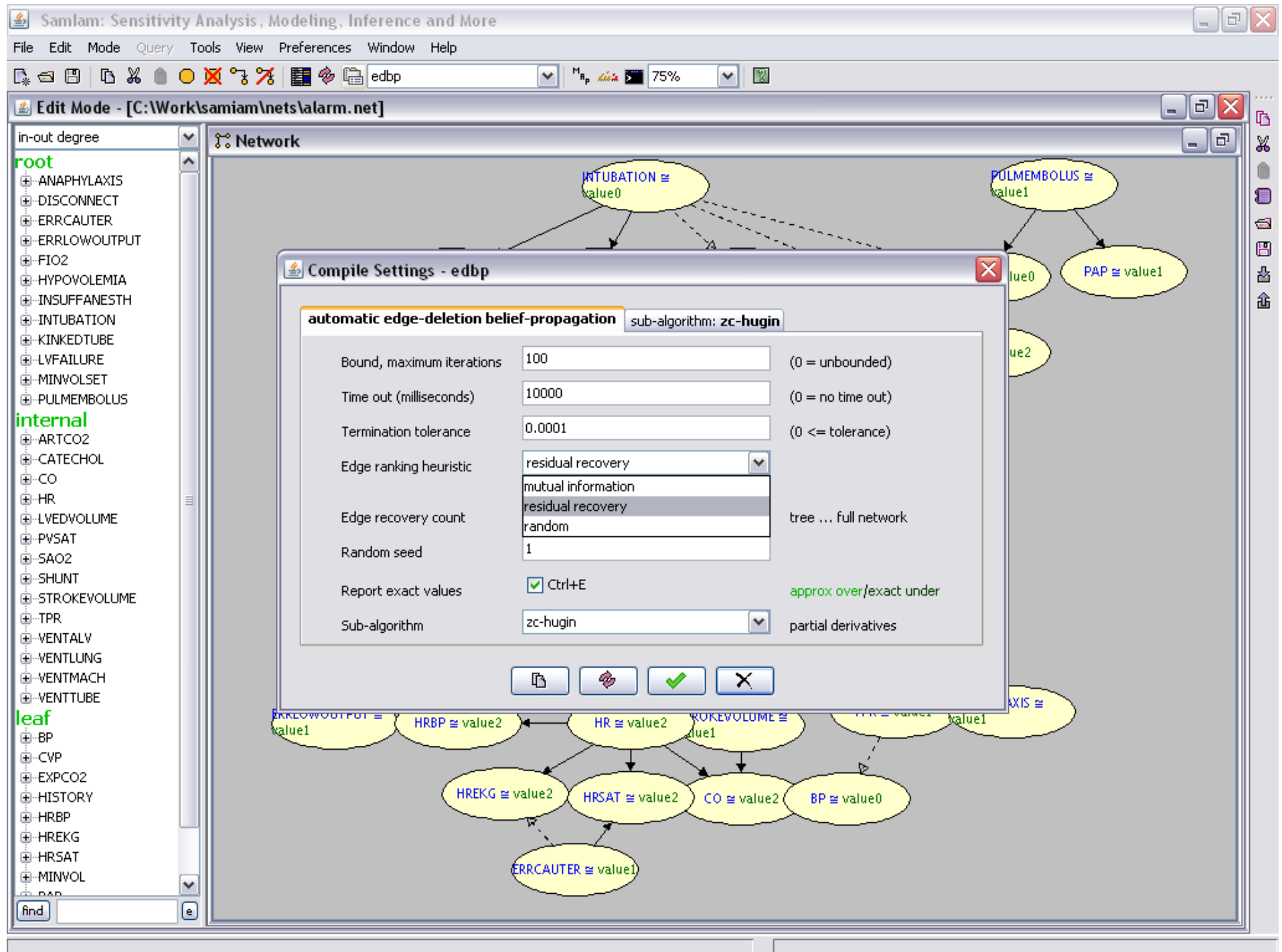
HREKG  $\cong$  value2

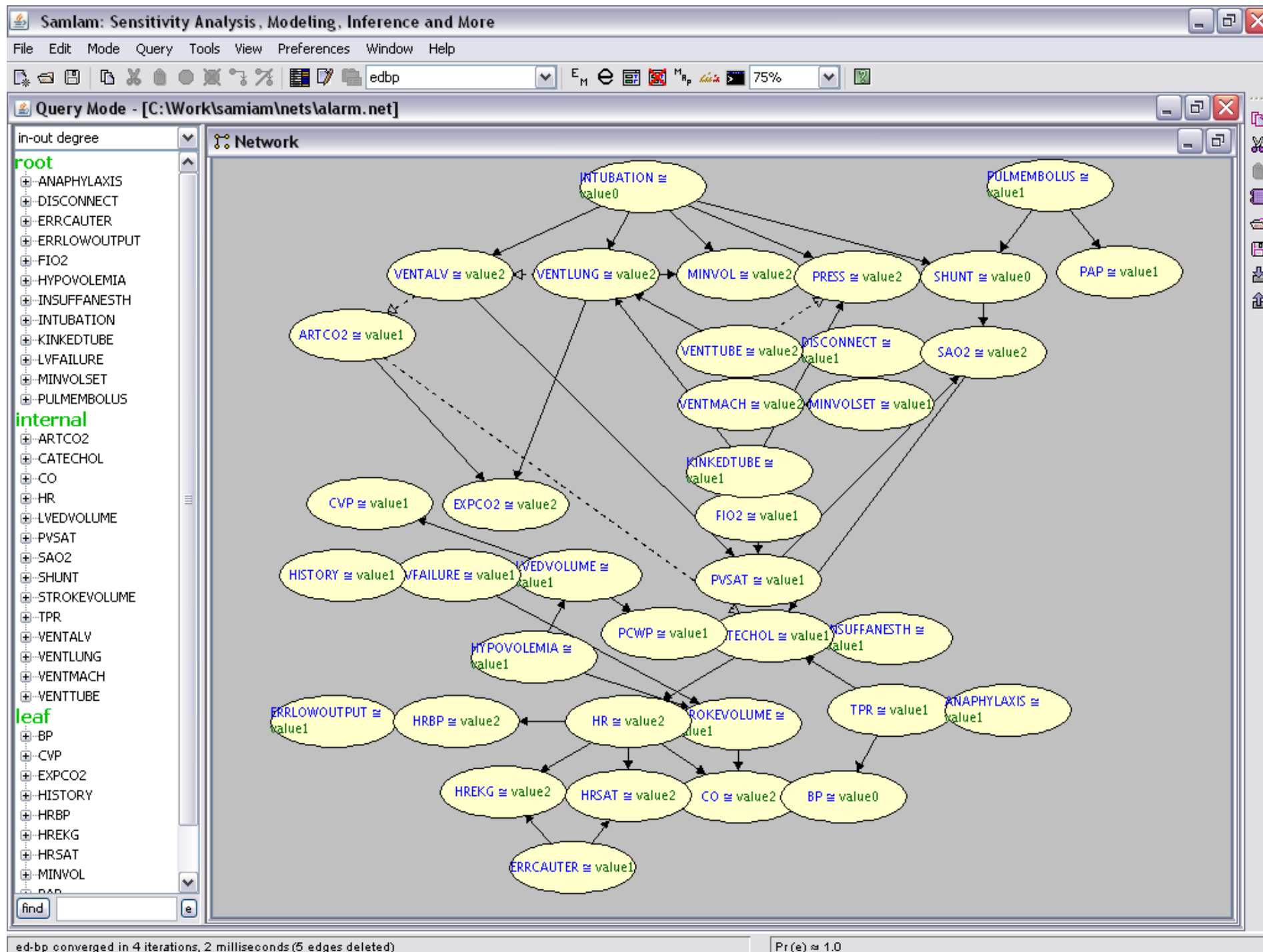
HRSAT  $\cong$  value2

CO  $\cong$  value2

BP  $\cong$  value0

find





Copyrighted Material

Adnan Darwiche

**MODELING AND REASONING**  
with  
**BAYESIAN NETWORKS**

CAMBRIDGE

Copyrighted Material



# AUTOMATED REASONING GROUP



UNIVERSITY OF CALIFORNIA LOS ANGELES

HOME :: PUBLICATIONS :: **LINKS** :: MEMBERS :: SOFTWARE :: RELATED COURSES

## SamIam

[Reasoning Group](#) [Download](#) [Online Help](#) [Report a Bug](#) [Credits](#) [Contact](#)

**S**ENSITIVITY **A**NALYSIS, **M**ODELING, **I**NFERENCE **A**ND **M**ORE

[BatchTool](#)  
[Code Bandit](#)  
[Editing Models](#)  
[EM Learning](#)  
[File Formats](#)  
[Inference](#)  
[MAP](#)  
[MPE](#)  
[Relational Models](#)  
[Sensitivity Analysis](#)  
[Time-Space Tradeoffs](#)  
[Timing MAP](#)



### Try ACE - a companion system for networks exhibiting local structure: determinism and CSI

SamIam is a comprehensive tool for modeling and reasoning with Bayesian networks, developed in Java by the Automated Reasoning Group of Professor Adnan Darwiche at UCLA.



SamIam includes two main components: a graphical user interface and a reasoning engine. The graphical interface allows users to develop Bayesian network models and to save them in a variety of formats. The reasoning engine supports many tasks including: classical inference; parameter estimation; time-space tradeoffs; sensitivity analysis; and explanation-generation based on MAP and MPE.

AR Group, UCLA

[Home](#) | [Screenshots](#) | [Sponsors](#) | [Suggestions](#) | [Videos](#) | [FAQ](#) | [Links](#)

Copyright © 2004-2005, Automated Reasoning Group, University Of California, Los Angeles, All Rights Reserved.

<http://reasoning.cs.ucla.edu>