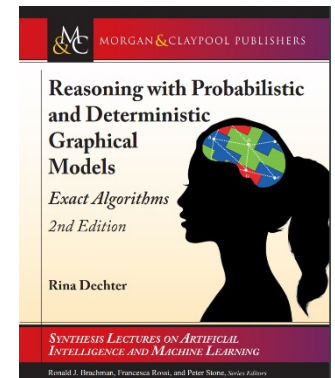


Reasoning with graphical models

Slides Set 7: AND/OR search for Probabilistic Networks

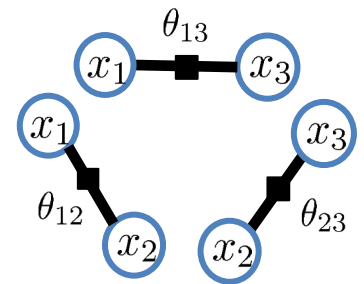
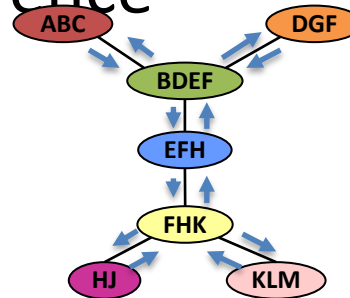
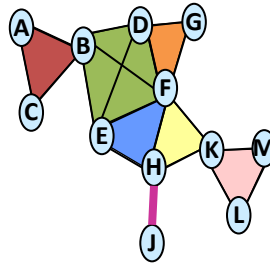
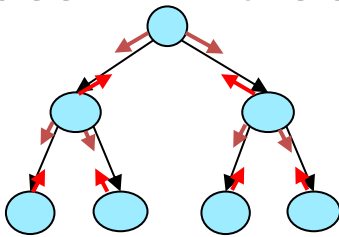
Rina Dechter

(Dechter chapters 6 and 7)

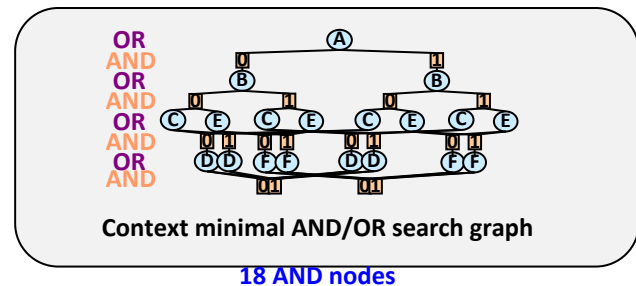
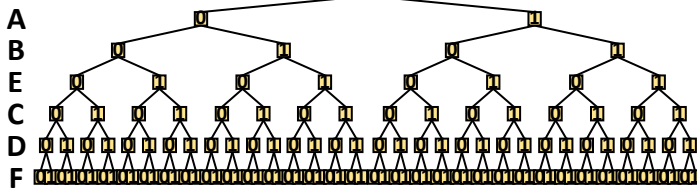


Overall Perspective

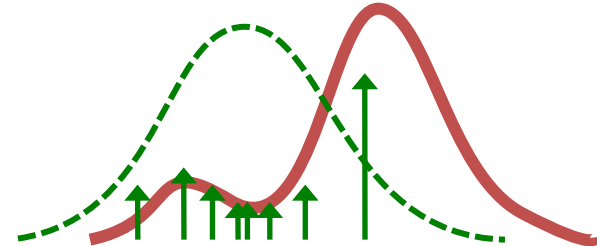
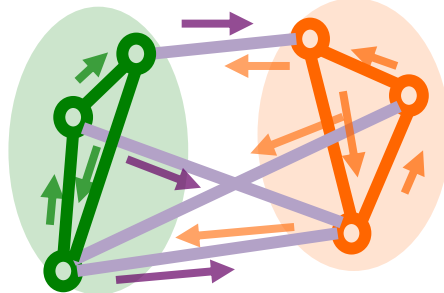
- Class 1: Introduction and Inference



- Class 2: Search

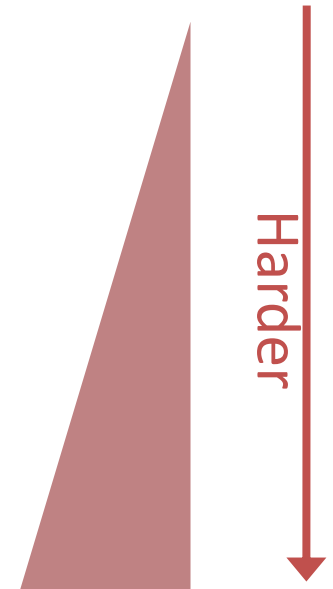


- Class 3: Variational Methods and Monte-Carlo Sampling



Types of queries

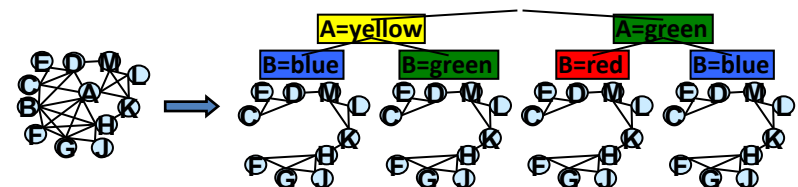
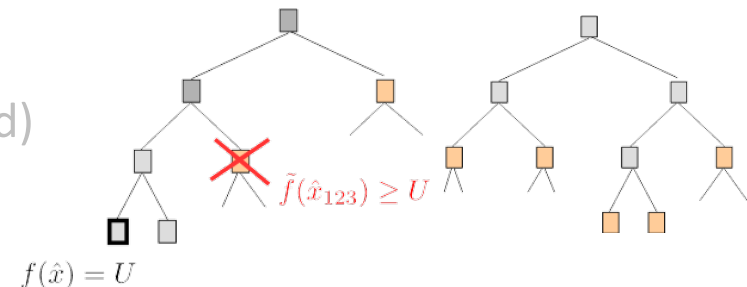
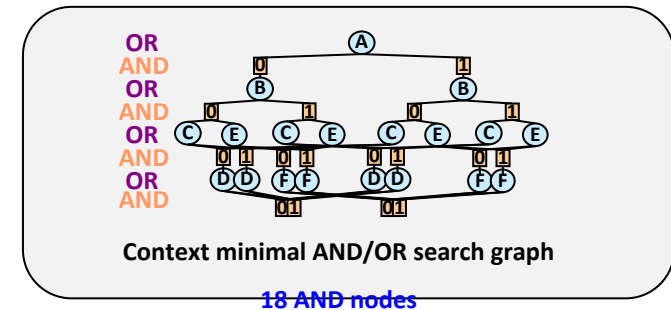
▶ Max-Inference	$f(\mathbf{x}^*) = \max_{\mathbf{x}} \prod_{\alpha} f_{\alpha}(\mathbf{x}_{\alpha})$
▶ Sum-Inference	$Z = \sum_{\mathbf{x}} \prod_{\alpha} f_{\alpha}(\mathbf{x}_{\alpha})$
▶ Mixed-Inference	$f(\mathbf{x}_M^*) = \max_{\mathbf{x}_M} \sum_{\mathbf{x}_S} \prod_{\alpha} f_{\alpha}(\mathbf{x}_{\alpha})$



- **NP-hard**: exponentially many terms
- We will focus on **approximation** algorithms
 - **Anytime**: very fast & very approximate ! Slower & more accurate

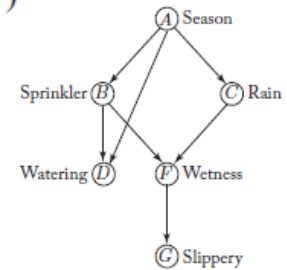
Outline: Search for Graphical Models

- Review Graphical Modes
- AND/OR search spaces, pseudo-trees
 - AND/OR search trees
 - AND/OR search graphs
 - Generating good pseudo-trees
 - Brute-force AND/OR
- Heuristic search (HS) for AND/OR spaces
 - Basic Heuristic search (Depth and Best)
 - AND/OR Depth-first HS (branch and bound)
 - AND/OR Best-first heuristic search
 - The Guiding MBE heuristic
 - Marginal Map (max-sum-product)
- Hybrids of search and Inference
- Summary and Class 2



Conditioning - the Probability Tree

$$P(D = 1, G = 0) = \sum_a P(a) \sum_c P(c|a) \sum_b P(b|a) \sum_f P(f|b, c) P(d = 1|b, a) P(g = 0|f)$$



(a) Directed Acyclic Graph

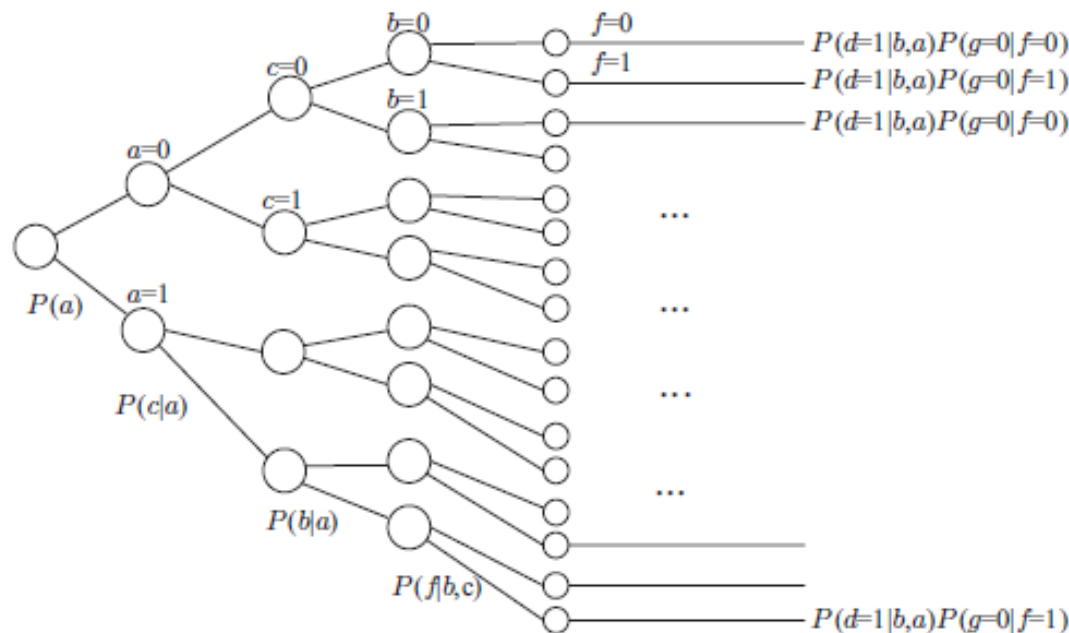
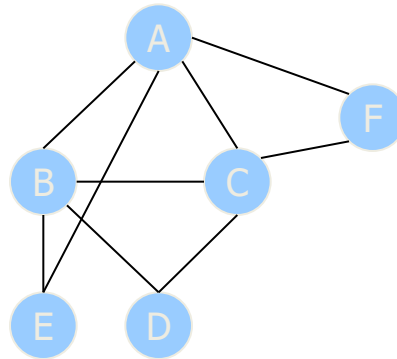


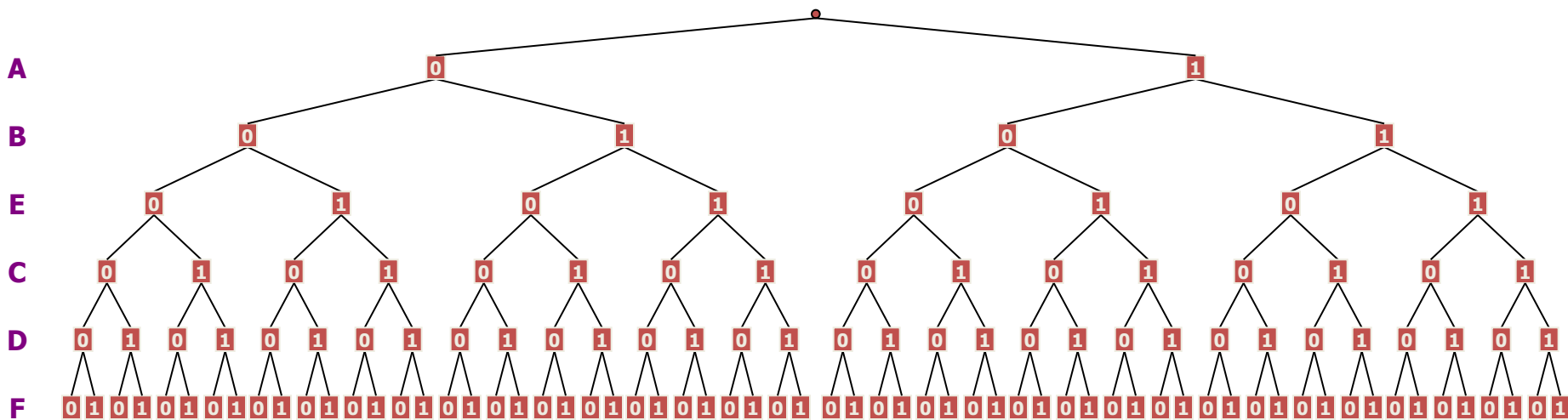
Figure 6.1: Probability tree for computing $P(d = 1, g = 0)$.

Complexity of conditioning: exponential time, linear space

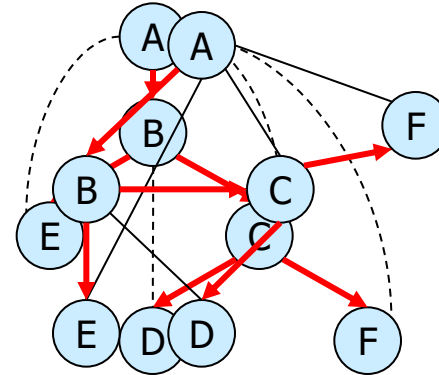
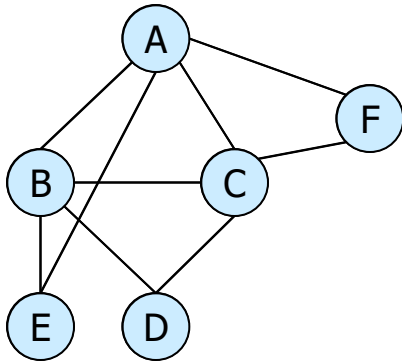
The Classic OR Search Space



Ordering: A B E C D F



AND/OR Search Space



OR

AND

OR

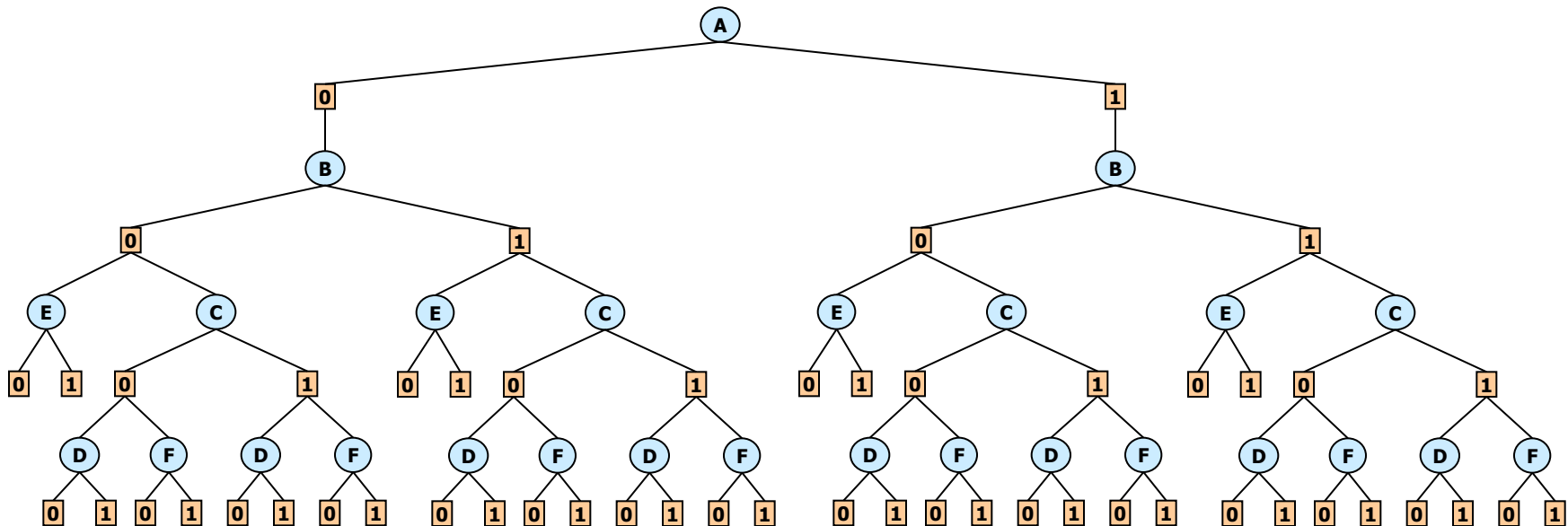
AND

OR

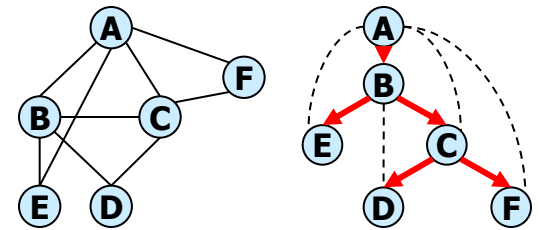
AND

OR

AND



AND/OR vs. OR



OR

AND

OR

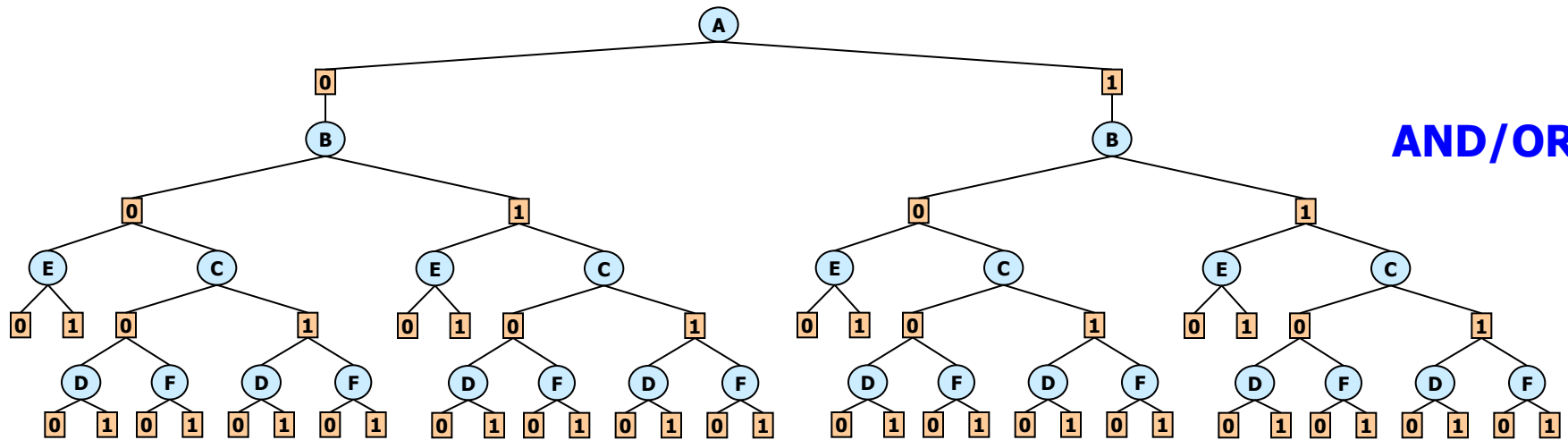
AND

OR

AND

OR

AND



AND/OR size: $\exp(4)$,
OR size $\exp(6)$

A

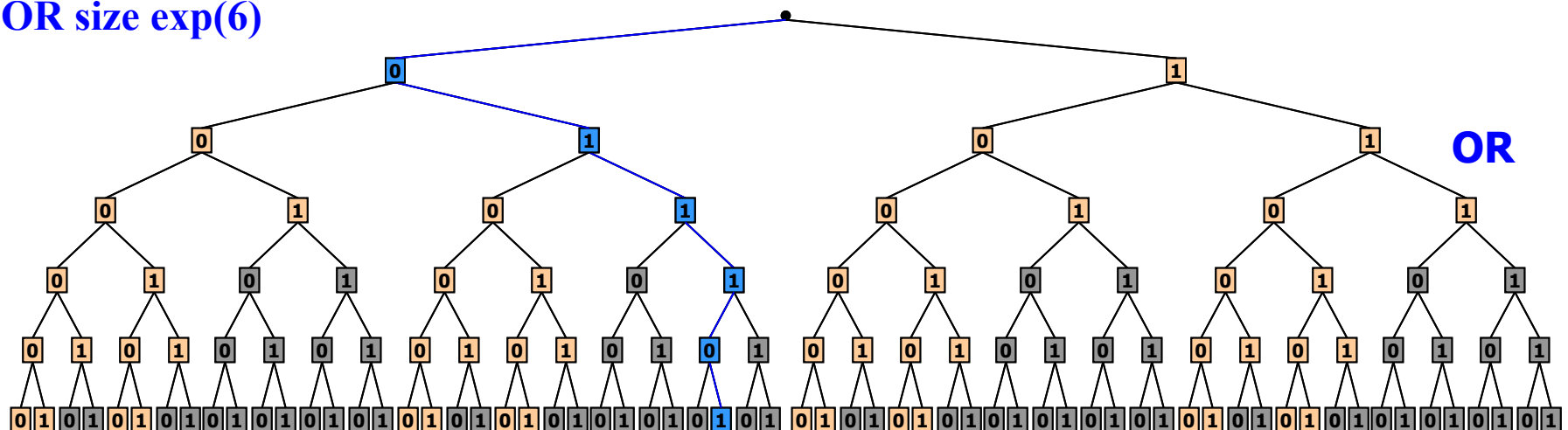
B

E

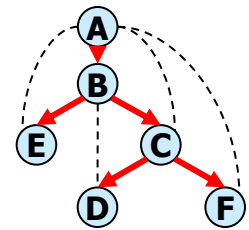
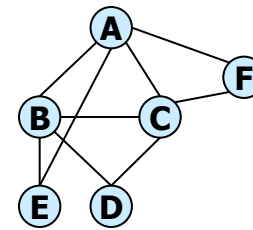
C

D

F



AND/OR vs. OR



OR

AND

OR

AND

OR

AND

OR

AND

A

B

E

C

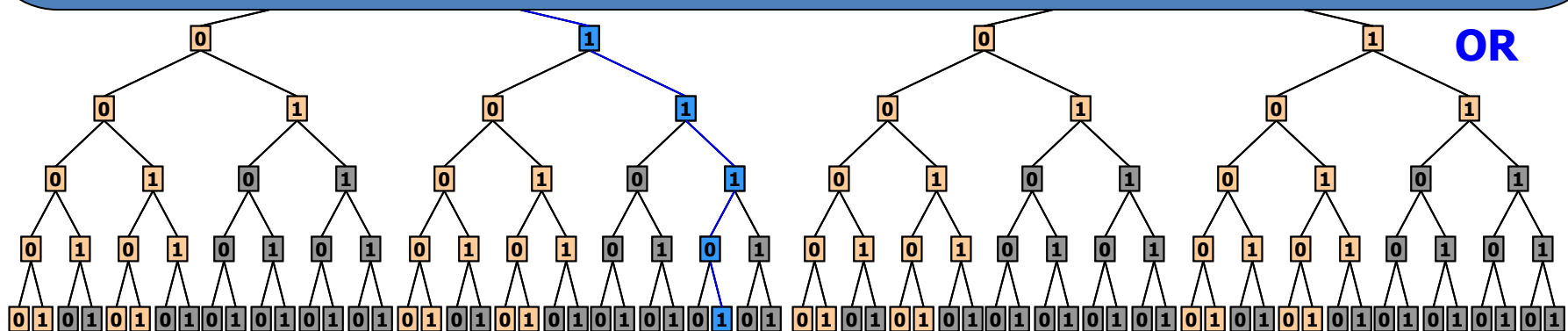
D

F

AND/OR

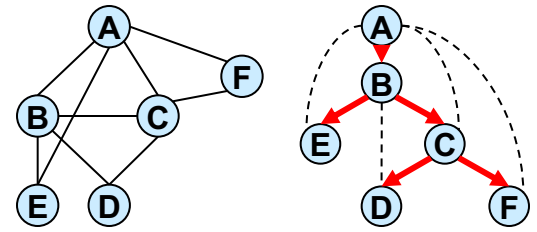
- *Size of tree $O(nk^h)$*
- *Can be traversed in*
 - *Time $O(nk^h)$, Space $O(n)$*
- *All solution trees = all configurations*

OR



AND/OR vs. OR

No-goods
(A=1,B=1)
(B=0,C=0)



OR

AND

OR

AND

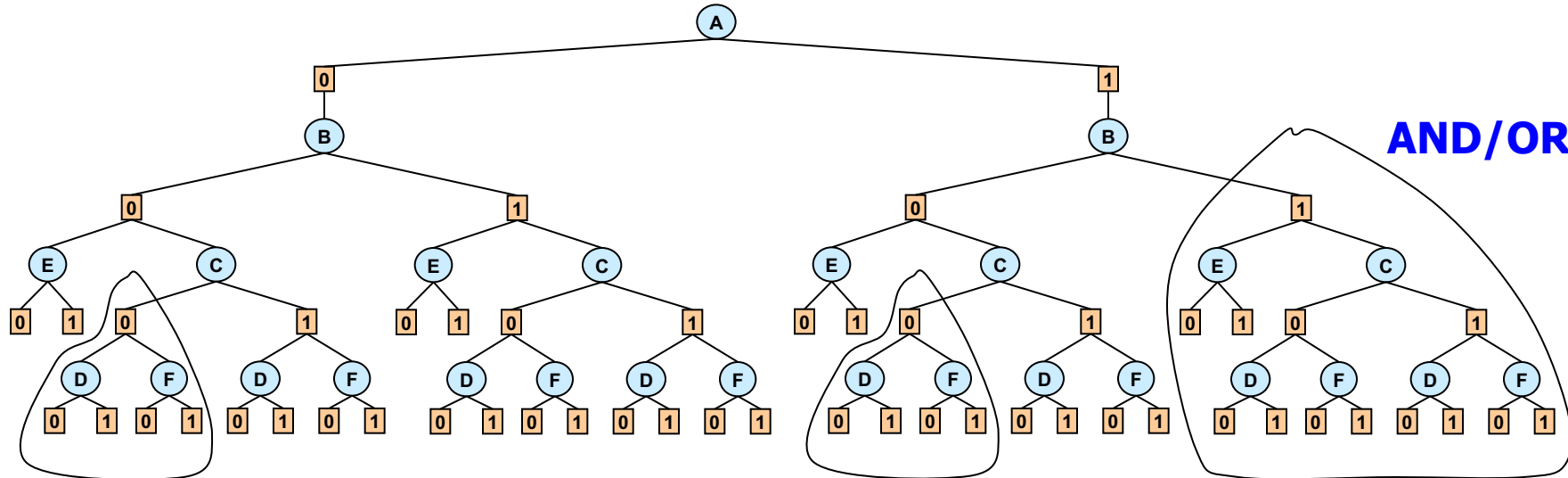
OR

AND

OR

AND

AND/OR



A

B

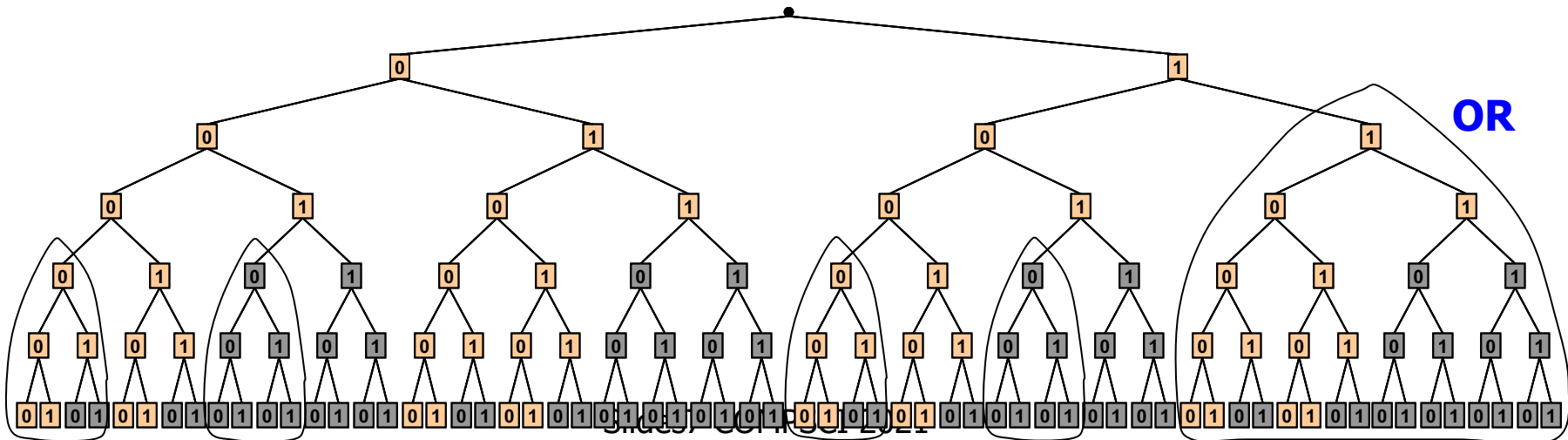
E

C

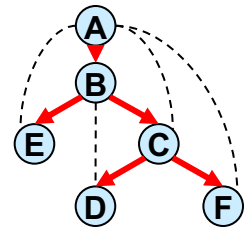
D

F

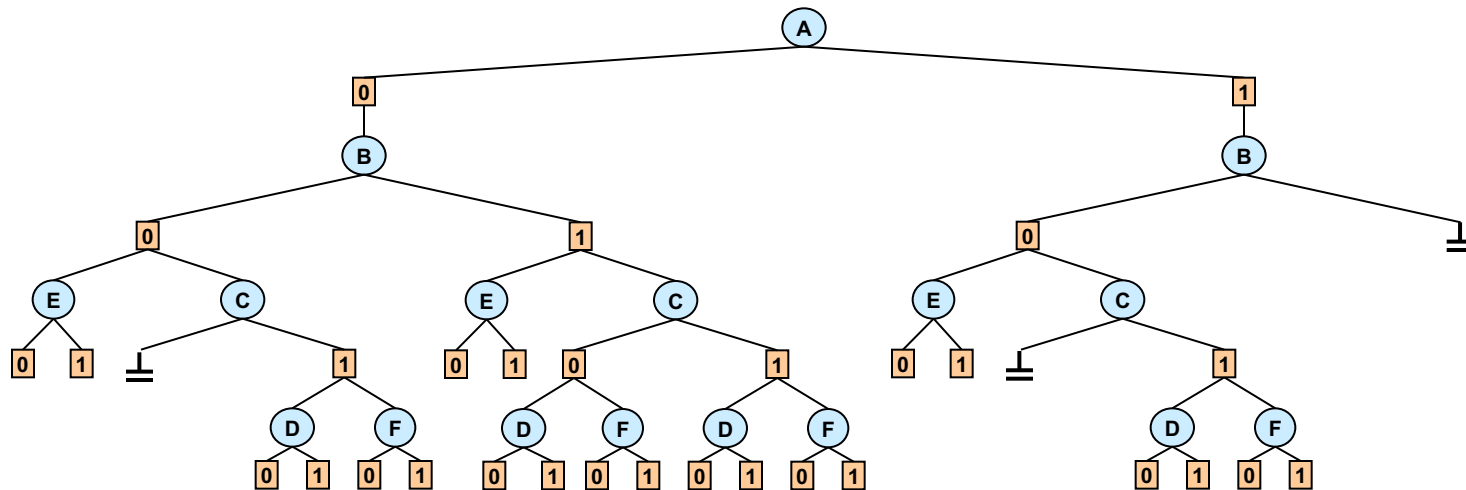
OR



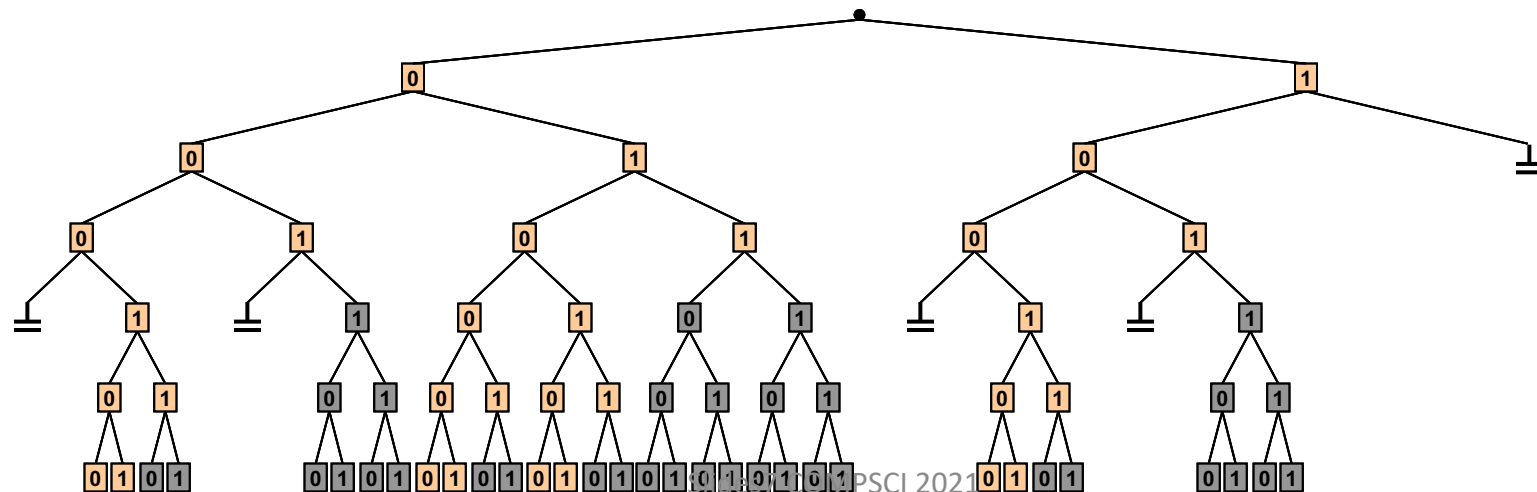
OR
AND
OR
AND
OR
AND
OR
AND



AND/OR



OR





Arc weights
Cost of a solution tree
The value function

Arc Weights for AND/OR Trees

$P(E | A, B)$

A	B	E=0	E=1
0	0	.4	.6
0	1	.5	.5
1	0	.7	.3
1	1	.2	.8

Evidence: E=0

$P(B | A)$

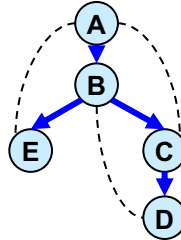
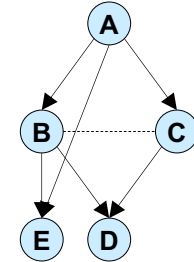
A	B=0	B=1
0	.4	.6
1	.1	.9

$P(C | A)$

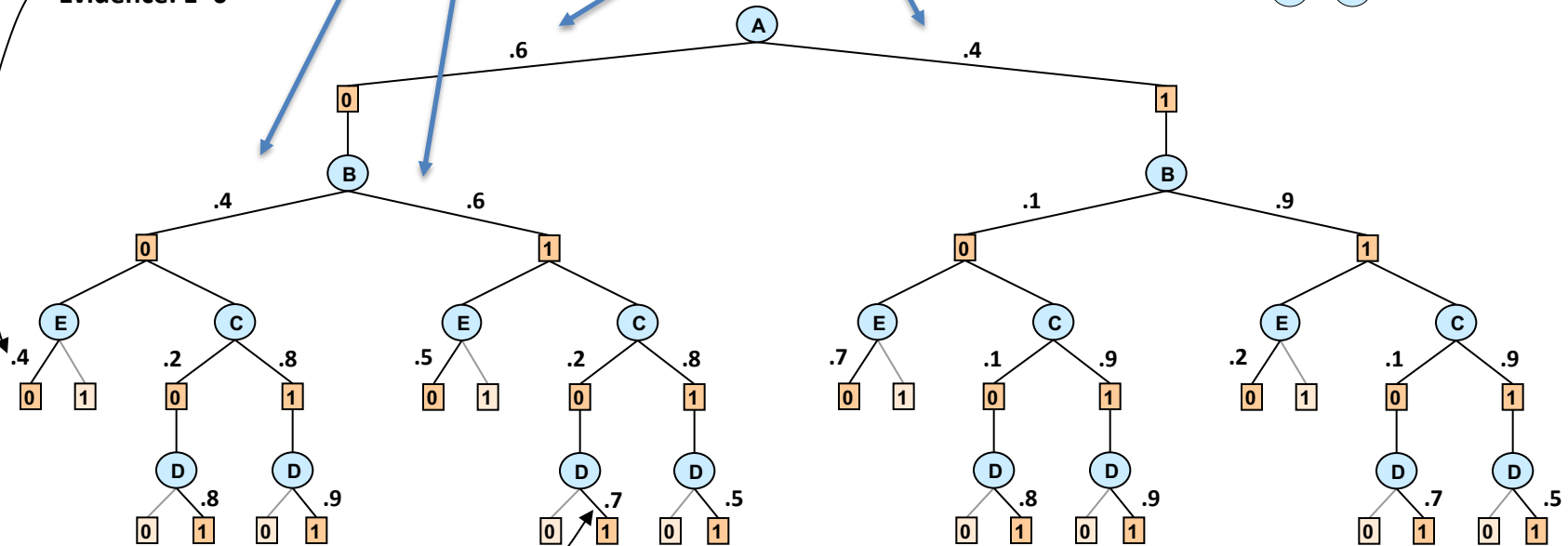
A	C=0	C=1
0	.2	.8
1	.7	.3

$P(A)$

A	P(A)
0	.6
1	.4



OR
AND
OR
AND
OR
AND
OR
AND



$P(D | B, C)$

B	C	D=0	D=1
0	0	.2	.8
0	1	.1	.9
1	0	.3	.7
1	1	.5	.5

Evidence: D=1

OR to AND arc weight $\langle X, x \rangle$ is the product of factors that all their arguments are just assigned at AND node $X=x$ but not before

Cost of a Solution Tree

$P(E | A, B)$

A	B	E=0	E=1
0	0	.4	.6
0	1	.5	.5
1	0	.7	.3
1	1	.2	.8

Evidence: E=0

$P(B | A)$

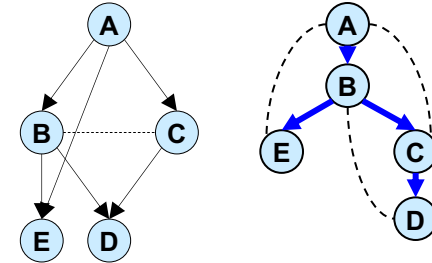
A	B=0	B=1
0	.4	.6
1	.1	.9

$P(C | A)$

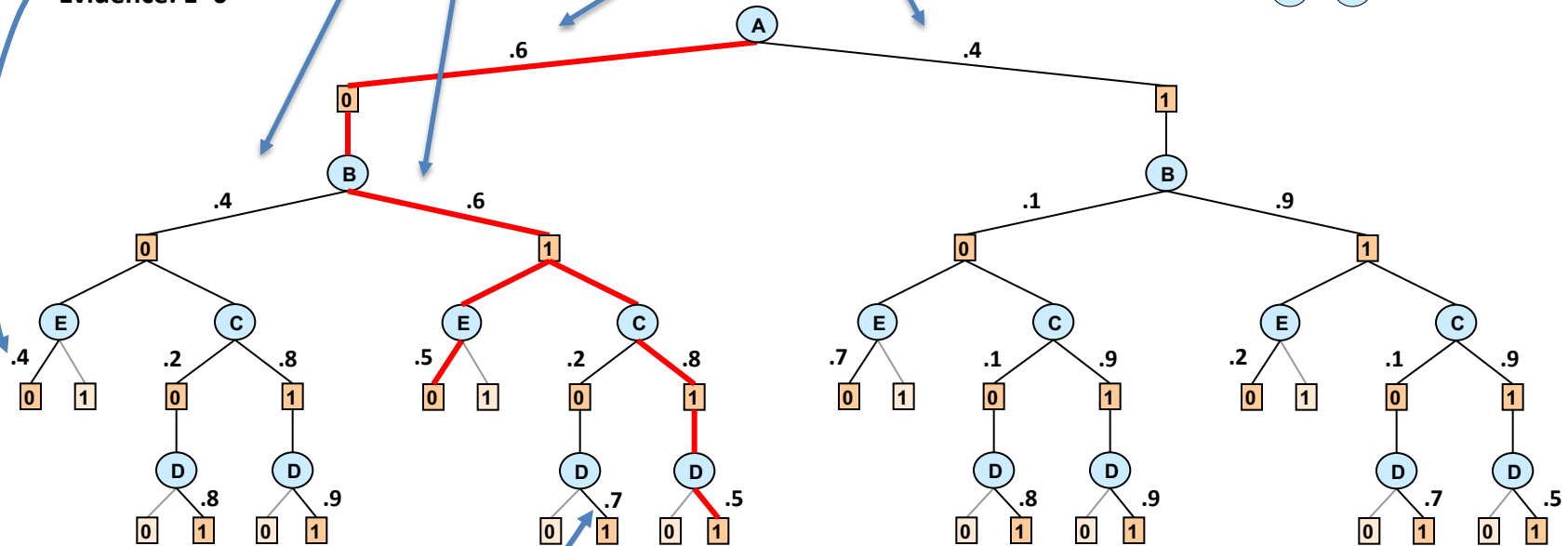
A	C=0	C=1
0	.2	.8
1	.7	.3

$P(A)$

A	P(A)
0	.6
1	.4



OR
AND
OR
AND
OR
AND
OR
AND



$P(D | B, C)$

B	C	D=0	D=1
0	0	.2	.8
0	1	.1	.9
1	0	.3	.7
1	1	.5	.5

Evidence: D=1

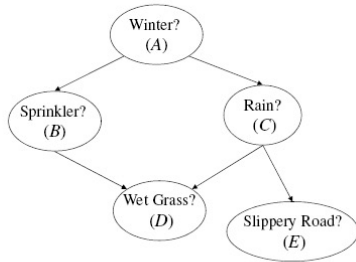
A solution tree includes the root and has a single child for any OR node, and all children of any of its AND nodes

Cost of the solution tree: the product of weights on its arcs

Cost of $(A=0, B=1, C=1, D=1, E=0) = 0.6 \cdot 0.6 \cdot 0.5 \cdot 0.8 \cdot 0.5 = 0.0720$

Arc Weights for AND/OR Trees

A Bayesian Network



A	Θ_A
true	.6
false	.4

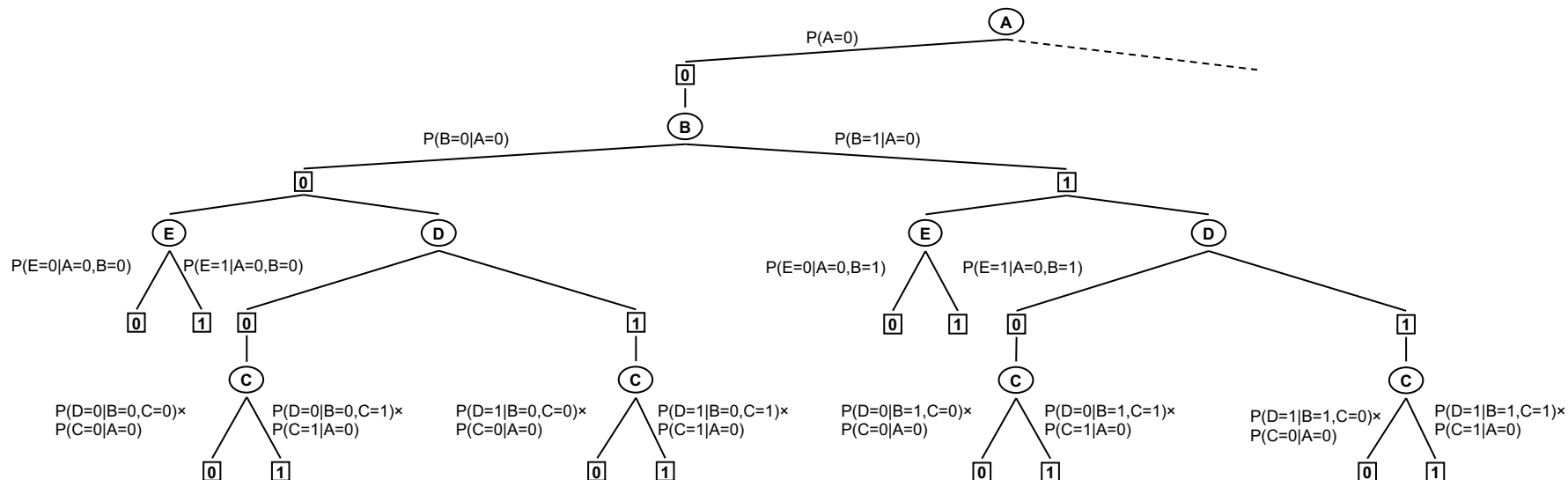
A	B	$\Theta_{B A}$
true	true	.2
true	false	.8
false	true	.75
false	false	.25

A	C	$\Theta_{C A}$
true	true	.8
true	false	.2
false	true	.1
false	false	.9

B	C	D	$\Theta_{D BC}$
true	true	true	.95
true	true	false	.05
true	false	true	.9
true	false	false	.1
false	true	true	.8
false	true	false	.2
false	false	true	0
false	false	false	1

C	E	$\Theta_{E C}$
true	true	.7
true	false	.3
false	true	0
false	false	1

OR to AND arc weight (X, x) is the product of factors f that are instantiated at the AND node $X=x$ but not before



The Value Function (Probability of Evidence)

$P(E | A, B)$

A	B	E=0	E=1
0	0	.4	.6
0	1	.5	.5
1	0	.7	.3
1	1	.2	.8

Evidence: E=0

$P(B | A)$

A	B=0	B=1
0	.4	.6
1	.1	.9

$P(C | A)$

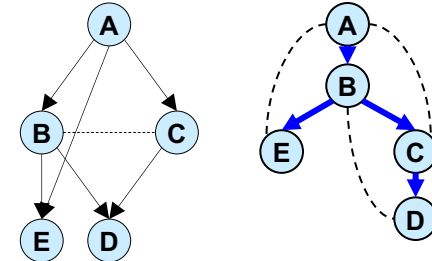
A	C=0	C=1
0	.2	.8
1	.7	.3

$P(A)$

A	P(A)
0	.6
1	.4

$P(D=1, E=0)=?$

.24408



OR

AND

OR

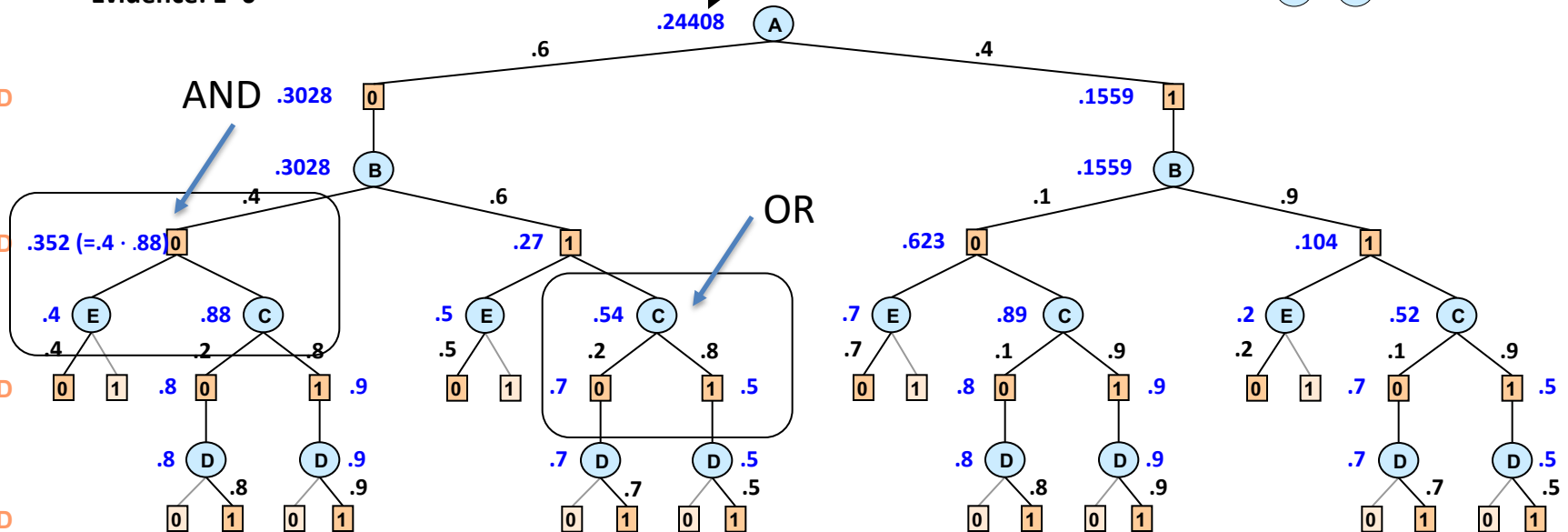
AND

OR

AND

OR

AND



$P(D | B, C)$

B	C	D=0	D=1
0	0	.2	.8
0	1	.1	.9
1	0	.3	.7
1	1	.5	.5

Evidence: D=1

Value of node = updated belief for sub-problem below

AND node: product

OR node: Marginalization by summation

$$\prod_{n' \in \text{children}(n)} v(n')$$

$$\sum_{n' \in \text{children}(n)} w(n, n') v(n')$$

The Value Function

$P(E | A, B)$

A	B	E=0	E=1
0	0	.4	.6
0	1	.5	.5
1	0	.7	.3
1	1	.2	.8

Evidence: E=0

$P(B | A)$

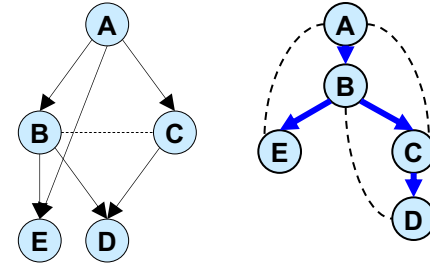
A	B=0	B=1
0	.4	.6
1	.1	.9

$P(C | A)$

A	C=0	C=1
0	.2	.8
1	.7	.3

$P(A)$

A	P(A)
0	.6
1	.4



OR

AND

OR

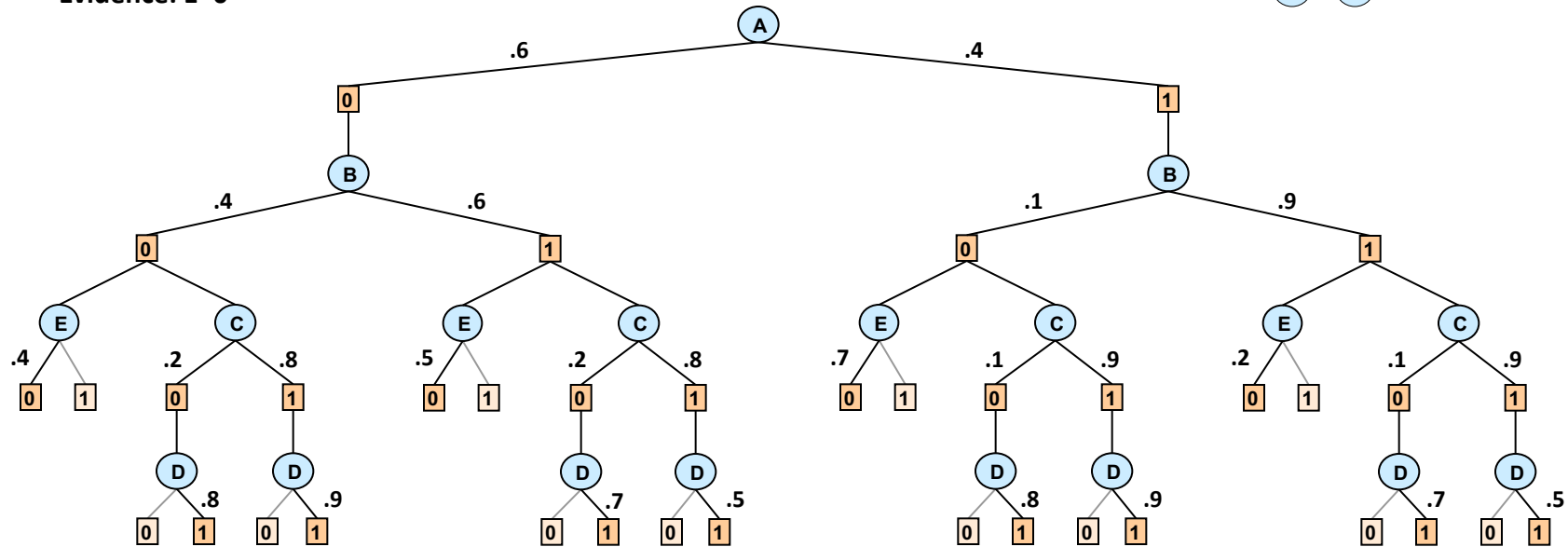
AND

OR

AND

OR

AND



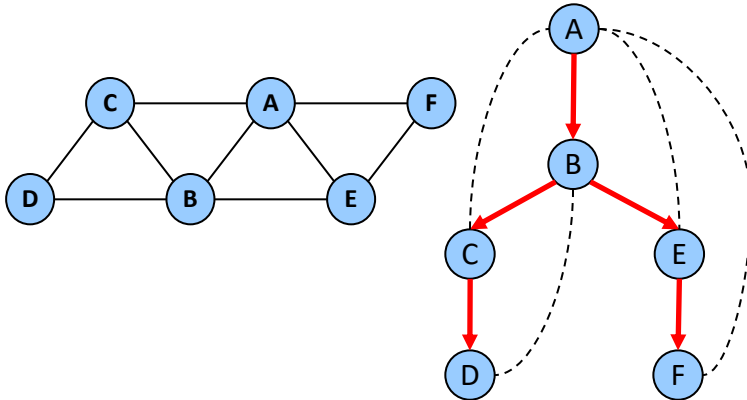
$P(D | B, C)$

B	C	D=0	D=1
0	0	.2	.8
0	1	.1	.9
1	0	.3	.7
1	1	.5	.5

Evidence: D=1, E=0

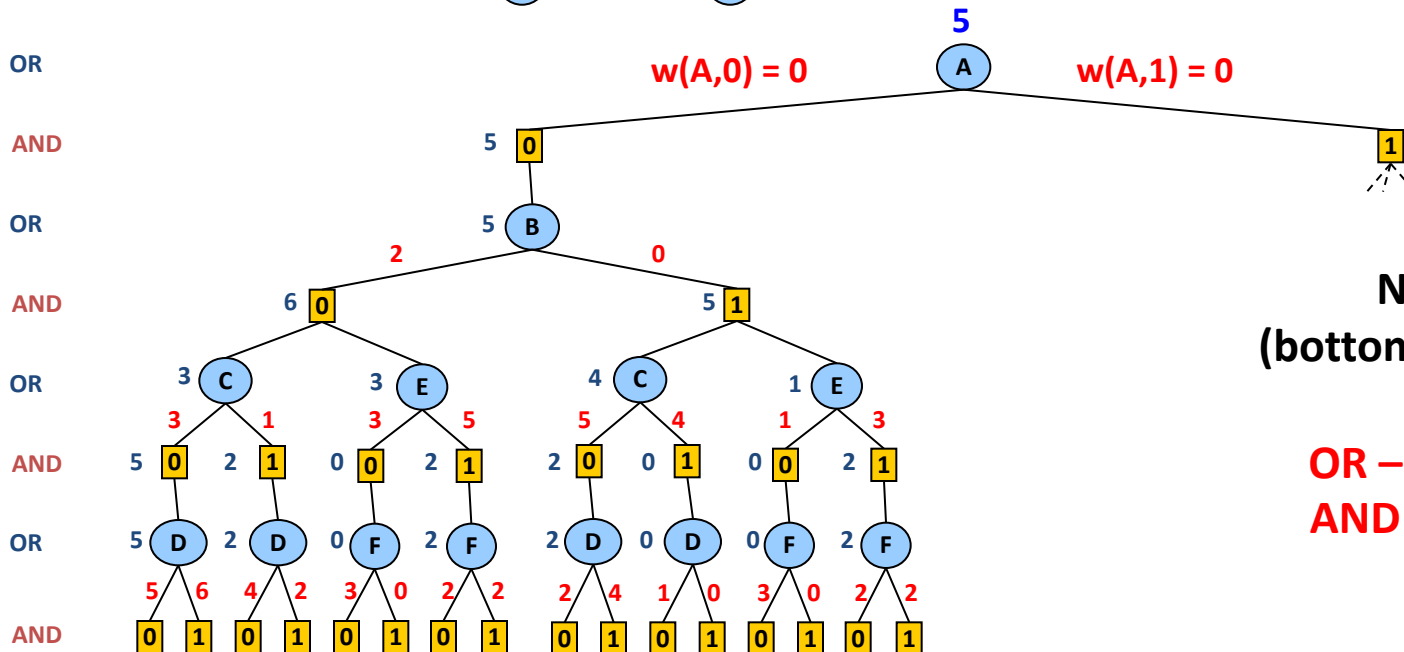
- $V(n)$ is dictated by the query of interest
- $V(n)$ the value of the sub-problem represented by $T(n)$
- For sum-inference it is the probability mass below n
- Can be computed recursively based on child values.

The Value Function for Optimization



A	B	f ₁	A	C	f ₂	A	E	f ₃	A	F	f ₄	B	C	f ₅	B	D	f ₆	B	E	f ₇	C	D	f ₈	E	F	f ₉
0	0	2	0	0	3	0	0	0	0	0	2	0	0	0	0	0	4	0	0	3	0	0	1	0	0	1
0	1	0	0	1	0	0	1	3	0	1	0	0	1	1	0	1	2	0	1	2	0	1	4	0	1	0
1	0	1	1	0	0	1	0	2	1	0	0	1	0	2	1	0	1	1	0	1	1	0	0	1	0	0
1	1	4	1	1	1	1	1	0	1	1	2	1	1	4	1	1	0	1	1	0	1	1	0	1	1	2

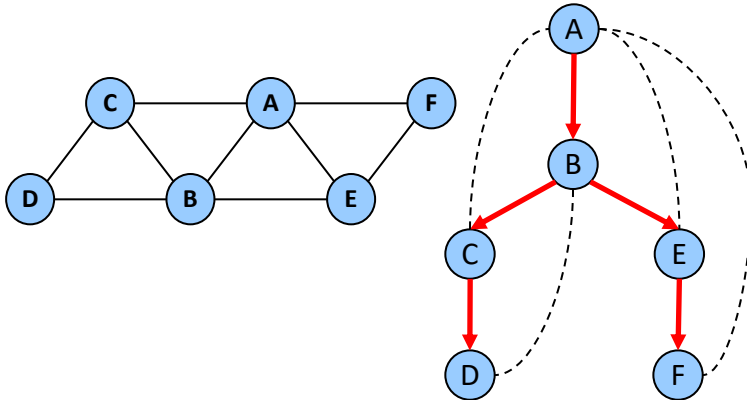
Objective function: $F^* = \min_x \sum_{\alpha} f_{\alpha}(x_{\alpha})$



Node Value
(bottom-up evaluation)

OR – minimization
AND – summation

The Value Function for Optimization



A	B	f_1	A	C	f_2	A	E	f_3	A	F	f_4	B	C	f_5	B	D	f_6	B	E	f_7	C	D	f_8	E	F	f_9
0	0	2	0	0	3	0	0	0	0	0	2	0	0	0	0	0	4	0	0	3	0	0	1	0	0	1
0	1	0	0	1	0	0	1	3	0	1	0	0	1	1	0	1	2	0	1	2	0	1	4	0	1	0
1	0	1	1	0	0	1	0	2	1	0	0	1	0	2	1	0	1	1	0	1	1	0	0	1	0	0
1	1	4	1	1	1	1	1	0	1	1	2	1	1	4	1	1	0	1	1	0	1	1	0	1	1	2

Objective function: $F^* = \min_x \sum_{\alpha} f_{\alpha}(x_{\alpha})$

OR

AND

OR

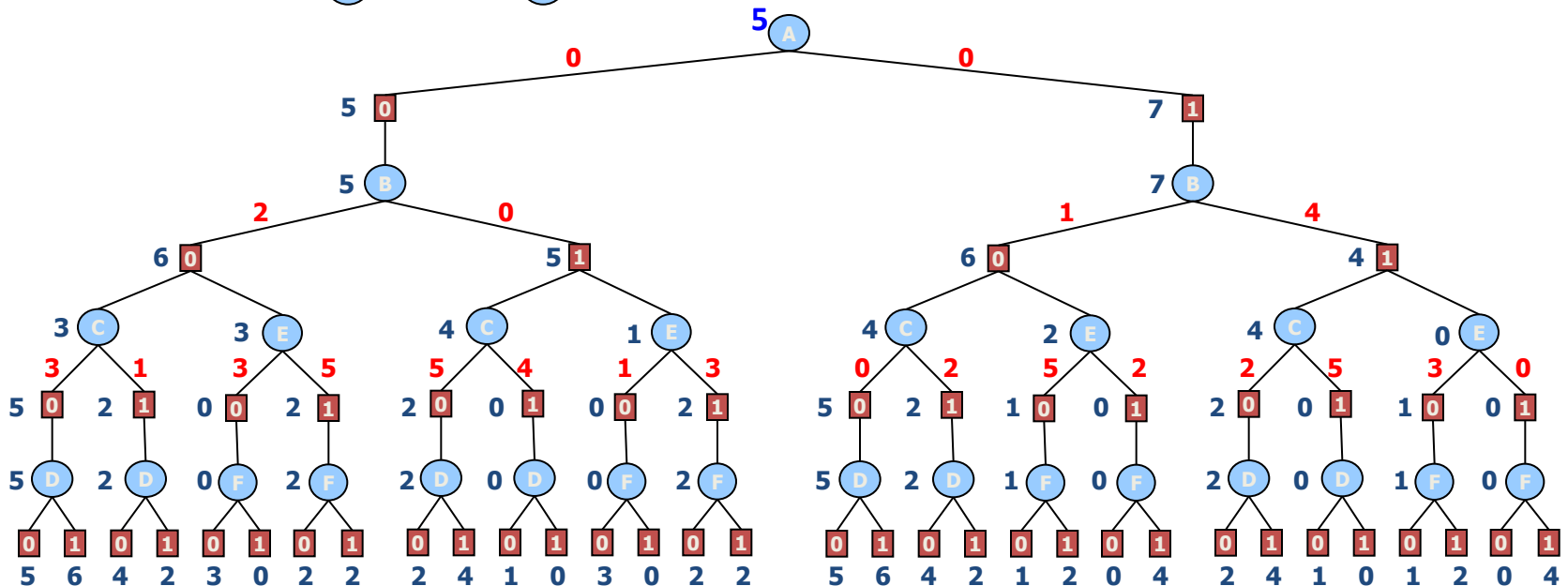
AND

OR

AND

OR

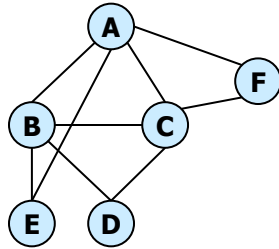
AND



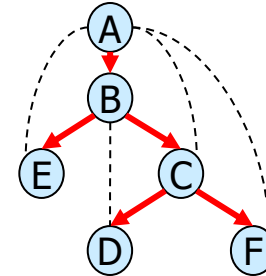
AND node = Combination operator (summation)

OR node = Marginalization operator (minimization)

The AND/OR Counting Value(#CSP)



solution



OR

AND

OR

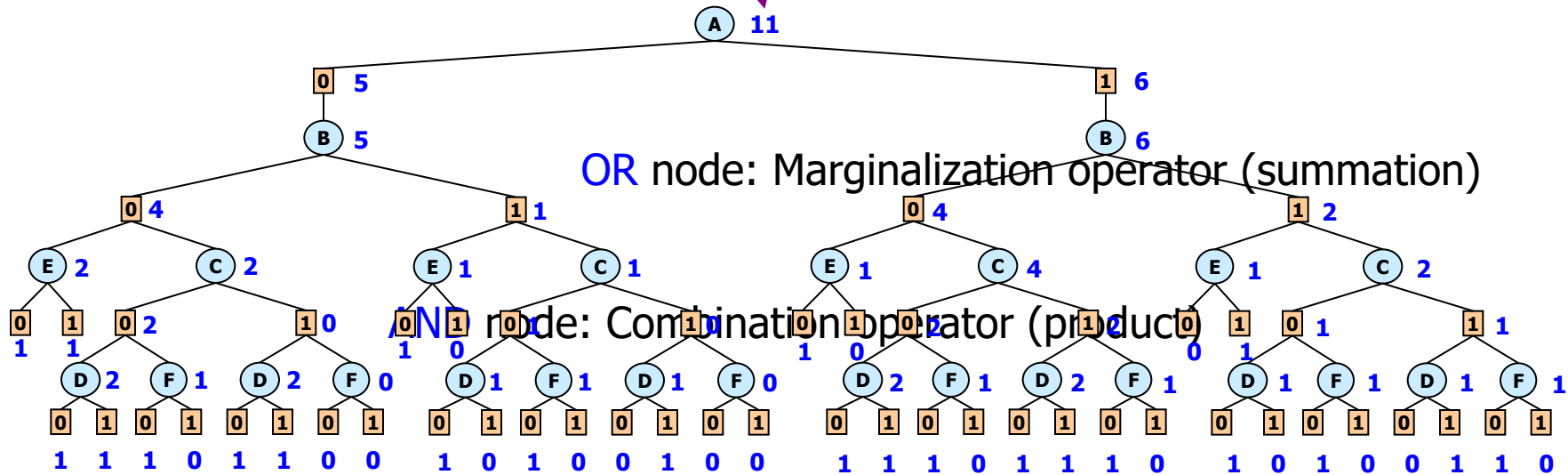
AND

OR

AND

OR

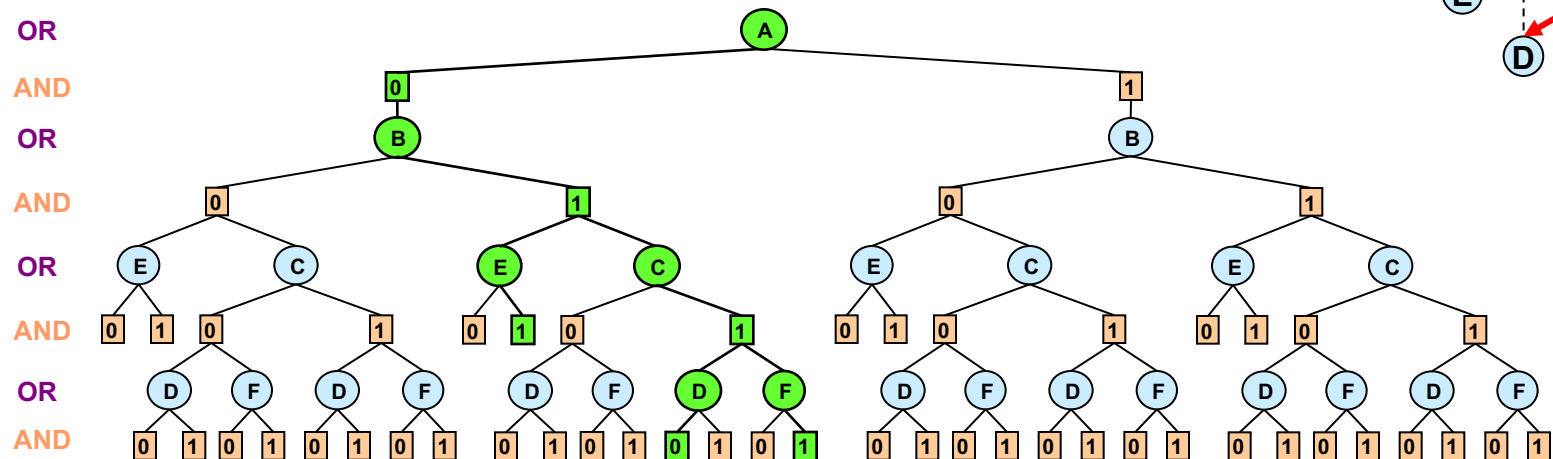
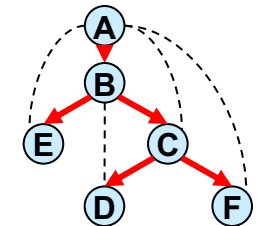
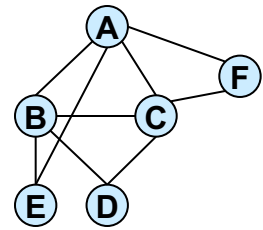
AND



Value of node = number of solutions below it

Summary: AND/OR Search Tree for GMs

- The AND/OR search tree of R relative to a pseudo-tree, T, has:
 - Alternating levels of: **OR** nodes (variables) and **AND** nodes (values)
- Successor function:
 - The successors of **OR nodes X** are all its consistent values along its path
 - The successors of **AND $\langle X, v \rangle$** are all X child variables in T
 - Arc-weight are assigned from the model factors
- A **solution** is a consistent subtree. Its cost, the product of the weights.
- Query:** compute the value of the root node



Size and Traversal of AND/OR Search Tree

	AND/OR tree	OR tree
Space	$O(n)$	$O(n)$
Size= Time	$O(n k^h)$ $O(n k^{w^* \log n})$ (Freuder & Quinn85), (Collin, Dechter & Katz91), (Bayardo & Miranker95), (Darwiche01)	$O(k^n)$

k = domain size

h = height of pseudo-tree

n = number of variables

w^* = treewidth

$$h \leq w^* \log n$$

AND/OR vs. OR Spaces

width	height	OR space		AND/OR space		
		Time (sec.)	Nodes	Time (sec.)	AND nodes	OR nodes
5	10	3.15	2,097,150	0.03	10,494	5,247
4	9	3.13	2,097,150	0.01	5,102	2,551
5	10	3.12	2,097,150	0.03	8,926	4,463
4	10	3.12	2,097,150	0.02	7,806	3,903
5	13	3.11	2,097,150	0.10	36,510	18,255

Random graphs with 20 nodes, 20 edges and 2 values per node



Pseudo-trees

PseudoTree Definition

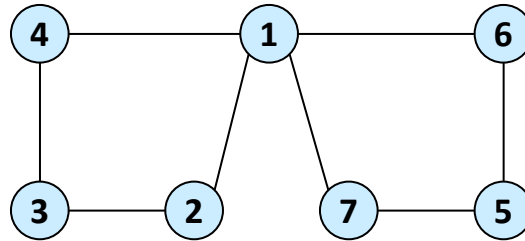
Given undirected graph $G = (V, E)$, a directed rooted tree $T = (V, E')$ defined on all its nodes is a pseudo tree if any arc of G which is not included in E' is a back-arc in T , namely it connects a node in T to an ancestor in T . The arcs in E' may not all be included in E .

Given a pseudo tree T of G , the extended graph of G relative to T includes also the arcs in E' that are not in E : as $GT = (V, E \cup E')$.

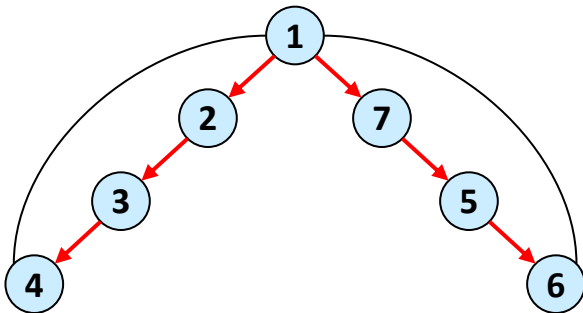
Pseudo Trees

A **pseudo-tree** of a graph is a tree spanning its nodes, where all arcs in the graph not in the tree are back-arcs

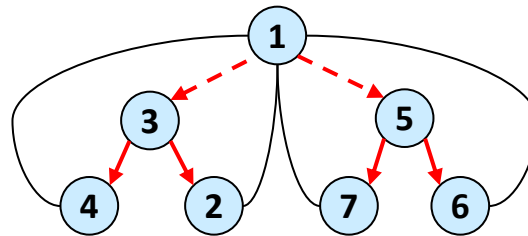
$$h \leq w^* \log n$$



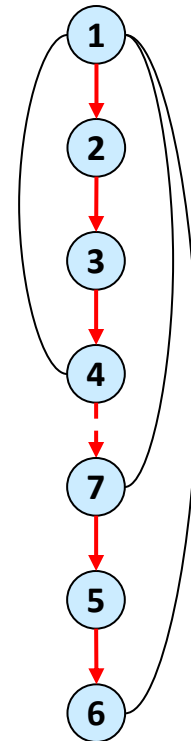
(a) Graph



(b) DFS tree
height=3

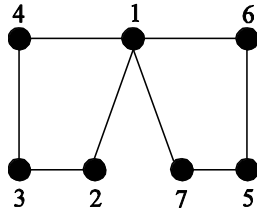


(c) Pseudo tree
height=2

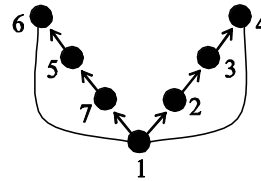


(d) Chain
height=6

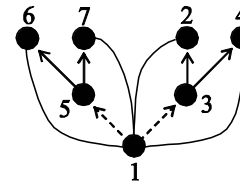
From DFS-Trees to Pseudo-Trees



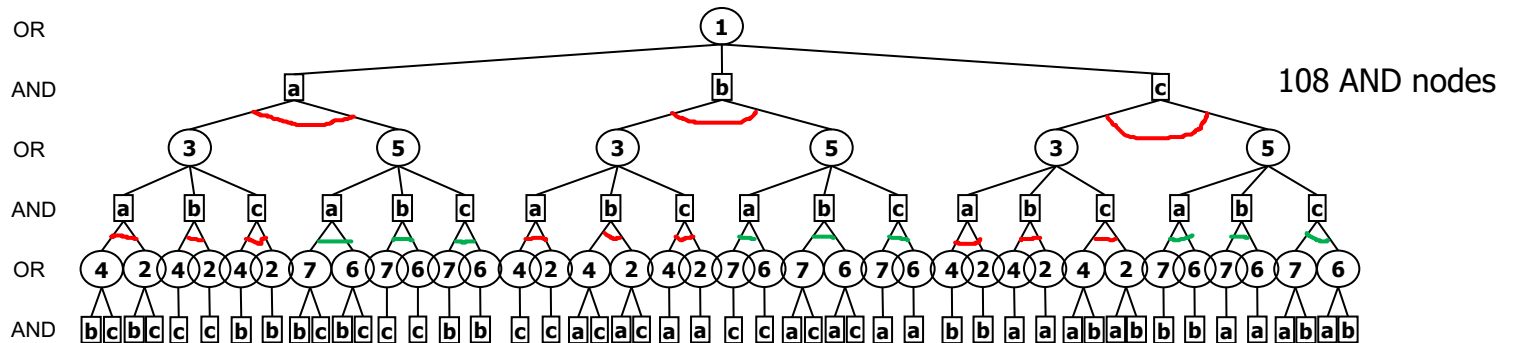
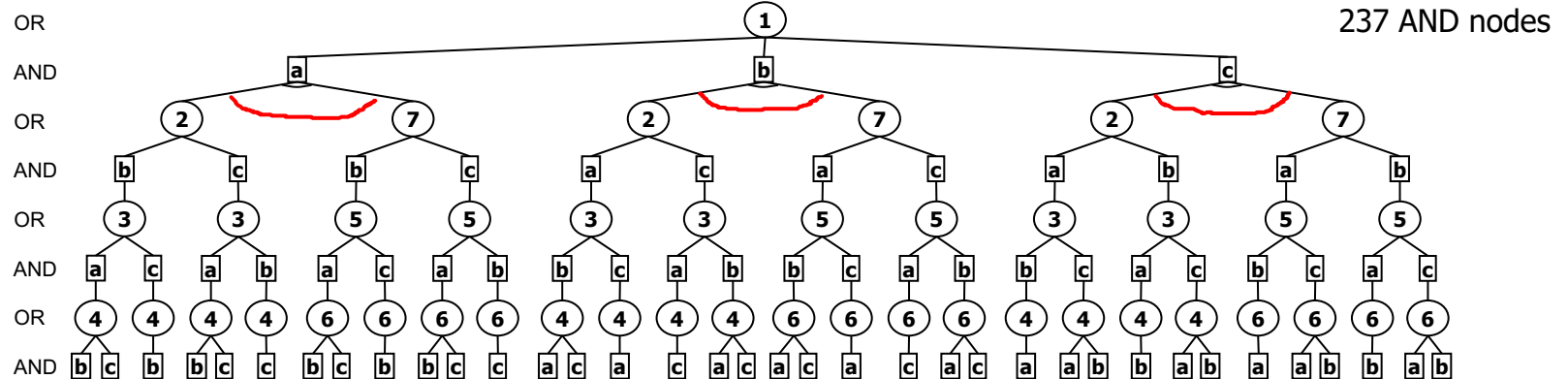
(a)



(b)

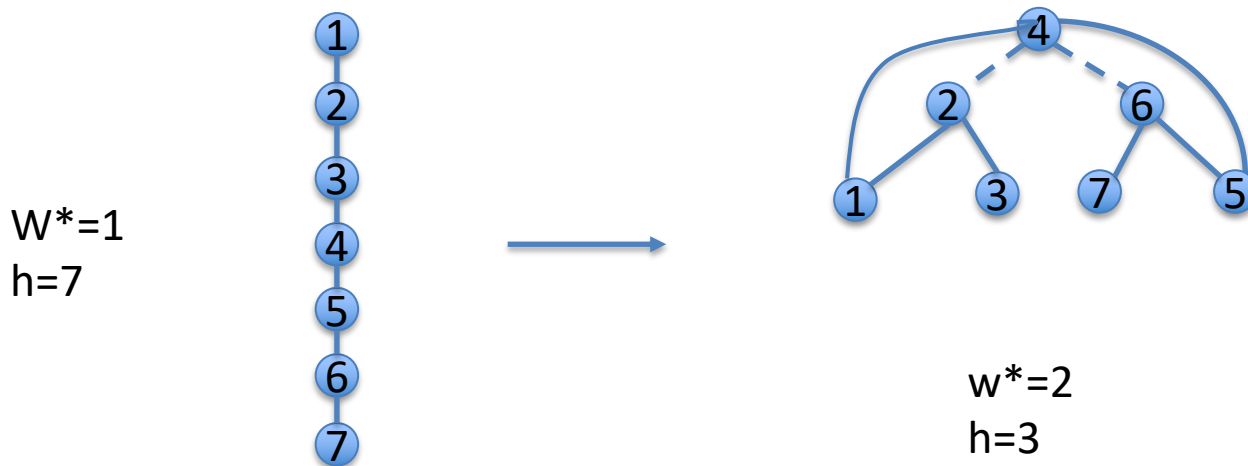


(c)



Finding Min-height Pseudo-Trees

- Finding a min height pseudo-tree is NP-complete, but:
- Given a tree-decomposition with treewidth w^* , there exists a pseudo-tree whose height satisfies
 - $h \leq w^* \log n$
- Optimality of h and w^* cannot be achieved at once.



AND/OR Search-Tree Properties

(k = domain size, h = pseudo-tree height. n = number of variables)

- **Theorem:** Any AND/OR search tree based on a pseudo-tree is sound and complete (expresses all and only solutions)
- **Theorem:** Size of AND/OR search tree is $O(n k^h)$
Size of OR search tree is $O(k^n)$
- **Theorem:** Size of AND/OR search tree can be bounded by $O(\exp(w^* \log n))$
- When the pseudo-tree is a chain we get an OR space

Summary: Queries and Value of Nodes

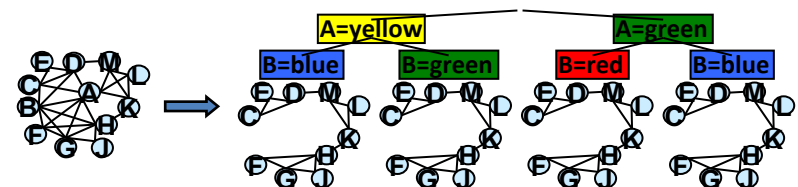
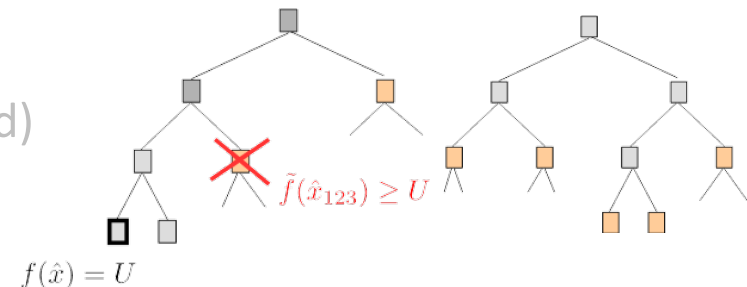
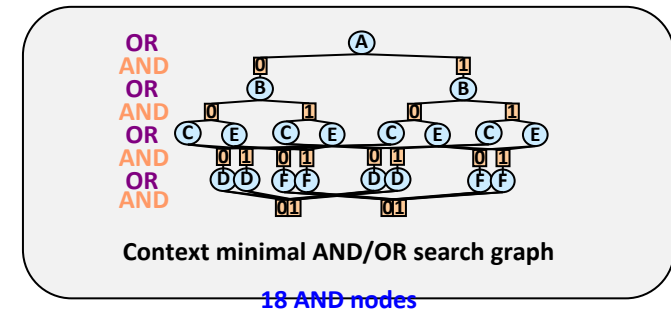
- $V(n)$ is the value of the tree $T(n)$ for the task:
 - Counting: $v(n)$ is number of solutions in $T(n)$
 - Consistency: $v(n)$ is 0 if $T(n)$ inconsistent, 1 otherwise.
 - Max-Inference: $v(n)$ is the optimal solution in $T(n)$
 - Sum-Inference: $v(n)$ is probability of evidence in $T(n)$.
 - Mixed-Inference: $v(n)$ is the marginal map in $T(n)$.
- Goal: compute the value of the root node recursively traversing the AND/OR tree.

Complexity of searching depth-first is

- Space: $O(n)$
- Time: $O(nk^h)$
- Time: $O(k^{w \cdot \log n})$

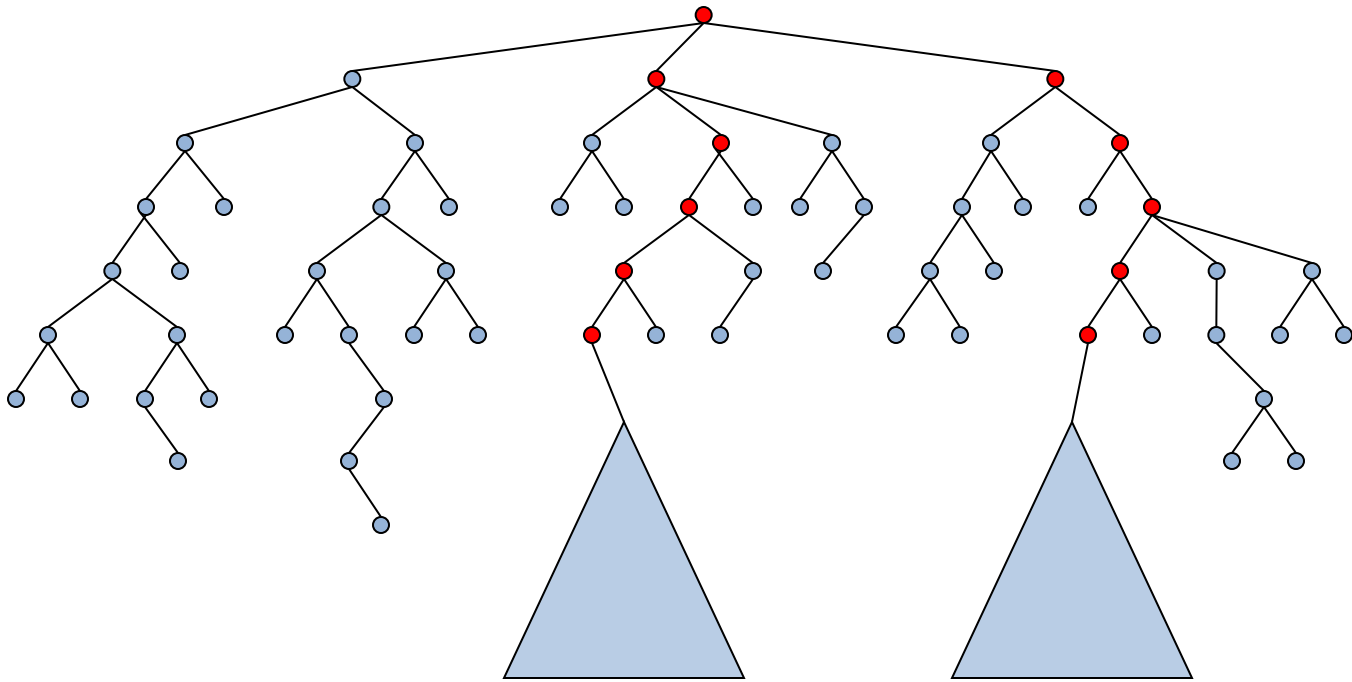
Outline: Search

- Review Graphical Models
- AND/OR search spaces, pseudo-trees
 - AND/OR search trees
 - **AND/OR search graphs**
 - Generating good pseudo-trees
 - Brute-force AND/OR
- Heuristic search (HS) for AND/OR spaces
 - Basic Heuristic search (Depth and Best)
 - AND/OR Depth-first HS (branch and bound)
 - AND/OR Best-first heuristic search
 - The Guiding MBE heuristic
 - Marginal Map (max-sum-product)
- Hybrids of search and Inference
- Summary and Class 2



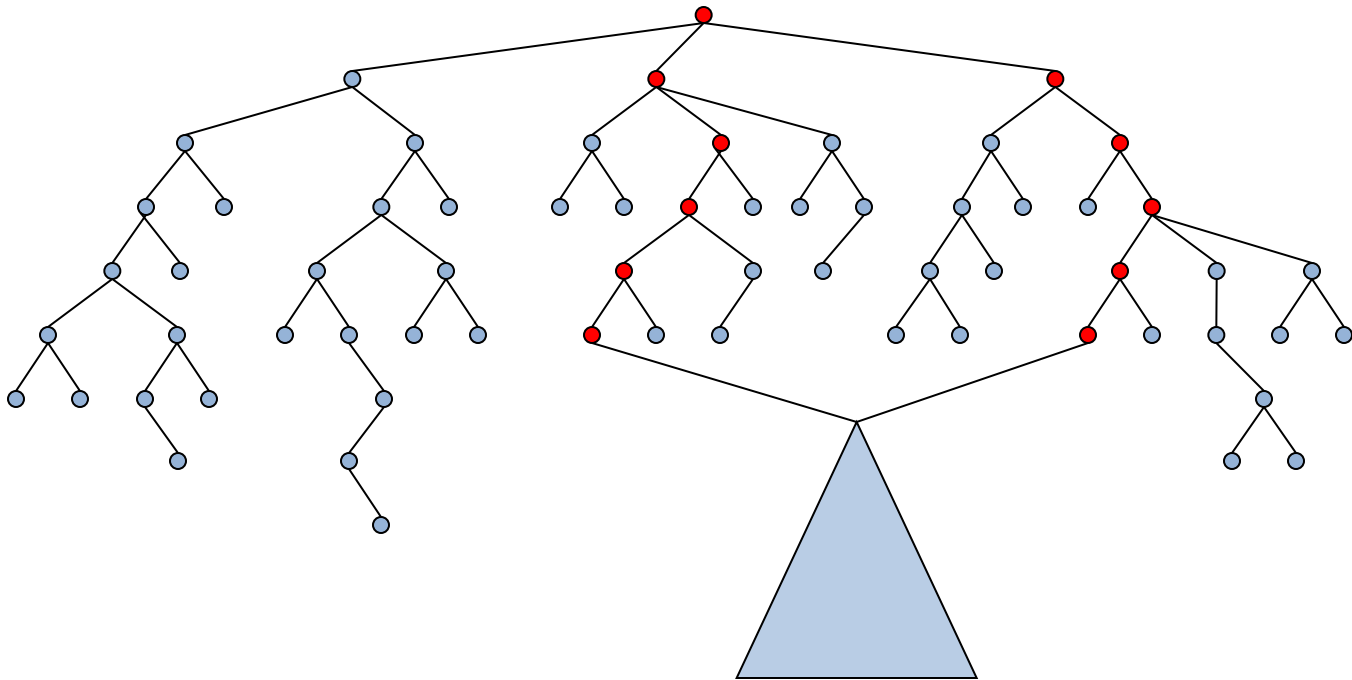
From Search Trees to Search Graphs

- Any two nodes that root **identical** subtrees or subgraphs can be **merged**

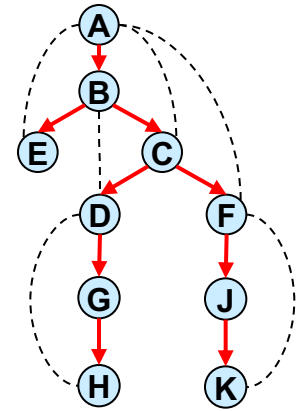
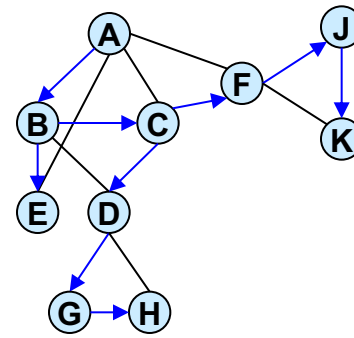


From Search Trees to Search Graphs

- Any two nodes that root **identical** subtrees or subgraphs can be **merged**



AND/OR Tree



OR

AND

OR

AND

OR

AND

OR

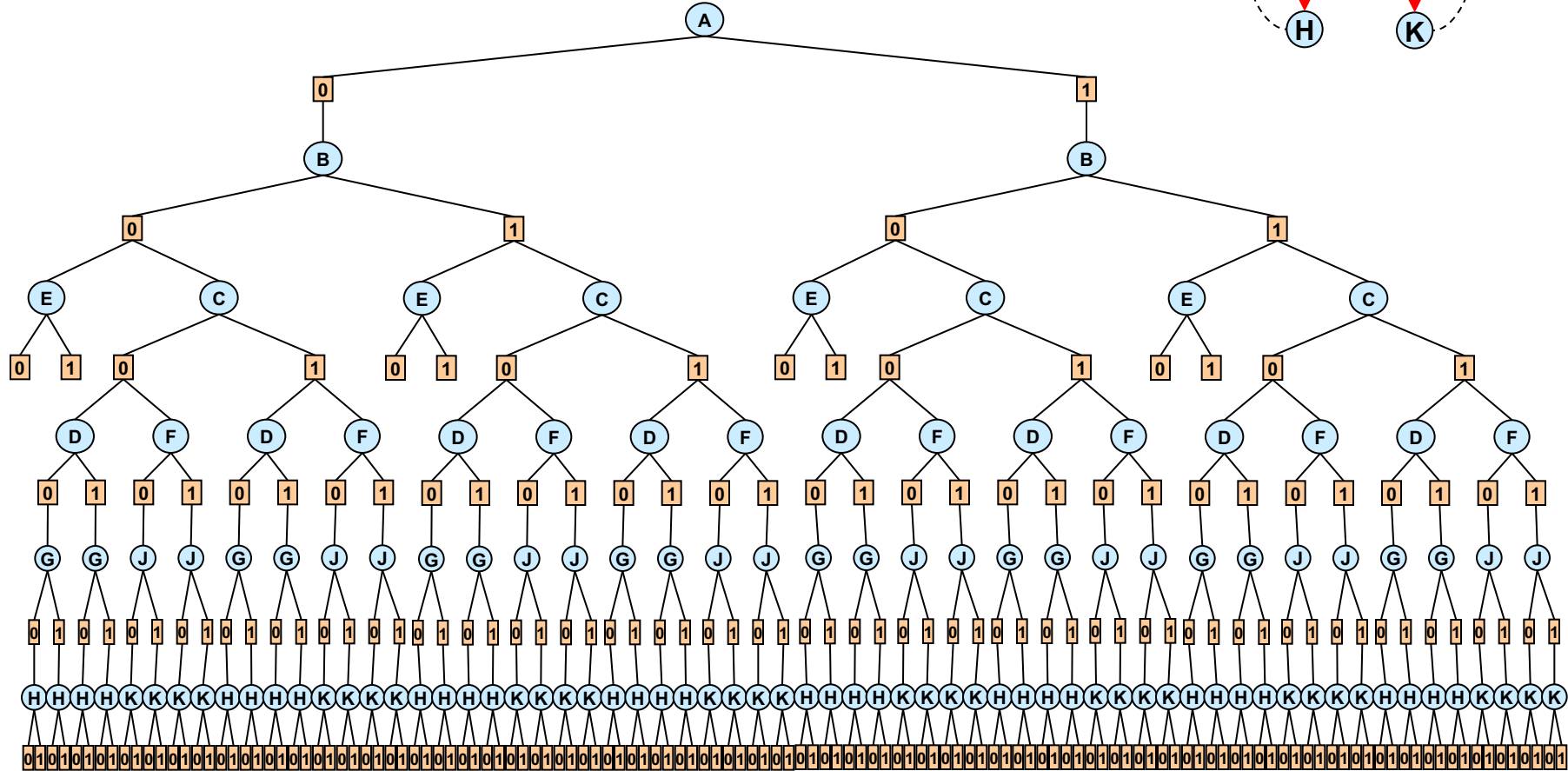
AND

OR

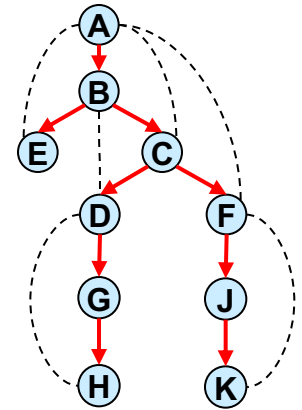
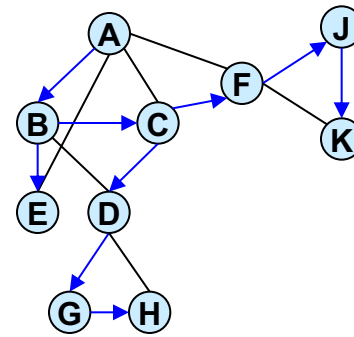
AND

OR

AND



AND/OR Graph



OR

AND

OR

AND

OR

AND

OR

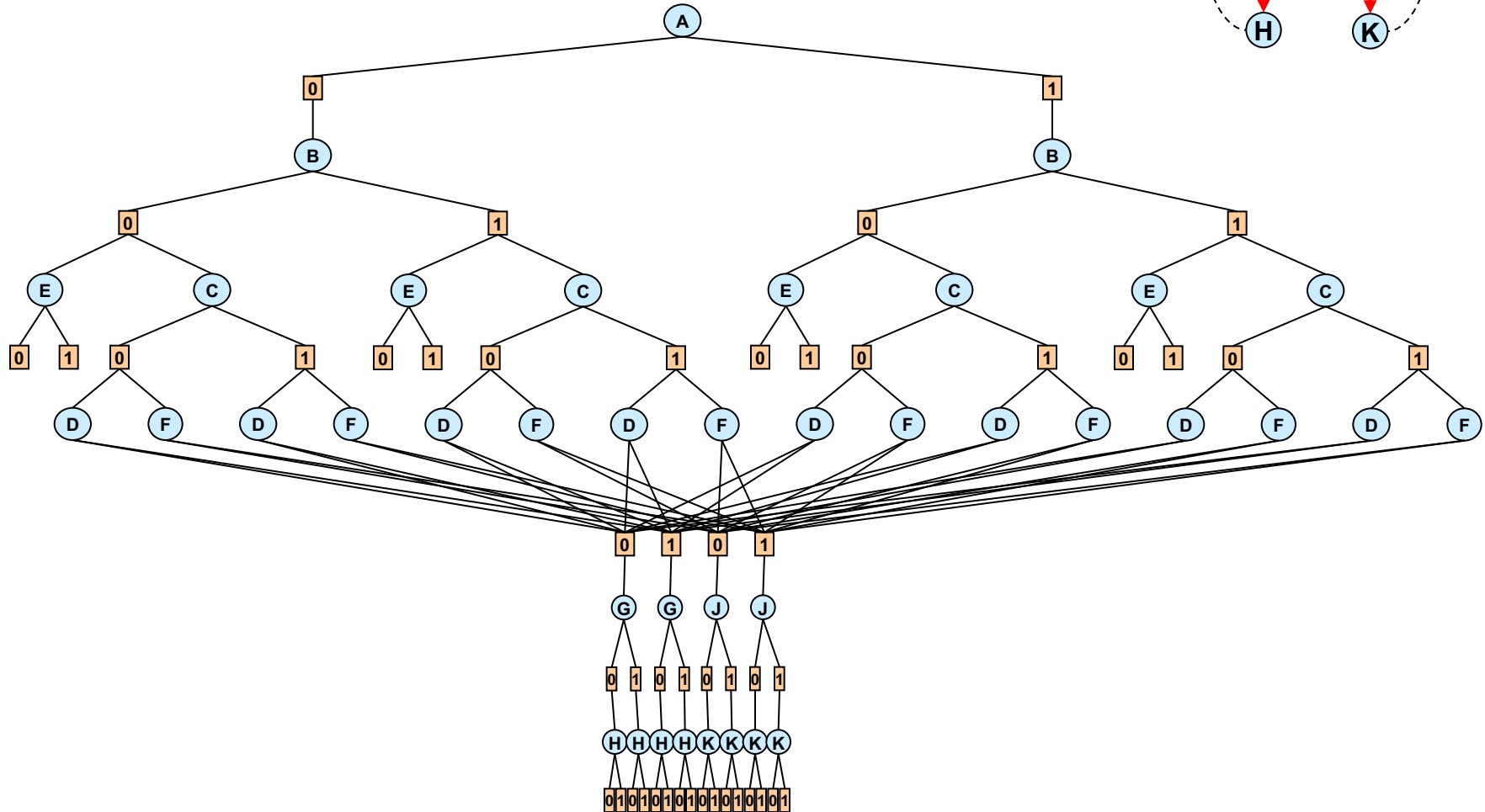
AND

OR

AND

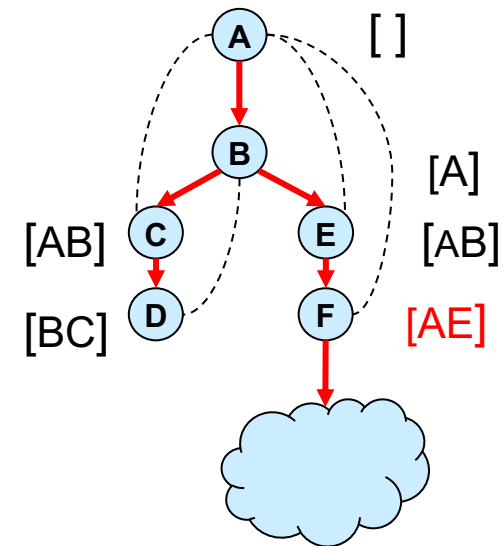
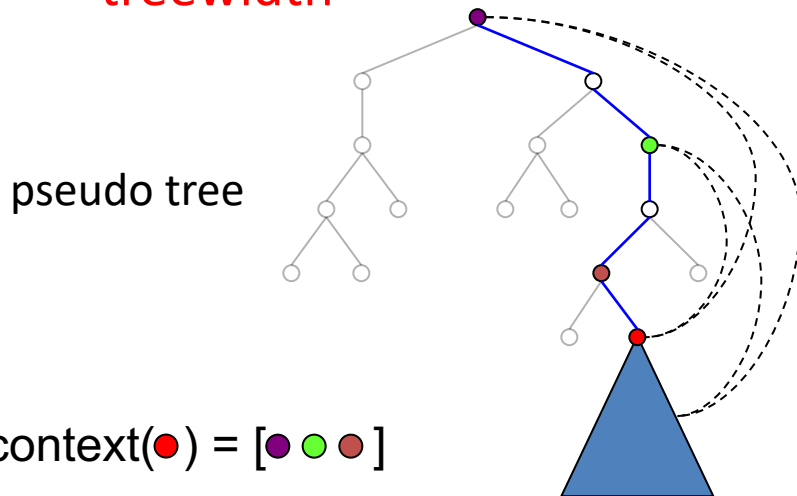
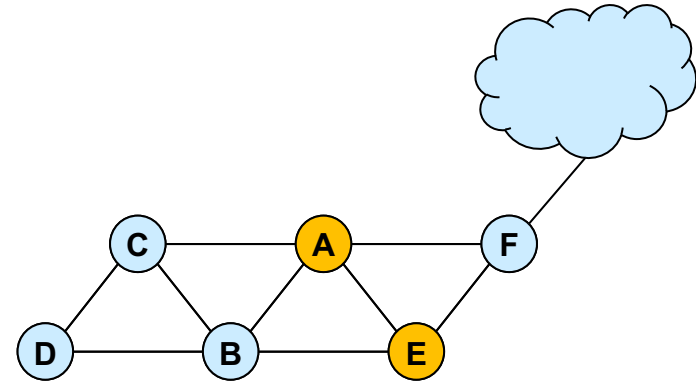
OR

AND



Merging Based on Context

- context (X) = ancestors of X in pseudo tree, connected to X, or to descendants of X
- context (X) = parents in the induced graph
- max |context| = induced width = treewidth



Context-Based Minimal AND/OR Search Graph

Definition 7.2.13 (context minimal AND/OR search graph) *The AND/OR search graph of M guided by a pseudo-tree T that is closed under context-based merge operator, is called the context minimal AND/OR search graph and is denoted by $C_T(R)$.*

AND/OR Tree DFS Algorithm (Value=Sum-Product)

$P(E | A, B)$

A	B	E=0	E=1
0	0	.4	.6
0	1	.5	.5
1	0	.7	.3
1	1	.2	.8

Evidence: E=0

$P(B | A)$

A	B=0	B=1
0	.4	.6
1	.1	.9

$P(C | A)$

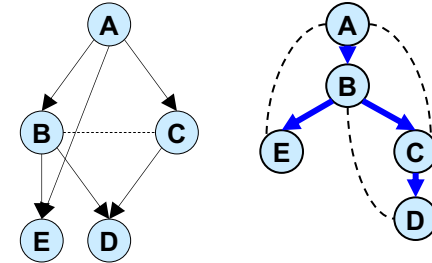
A	C=0	C=1
0	.2	.8
1	.7	.3

$P(A)$

A	P(A)
0	.6
1	.4

Result: $P(D=1, E=0)$

.24408



OR

AND

OR

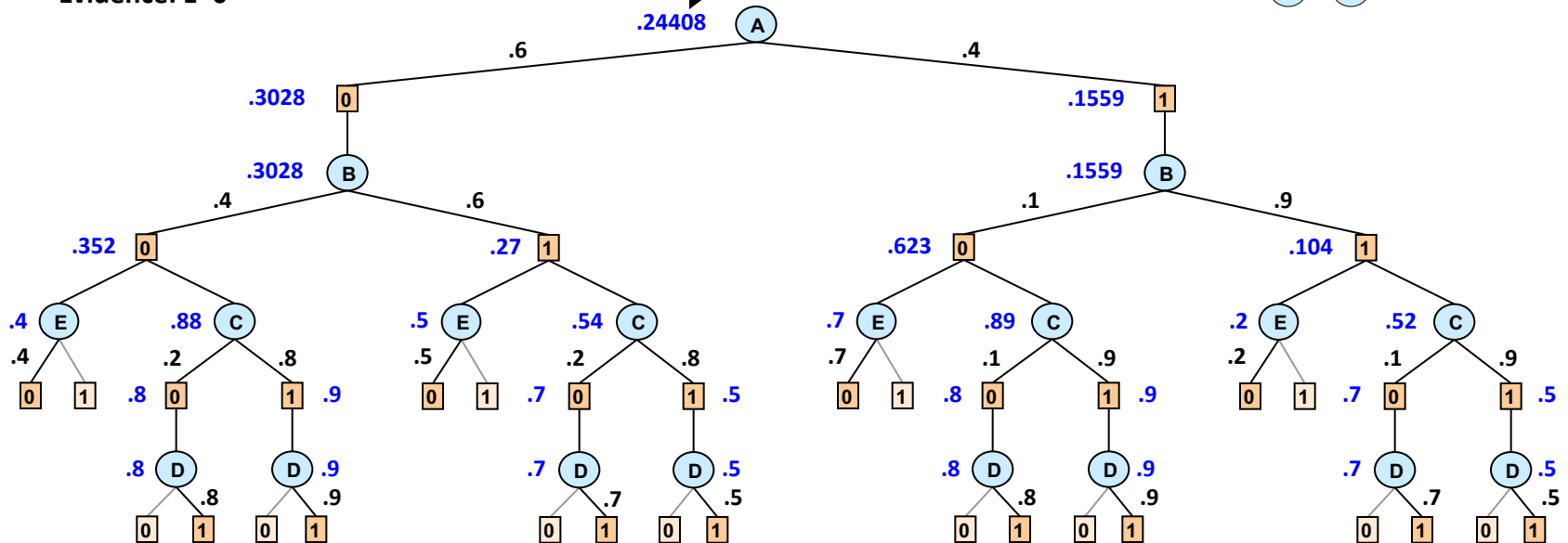
AND

OR

AND

OR

AND



$P(D | B, C)$

B	C	D=0	D=1
0	0	.2	.8
0	1	.1	.9
1	0	.3	.7
1	1	.5	.5

Evidence: D=1, E=0

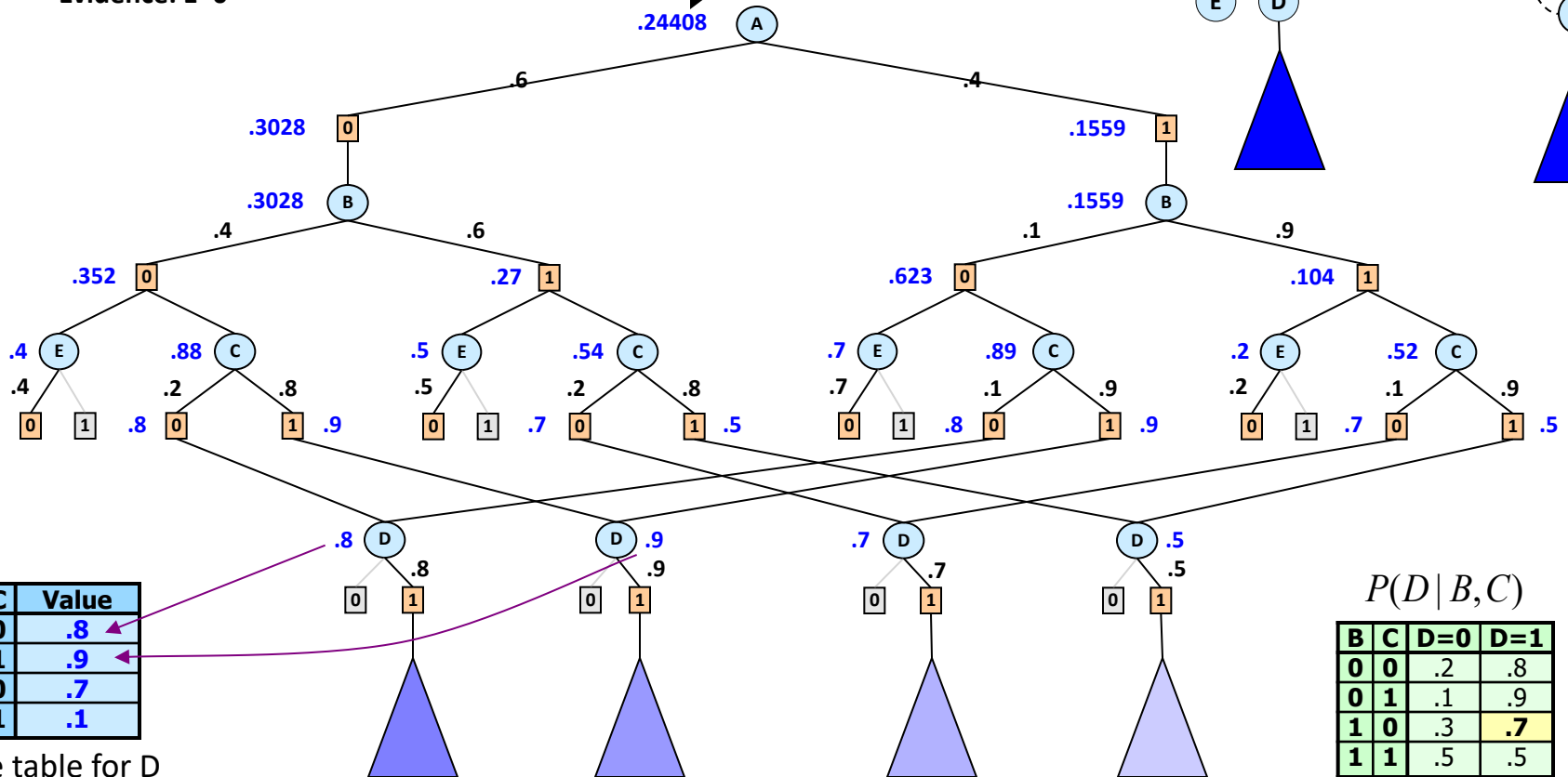
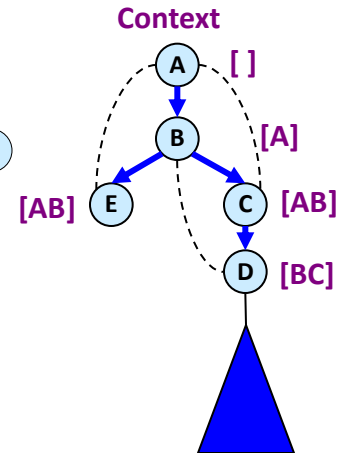
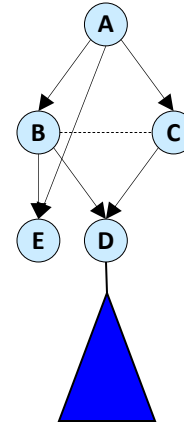
A	B	E=0	E=1
0	0	.4	.6
0	1	.5	.5
1	0	.7	.3
1	1	.2	.8

A	B=0	B=1
0	.4	.6
1	.1	.9

A	C=0	C=1
0	.2	.8
1	.7	.3

A	P(A)
0	.6
1	.4

.24408



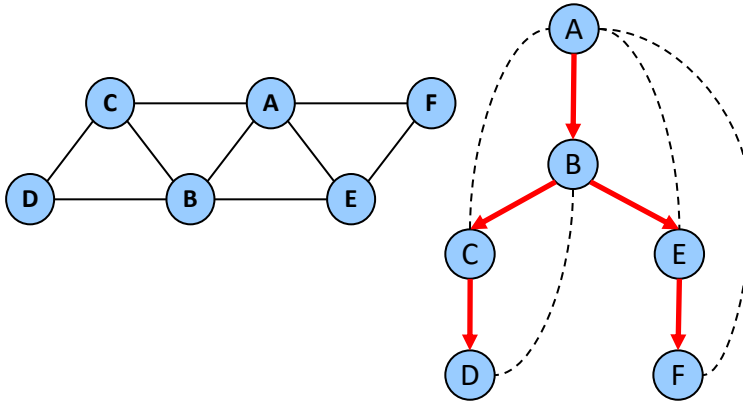
B	C	Value
0	0	.8
0	1	.9
1	0	.7
1	1	.1

Cache table for D

B	C	D=0	D=1
0	0	.2	.8
0	1	.1	.9
1	0	.3	.7
1	1	.5	.5

Evidence: D=1,E=0

AND/OR Search Graph (Optimization)



A	B	f ₁	A	C	f ₂	A	E	f ₃	A	F	f ₄	B	C	f ₅	B	D	f ₆	B	E	f ₇	C	D	f ₈	E	F	f ₉
0	0	2	0	0	3	0	0	0	0	0	2	0	0	0	0	0	4	0	0	3	0	0	1	0	0	1
0	1	0	0	1	0	0	1	3	0	1	0	0	1	1	0	1	2	0	1	2	0	1	4	0	1	0
1	0	1	1	0	0	1	0	2	1	0	0	1	0	2	1	0	1	1	0	1	1	0	0	1	0	0
1	1	4	1	1	1	1	1	0	1	1	2	1	1	4	1	1	0	1	1	0	1	1	0	1	1	2

Objective function: $F^* = \min_x \sum_{\alpha} f_{\alpha}(x_{\alpha})$

OR

AND

OR

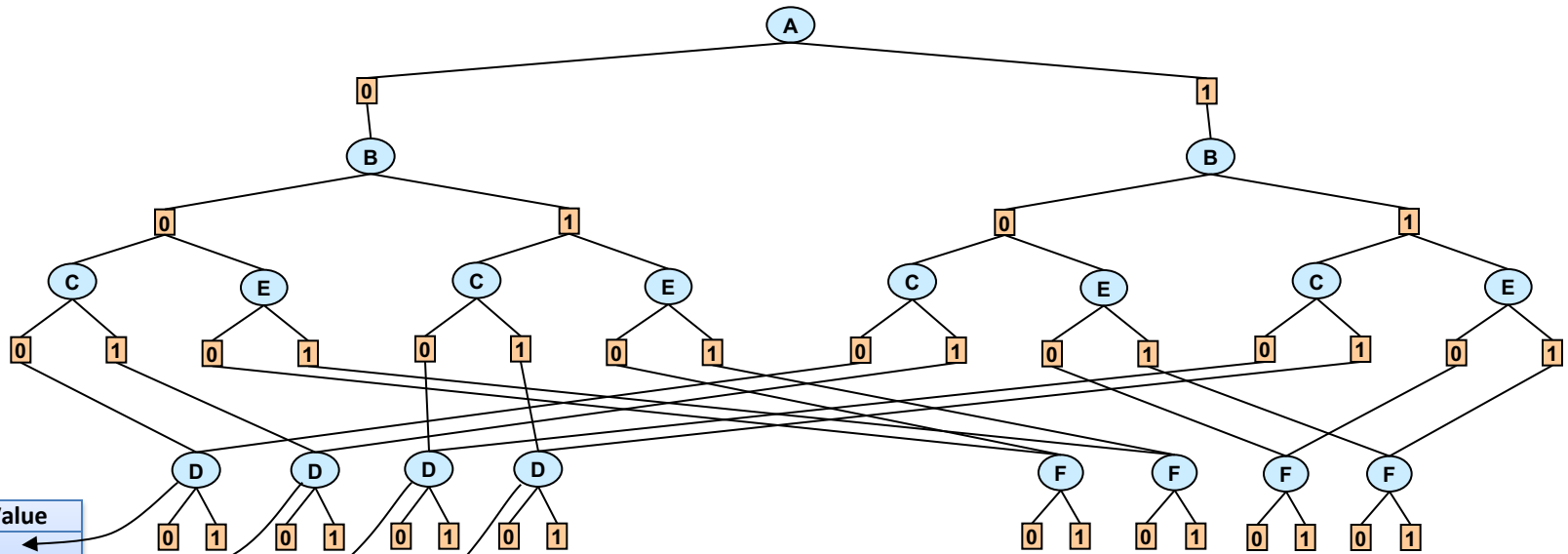
AND

OR

AND

OR

AND



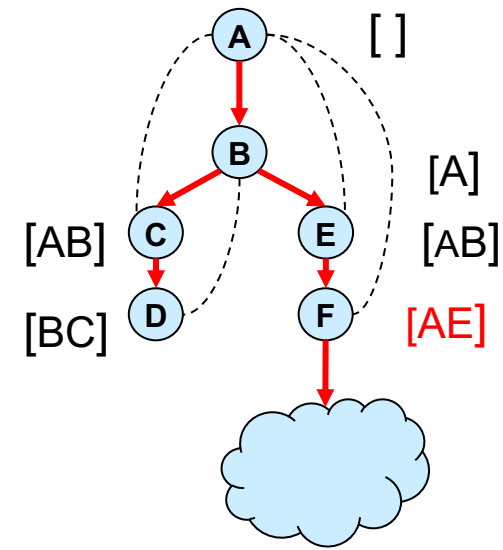
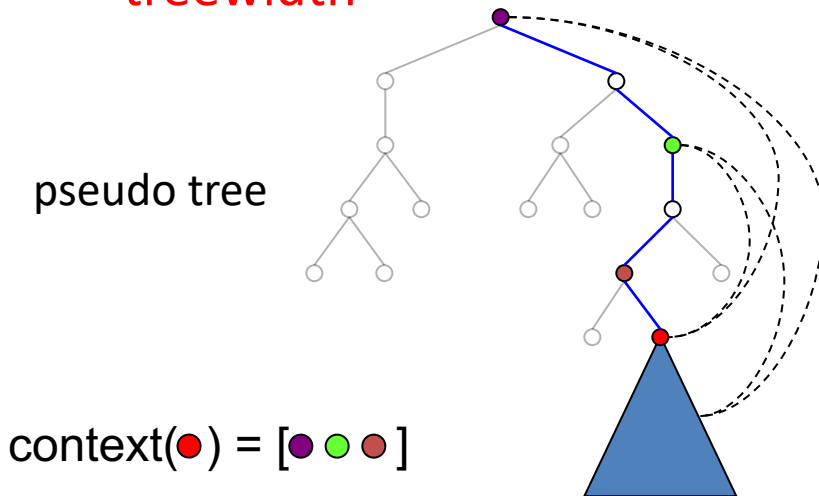
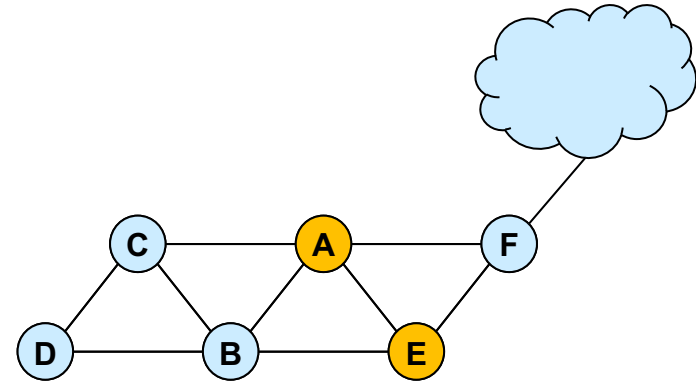
B	C	Value
0	0	←
0	1	←
1	0	←
1	1	←

Context minimal AND/OR search graph

Cache table for D

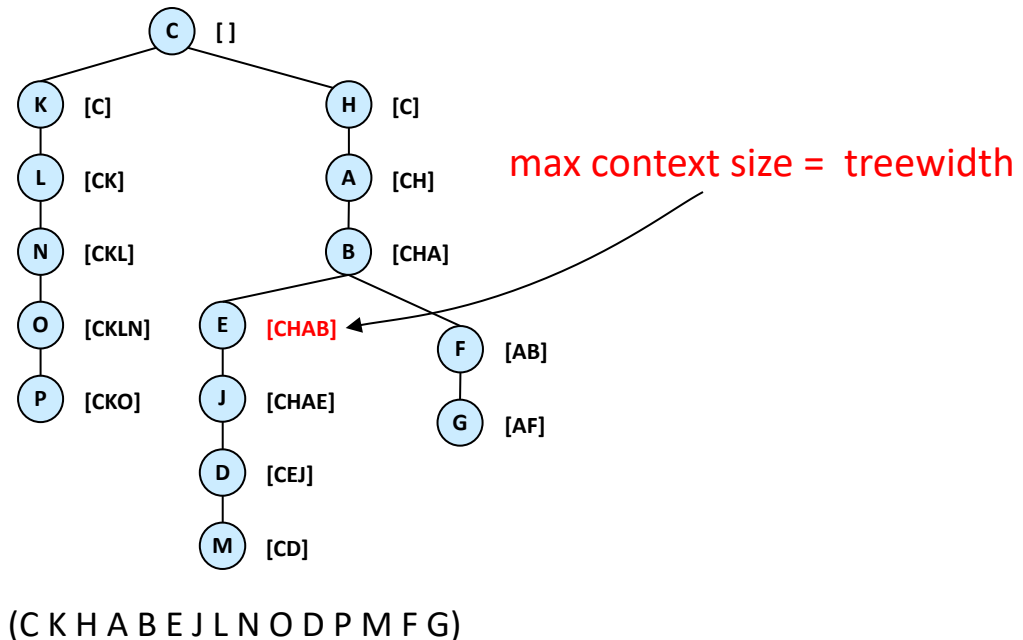
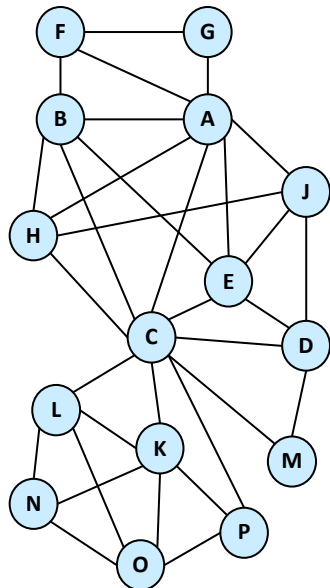
Merging Based on Context

- (OR) context (X) = ancestors of X in pseudo tree, connected to X, or to descendants of X
- context (X) = parents in the induced graph
- $\max |\text{context}| = \text{induced width} = \text{treewidth}$

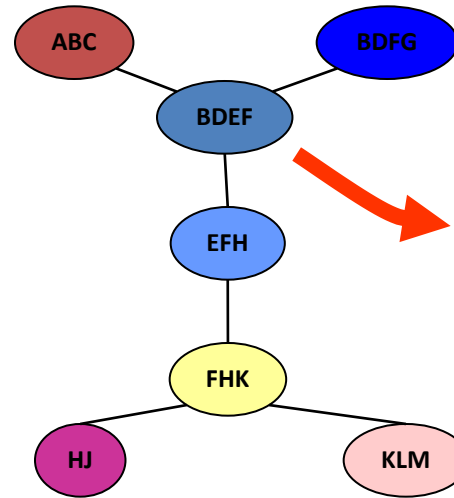
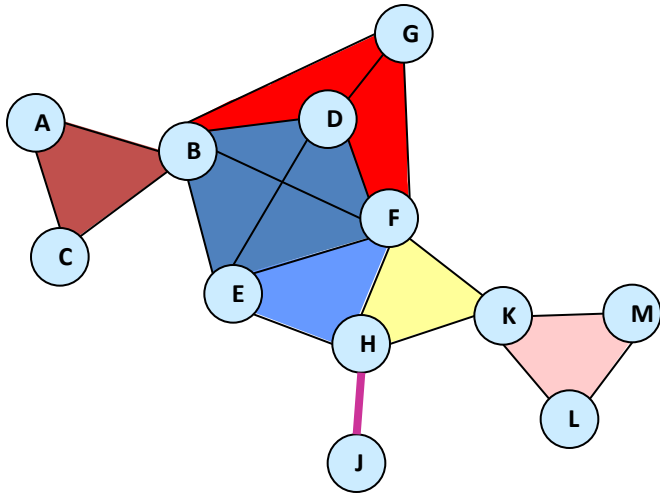


How Big Is The Context?

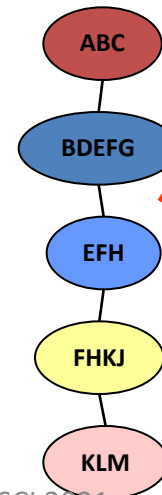
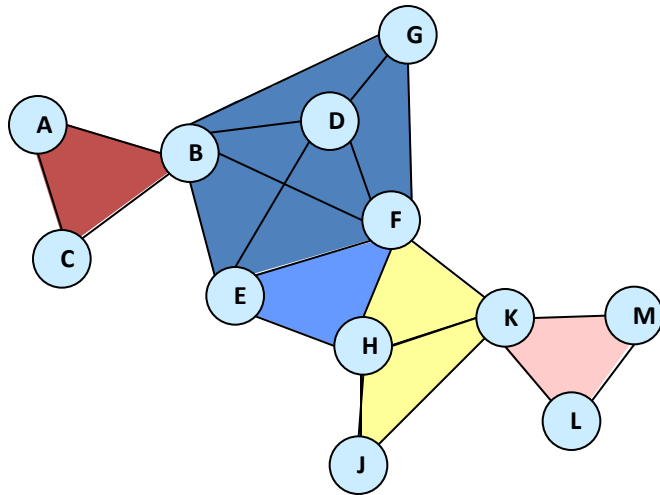
- Theorem:** The maximum context-size of a pseudo-tree equals the **treewidth** along the pseudo tree.



Treewidth vs. Pathwidth

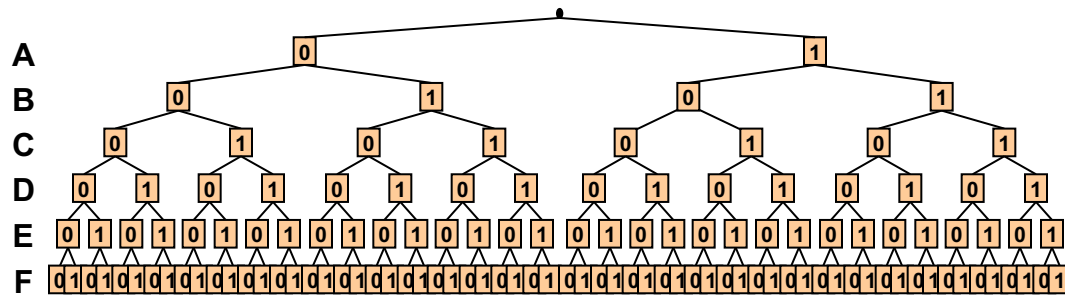
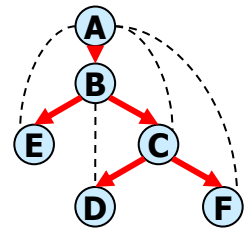


treewidth = 3
= (max cluster size) - 1



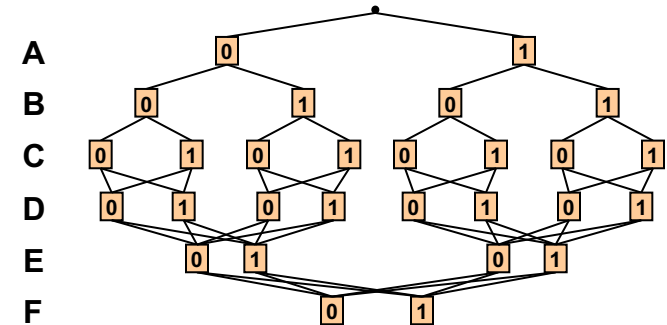
pathwidth = 4
= (max cluster size) - 1

All Four Search Spaces



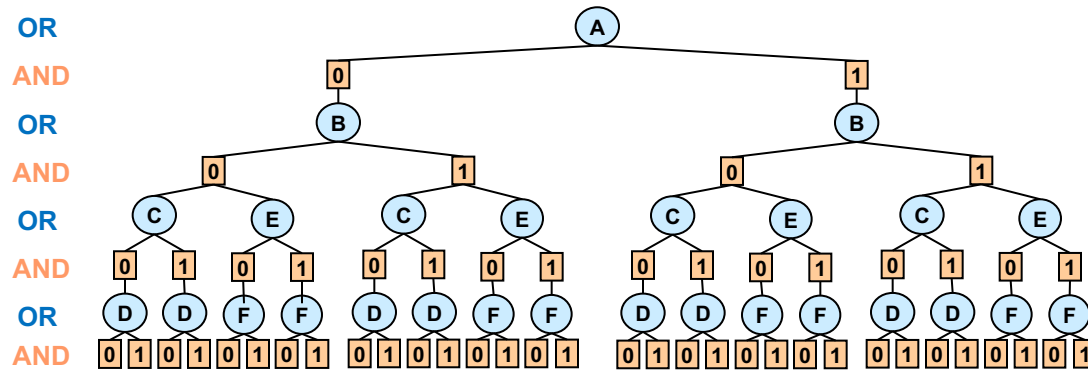
Full OR search tree

126 nodes



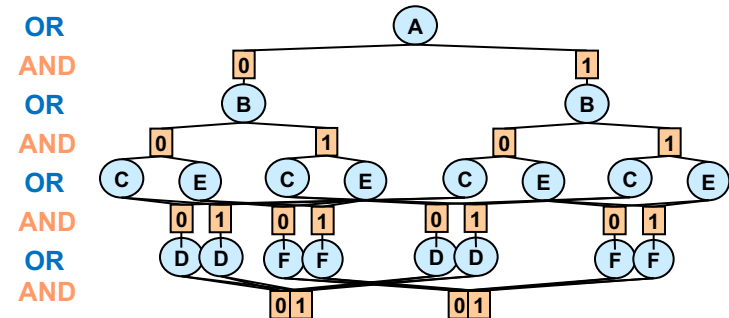
Context minimal OR search graph

28 nodes



Full AND/OR search tree

54 AND nodes



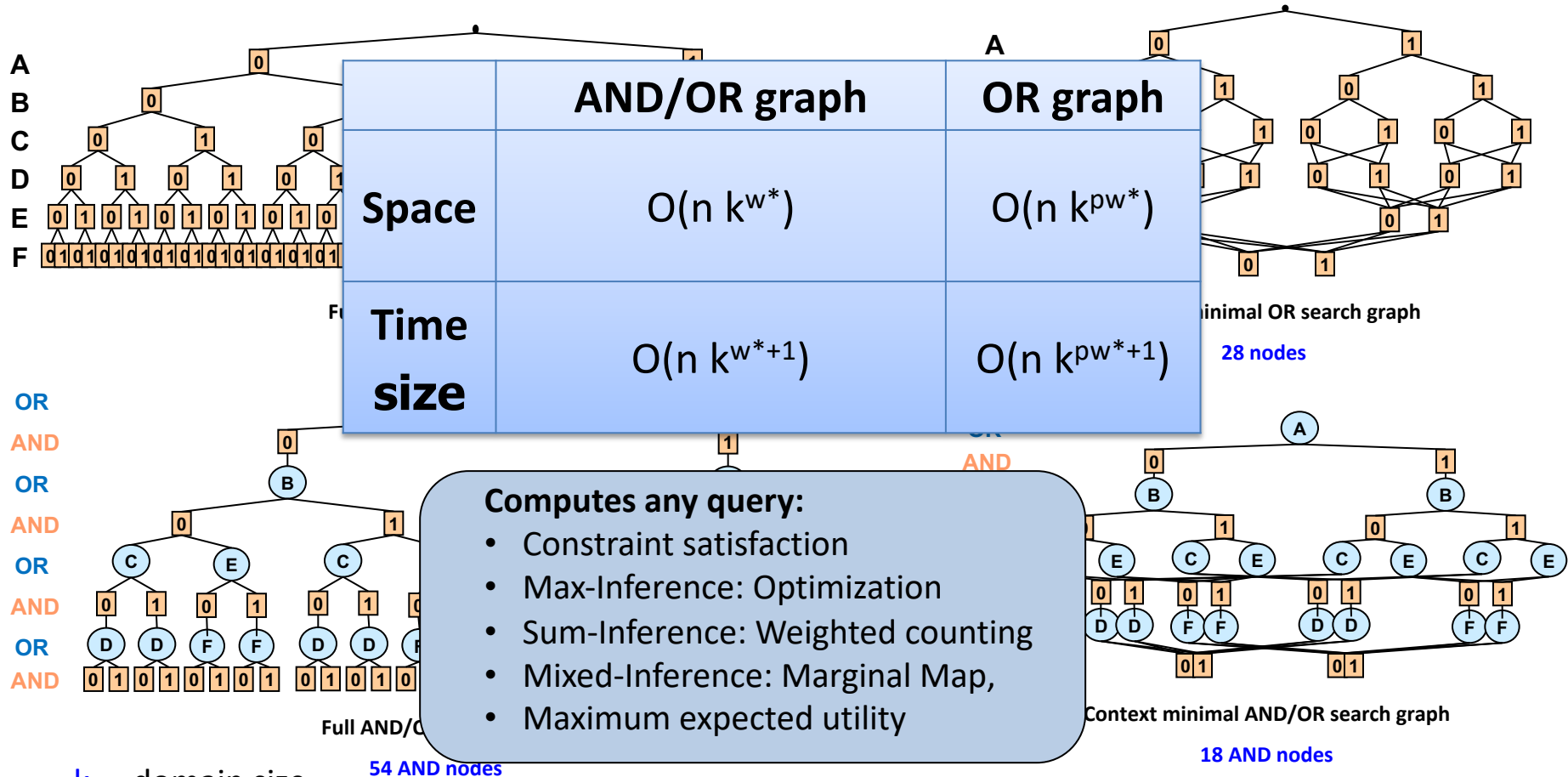
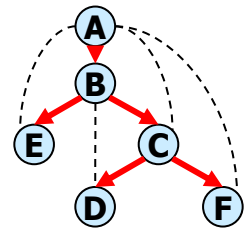
Context minimal AND/OR search graph

18 AND nodes

k = domain size
 n = number of variables
 w^* = treewidth
 pw^* = pathwidth

Any query is best computed
 over the context-minimal AND/OR space

All Four Search Spaces

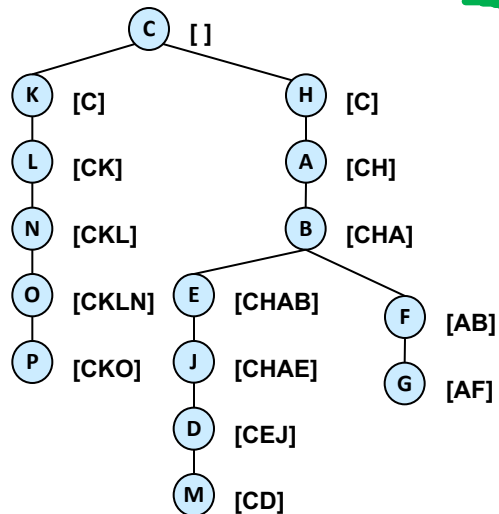
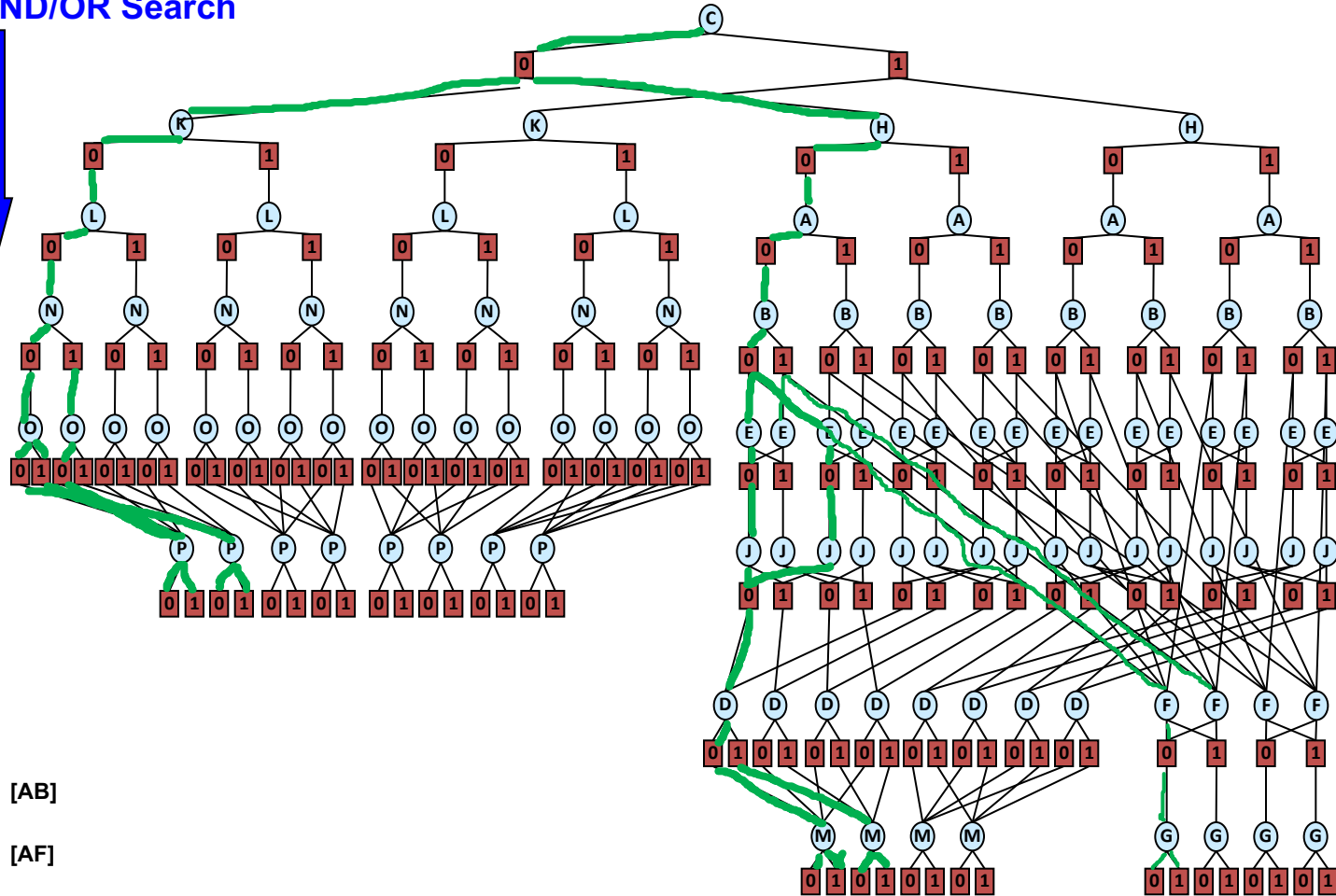
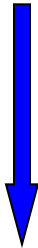
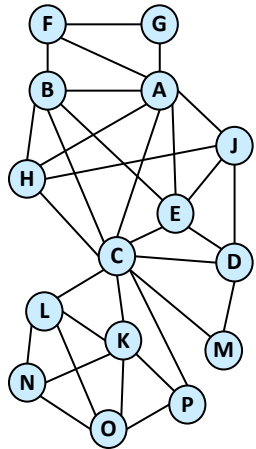


k = domain size
 n = number of variables
 w^* = treewidth
 pw^* = pathwidth

Any query is best computed over the context-minimal AND/OR space

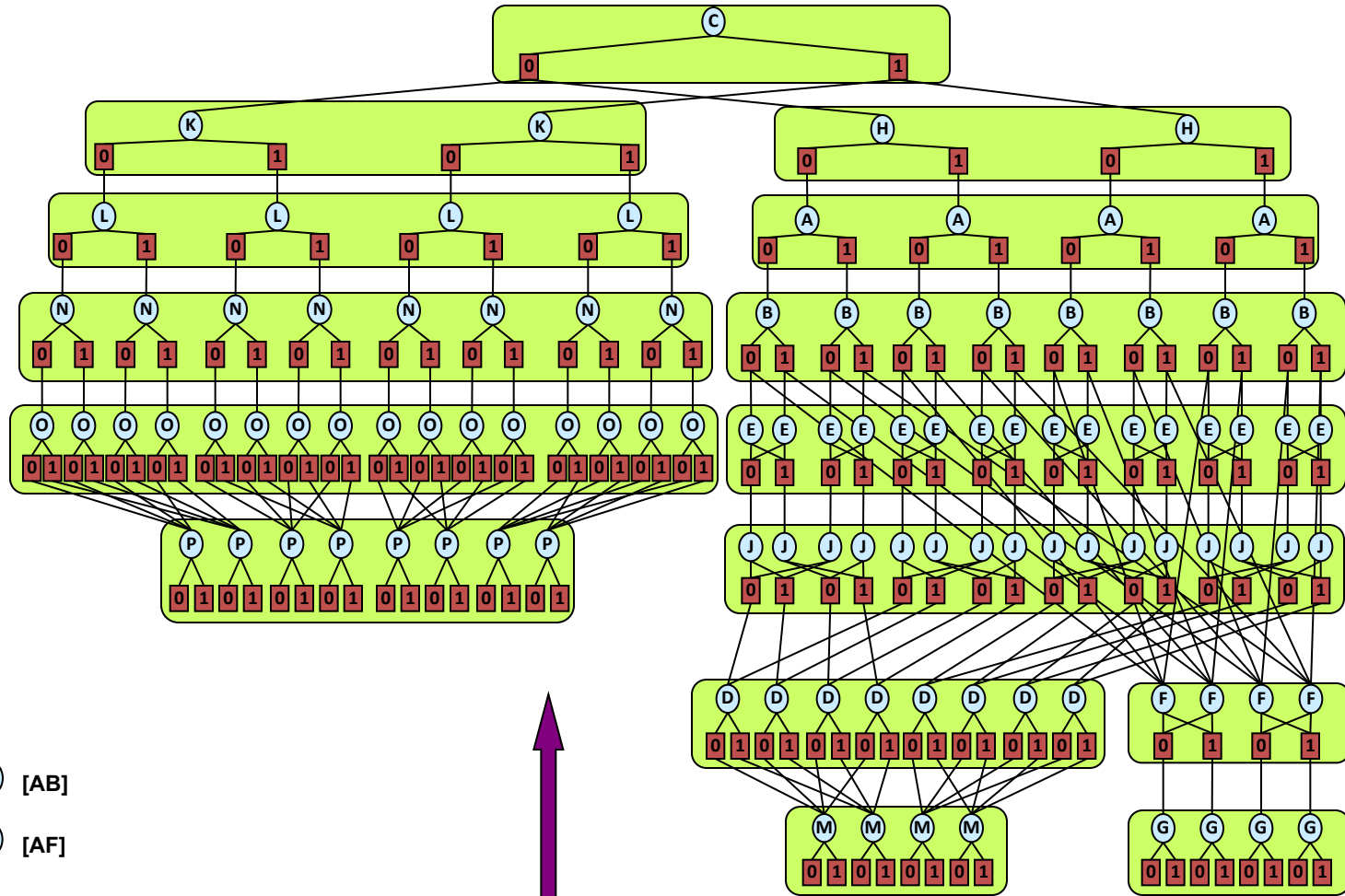
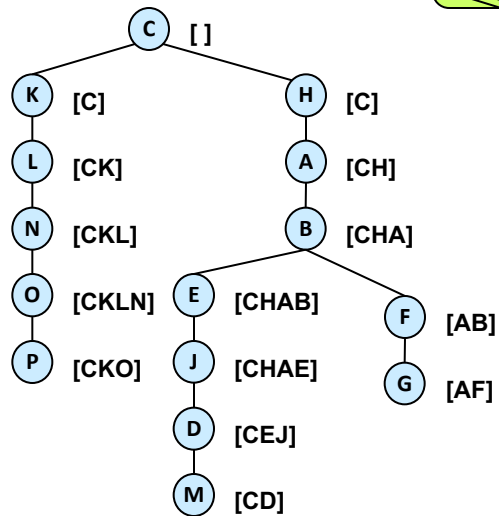
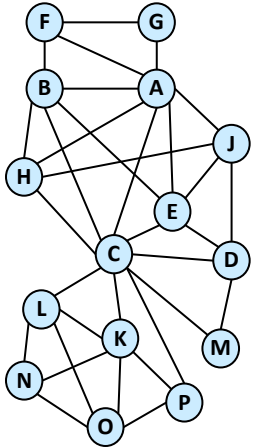
AND/OR Search and Variable Elimination

AND/OR Search



(CKHABEJLNODPMFG)

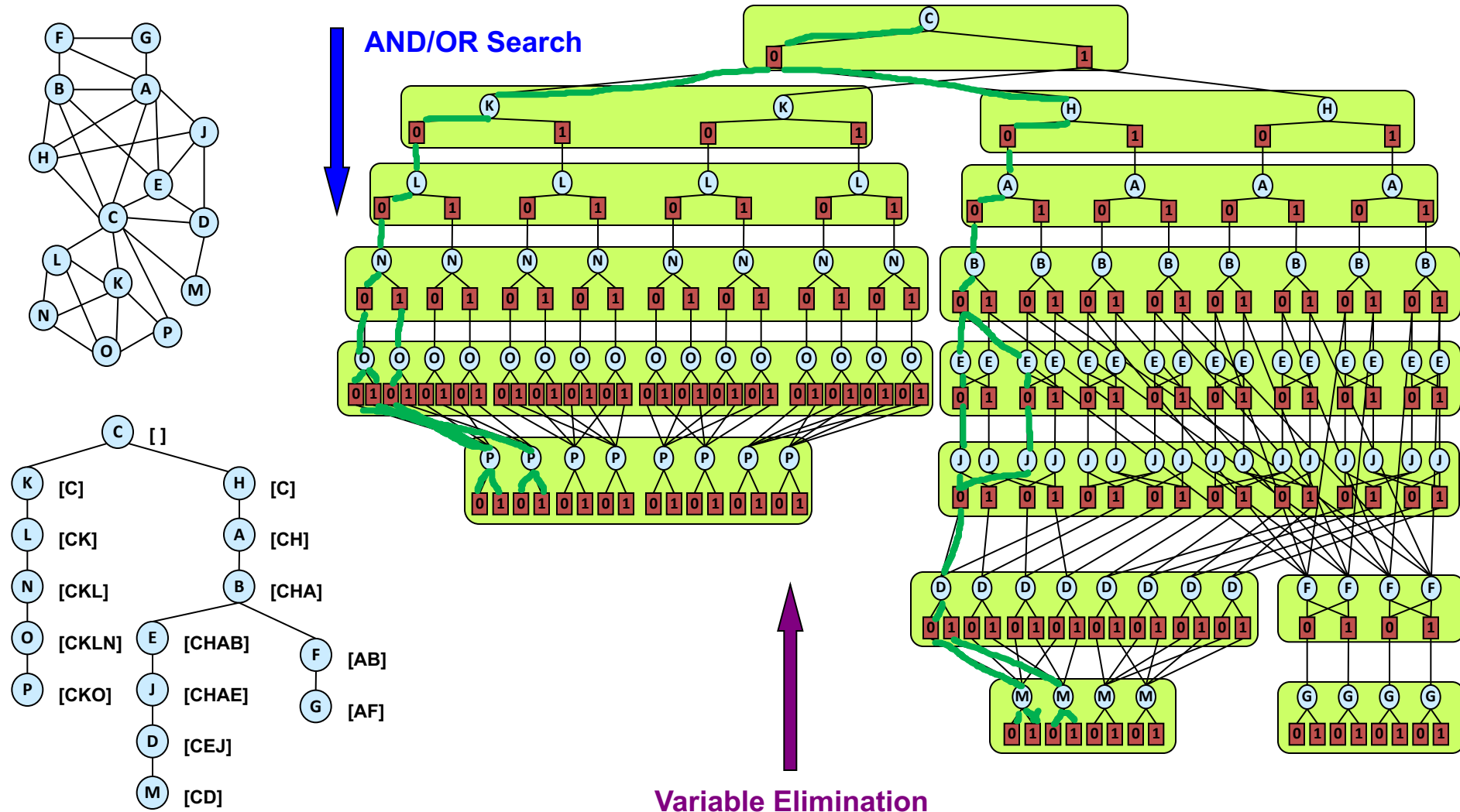
AND/OR Search and Variable Elimination



Variable Elimination

(CKHABEJLNODPMFG)

AND/OR Search and Variable Elimination



(CKHABEJLNODPMFG)

AND Merge vs OR Merge

Example 6.27 As another illustration consider the example given in Figure 6.11a. The AND contexts of each node in the pseudo tree is given in square brackets. The context minimal AND/OR search graph (based on AND merging) is given in Figure 6.11b. Similarly, Figure 6.11d shows the context minimal AND/OR graph based on the OR contexts given in Figure 6.11c.

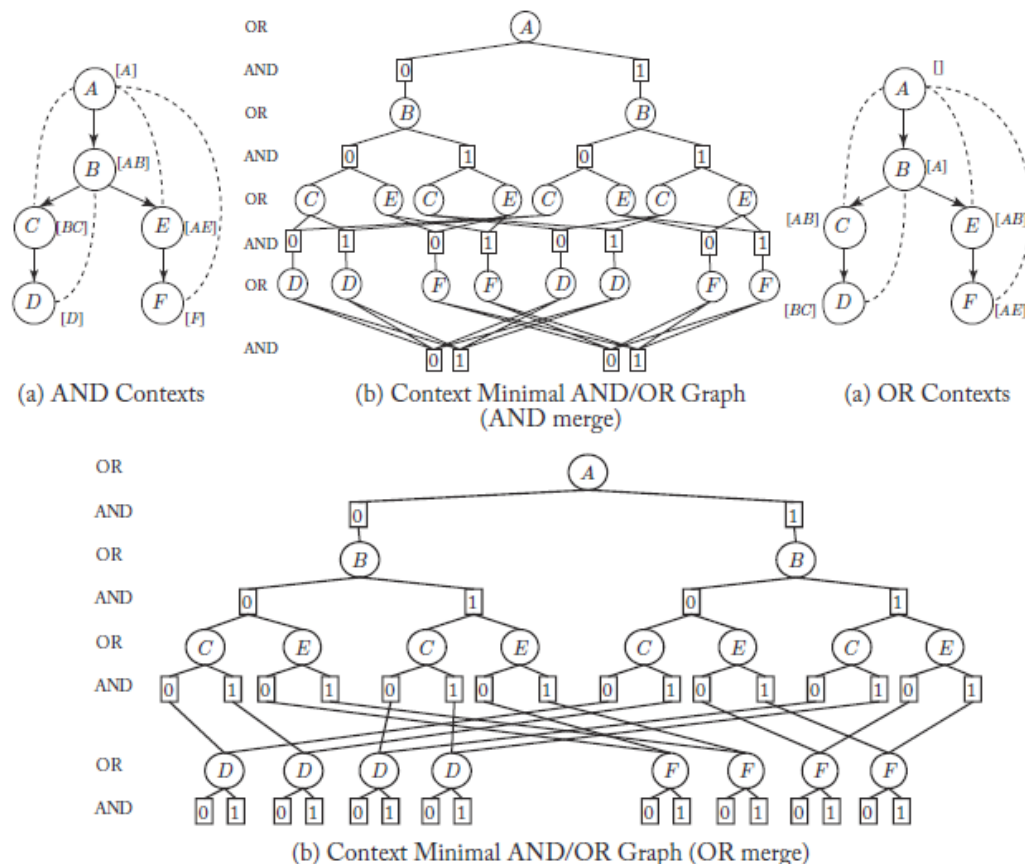
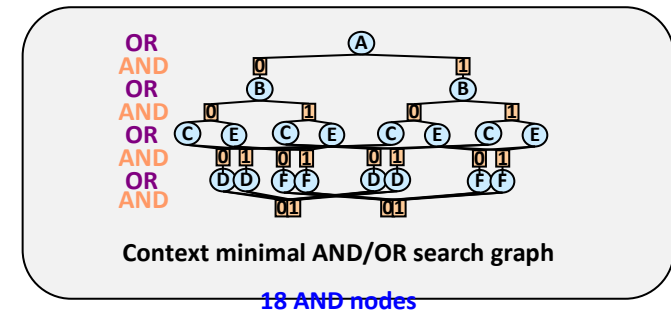
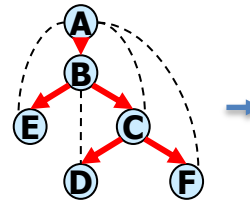


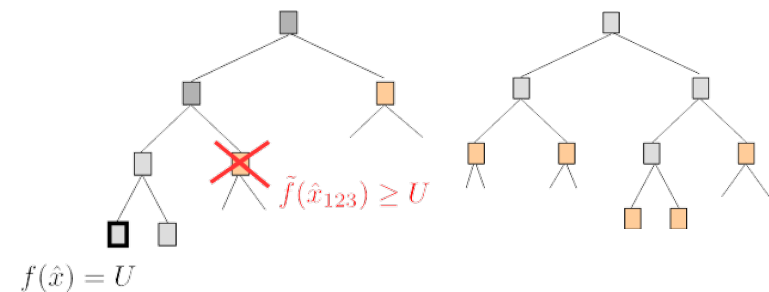
Figure 6.11: AND/OR search graph for graphical models.

Road Map: Search

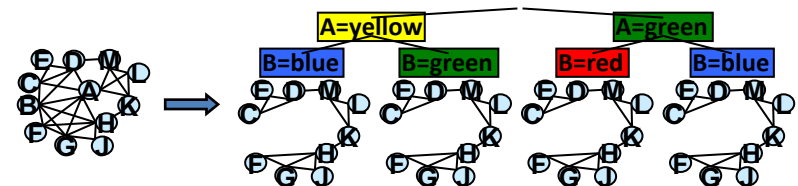
- Review Graphical Modes
- AND/OR search spaces, pseudo-trees
 - AND/OR search trees
 - AND/OR search graphs
 - **Generating good pseudo-trees**
 - Brute-force AND/OR



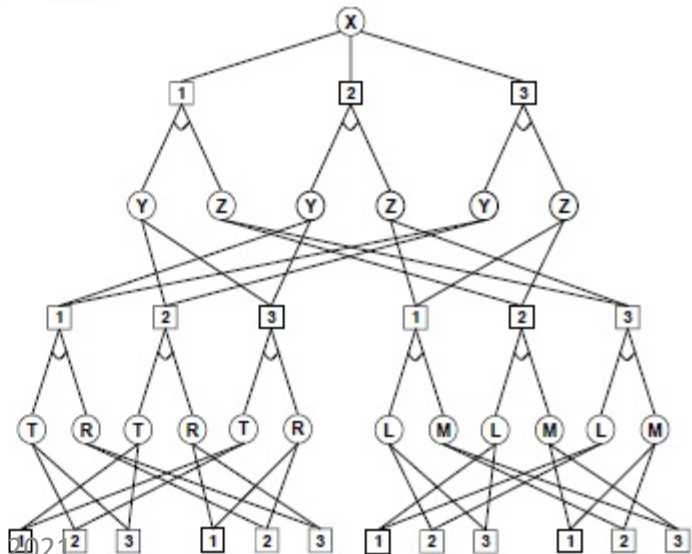
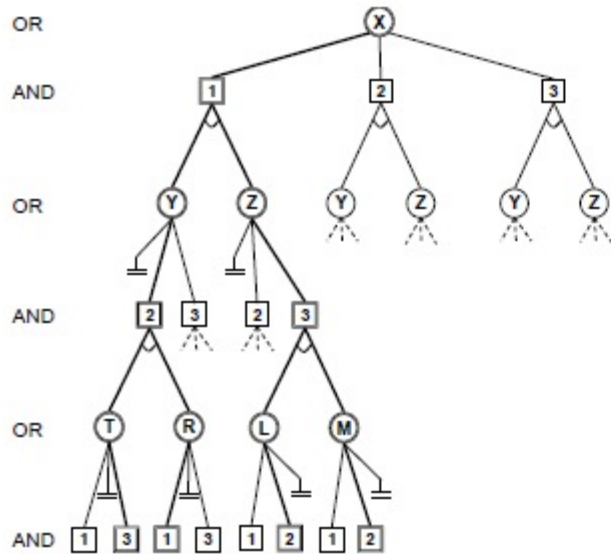
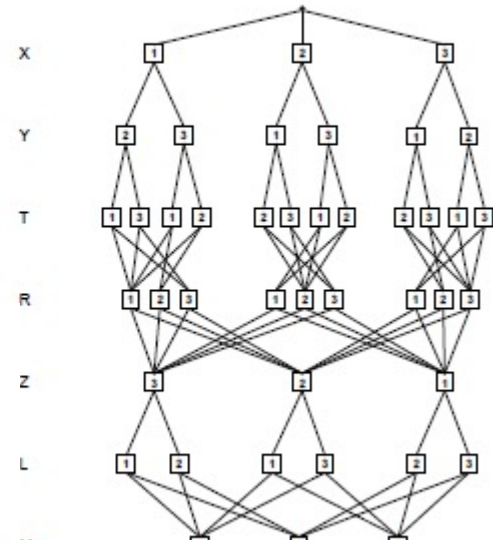
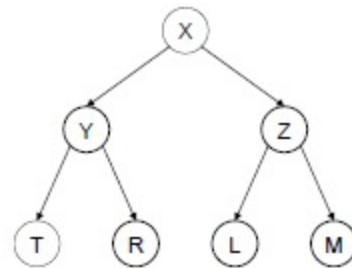
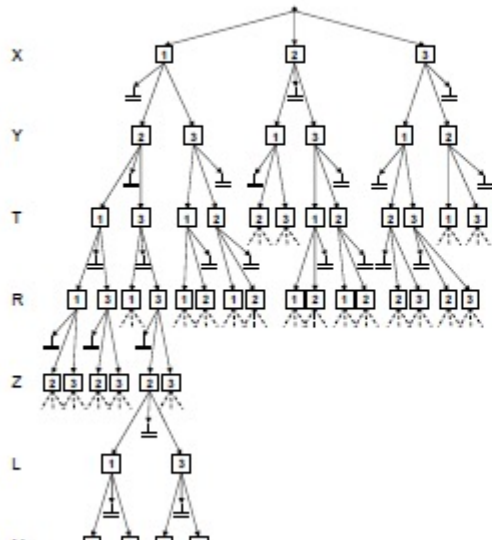
- Heuristic search for AND/OR spaces
 - Basic Heuristic search (Depth and Best)
 - Depth-first AND/OR branch and bound
 - Best-first AND/OR search
 - The Guiding MBE heuristic
 - Marginal Map (max-sum-product)



- Hybrids of search and Inference
- Summary and Class 2



Search Spaces for a Tree GM





Finding Good Pseudo-Trees

Finding Min-Height Pseudo-Trees

- Finding min height pseudo tree is NP-complete, but:
- Given a tree-decomposition whose tree-width is w^* , there exists a pseudo-tree T of G whose depth, satisfies $h \leq w^* \log n$,

Constructing Pseudo-Trees

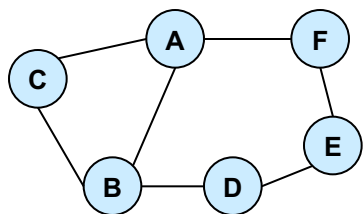
- **Min-Fill** [Kjaerulff, 1990]
 - Depth-first traversal of the induced graph obtained along the **min-fill** elimination order
 - Variables ordered according to the smallest “fill-set”
- **Hypergraph Partitioning** [Karypis and Kumar, 2000]
 - Functions are vertices in the hypergraph and variables are hyperedges
 - Recursive decomposition of the hypergraph while minimizing the separator size at each step
 - Using state-of-the-art software package **hMeTiS**

Variable Orderings and Pseudo-Trees

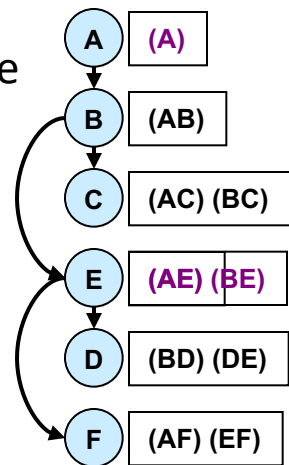
Note: we order from top to

Note: we order from top to bottom here

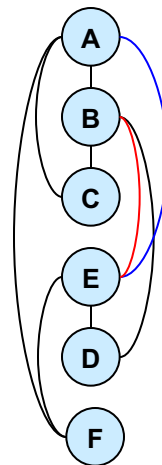
Bucket-tree = pseudo-tree



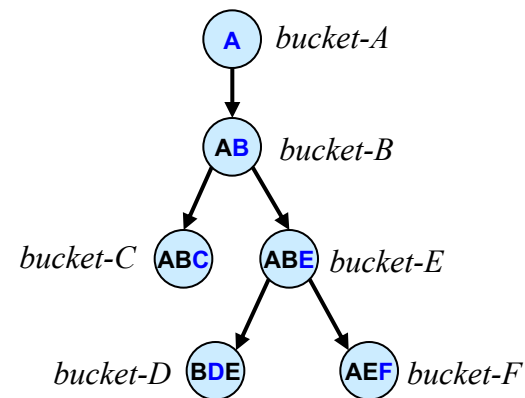
***d*: A B C E D F**



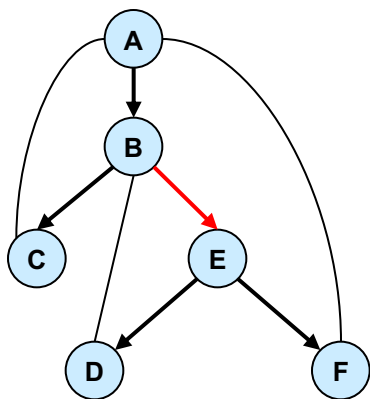
Bucket-tree based on d



Induced graph

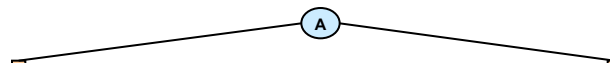


Bucket-tree



Bucket-tree used as pseudo-tree

OR



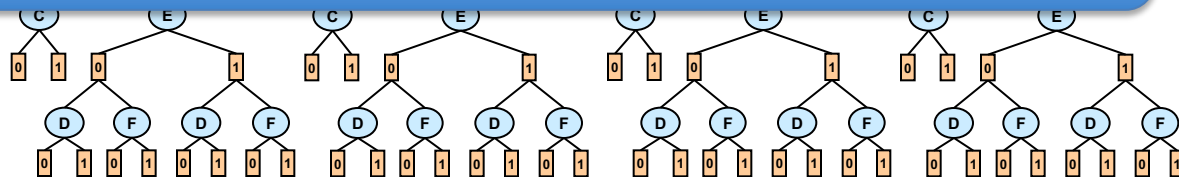
Finding small height or small width pseudo-trees is NP-hard
So, which orderings would give good pseudo-trees?

OR

AND

OR

AND



AND/OR search tree

Quality of Pseudo-Trees

Network	hypergraph		min-fill	
	width	depth	width	depth
barley	7	13	7	23
diabetes	7	16	4	77
link	21	40	15	53
mildew	5	9	4	13
munin1	12	17	12	29
munin2	9	16	9	32
munin3	9	15	9	30
munin4	9	18	9	30
water	11	16	10	15
pigs	11	20	11	26

Bayesian Networks Repository

Network	hypergraph		min-fill	
	width	depth	width	depth
spot5	47	152	39	204
spot28	108	138	79	199
spot29	16	23	14	42
spot42	36	48	33	87
spot54	12	16	11	33
spot404	19	26	19	42
spot408	47	52	35	97
spot503	11	20	9	39
spot505	29	42	23	74
spot507	70	122	59	160

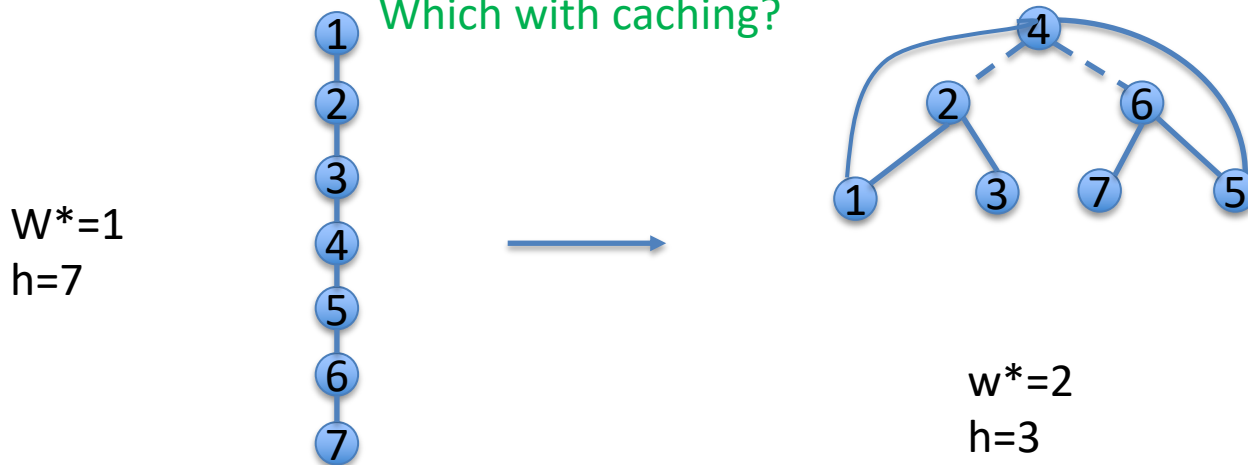
SPOT5 Benchmarks

For more see [Dechter 2013]

Finding Min-height Pseudo-Trees

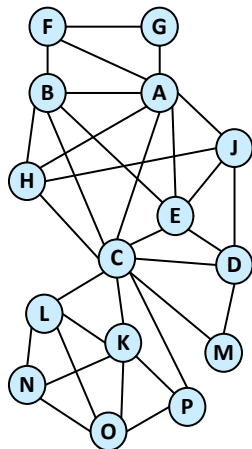
- Finding a min height pseudo-tree is NP-complete, but:
- Given a tree-decomposition with treewidth w^* , there exists a pseudo-tree whose height satisfies
 - $h \leq w^* \log n$
- Optimality of h and w^* cannot be achieved at once.

Which pseudo-tree if no caching?
Which with caching?



Hypergraphs Decomposition

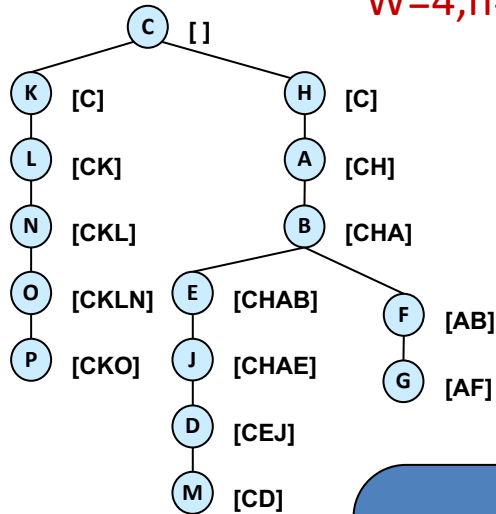
It is well known that the problem of finding the minimal size hypergraph separator is hard. However, heuristic approaches were developed over the years.¹. Generating a pseudo tree $\mathcal{T}$ (yielding also a tree-decomposition) for $\mathcal{M}$ using hypergraph decomposition is fairly straightforward. The vertices of the hypergraph are partitioned into two balanced (roughly equal-sized) parts, denoted by $\mathcal{H}_{left}$ and $\mathcal{H}_{right}$, respectively, while minimizing the number of hyperedges across. A small number of crossing edges translates into a small number of variables shared between the two sets of functions. $\mathcal{H}_{left}$ and $\mathcal{H}_{right}$ are then each recursively partitioned in the same fashion, until they contain a single vertex. The result of this process is a tree of hypergraph separators which can be shown to also be a pseudo tree of the original model where each separator corresponds to a subset of variables connected by a chain.



Let's try a hypergraph decomposition

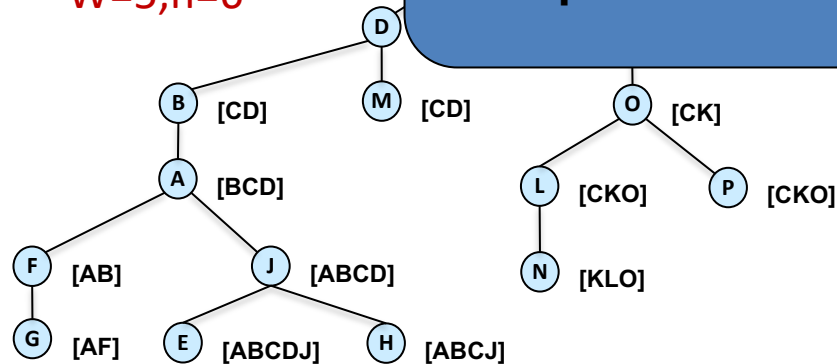
The Impact of the Pseudo-Tree

$W=4, h=8$



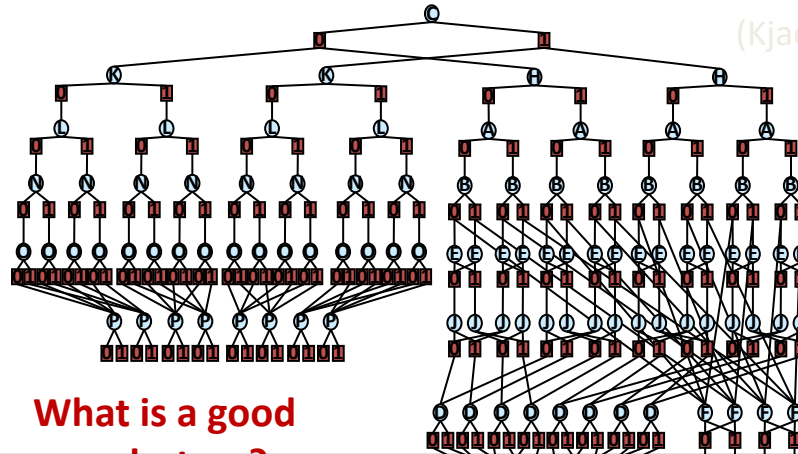
(C K H A B E J L N O)

$W=5, h=6$

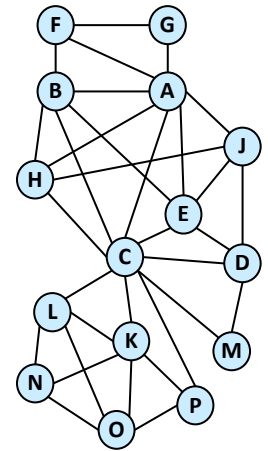


(C D K B A O M L N P J H E F G)

Min-Fill
(Kjaerulff90)

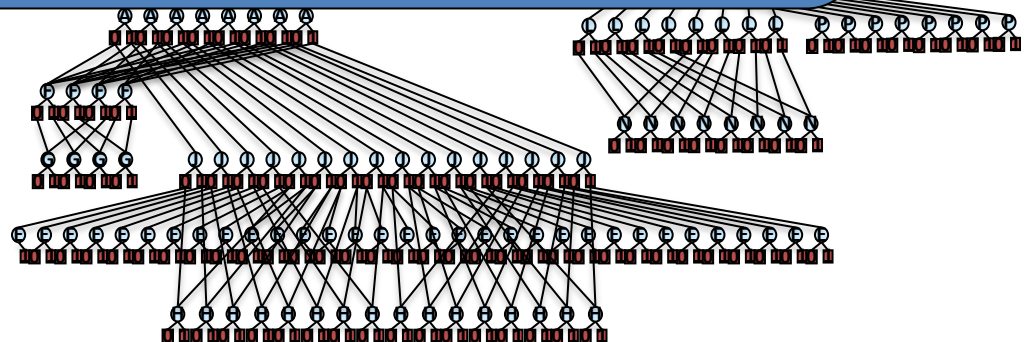


What is a good

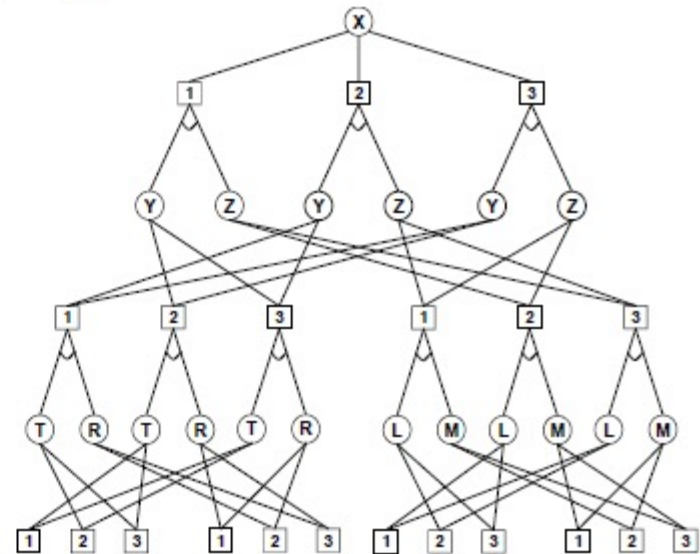
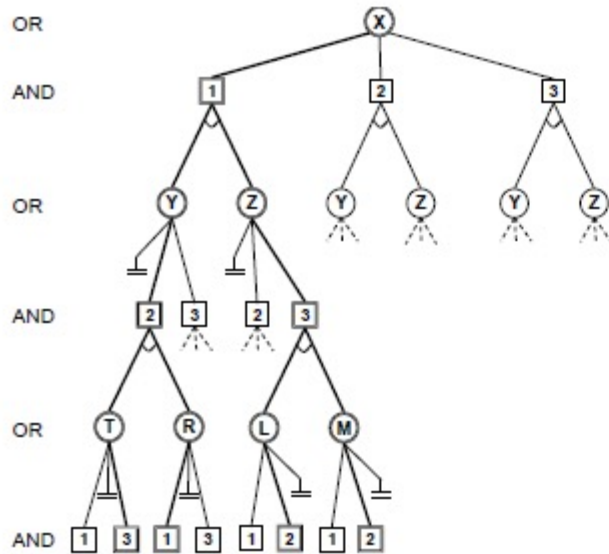
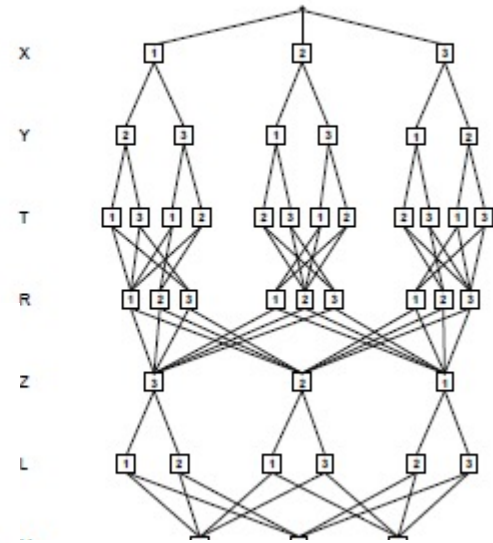
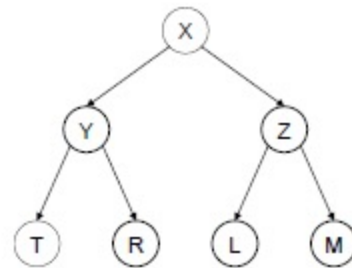
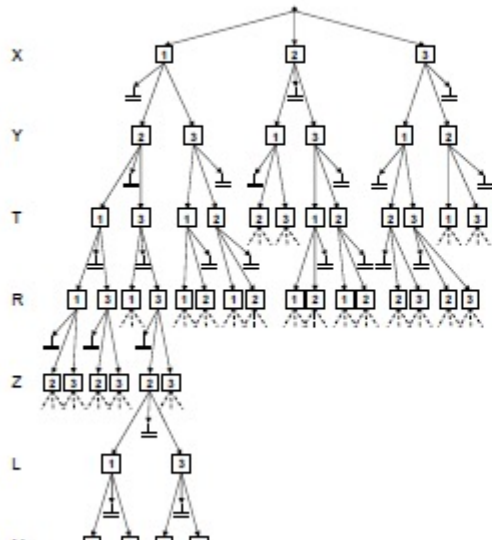


graph
onning
(s)

- Choose pseudo-tree with a minimal search graph
- But determinism and pruning for optimization is unpredictable

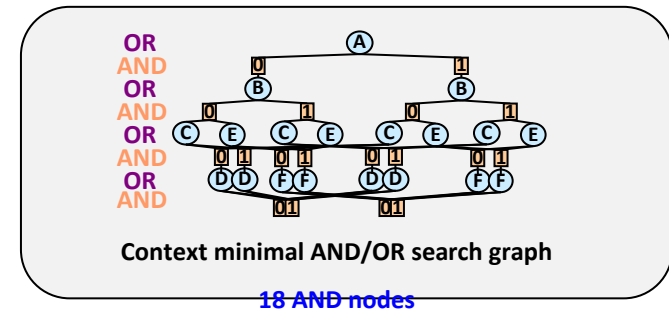


Search Spaces for a Tree GM



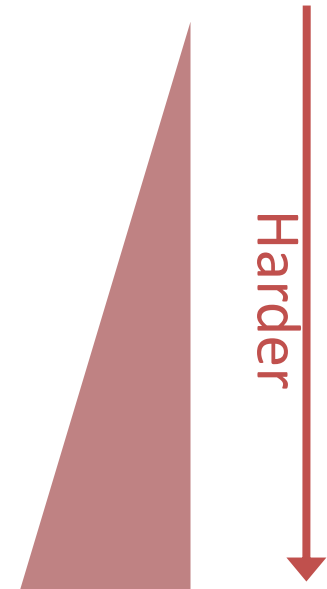
Road Map: Search

- Review Graphical Modes
- AND/OR search spaces, pseudo-trees
 - AND/OR search trees
 - AND/OR search graphs
 - Generating good pseudo-trees
 - **Brute-force AND/OR**
- Heuristic search (HS) for AND/OR spaces
 - Basic Heuristic search (Depth and Best)
 - AND/OR Depth-first HS (branch and bound)
 - AND/OR Best-first heuristic search
 - The Guiding MBE heuristic
 - Marginal Map (max-sum-product)
- Hybrids of search and Inference
- Summary and Class 2



Types of queries

▶ Max-Inference	$f(\mathbf{x}^*) = \max_{\mathbf{x}} \prod_{\alpha} f_{\alpha}(\mathbf{x}_{\alpha})$
▶ Sum-Inference	$Z = \sum_{\mathbf{x}} \prod_{\alpha} f_{\alpha}(\mathbf{x}_{\alpha})$
▶ Mixed-Inference	$f(\mathbf{x}_M^*) = \max_{\mathbf{x}_M} \sum_{\mathbf{x}_S} \prod_{\alpha} f_{\alpha}(\mathbf{x}_{\alpha})$



- All solved by AND/OR Depth-first search,
 - Linear memory, $\exp(h)$ time or
 - $\exp(w^*)$ memory and time
- But, we can do better by:
 - Pruning while searching
 - Generating upper and lower bounds anytime

AND/OR Tree DFS Algorithm (Belief Updating)

$P(E | A, B)$

A	B	E=0	E=1
0	0	.4	.6
0	1	.5	.5
1	0	.7	.3
1	1	.2	.8

Evidence: E=0

$P(B | A)$

A	B=0	B=1
0	.4	.6
1	.1	.9

$P(C | A)$

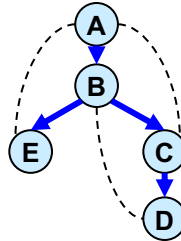
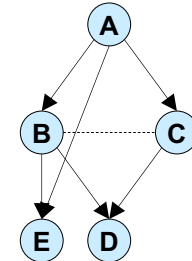
A	C=0	C=1
0	.2	.8
1	.7	.3

$P(A)$

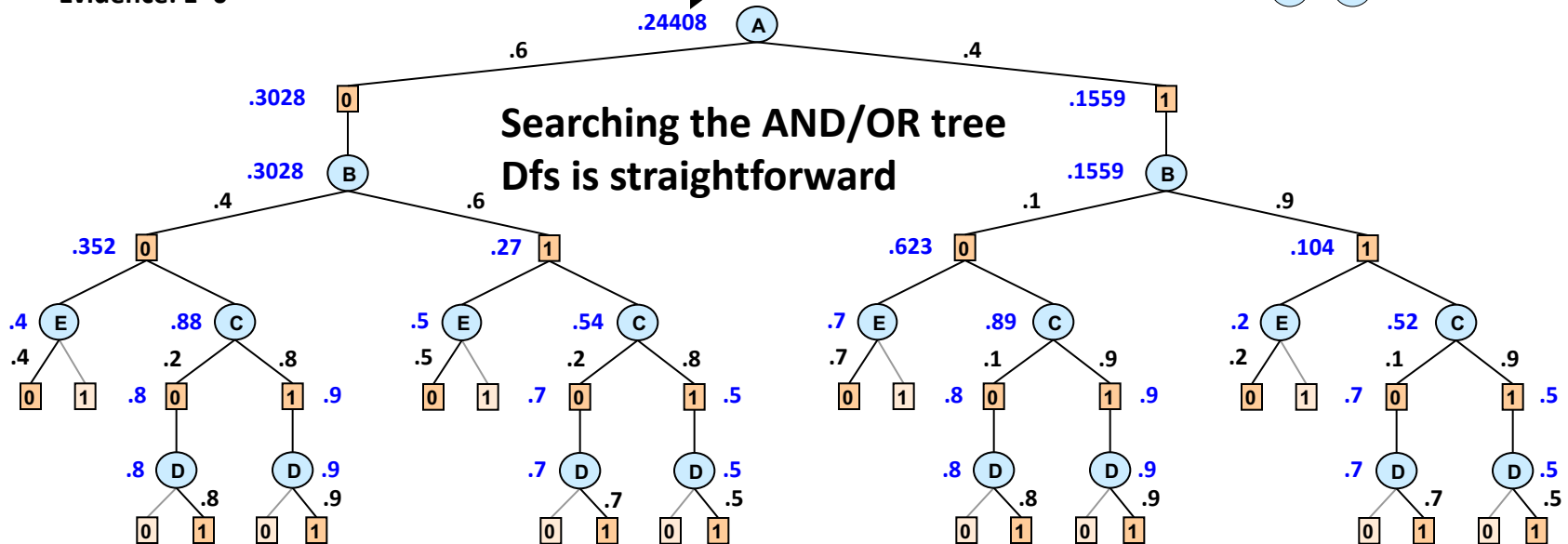
A	P(A)
0	.6
1	.4

Result: $P(D=1, E=0)$

.24408



Searching the AND/OR tree
Dfs is straightforward



$P(D | B, C)$

B	C	D=0	D=1
0	0	.2	.8
0	1	.1	.9
1	0	.3	.7
1	1	.5	.5

Evidence: D=1

OR node: Marginalization operator (summation)

AND node: Combination operator (product)

Value of node = updated belief for sub-problem below

AND/OR Graph DFS Algorithm (Belief Updating)

$P(E | A, B)$

A	B	E=0	E=1
0	0	.4	.6
0	1	.5	.5
1	0	.7	.3
1	1	.2	.8

Evidence: E=0

$P(B | A)$

A	B=0	B=1
0	.4	.6
1	.1	.9

$P(C | A)$

A	C=0	C=1
0	.2	.8
1	.7	.3

$P(A)$

A	P(A)
0	.6
1	.4

Result: $P(D=1, E=0)$

.24408

Searching the AND/OR graph should avoid dead caches, less simple

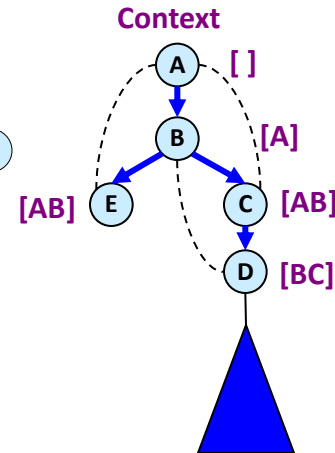
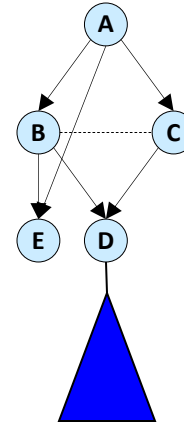
B	C	Value
0	0	.8
0	1	.9
1	0	.7
1	1	.1

Cache table for D

$P(D | B, C)$

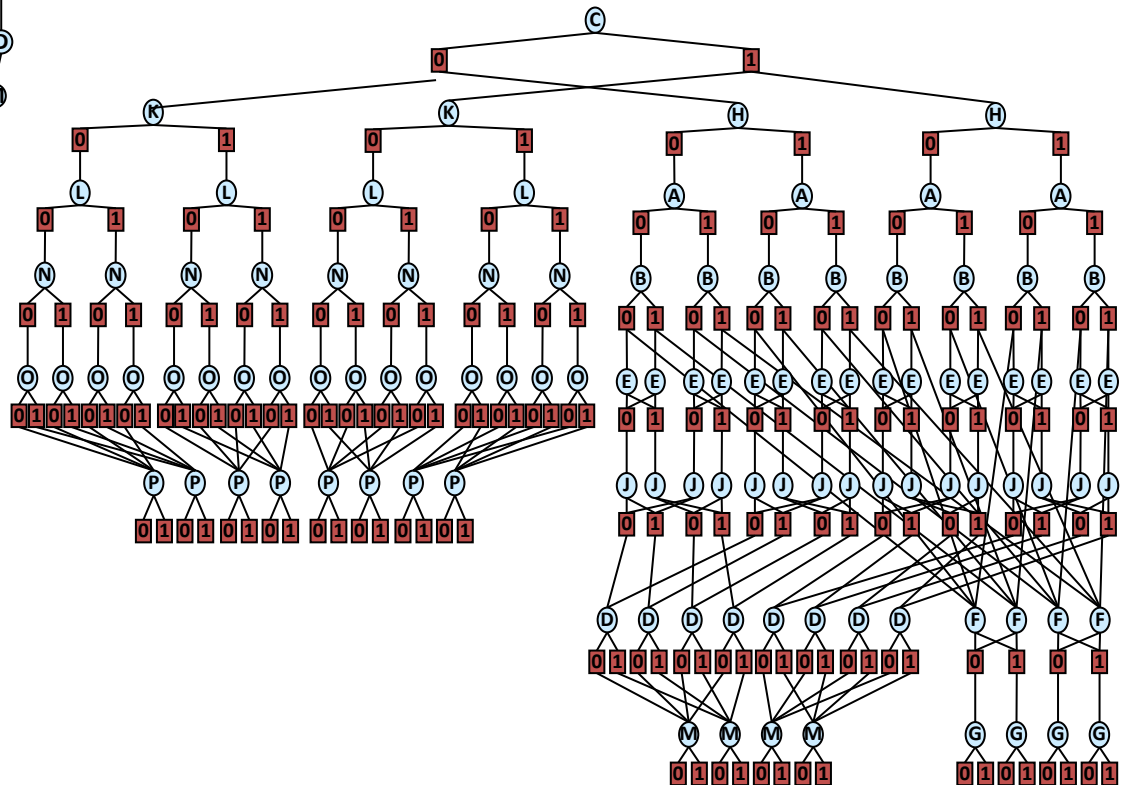
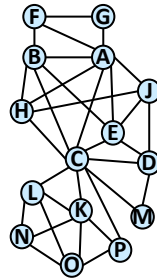
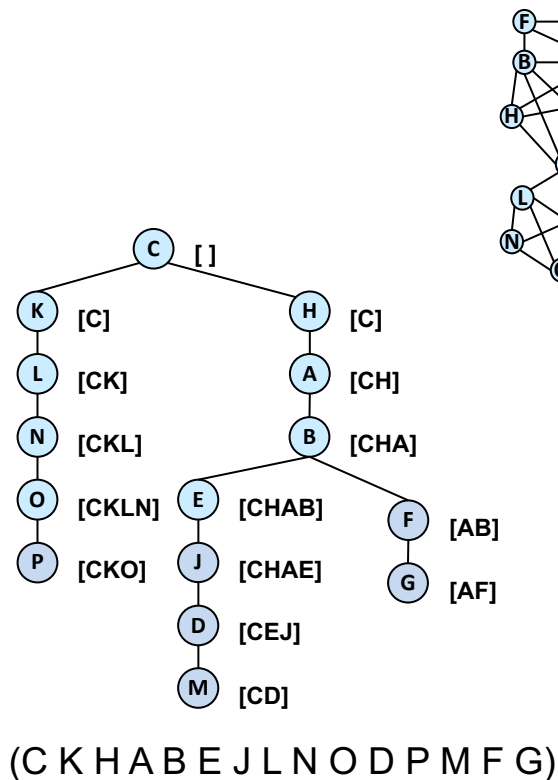
B	C	D=0	D=1
0	0	.2	.8
0	1	.1	.9
1	0	.3	.7
1	1	.5	.5

Evidence: D=1



Dead Caches

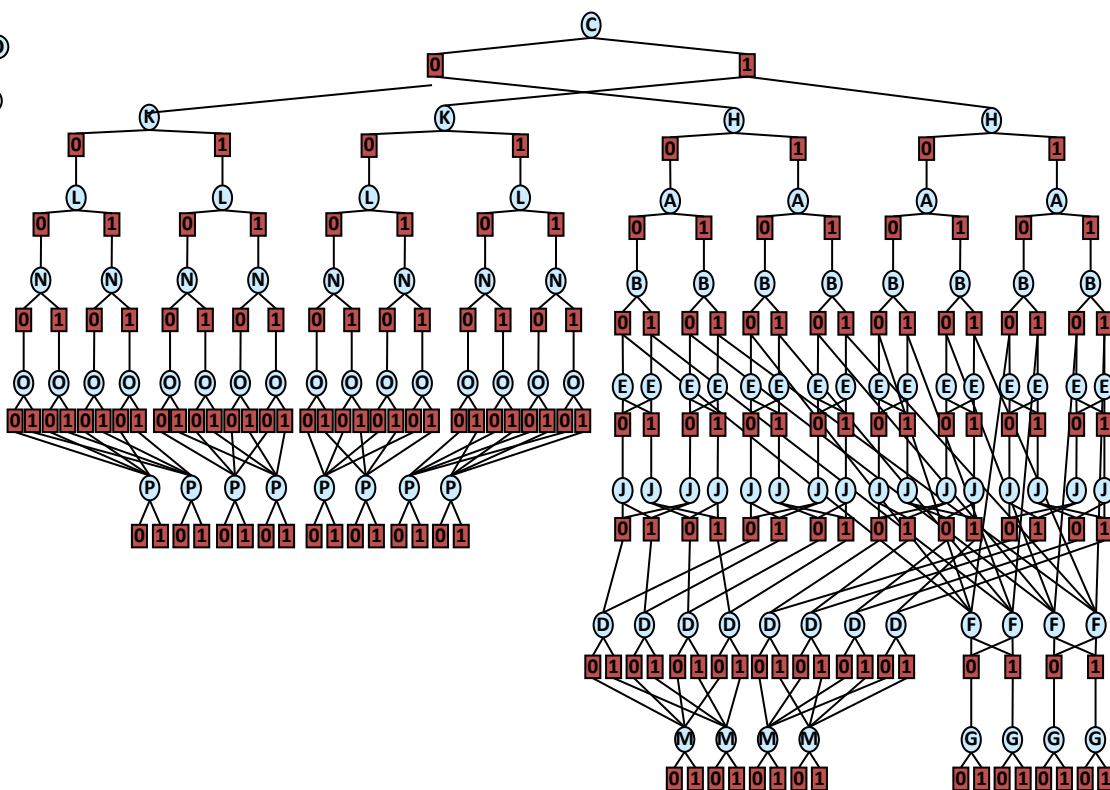
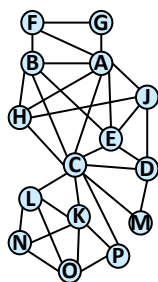
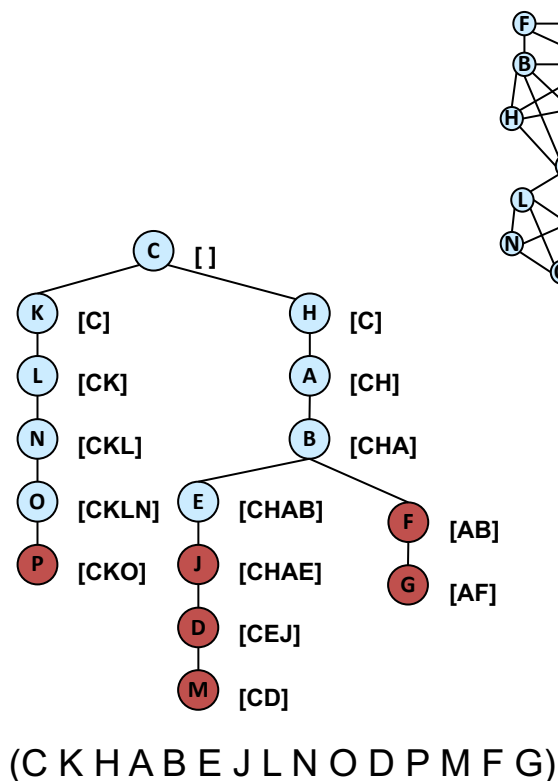
Definition 8.1.9 (dead cache) *If X is the parent of Y in pseudo-tree $\mathcal{T}$, and $\text{context}(X) \subset \text{context}(Y)$, then $\text{context}(Y)$ represents a dead cache.*



(Darwiche 2000)

Dead Caches

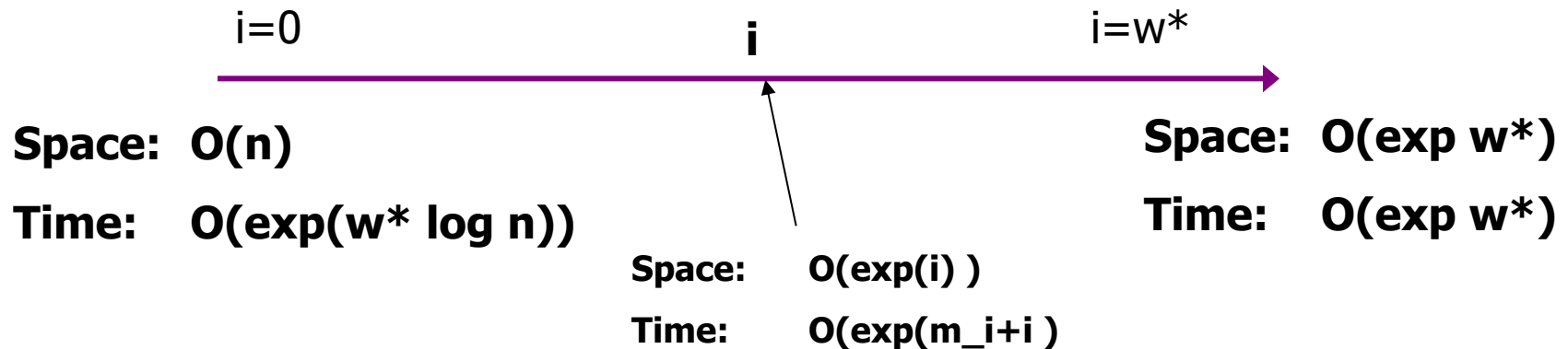
Definition 8.1.9 (dead cache) *If X is the parent of Y in pseudo-tree $\mathcal{T}$, and $\text{context}(X) \subset \text{context}(Y)$, then $\text{context}(Y)$ represents a dead cache.*



(Darwiche 2000)

Searching AND/OR Graphs

- AND/OR(i): searches depth-first, cache i -context
 - i = the max size of a cache table (i.e. number of variables in a context)



m_i is related to the size of the i -cutset.

Different Levels of Caching

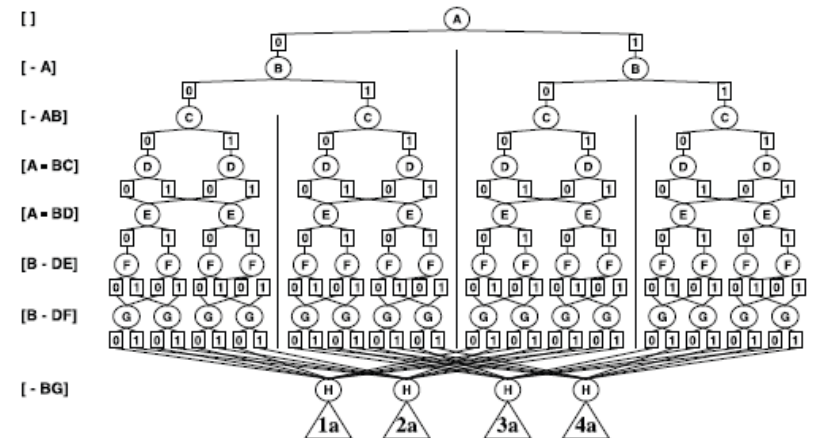
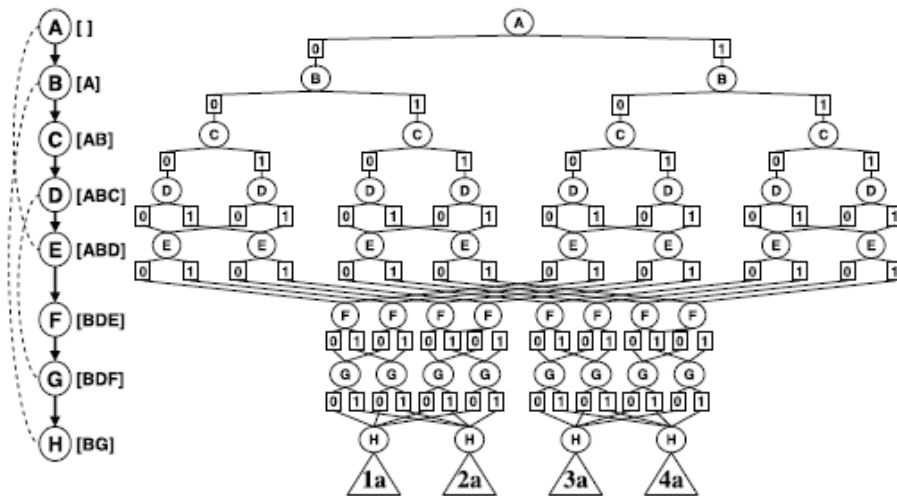


Figure 7.11:
AOC(2) graph (adaptive caching).

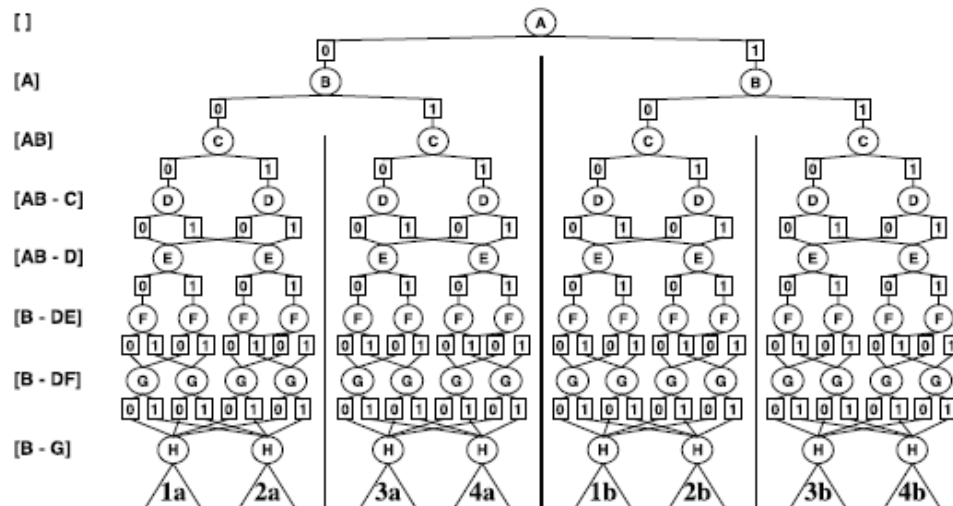


Figure 7.12: AOCutset(2) graph (AND/OR Cutset).

Search for Mixed Deterministic and Probabilistic Graphical Models

AND/OR Search for Mixed Networks

Definition 8.2.1 (backtrack-free AND/OR search tree) *Given graphical model $\mathcal{M}$ and given an AND/OR search tree $S_{\mathcal{T}}(\mathcal{M})$, the backtrack-free AND/OR search tree of $\mathcal{M}$ based on $\mathcal{T}$, denoted $BF_{\mathcal{T}}(\mathcal{M})$, is obtained by pruning from $S_{\mathcal{T}}(\mathcal{M})$ all inconsistent subtrees, namely all nodes that root no consistent partial solution.*

- Graph-based No-good and good learning are automatically performed by AND/OR (backjumping) and by caching.

AND/OR Backtrack-Free

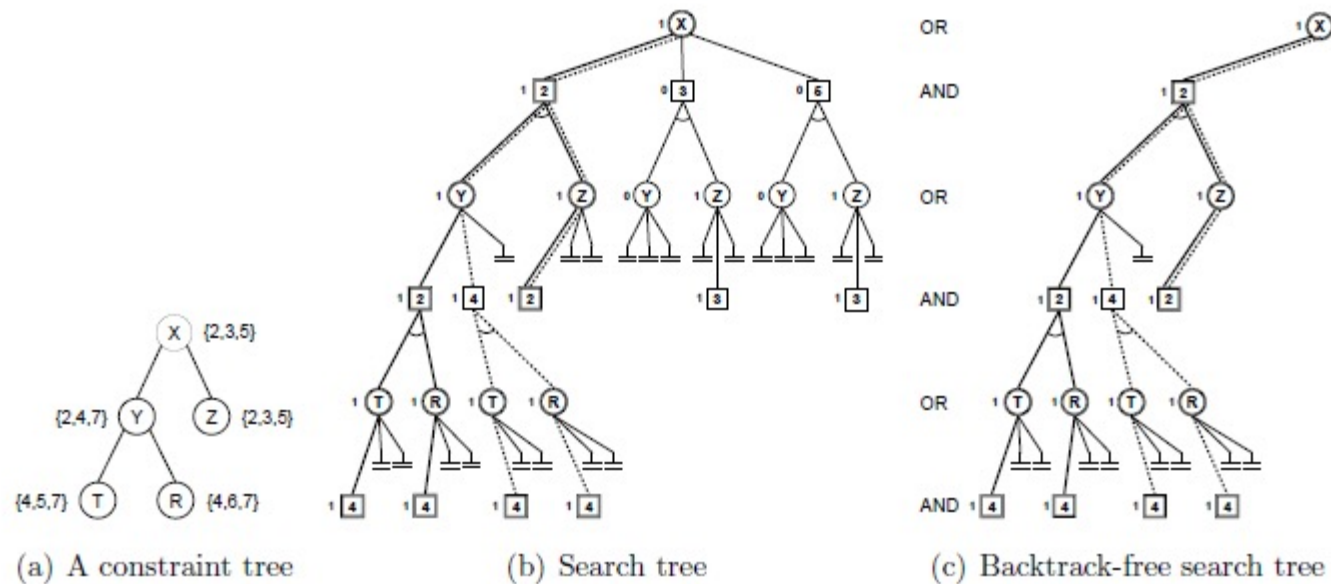


Figure 8.1: AND/OR search tree and backtrack-free tree

AND/OR CPE (Constraint Probability Evaluation)

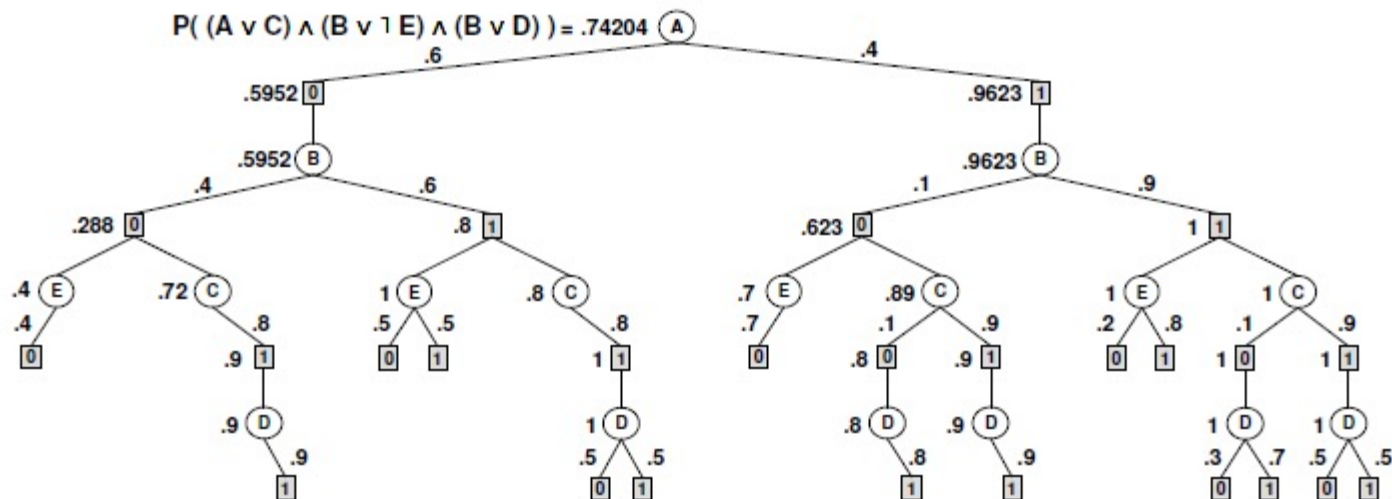


Figure 8.2: Mixed network defined by the query $\varphi = (A \vee C) \wedge (B \vee \neg E) \wedge (B \vee D)$

Example 8.2.6 We refer back to the example in Figure 7.4. Consider a constraint network that is defined by the CNF formula $\varphi = (A \vee C) \wedge (B \vee \neg E) \wedge (B \vee D)$. The trace of algorithm AND-OR-CPE without caching is given in Figure 8.2. Notice that the clause $(A \vee C)$ is not satisfied if $A = 0$ and $C = 0$, therefore the paths that contain this assignment cannot be part of a solution of the mixed network. The value of each node is shown to its left (the leaf nodes assume a dummy value of 1, not shown in the figure). The value of the root node is the probability of φ . Figure 8.2 is similar to Figure 7.4. In Figure 7.4 the evidence can be modeled as the CNF formula with unit clauses $D \wedge \neg E$. $\square$

AND/OR CPE (Constraint Probability Evaluation)

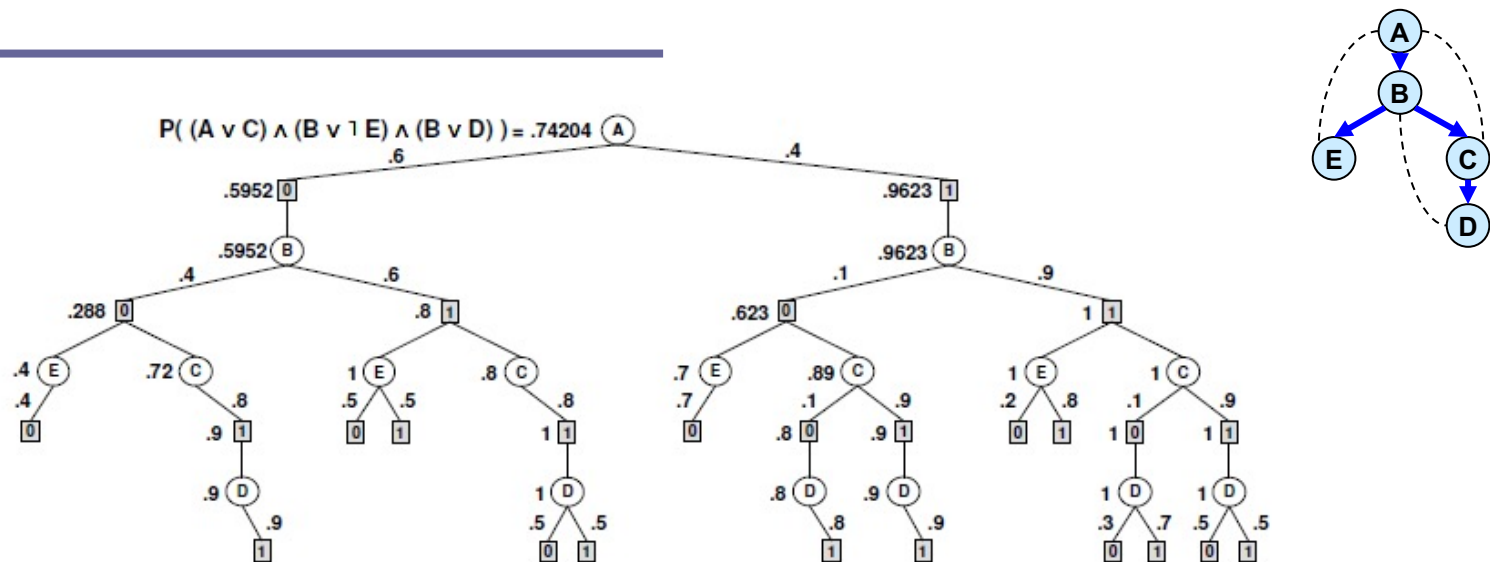
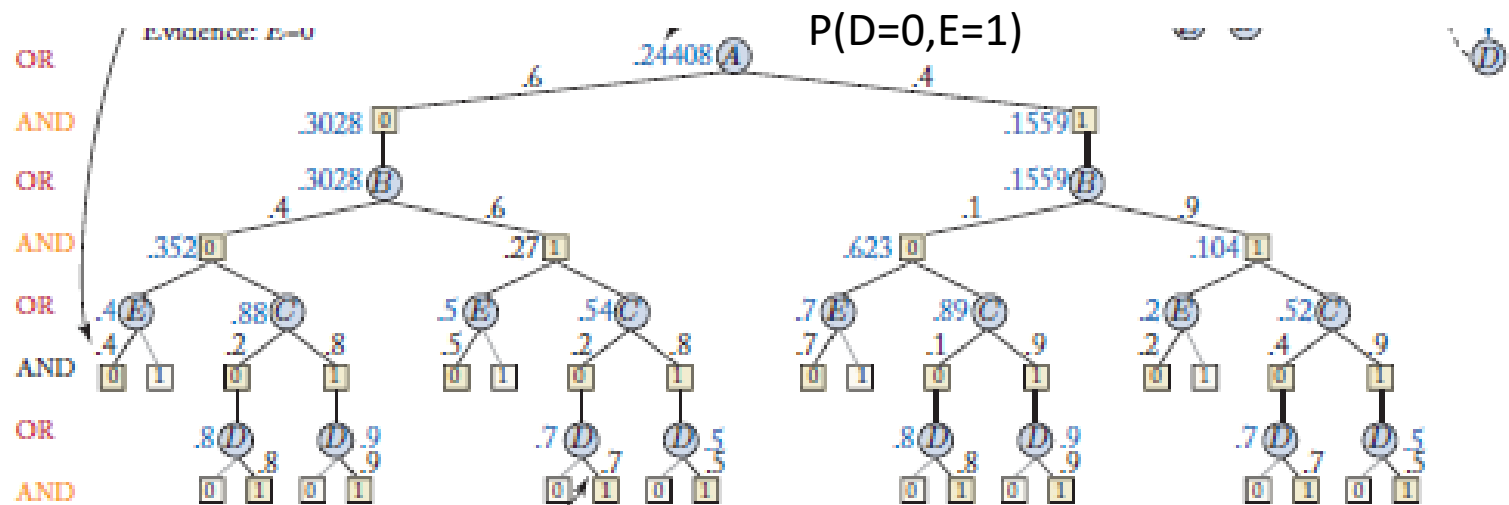
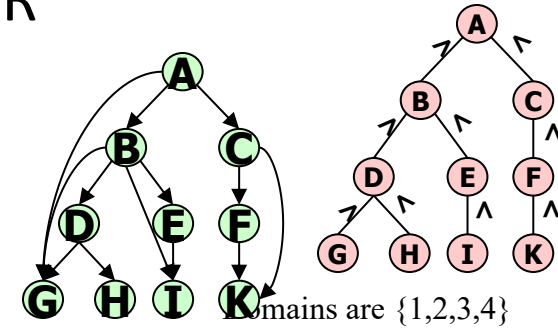
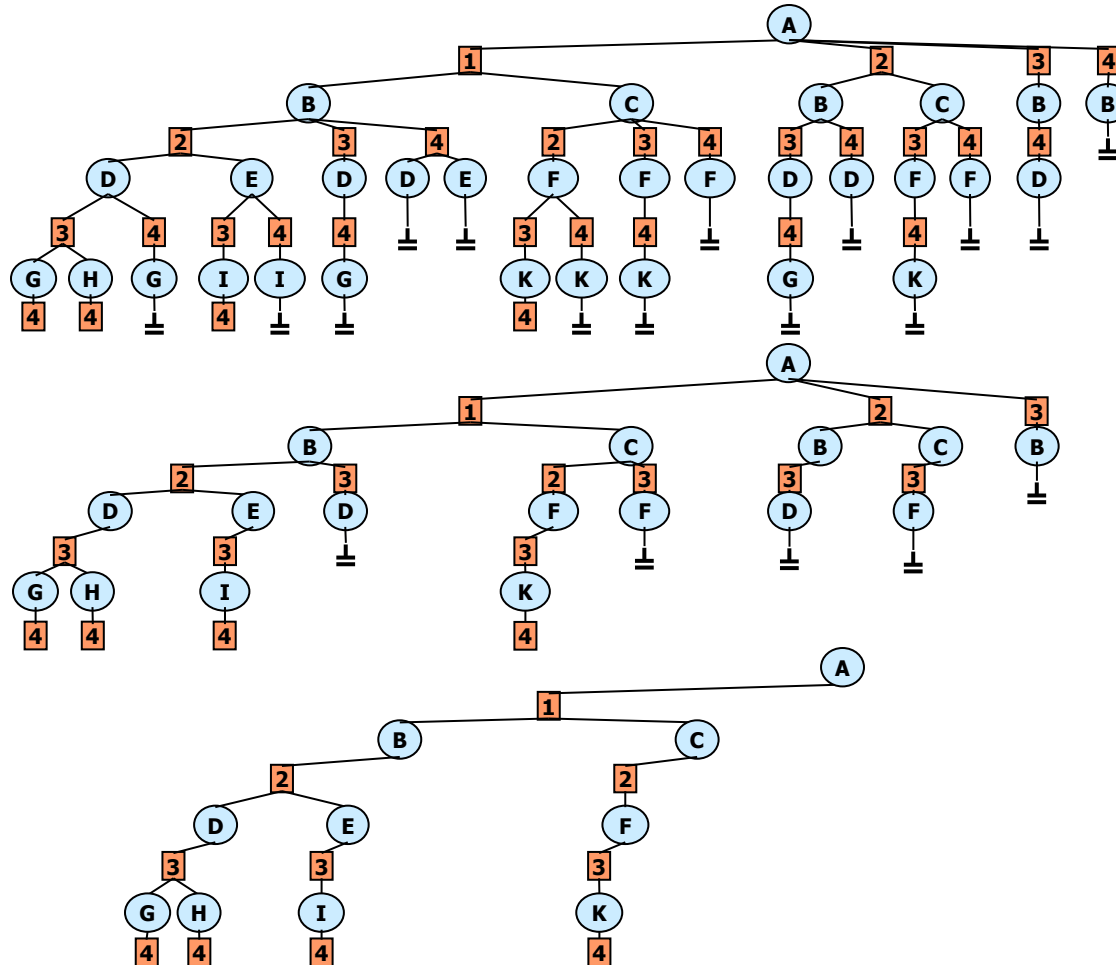


Figure 8.2: Mixed network defined by the query $\varphi = (A \vee C) \wedge (B \vee \neg E) \wedge (B \vee D)$



The Effect of Constraint Propagation in AND/OR



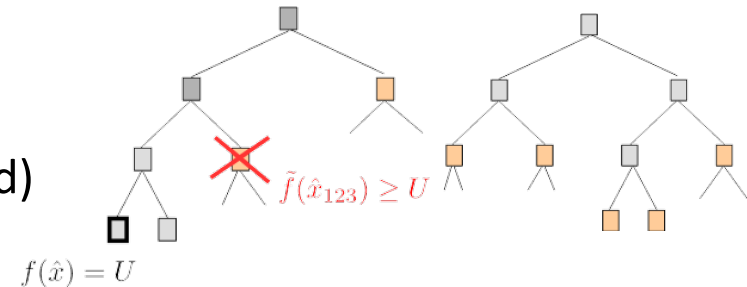
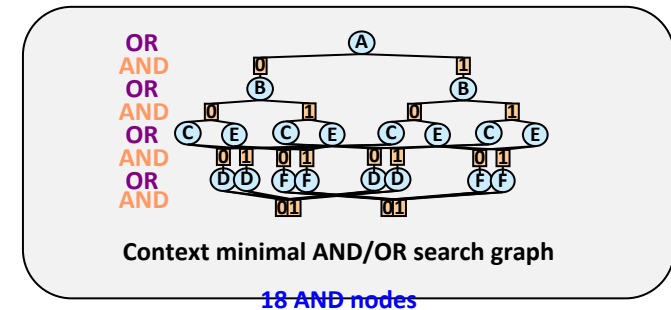
CONSTRAINTS ONLY

FORWARD CHECKING

MAINTAINING ARC
CONSISTENCY

Outline: Search

- Review Graphical Modes
- AND/OR search spaces, pseudo-trees
 - AND/OR search trees
 - AND/OR search graphs
 - **Generating good pseudo-trees**
 - Brute-force search
- Heuristic search (HS) for AND/OR spaces
 - **Basic Heuristic search (Depth and Best)**
 - AND/OR Depth-first HS (branch and bound)
 - AND/OR Best-first heuristic search
 - The Guiding MBE heuristic
 - Marginal Map (max-sum-product)
- Hybrids of search and Inference
- Summary and Class 2



Basic Heuristic Search

We assume min-sum problems in the following

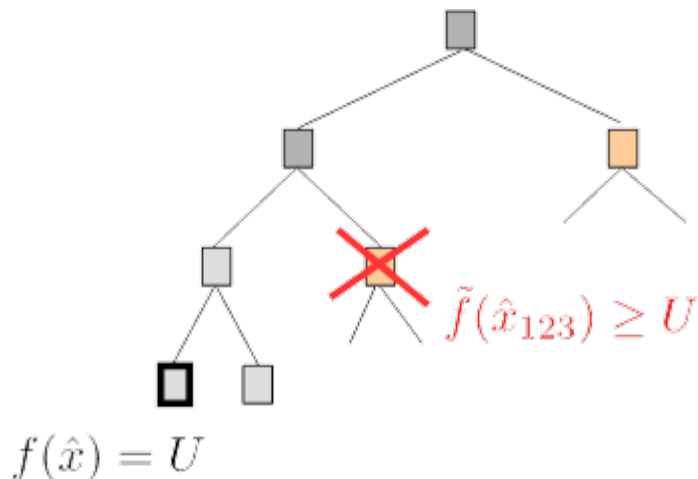
Heuristic function $\tilde{f}(\hat{x}_p)$ computes a lower bound on the best extension of partial configuration $\hat{x}_p$ and can be used to guide heuristic search.

We focus on:

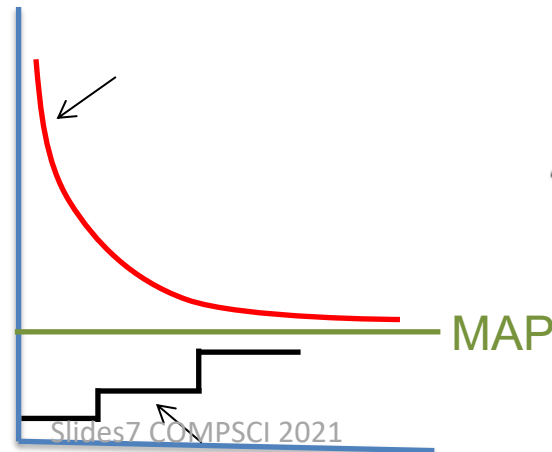
1. Branch-and-Bound

Use heuristic function $\tilde{f}(\hat{x}_p)$ to prune the depth-first search tree

Linear space



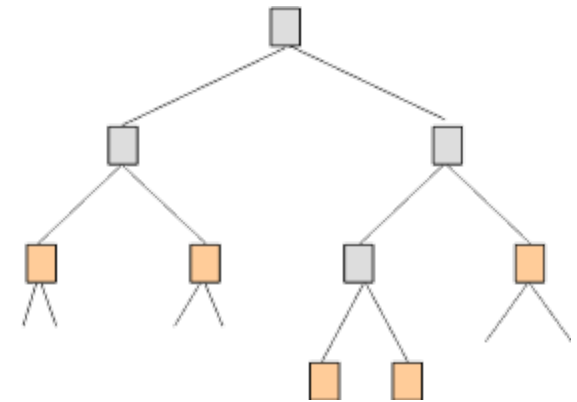
BnB is upper-bound anytime



2. Best-First Search

Always expand the node with the lowest heuristic value $\tilde{f}(\hat{x}_p)$

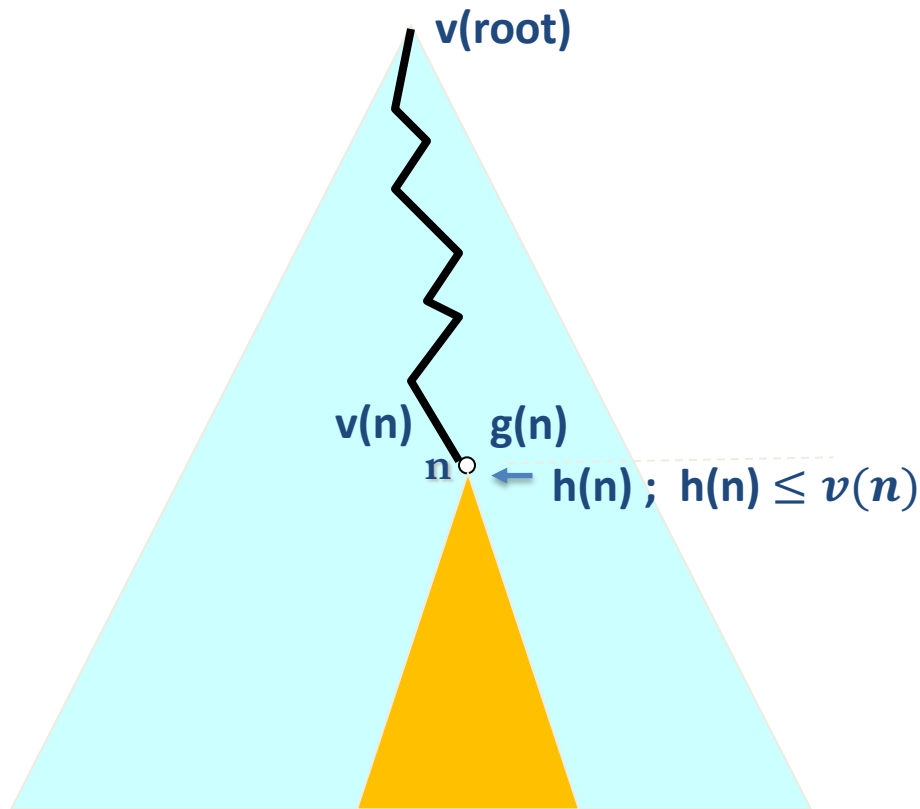
Needs lots of memory



Basic Heuristic Search; Best-First

Task: compute $v(\text{root})$: MPE, MAP(MMAP)

Each node is a sub-problem
(defined by current conditioning)

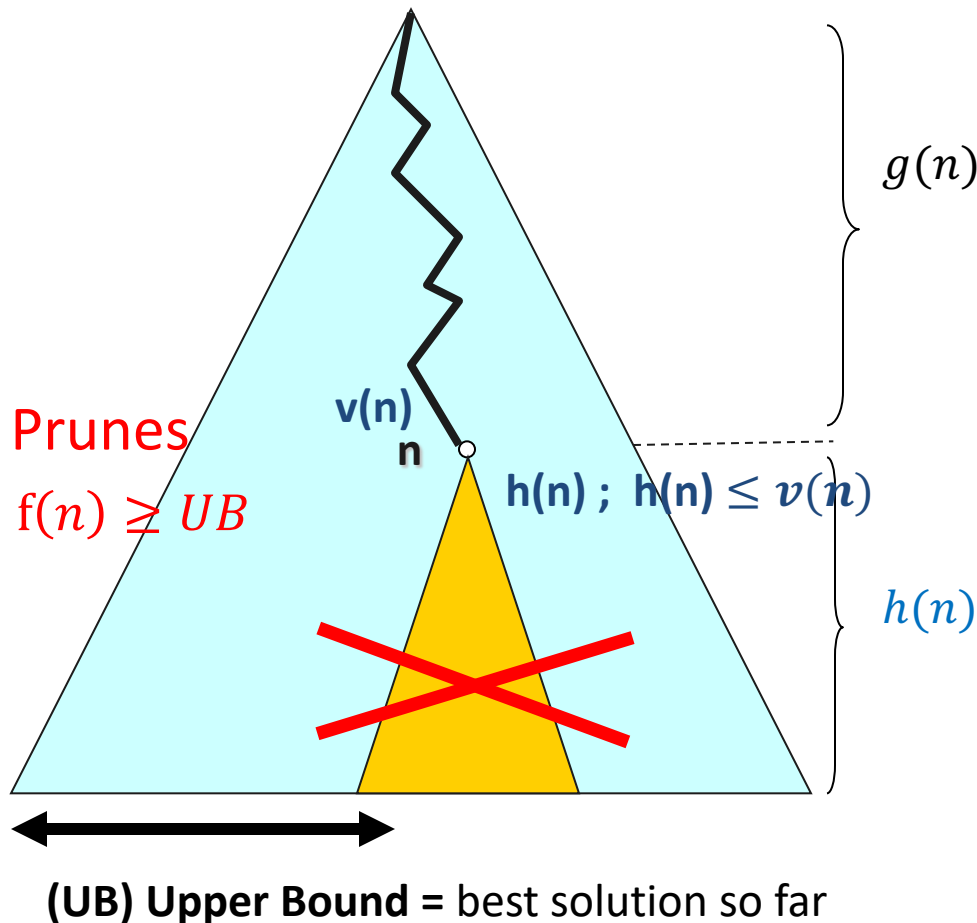


$$f(n) = g(n) + h(n) \leq g(n) + v(n) = f^*(n)$$

$f(n)$ is a lower bound on best cost through n

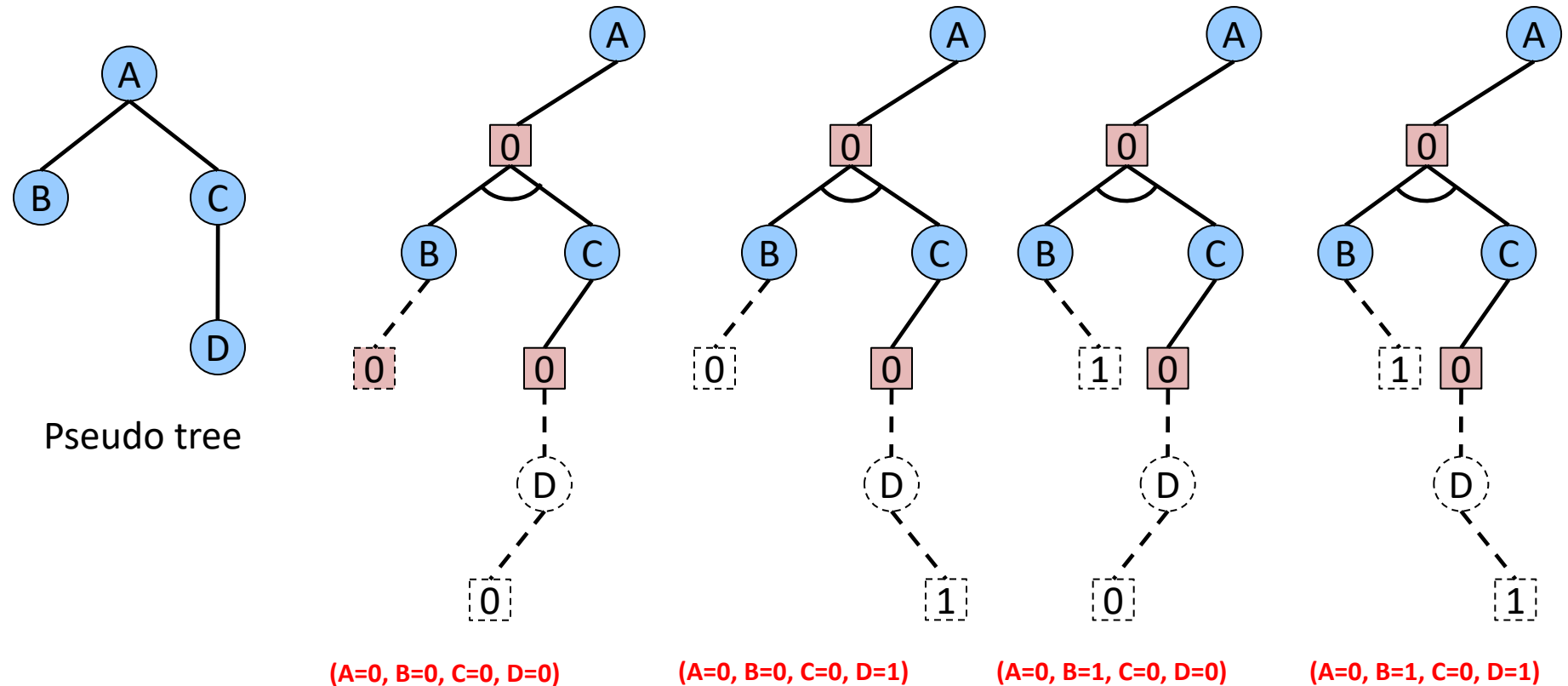
- **Best-First Algorithms, (A*)**
 - Expand nodes in OPEN list in order of $\min f(n)$
 - Terminates with first full solution (for mpe)
- **Properties**
 - Optimal, if $h(n) \leq v(n)$
 - Expands least set of nodes
 - exponential memory
 - **Not anytime solution for MPE**
 - **But, yields lower bounds on value, anytime**

Basic Heuristic Search; Depth-First



- **Depth-First (B&B for MAP)**
 - Expand in dfs order
 - Update UB with each solution
 - Prunes if $f(n) \geq UB$
- **Properties**
 - Can use only linear memory
 - Yields upper bounds anytime

Partial Solution Tree for AND/OR



Extension(T') – solution trees that extend T'

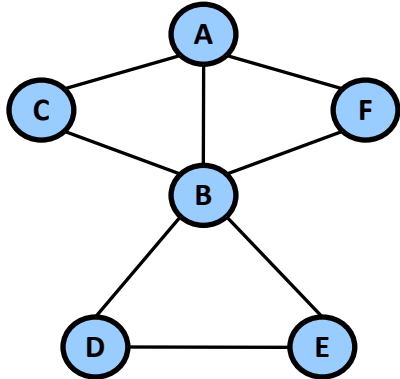
$g(T')$ = conditioned value of a node

$V(T')$ = the combined value below T'

$f^*(T')$ = conditioned value through T'

Exact Evaluation Function

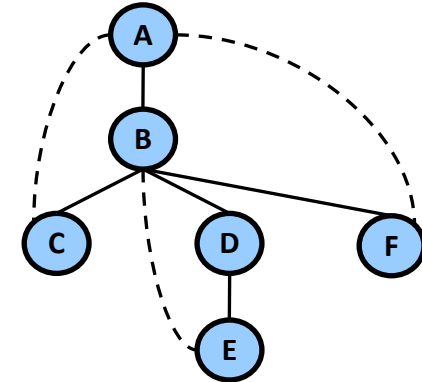
Conditioned value of a node



A	B	C	$f_1(ABC)$
0	0	0	2
0	0	1	5
0	1	0	3
0	1	1	5
1	0	0	9
1	0	1	3
1	1	0	7
1	1	1	2

A	B	F	$f_2(ABF)$
0	0	0	3
0	0	1	5
0	1	0	1
0	1	1	4
1	0	0	6
1	0	1	5
1	1	0	6
1	1	1	5

B	D	E	$f_3(BDE)$
0	0	0	6
0	0	1	4
0	1	0	8
0	1	1	5
1	0	0	9
1	0	1	3
1	1	0	7
1	1	1	4



OR

AND

OR

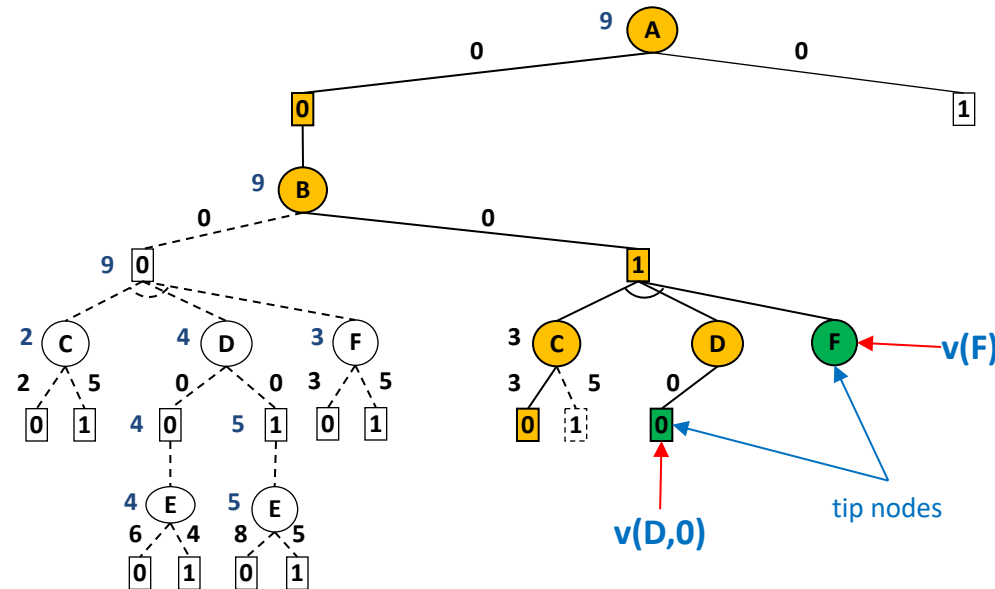
AND

OR

AND

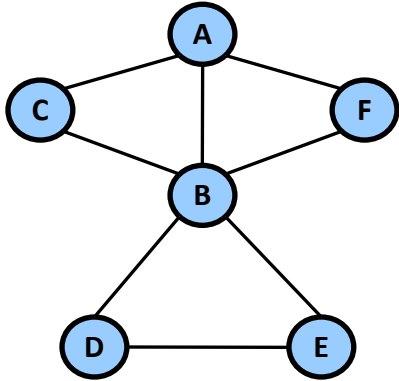
OR

AND



$$f^*(T') = w(A,0) + w(B,1) + w(C,0) + w(D,0) + v(D,0) + v(F)$$

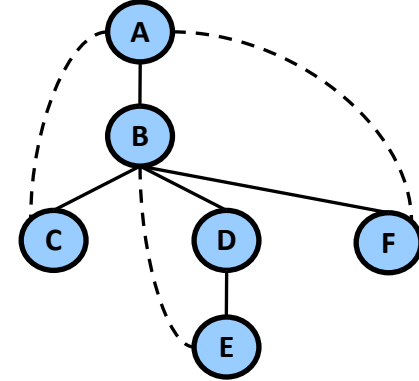
Heuristic Evaluation Function



A	B	C	$f_1(ABC)$
0	0	0	2
0	0	1	5
0	1	0	3
0	1	1	5
1	0	0	9
1	0	1	3
1	1	0	7
1	1	1	2

A	B	F	$f_2(ABF)$
0	0	0	3
0	0	1	5
0	1	0	1
0	1	1	4
1	0	0	6
1	0	1	5
1	1	0	6
1	1	1	5

B	D	E	$f_3(BDE)$
0	0	0	6
0	0	1	4
0	1	0	8
0	1	1	5
1	0	0	9
1	0	1	3
1	1	0	7
1	1	1	4



OR

AND

OR

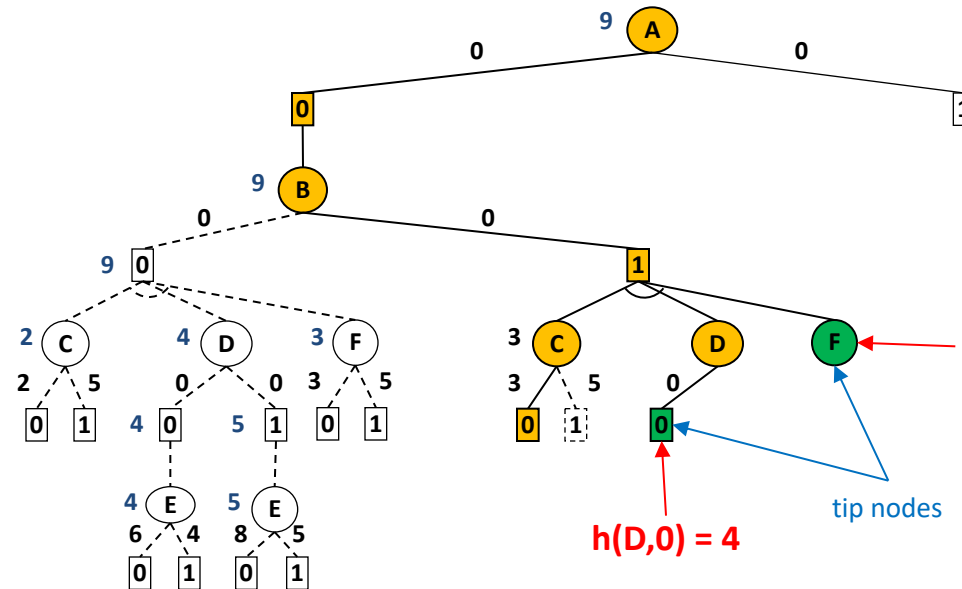
AND

OR

AND

OR

AND



$$h(n) \leq v(n)$$

$$h(F) = 5$$

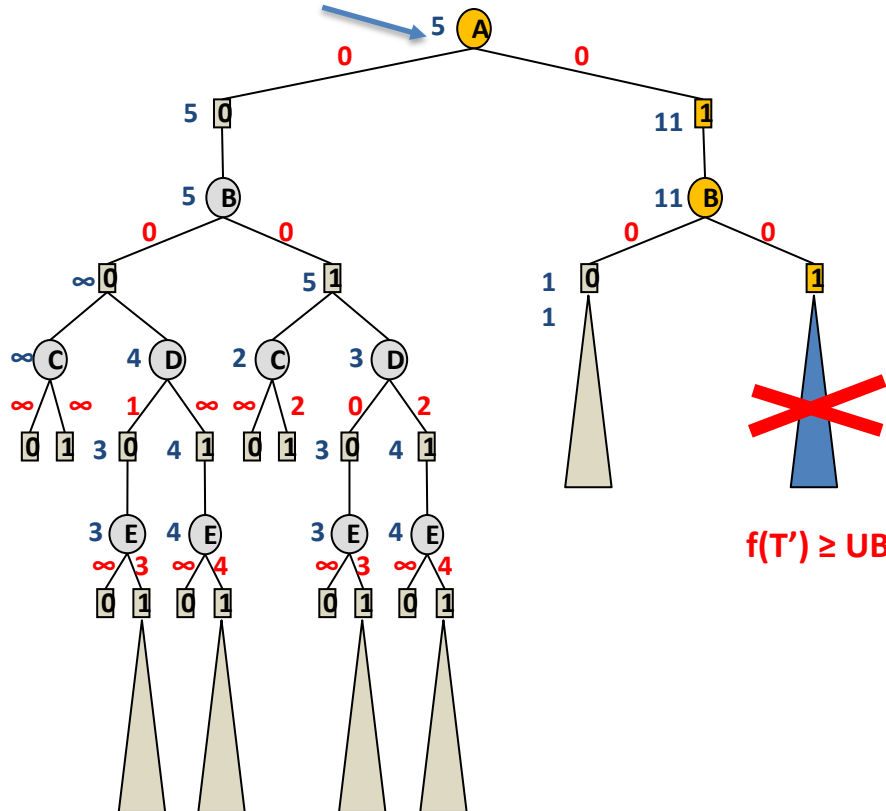
$$h(D,0) = 4$$

tip nodes

$$f(T') = w(A,0) + w(B,1) + w(C,0) + w(D,0) + h(D,0) + h(F) = 12 \leq f^*(T')$$

Depth-First AND/OR Branch-and-Bound

UB (best solution so far)



- Associate each node n with a heuristic lower bound $h(n)$ on $v(n)$

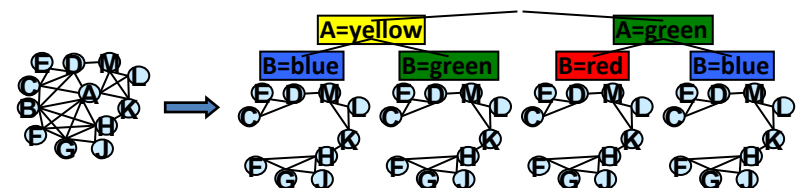
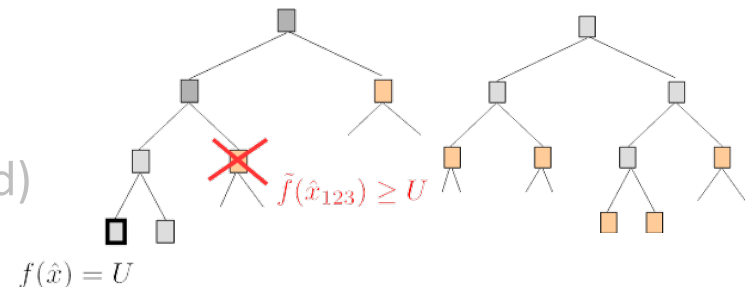
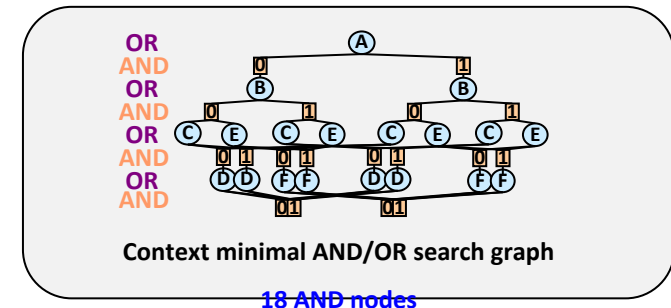
Algorithm AOBB:

- **EXPAND** (top-down)
 - Evaluate $f(T')$ and prune search if $f(T') \geq UB$
 - If not in cache, generate successors of the tip node n
- **PROPAGATE** (bottom-up)
 - Update value of the parent p of n
 - OR nodes: minimization
 - AND nodes: summation
 - Cache value of n based on context only if fully explored

Outline: Search

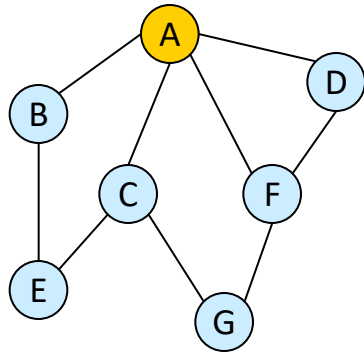
- Review Graphical Models
- AND/OR search spaces, pseudo-trees
 - AND/OR search trees
 - AND/OR search graphs
 - Generating good pseudo-trees
 - Brute-force search
- Heuristic search (HS) for AND/OR spaces
 - Basic Heuristic search (Depth and Best)
 - AND/OR Depth-first HS (branch and bound)
 - AND/OR Best-first heuristic search
 - The Guiding MBE heuristic
 - Marginal Map (max-sum-product)
- **Hybrids of search and Inference**
- Summary and Class 2

Chapter 7 in Dechter1



Conditioning versus Elimination

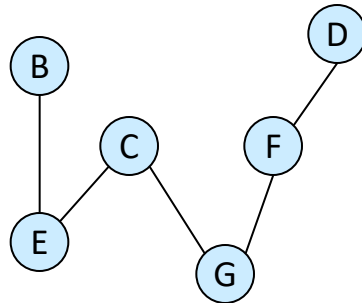
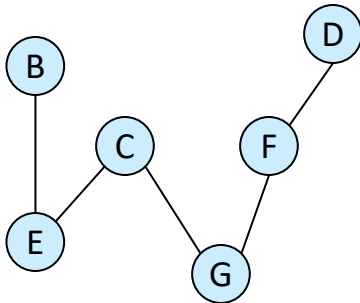
Conditioning (search)



$A=1$

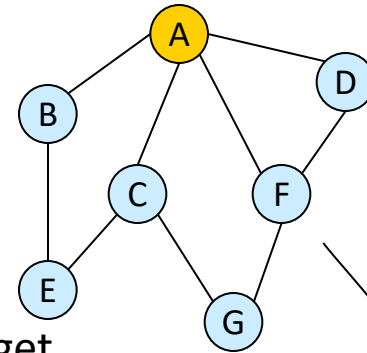
...

$A=k$

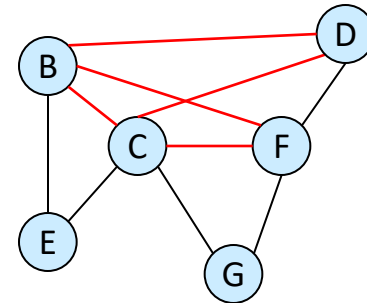


k "sparser" problems

Elimination (inference)



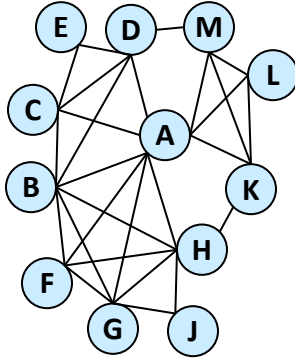
The main target
In conditioning is to
Trade memory for time.
It is not really possible to do better
Then elimination



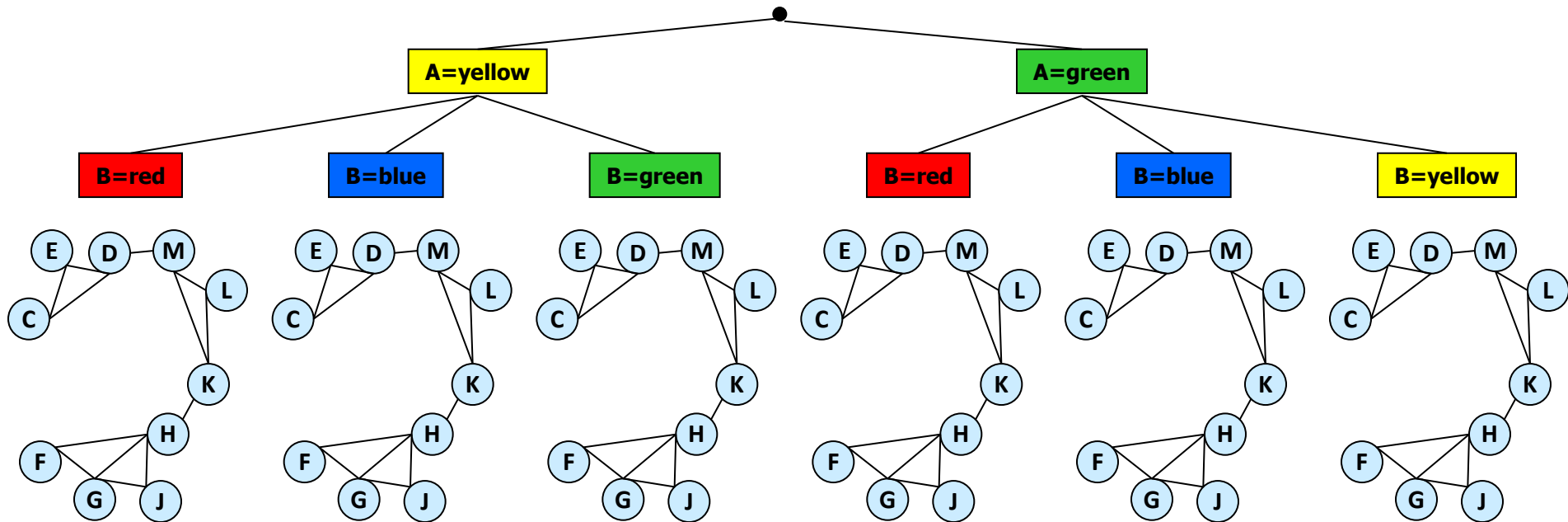
1 "denser" problem

OR w-Cutset

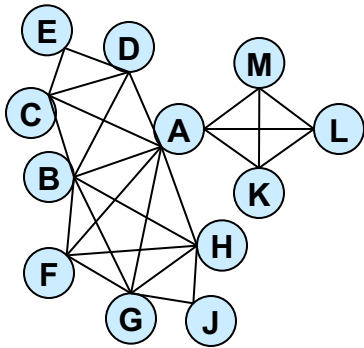
Graph
Coloring
problem



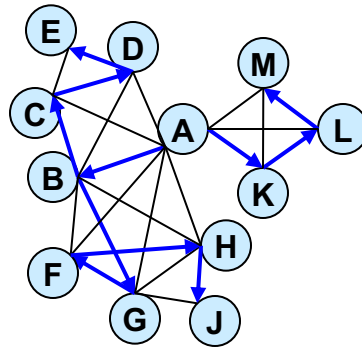
- Inference may require too much memory
- **Condition** on some of the variables



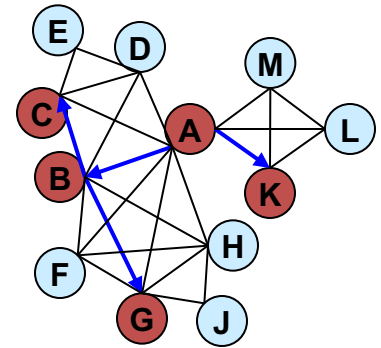
AND/OR w-cutset



graphical model

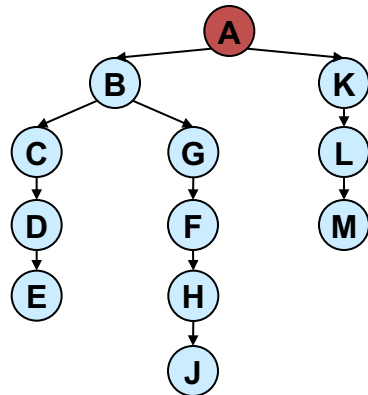
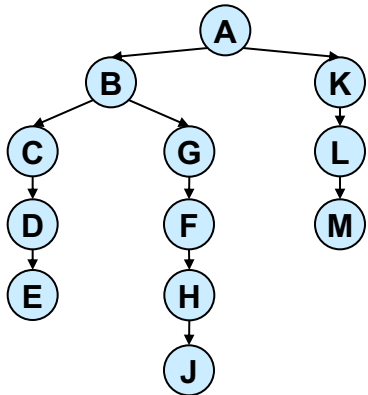
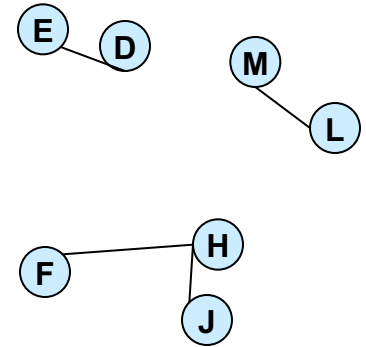
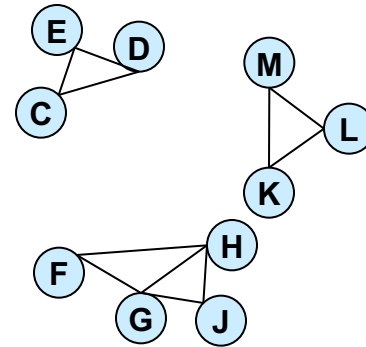
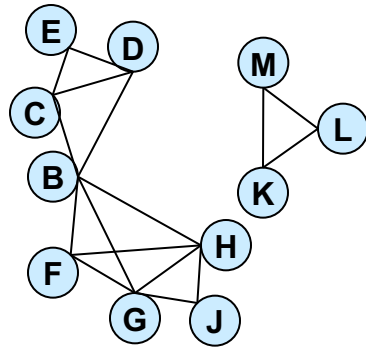
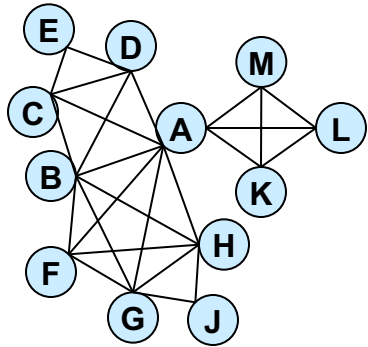


pseudo tree

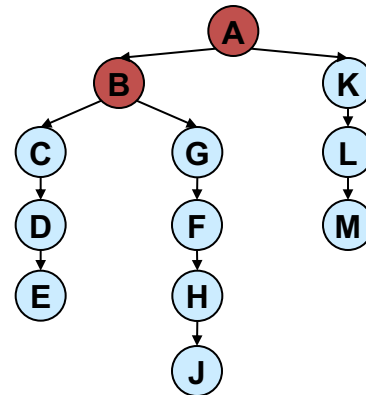


1-cutset tree

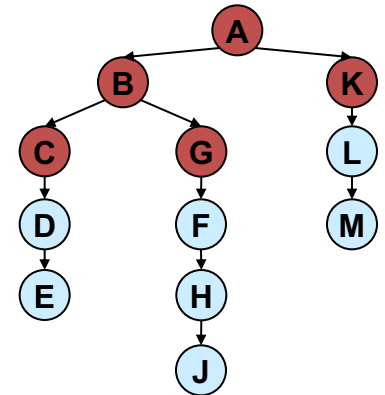
AND/OR w-cutset



3-cutset



2-cutset



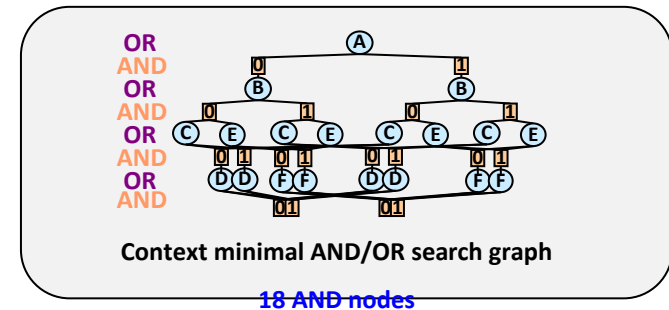
1-cutset

Summary: AND/OR Cutset-Conditioning

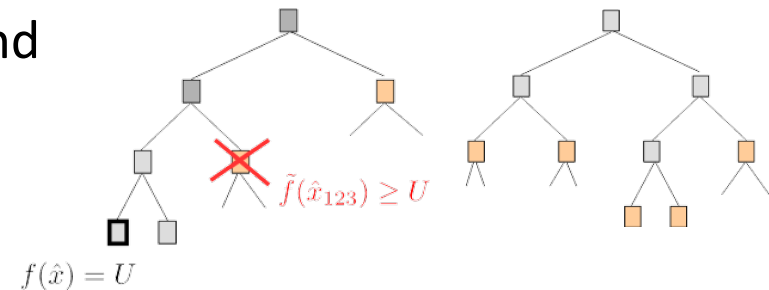
- Trade memory for time.
- We never improve time: cycle-cutset size is larger of equal to treewidth+1
- Sometime we do not worsen the time and memory can be much better (e.g., when the induced-width is high)

Summary of Search

- AND/OR search spaces, pseudo-trees
 - AND/OR search trees
 - AND/OR search graphs
 - Generating good pseudo-trees
 - Brute-force search



- Heuristic search for AND/OR spaces
 - Depth-first AND/OR branch and bound
 - Best-first AND/OR search
 - The Guiding MBE heuristic
 - Marginal Map (max-sum-product)



- Hybrids of search and Inference
- **Summary and Class 2**

