Exact Reasoning: AND/OR Search and Hybrids

COMPSCI 276, Fall 2013

Set 8, Rina dechter

Probabilistic Inference Tasks

- Belief updating:

$$BEL(X_i) = P(X_i = x_i \mid evidence)$$

- Finding most probable explanation (MPE)

$$\bar{x}^* = \arg\max_{\bar{x}} P(\bar{x}, e)$$

- Finding maximum a-posteriory hypothesis

$$(a_1^*, \ldots, a_k^*) = \arg\max_{\bar{a}} \sum_{X/A} P(\bar{x}, e)$$

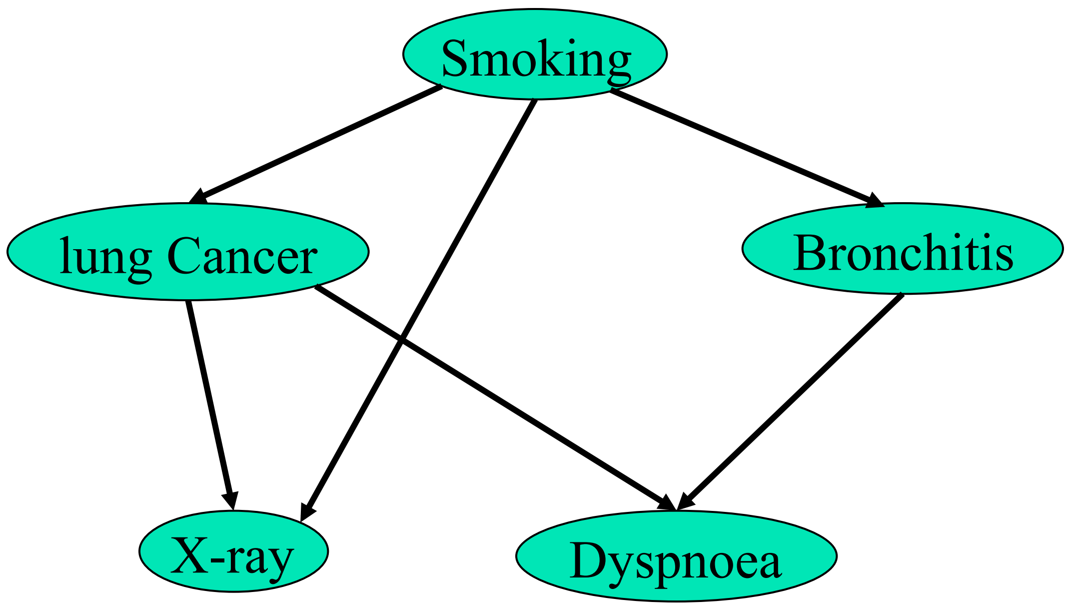
$A \subseteq X$: **hypothesis variables**

- Finding maximum-expected-utility (MEU) decision

$$(d_1^*, \ldots, d_k^*) = \arg\max_{\bar{d}} \sum_{X/D} P(\bar{x}, e) U(\bar{x})$$

$D \subseteq X$: **decision variables**
$U(\bar{x})$: **utility function**

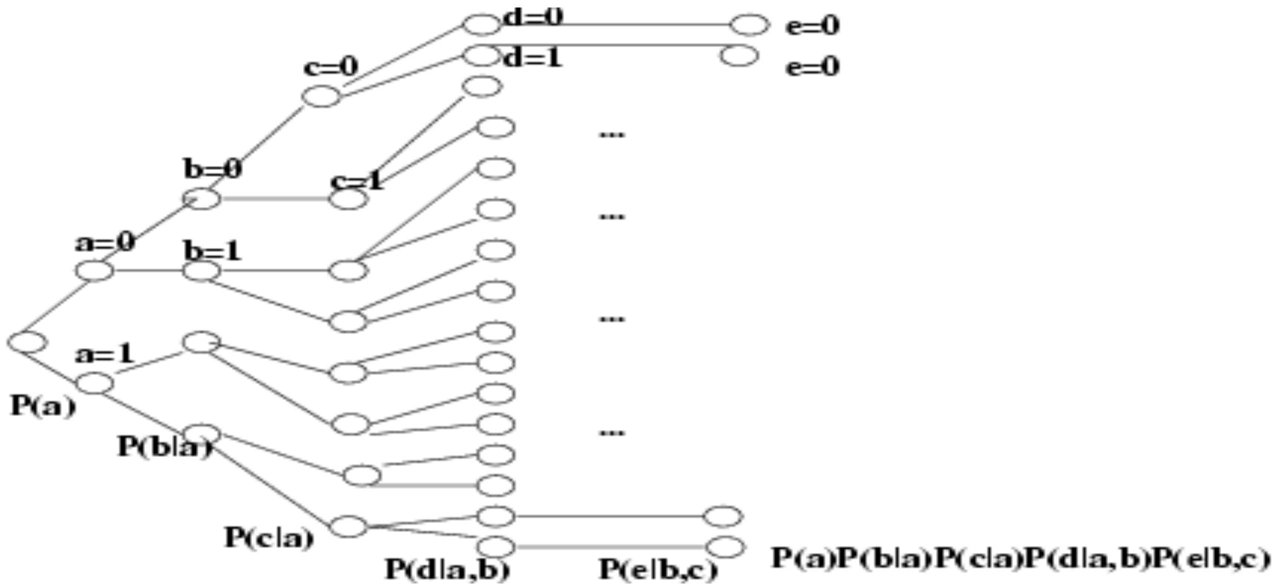
Belief Updating



P (lung cancer=yes | smoking=no, dyspnoea=yes) = ?

Conditioning generates the probability tree

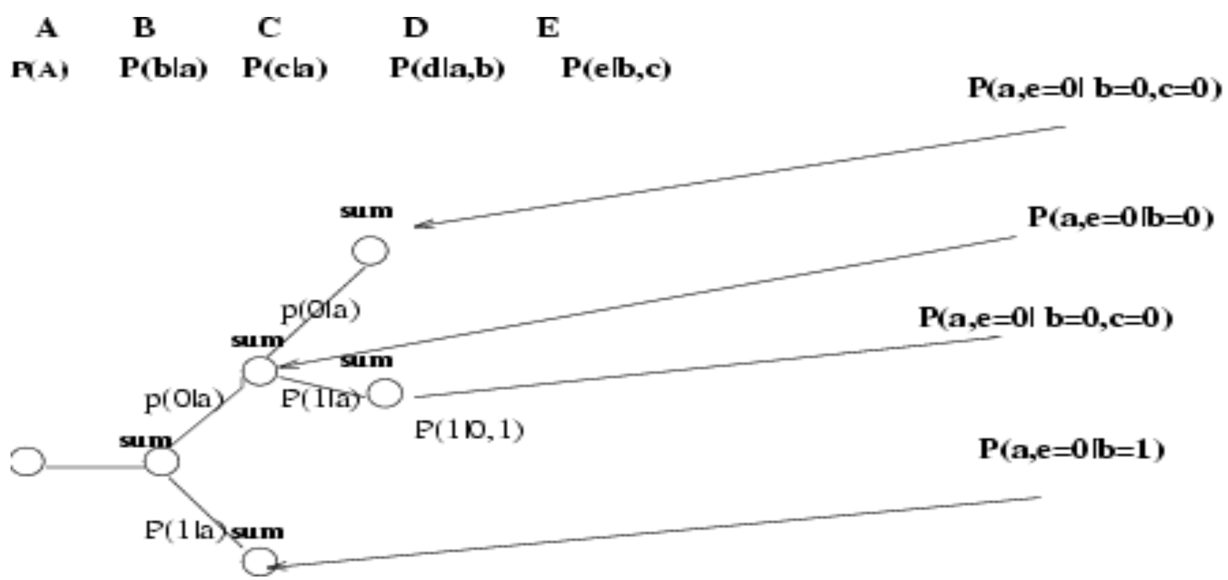
$$P(a, e=0) = P(a)\sum_b P(b \mid a)\sum_c P(c \mid a)\sum_b P(d \mid a, b)\sum_{e=0} P(e \mid b, c)$$



Complexity of conditioning: exponential time, linear space

Conditioning+Elimination

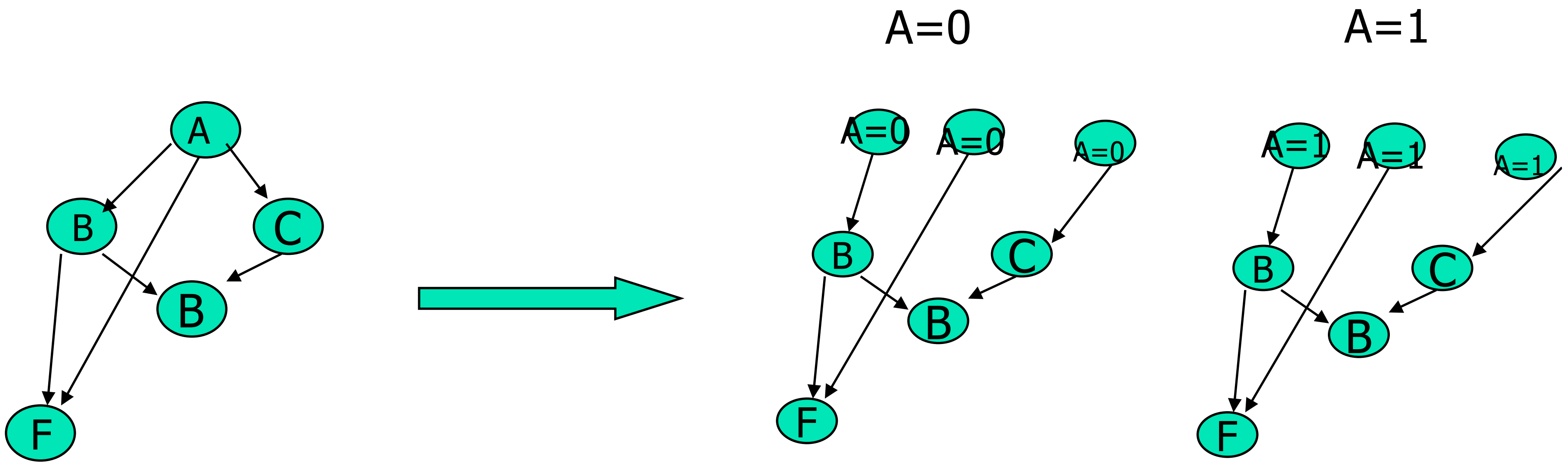
$$P(a,e=0) = P(a)\sum_b P(b|a)\sum_c P(c|a)\sum_d P(d|a,b)\sum_{e=0} P(e|b,c)$$



Idea: conditioning until w^* of a (sub)problem gets small

Loop-cutset decomposition

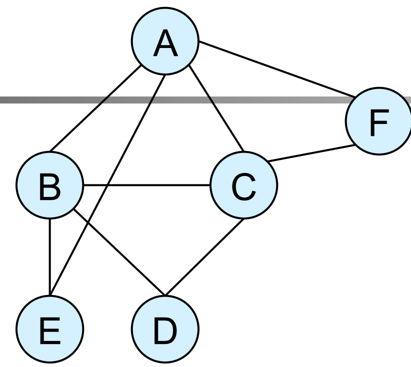
- You condition until you get a polytree



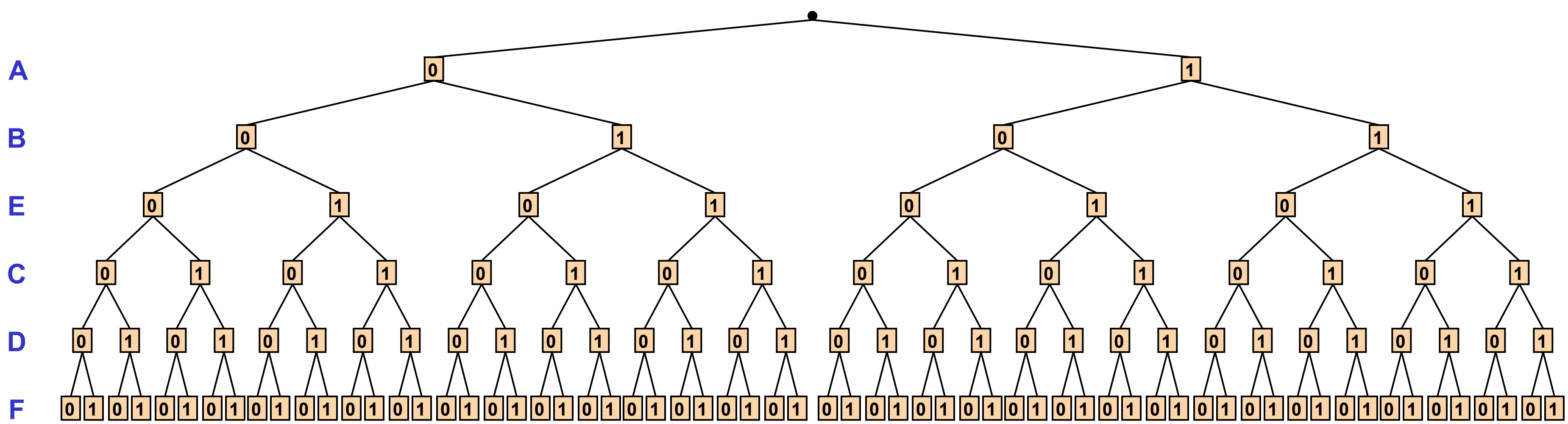
P(B|F=0) = P(B, A=0|F=0)+P(B,A=1|F=0)

Loop-cutset method is time exp in loop-cutset size
And linear space

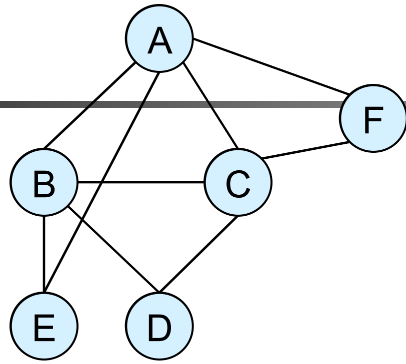
OR search space



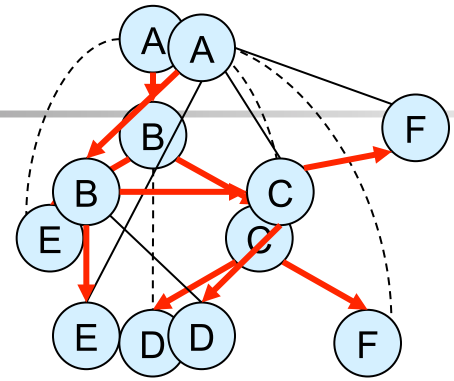
Ordering: A B E C D F



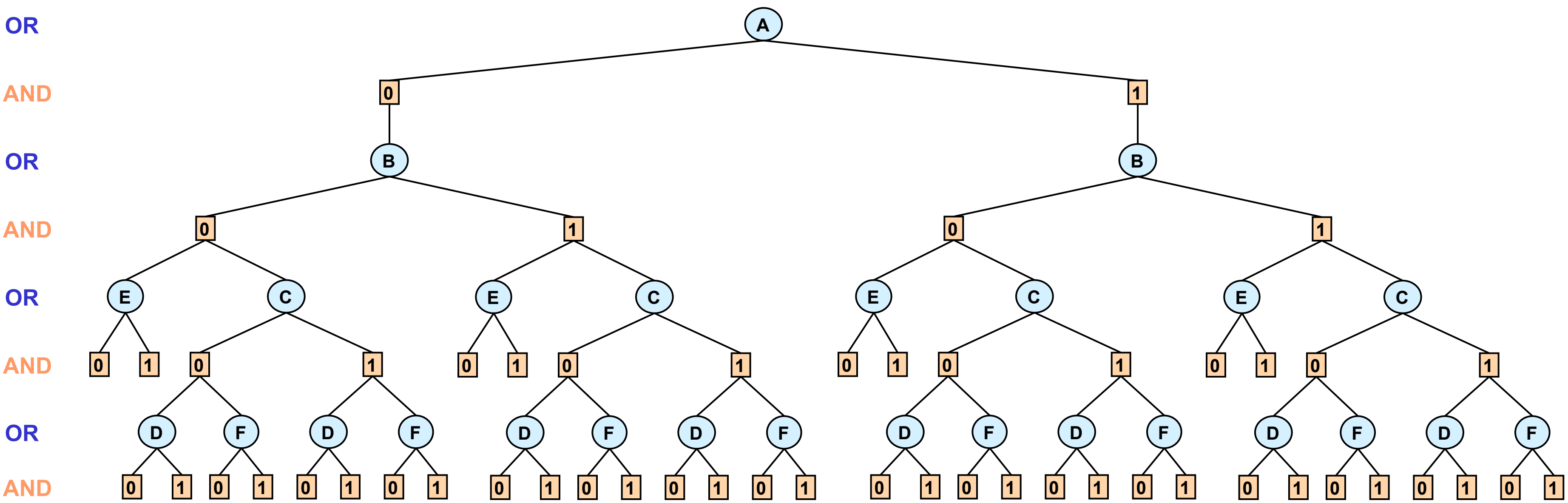
AND/OR search space



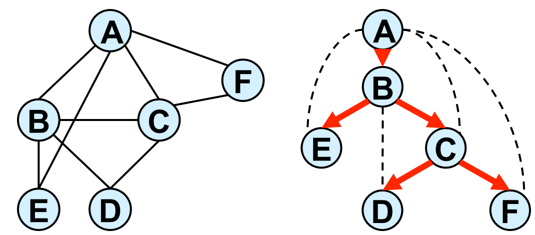
Primal graph

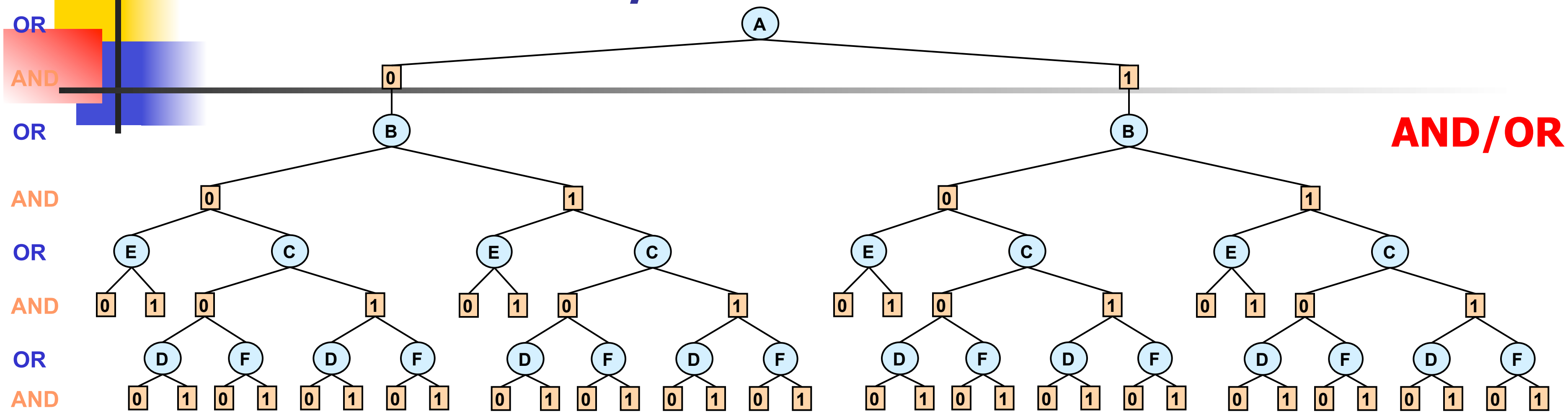


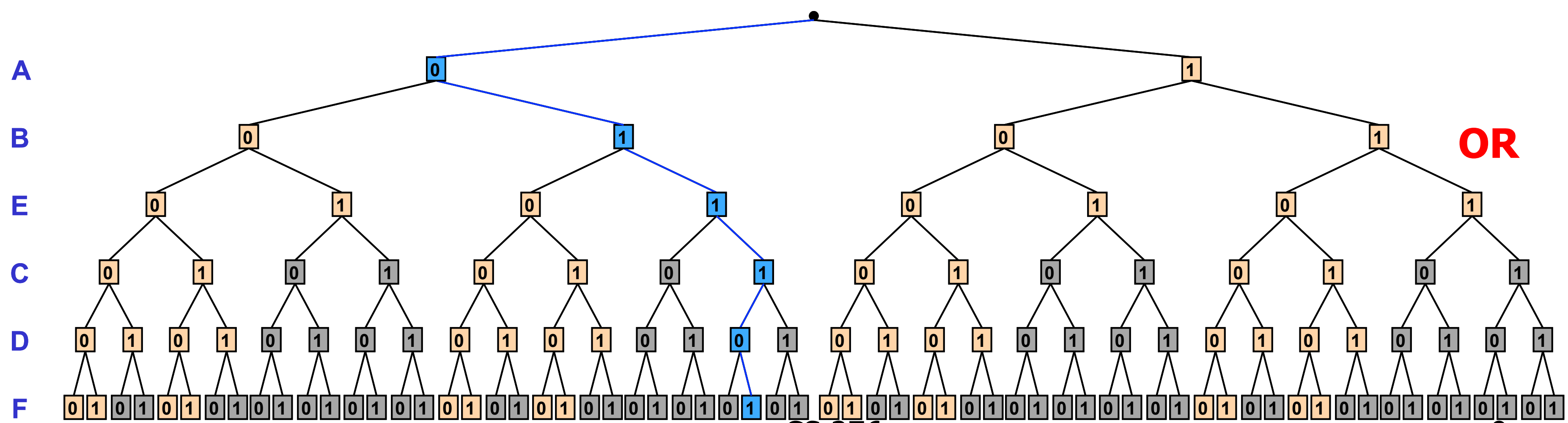
DFS tree



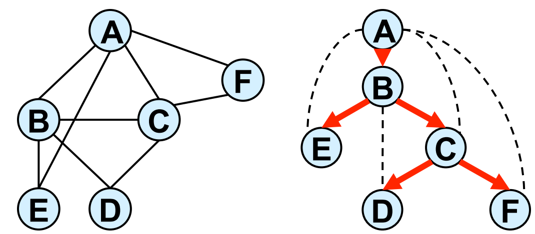
OR vs AND/OR

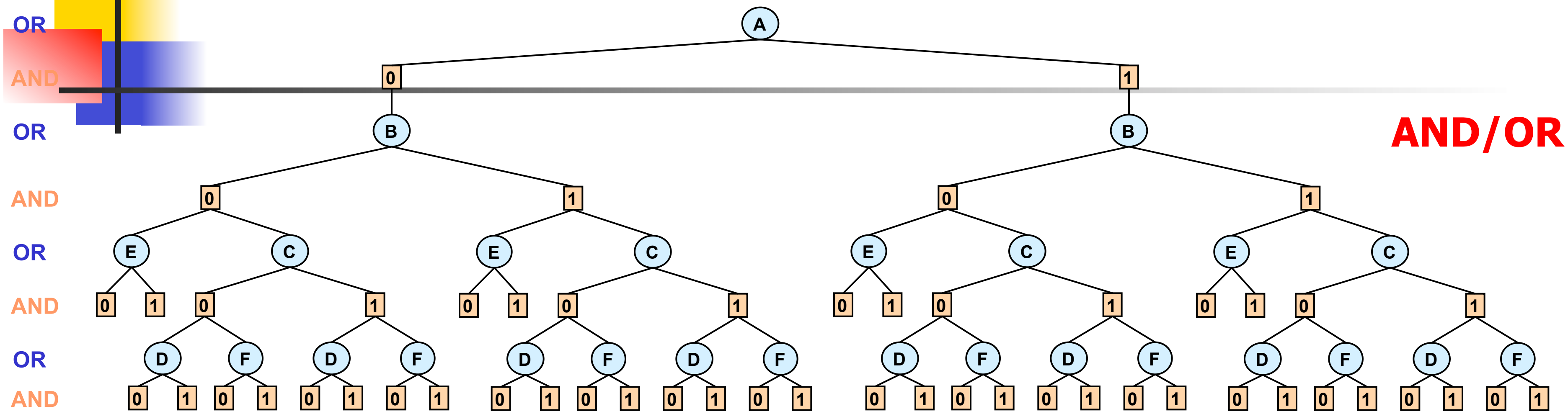




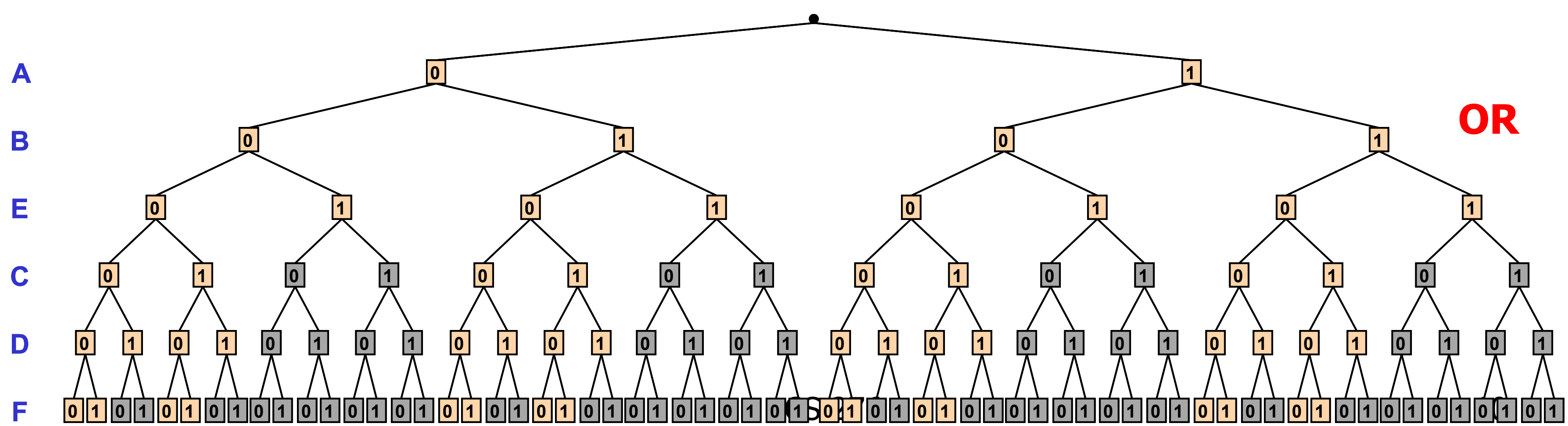


AND/OR vs. OR



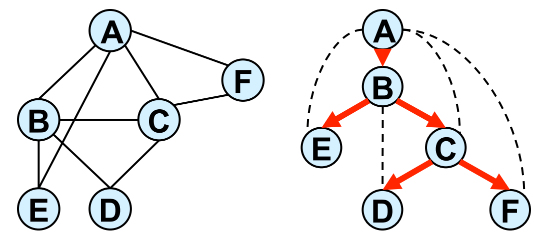


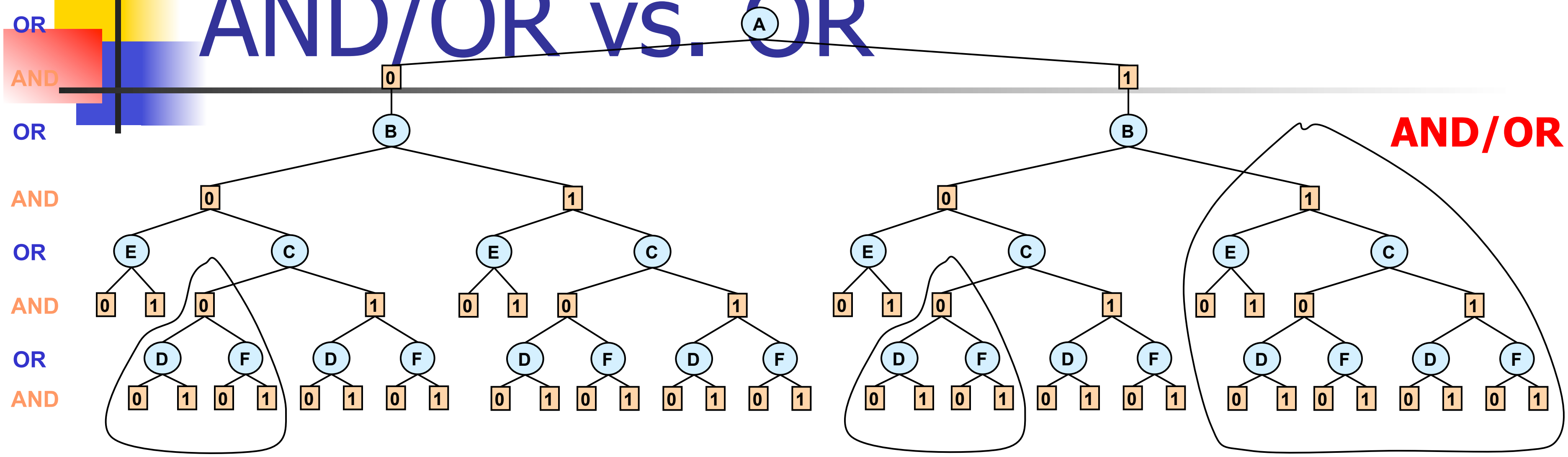
AND/OR size: exp(4), OR size exp(6)

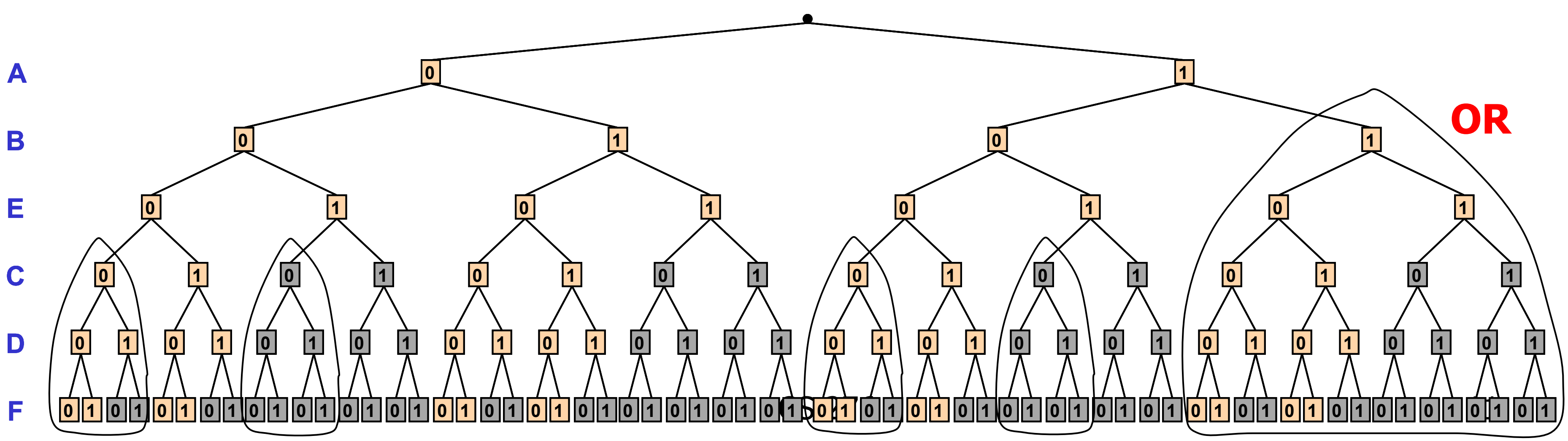


AND/OR vs. OR

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

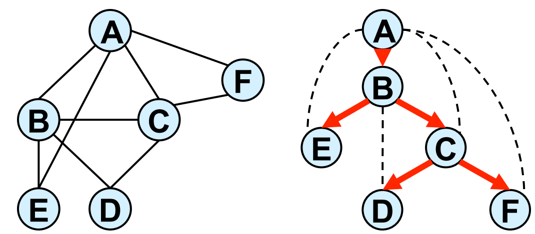


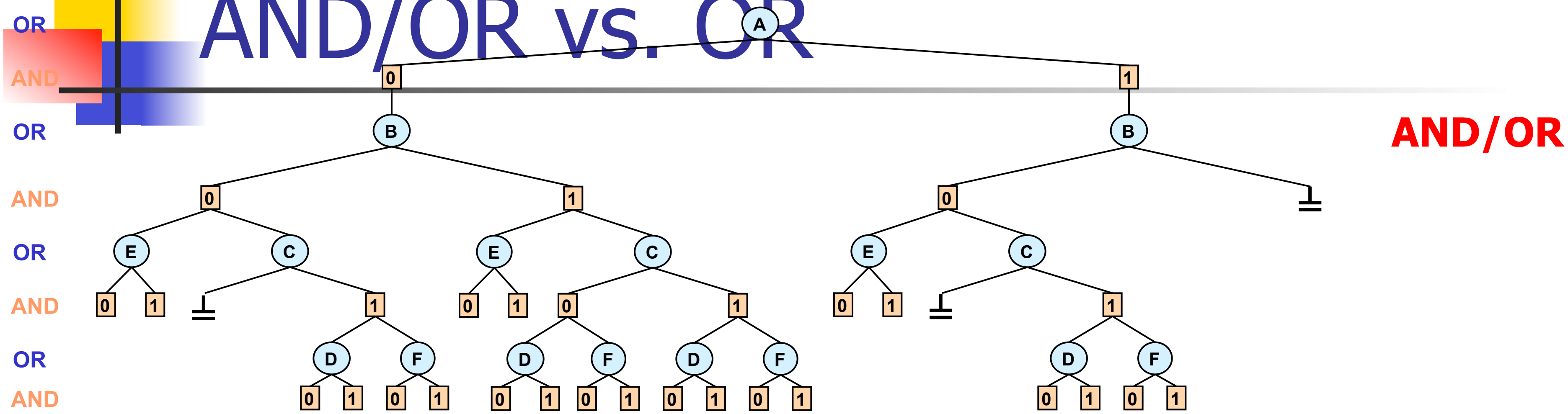


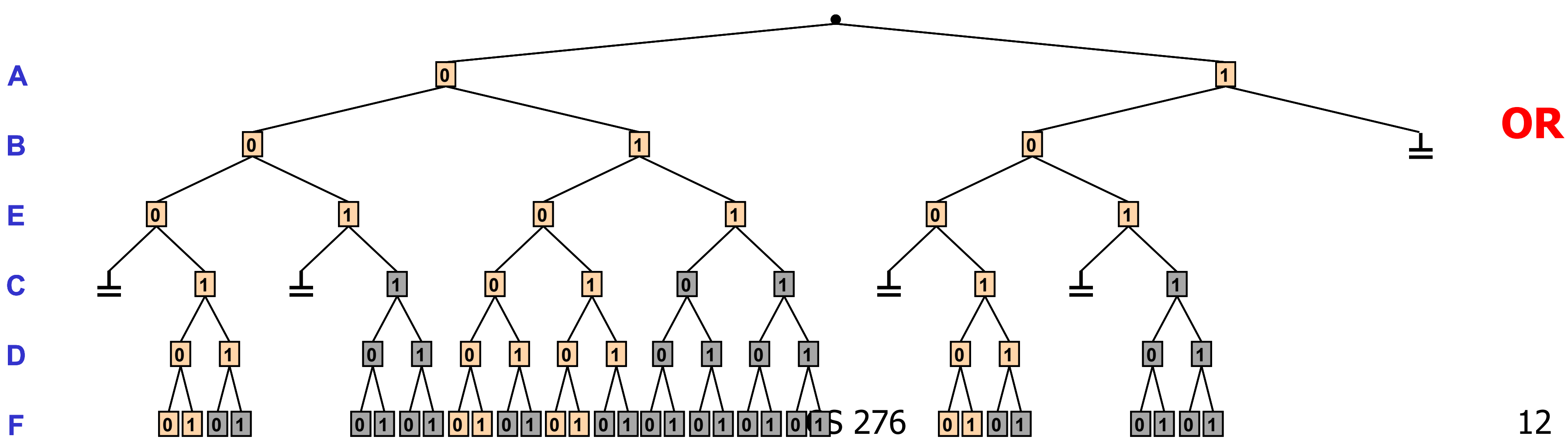


AND/OR vs. OR

(A=1,B=1)
(B=0,C=0)





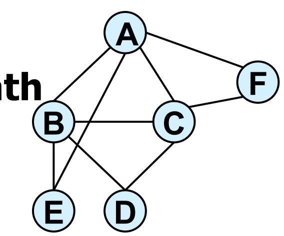


OR space vs. AND/OR space

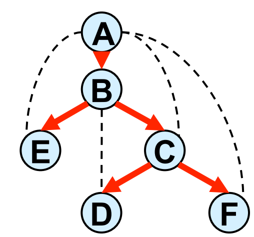
width	height	OR space			AND/OR space		
		time(sec.)	nodes	backtracks	time(sec.)	AND nodes	OR nodes
5	10	3.154	2,097,150	1,048,575	0.03	10,494	5,247
4	9	3.135	2,097,150	1,048,575	0.01	5,102	2,551
5	10	3.124	2,097,150	1,048,575	0.03	8,926	4,463
4	10	3.125	2,097,150	1,048,575	0.02	7,806	3,903
5	13	3.104	2,097,150	1,048,575	0.1	36,510	18,255
5	10	3.125	2,097,150	1,048,575	0.02	8,254	4,127
6	9	3.124	2,097,150	1,048,575	0.02	6,318	3,159
5	10	3.125	2,097,150	1,048,575	0.02	7,134	3,567
5	13	3.114	2,097,150	1,048,575	0.121	37,374	18,687
5	10	3.114	2,097,150	1,048,575	0.02	7,326	3,663

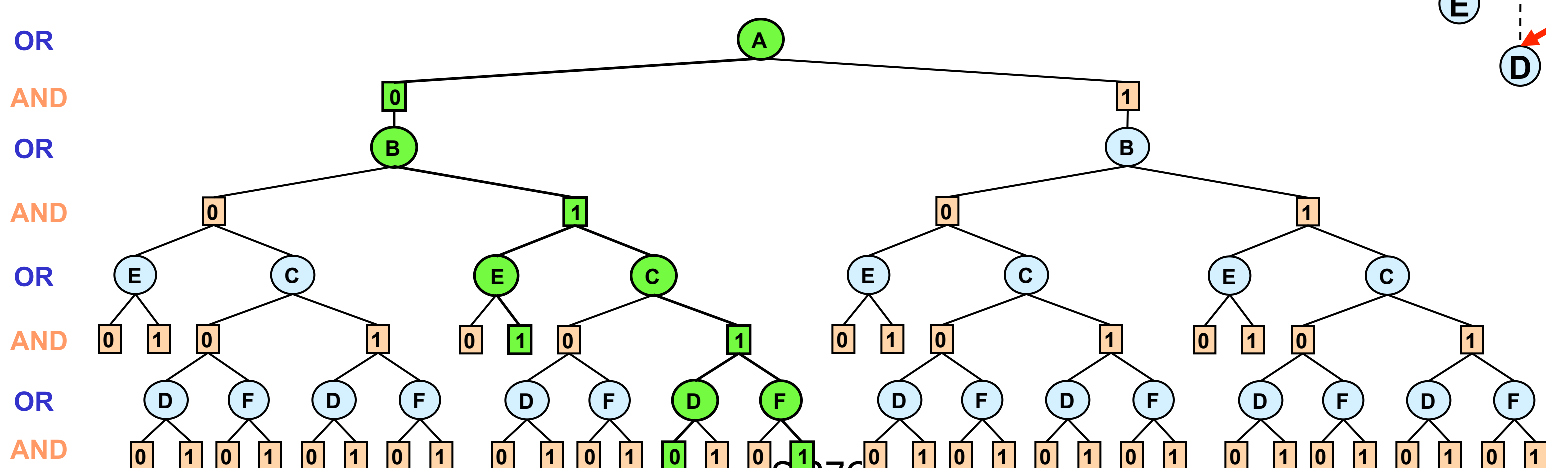
AND/OR search tree for graphical models

- **The AND/OR search tree of a GM relative to a spanning-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 <X,v> are all X child variables in T**

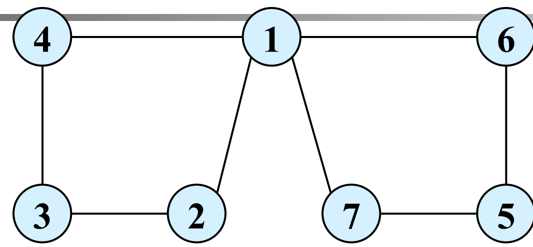


- **A solution is a consistent subtree**
- **Task: compute the value of the root node**

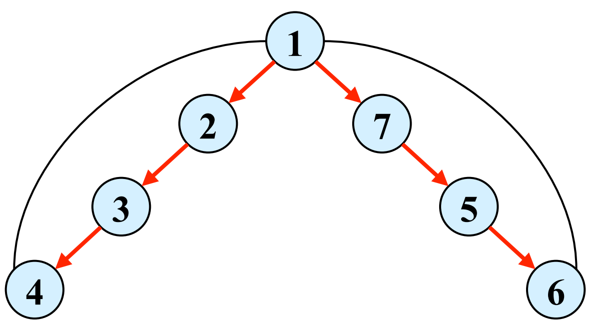




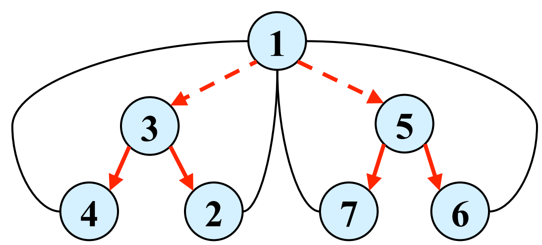
From DFS trees to pseudo-trees (Freuder 85, Bayardo 95)



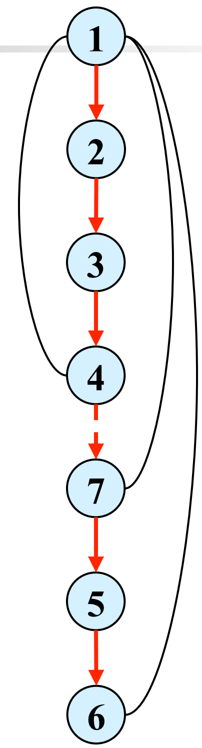
(a) Graph



(b) DFS tree
depth=3



(c) pseudo- tree
depth=2

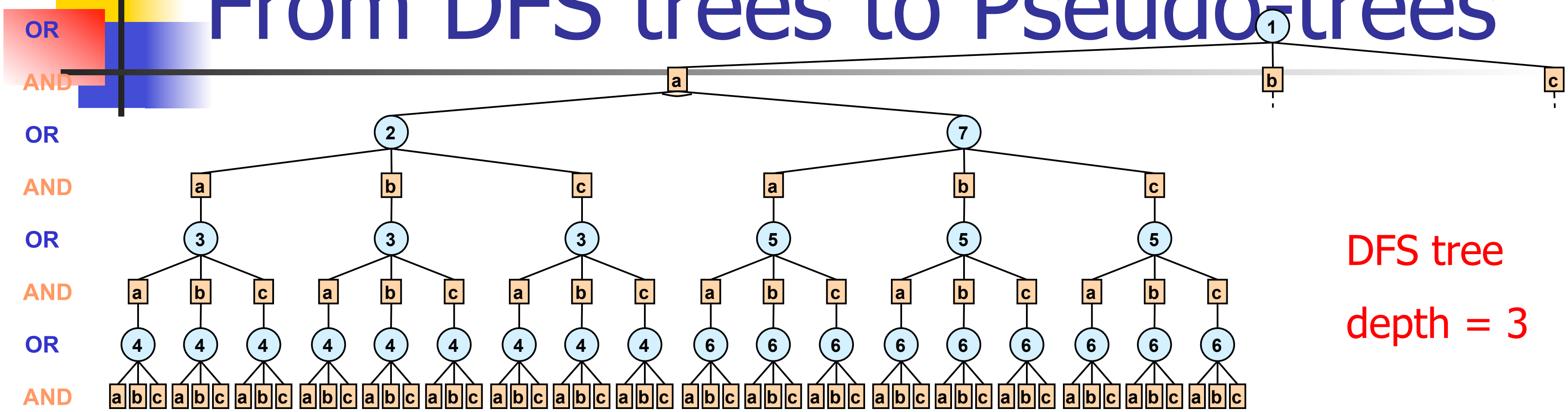


(d) Chain
depth=6

Pseudo-tree definition

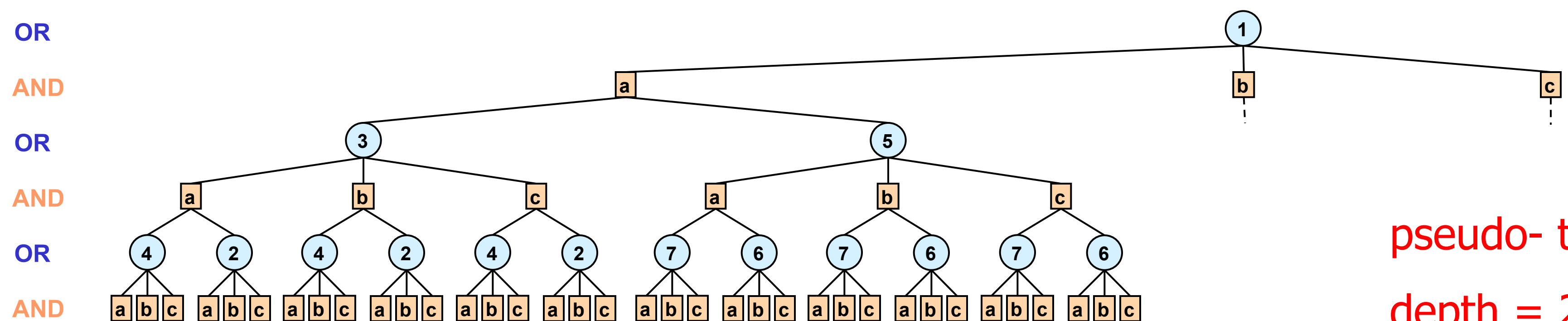
Given undirected graph G = (V, E), a directed rooted tree T = (V, E′) defined on all its nodes is a pseudo tree if any arc of G which is not included in E′ is a back-arc in T, namely it connects a node in T to an ancestor in T. The arcs in E′ may not all be included in E. Given a pseudo tree T of G, the extended graph of G relative to T includes also the arcs in E′ that are not in E. Namely the extended graph is defined as GT = (V, E ∪ E′).

From DFS trees to Pseudo-trees



DFS tree

depth = 3



pseudo- tree

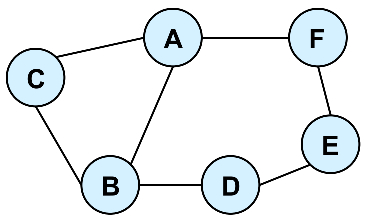
depth = 2

Finding min-depth Pseudo-trees

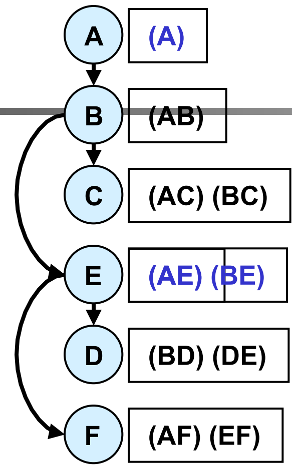
- Finding min depth DFS, or pseudo tree is NP-complete, but:
- Given a tree-decomposition whose tree-width is w*, there exists a pseudo -tree T of G whose depth, satisfies m <= w* log n,

Generating pseudo-trees from Bucket trees

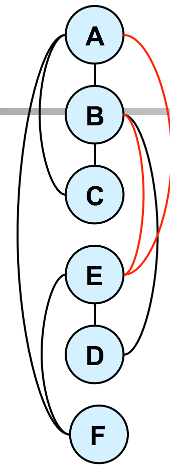
Note: we plot order from top to bottom here



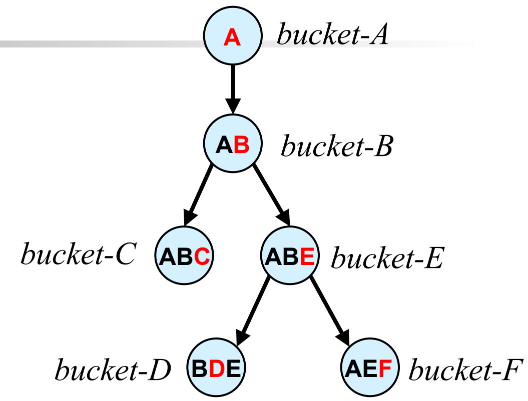
d: A B C E D F



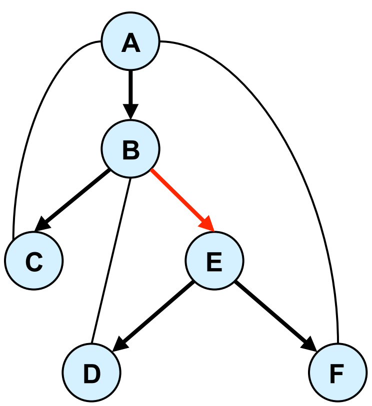
Bucket-tree based on *d*



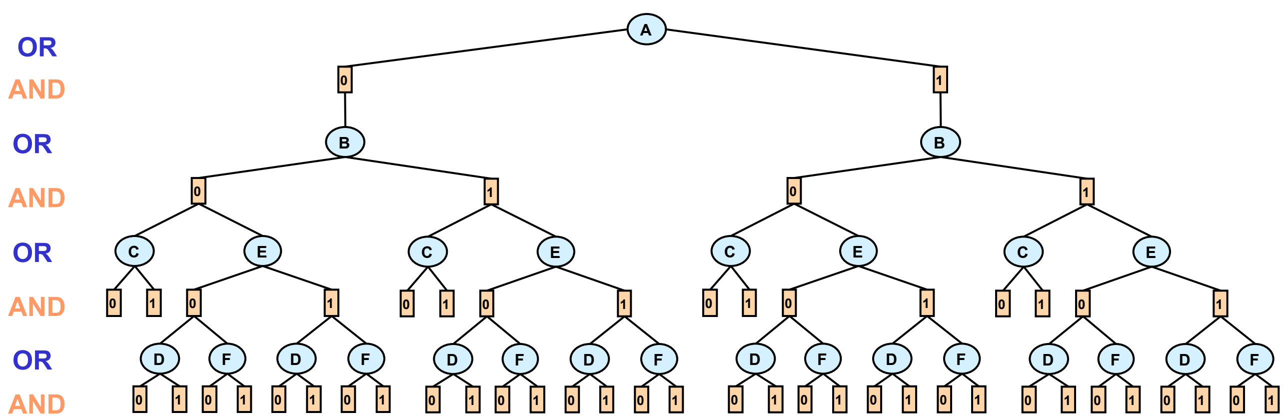
Induced graph



Bucket-tree



Bucket-tree used as pseudo-tree



AND/OR search tree

Constructing Pseudo Trees

- **Min-Fill** (Kjaerulff, 1990)
  - Depth-first traversal of the induced graph obtained along the min-fill elimination order, or generate the bucket-tree
  - 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

Quality of the 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

yesian 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

AND/OR Search-tree properties

(*k* = domain size, *m* = pseudo-tree depth. *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 $O(n\ k^m)$
  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

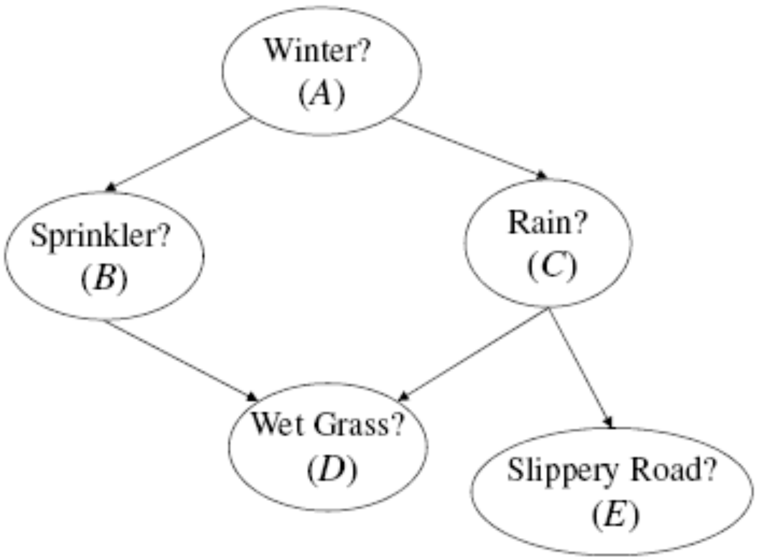
Tasks and value of nodes

- **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 othewise.
  - **Optimization: v(n) is the optimal solution in T(n)**
  - **Belief updating: v(n), probability of evidence in T(n).**
  - **Partition function: v(n) is the total probability in T(n).**

- **Goal: compute the value of the root node recursively using dfs search of the AND/OR tree.**

- **Theorem: Complexity of AO dfs search is**
  - **Space: O(n)**
  - **Time: $O(n\ k^m)$**
  - **Time: O(exp(w* log n))**

A Bayesian Network



A	Θ_A
true	.6
false	.4

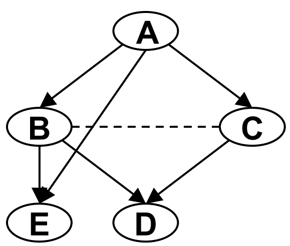
A	B	$\Theta_{B\|A}$
true	true	.2
true	false	.8
false	true	.75
false	false	.25

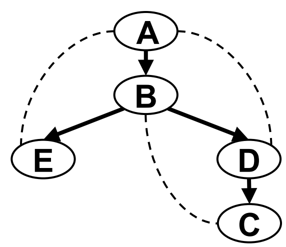
A	C	$\Theta_{C\|A}$
true	true	.8
true	false	.2
false	true	.1
false	false	.9

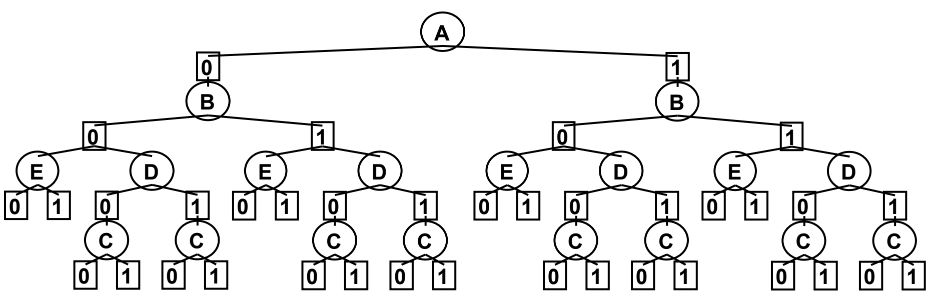
B	C	D	$\Theta_{D\|BC}$
true	true	true	.95
true	true	false	.05
true	false	true	.9
true	false	false	.1
false	true	true	.8
false	true	false	.2
false	false	true	0
false	false	false	1

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

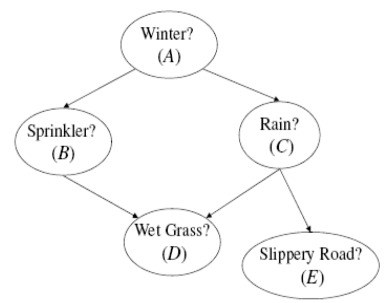
Belief-updating on example







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

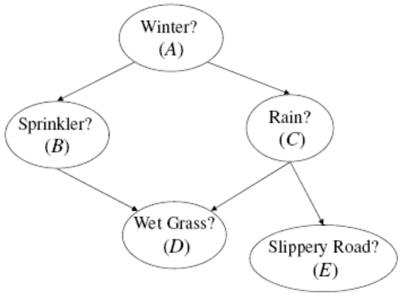
C	E	$\Theta_{E\|C}$
true	true	.7
true	false	.3
false	true	0
false	false	1

A weight of a Solution Tree

Definition 7.1.9 (weight of a solution subtree) *Given a weighted AND/OR tree* $S_{\mathcal{T}}(\mathcal{M})$*, of a graphical model* $\mathcal{M}$*, and given a solution subtree* t*, the weight of* t *is* $w(t) = \bigotimes_{e \in arcs(t)} w(e)$*, where* $arcs(t)$ *is the set of arcs in subtree* t*.*

Buckets relative to a pseudo-tree:
BT (Xi) = {f ∈ F |Xi ∈ scope(f), scope(f) ⊆ pathT (Xi)}.

A Bayesian Network



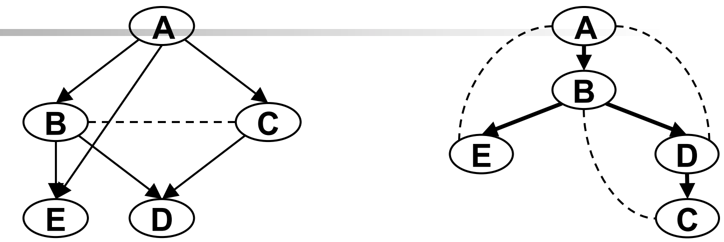
A	Θ_A
true	.6
false	.4

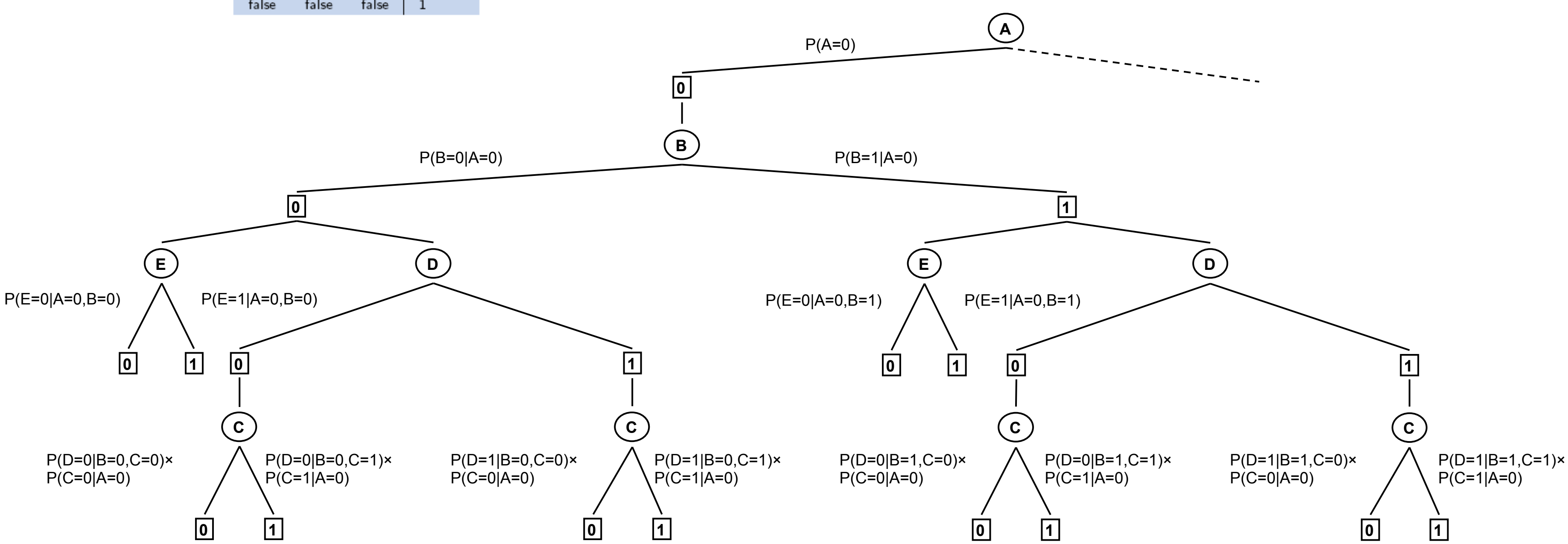
A	B	$\Theta_{B\|A}$
true	true	.2
true	false	.8
false	true	.75
false	false	.25

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





AND/OR Tree DFS Algorithm (Belief Updating)

$P(E \mid A,B)$

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

$P(B \mid A)$

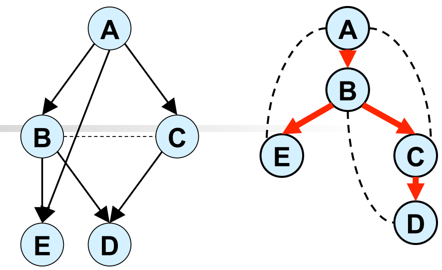
A	B=0	B=1
0	.4	.6
1	.1	.9

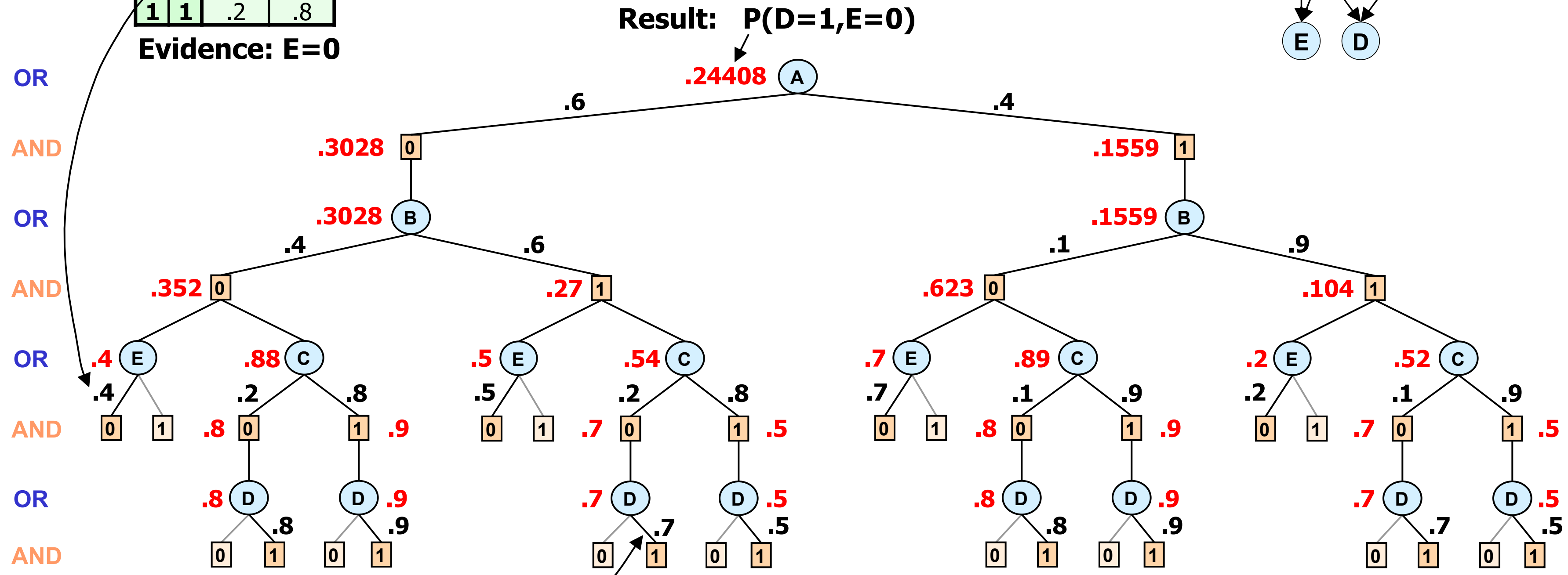
$P(C \mid A)$

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

$P(A)$

A	P(A)
0	.6
1	.4





$P(D \mid B,C)$

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

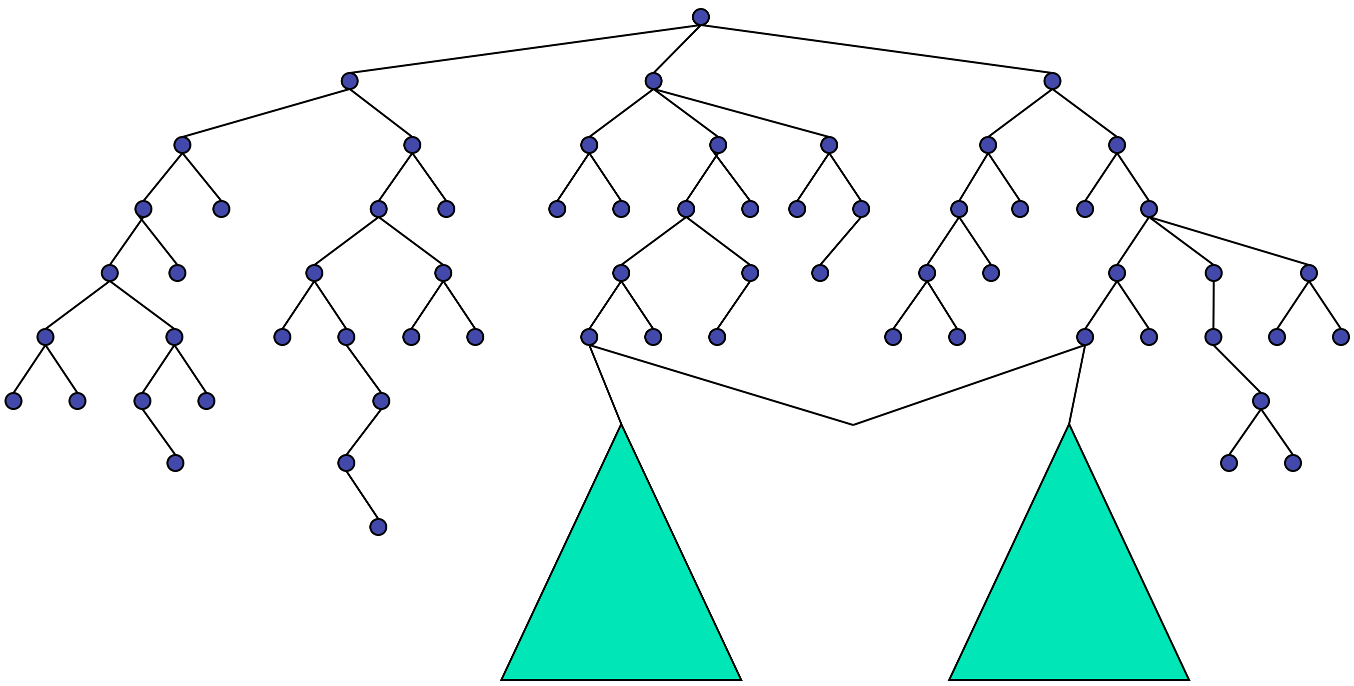
Complexity of AND/OR Tree Search

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

k = domain size
m = depth of pseudo-tree
n = number of variables
w*= treewidth

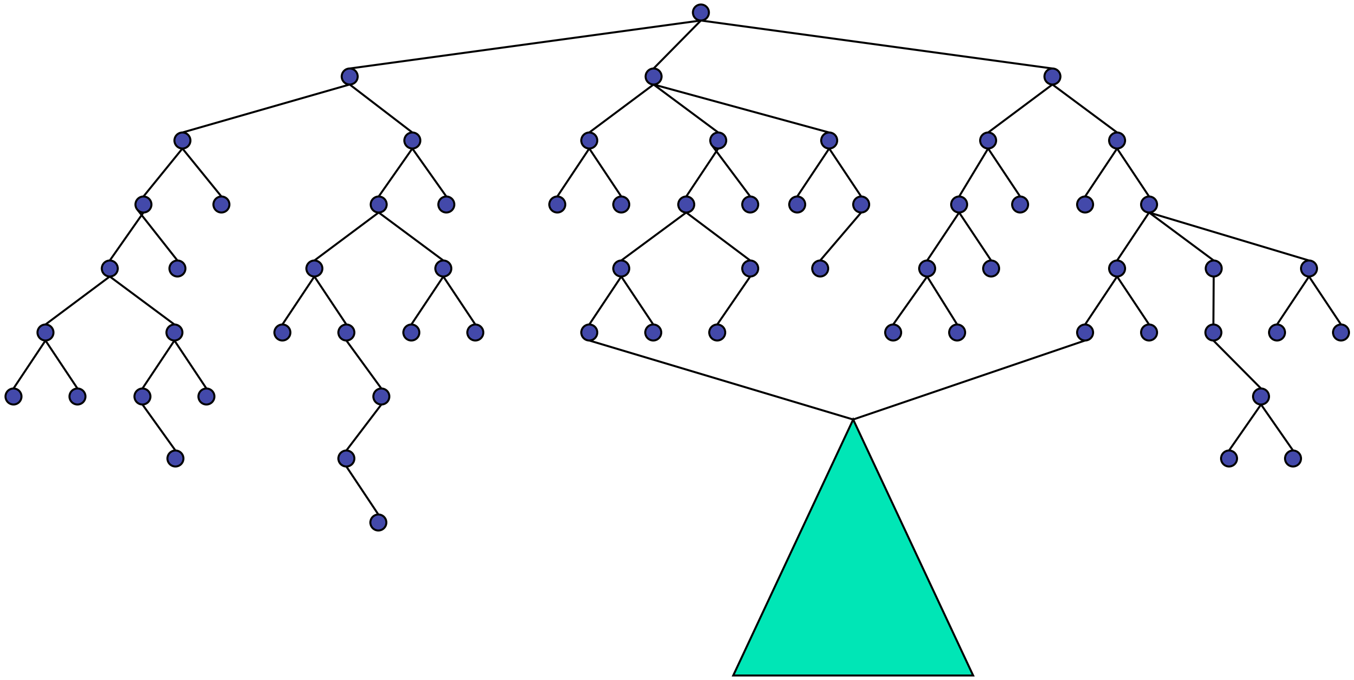
From Search Trees to Search Graphs

- Any two nodes that root identical subtrees (subgraphs) can be merged

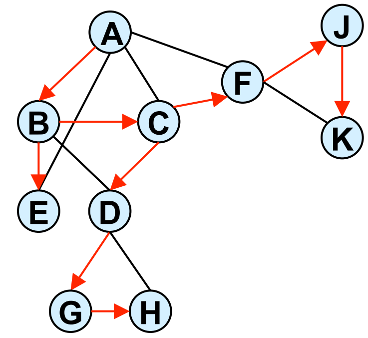


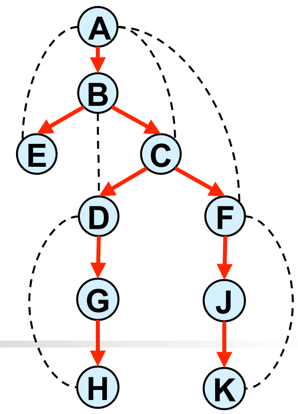
From Search Trees to Search Graphs

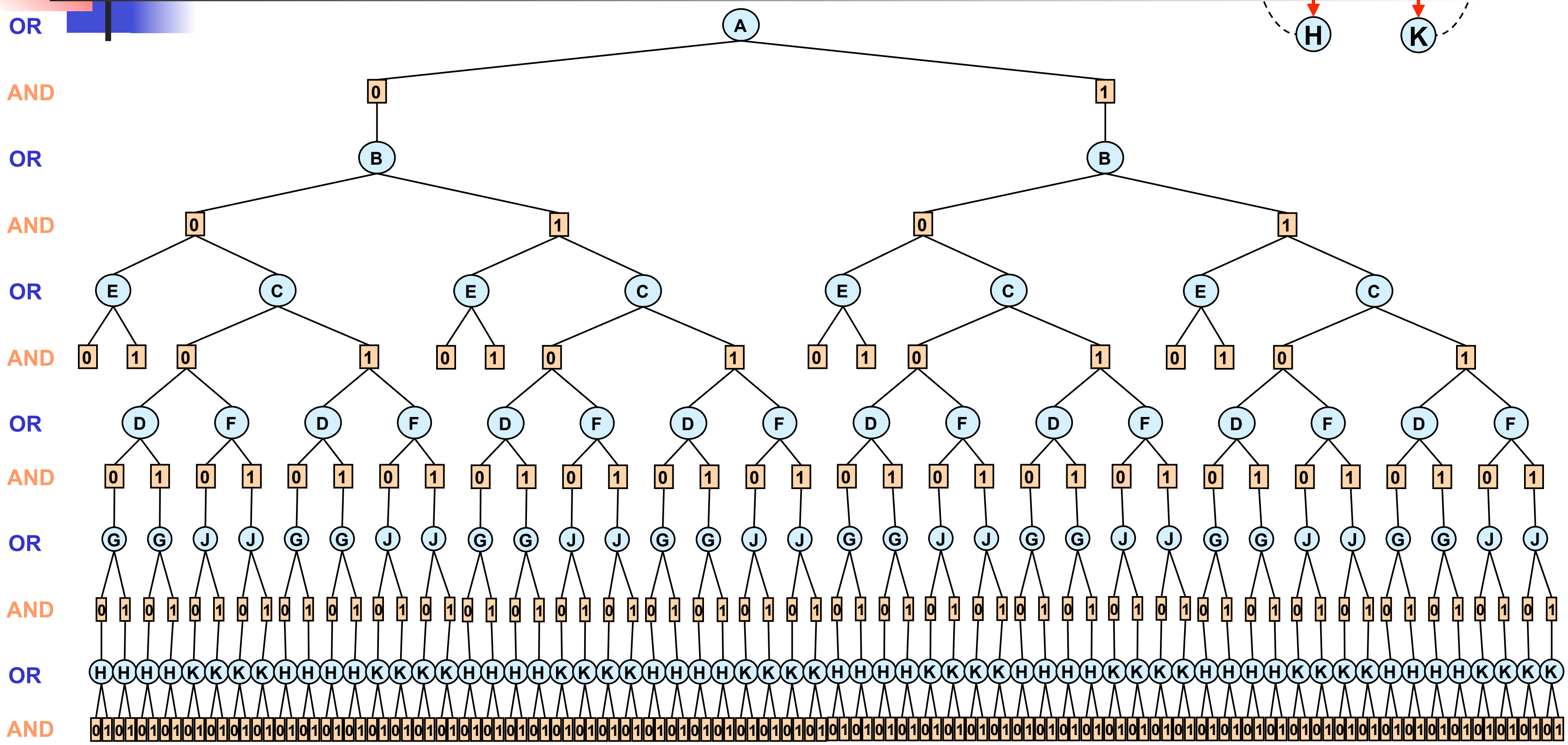
- Any two nodes that root identical subtrees (subgraphs) can be merged



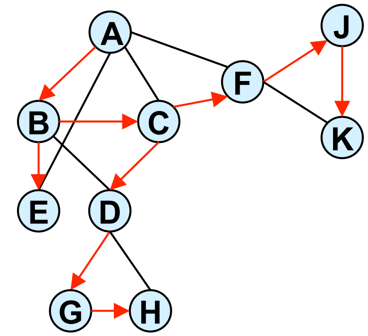
AND/OR Tree

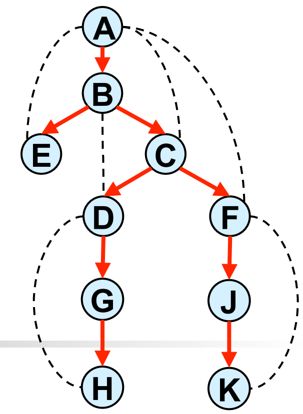

A
B
C
D
E
F
G
H
J
K

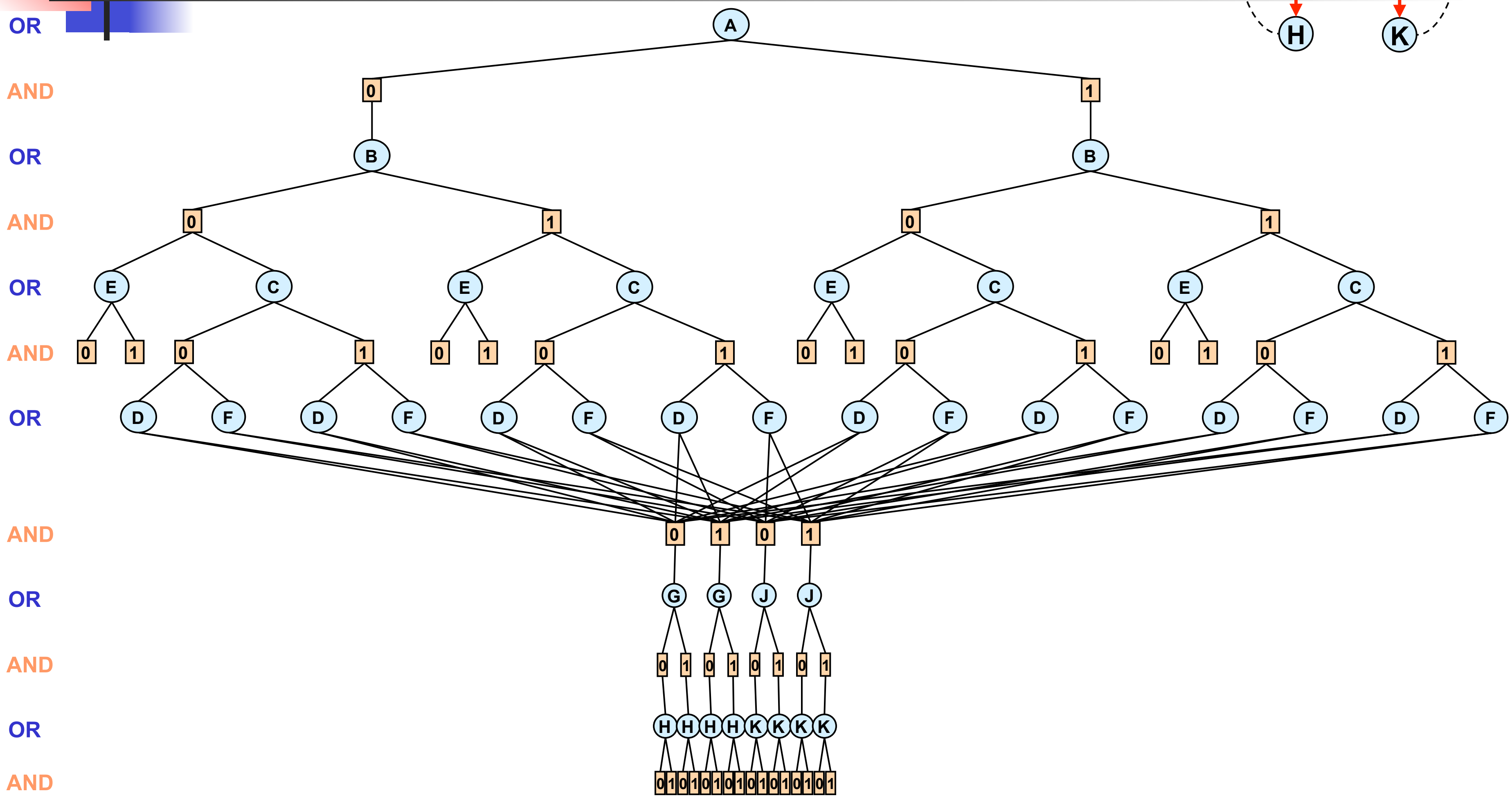

A
B
C
D
E
F
G
H
J
K


OR
AND
OR
AND
OR
AND
OR
AND
OR
AND
OR
AND

An AND/OR graph



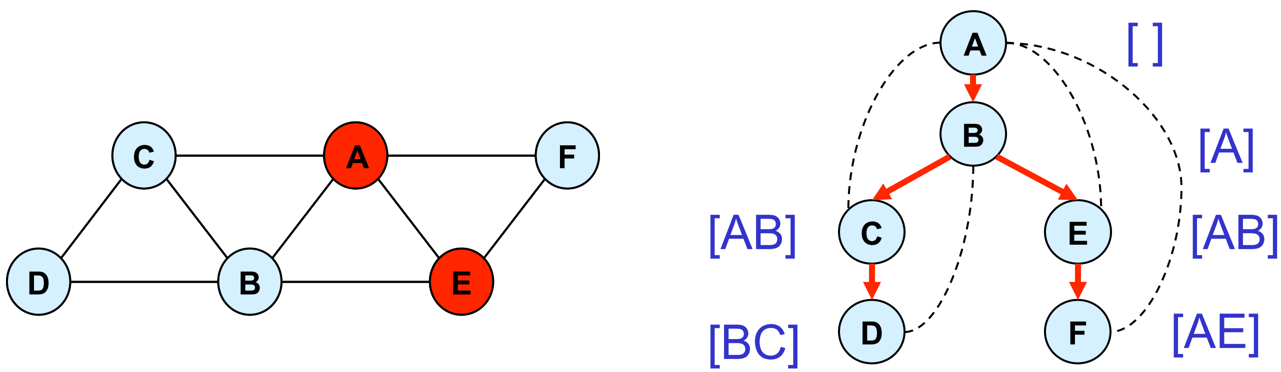




Merging Based on Context

One way of recognizing nodes that can be merged:

context (X) = ancestors of X in pseudo tree that are connected to X, or to descendants of X



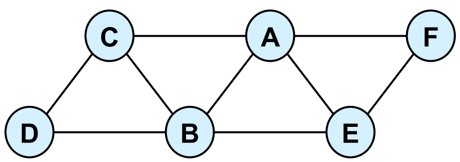
pseudo tree

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 Search Graph

Constraint Satisfaction – Counting Solutions

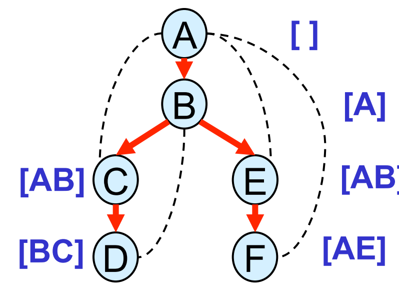


A	B	C	R_{ABC}
0	0	0	**1**
0	0	1	**1**
0	1	0	**0**
0	1	1	**1**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**

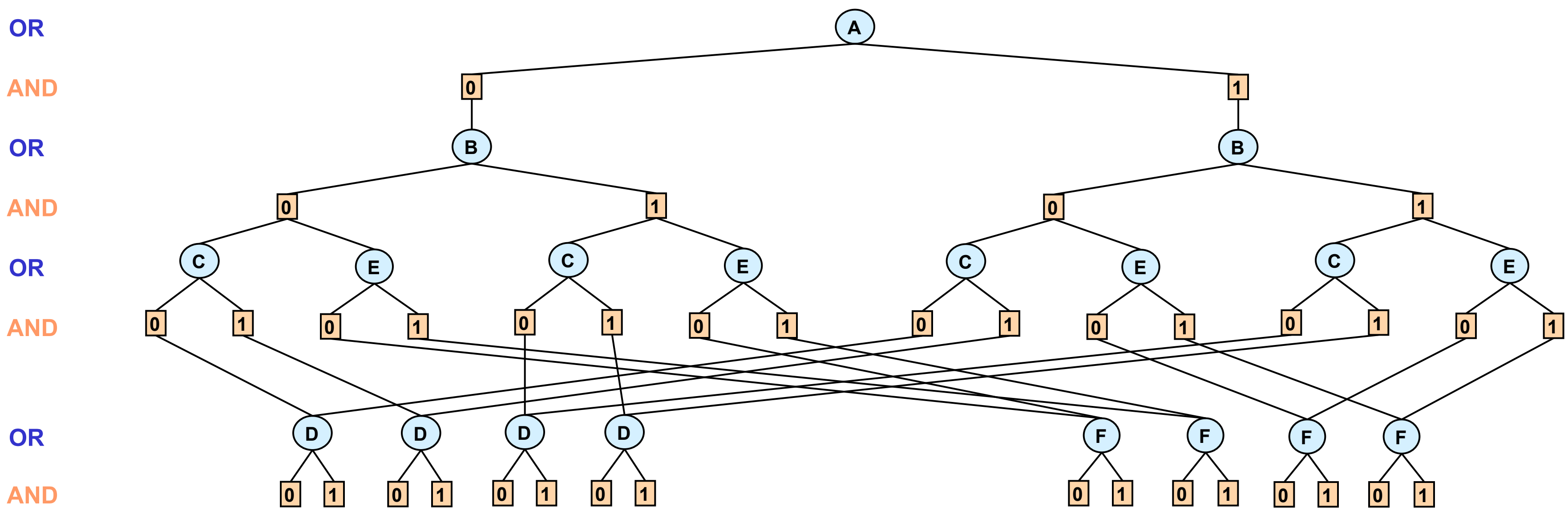
B	C	D	R_{BCD}
0	0	0	**1**
0	0	1	**1**
0	1	0	**1**
0	1	1	**0**
1	0	0	**1**
1	0	1	**0**
1	1	0	**1**
1	1	1	**1**

A	B	E	R_{ABE}
0	0	0	**1**
0	0	1	**0**
0	1	0	**1**
0	1	1	**1**
1	0	0	**0**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**

A	E	F	R_{AEF}
0	0	0	**0**
0	0	1	**1**
0	1	0	**1**
0	1	1	**1**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**



pseudo tree



context minimal graph

AND/OR Tree DFS Algorithm (Belief Updating)

$P(E \mid A,B)$

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

$P(B \mid A)$

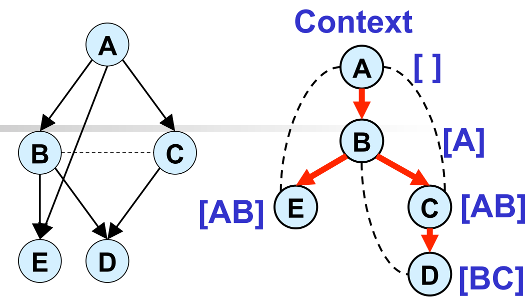
A	B=0	B=1
0	.4	.6
1	.1	.9

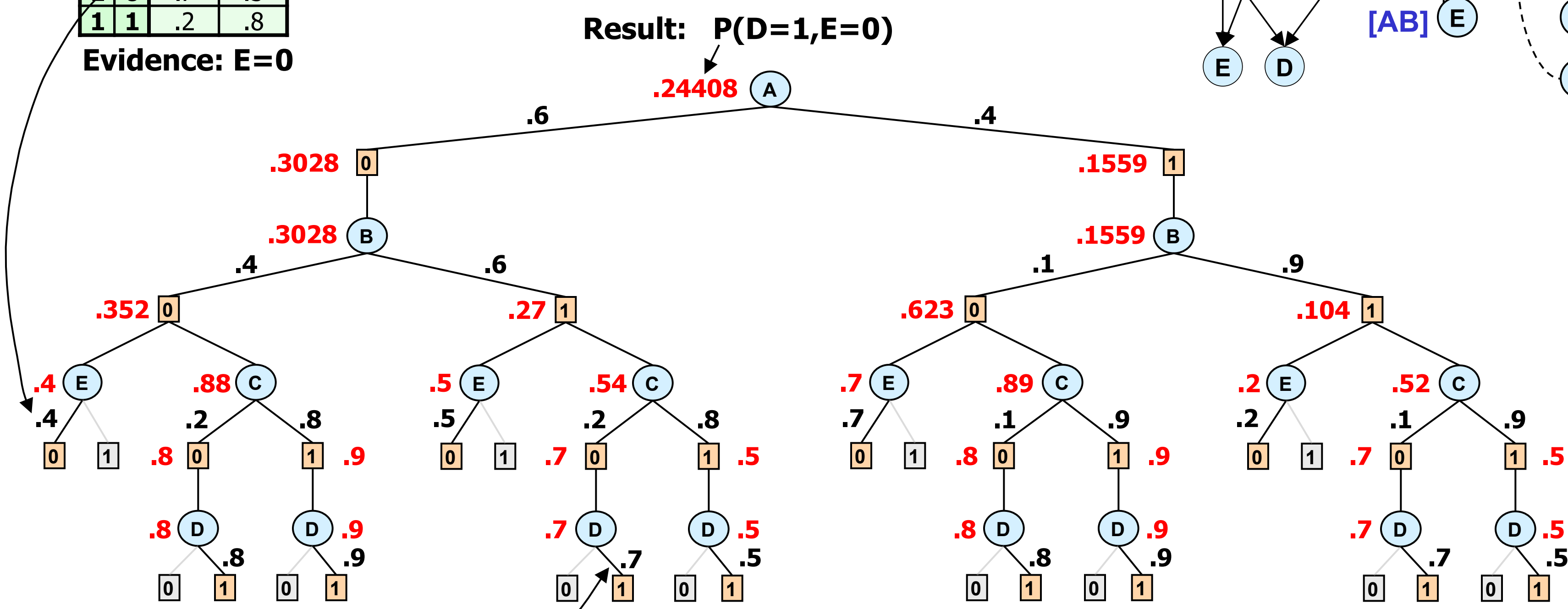
$P(C \mid A)$

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

$P(A)$

A	P(A)
0	.6
1	.4





$P(D \mid B,C)$

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)

$P(E \mid A,B)$

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

$P(B \mid A)$

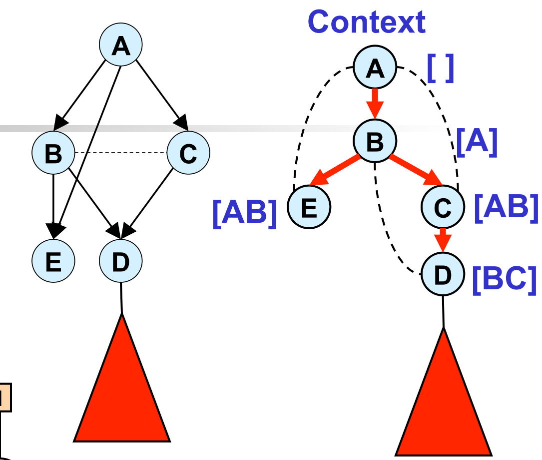
A	B=0	B=1
0	.4	.6
1	.1	.9

$P(C \mid A)$

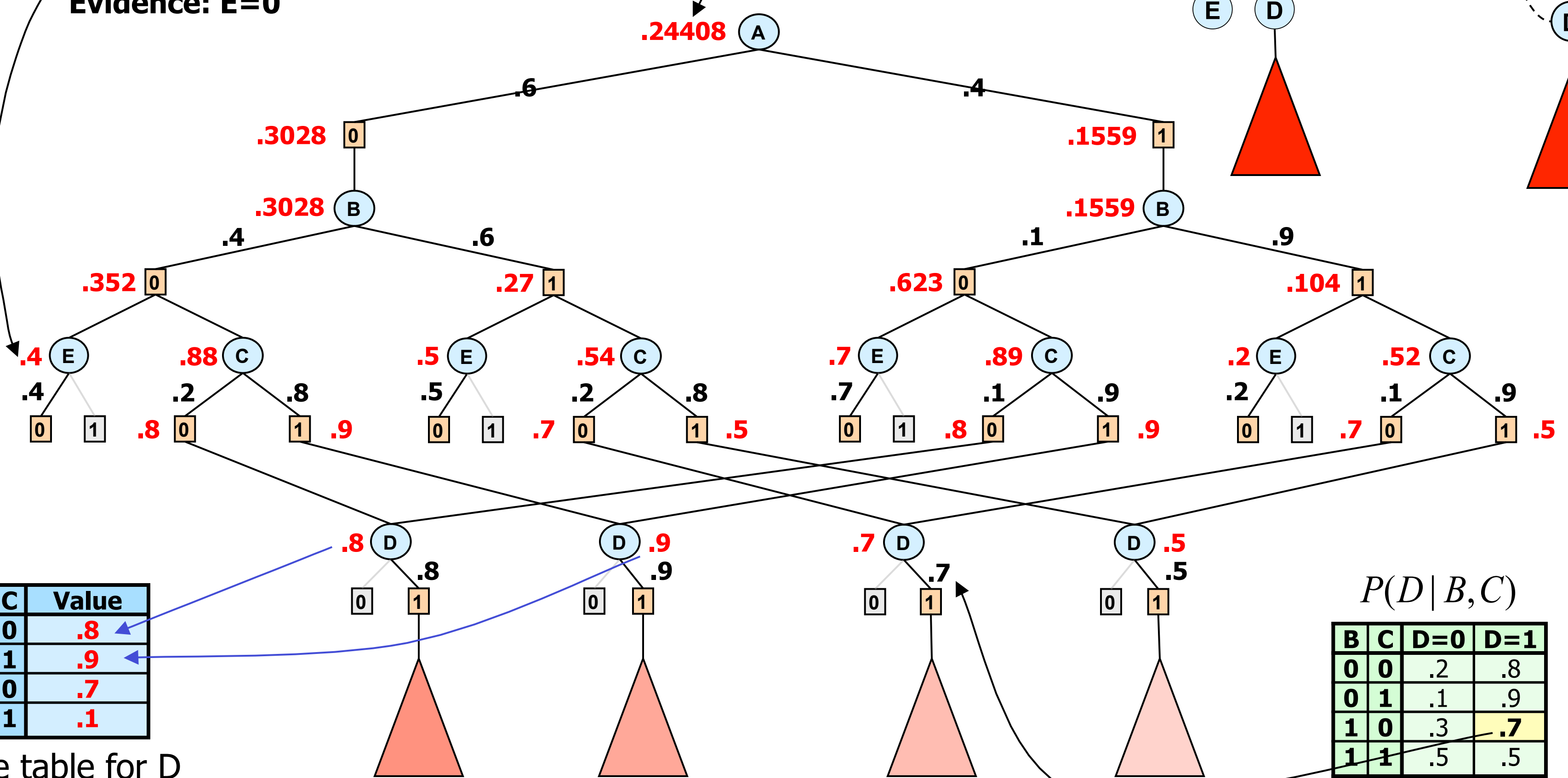
A	C=0	C=1
0	.2	.8
1	.7	.3

$P(A)$

A	P(A)
0	.6
1	.4



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



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

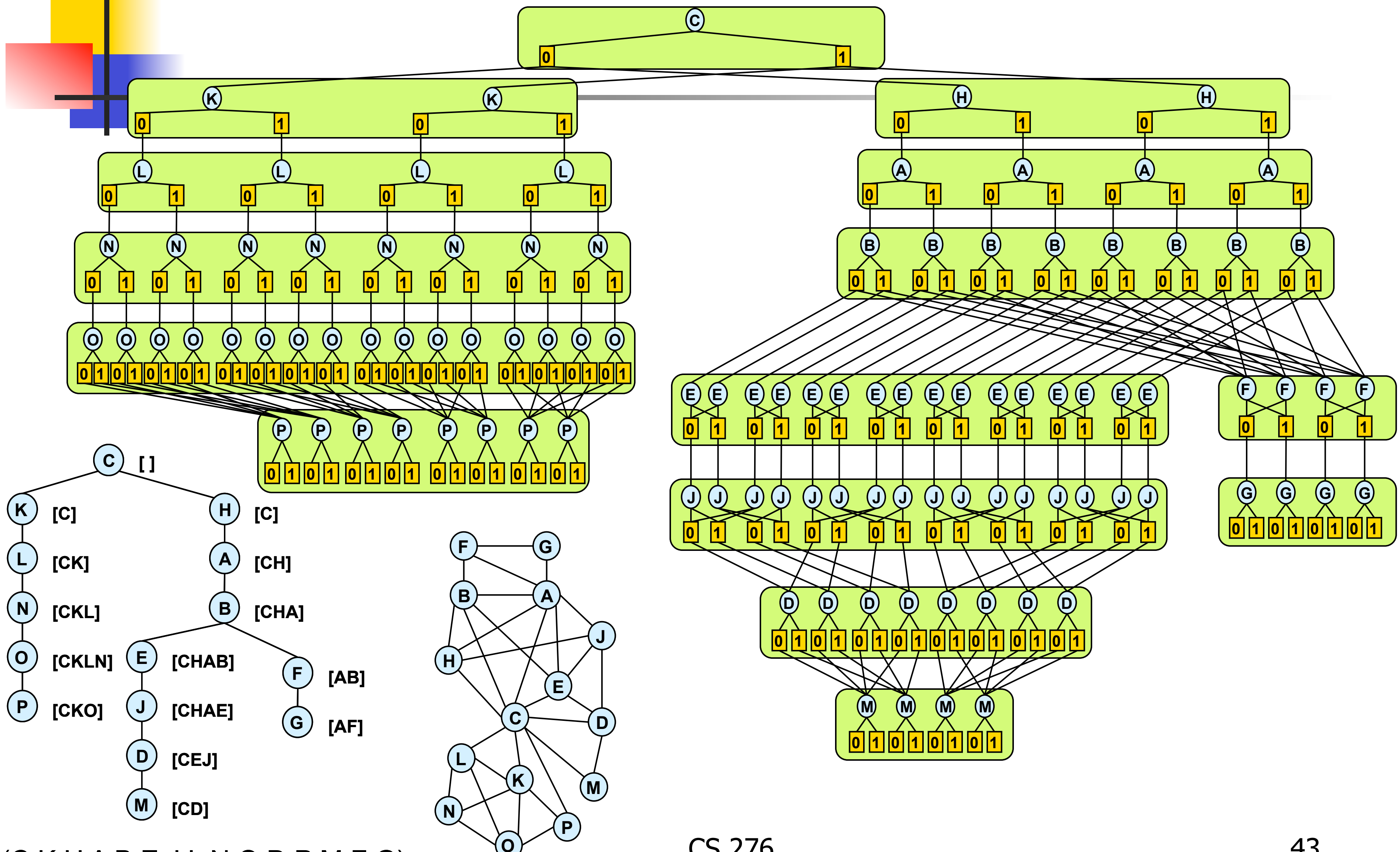
Cache table for D

$P(D \mid B,C)$

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

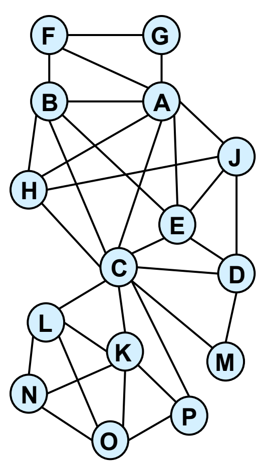
AND/OR context minimal graph

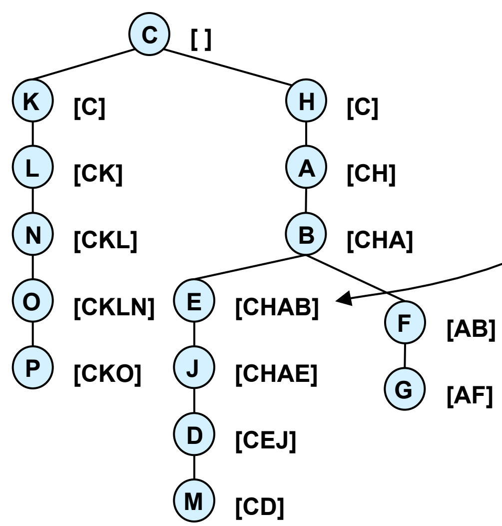


(C K H A B E J L N O D P M F G)

How Big Is the Context?

Theorem: *The maximum context size for a pseudo tree is equal to the treewidth of the graph along the pseudo tree.*

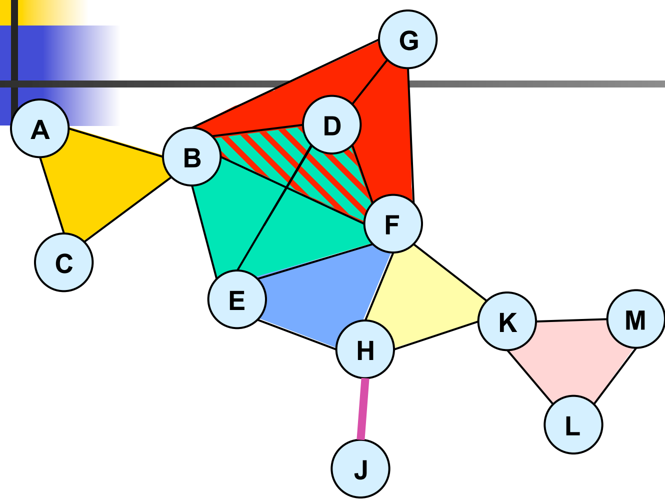


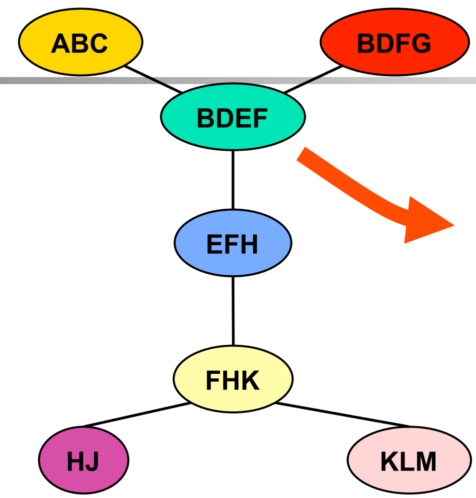


max context size = treewidth

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

Treewidth vs. Pathwidth

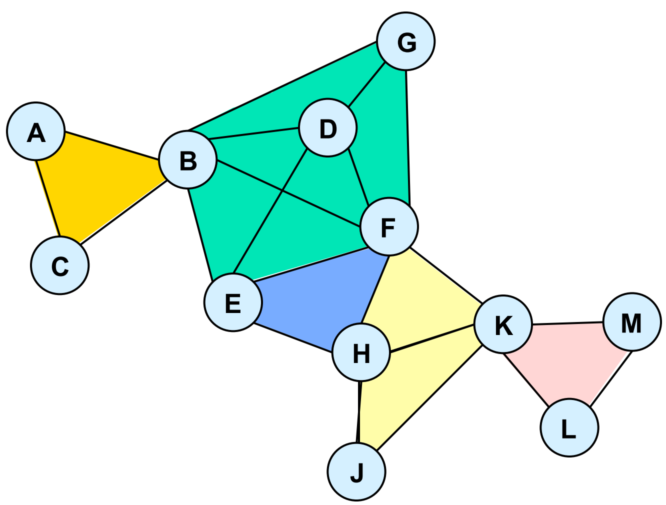


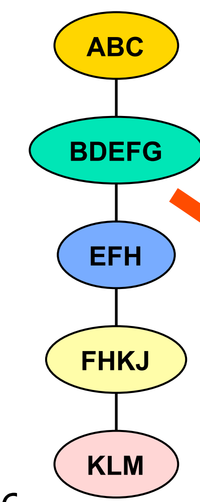


TREE

***treewidth* = 3**

= (max cluster size) - 1





CHAIN

***pathwidth* = 4**

= (max cluster size) - 1

Tasks 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 othewise.
  - **Optimization: v(n) is the optimal solution in T(n)**
  - **Belief updating: v(n), probability of evidence in T(n).**
  - **Partition function: v(n) is the total probability in T(n).**
- **Theorem: Complexity of AO dfs search tree is**
  - **Space: O(n)**
  - **Time: $O(n\ k^m)$**
  - **Time: O(exp(w* log n))**
- **Theorem: Complexity of AO dfs search tree is**
  - **Space: $O(n\ k^{w*})$**
  - **Time: $O(n\ k^{w*})$**
- We can have hybrids trading space for time

Complexity of AND/OR Graph Search

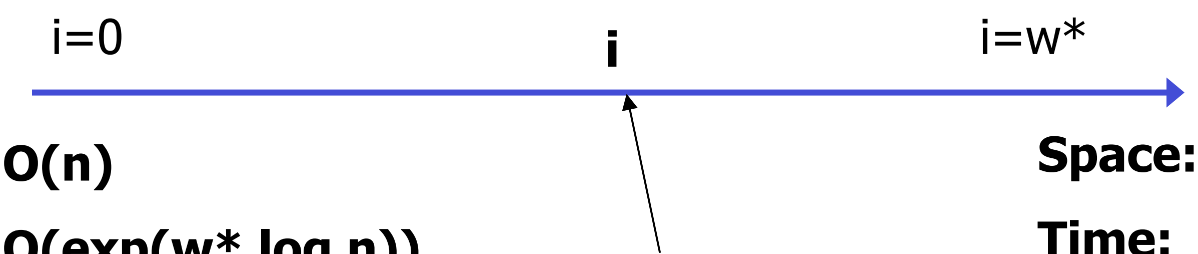
	AND/OR graph	OR graph
Space	$O(n\ k^{w^*})$	$O(n\ k^{pw^*})$
Time	$O(n\ k^{w^*})$	$O(n\ k^{pw^*})$

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

$$w^* \le pw^* \le w^* \log n$$

Searching AND/OR Graphs

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



Space: O(n)

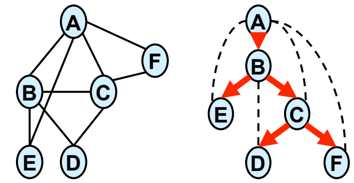
Time: O(exp(w* log n))

Space: O(exp(i))

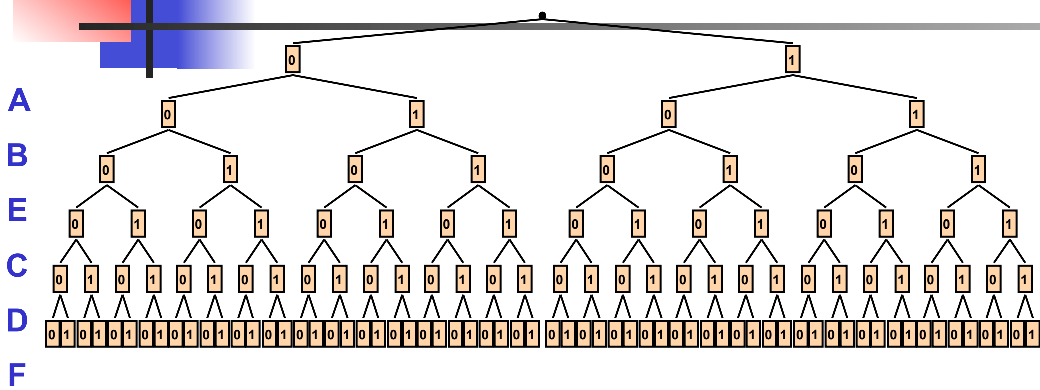
Time: O(exp(m_i+i)

Space: O(exp w*)

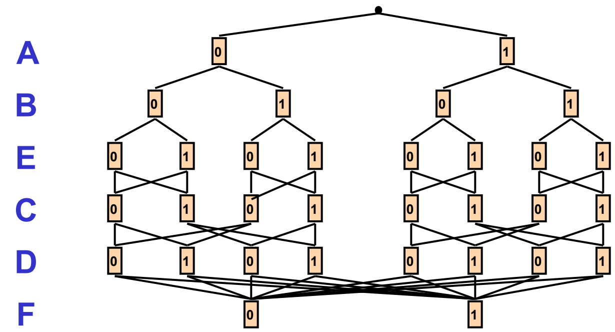
Time: O(exp w*)



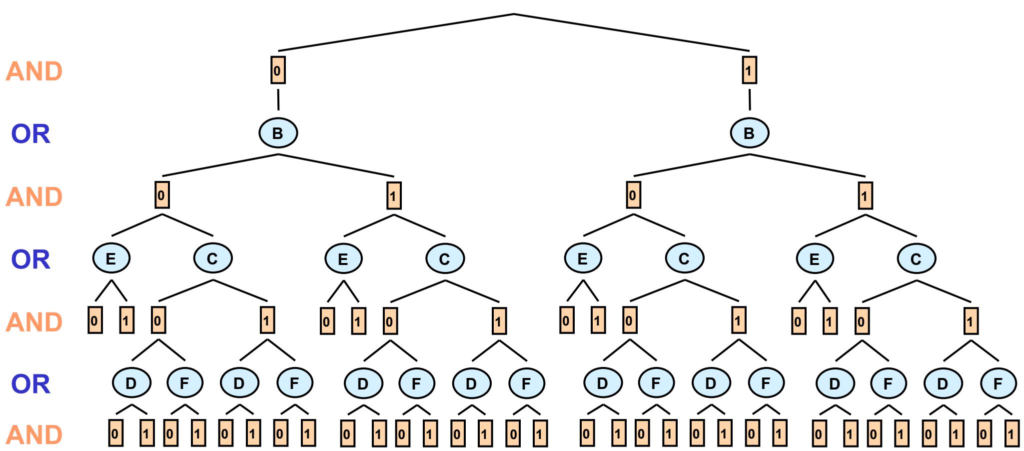
All four search spaces



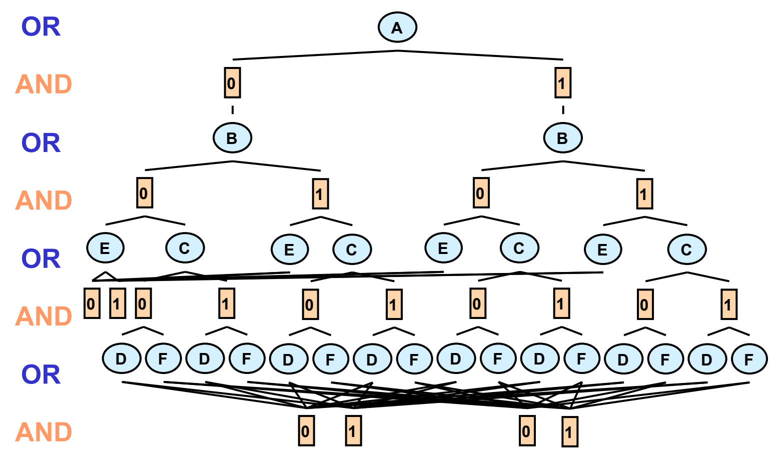
Full OR search tree



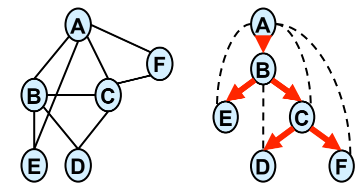
Context minimal OR search graph



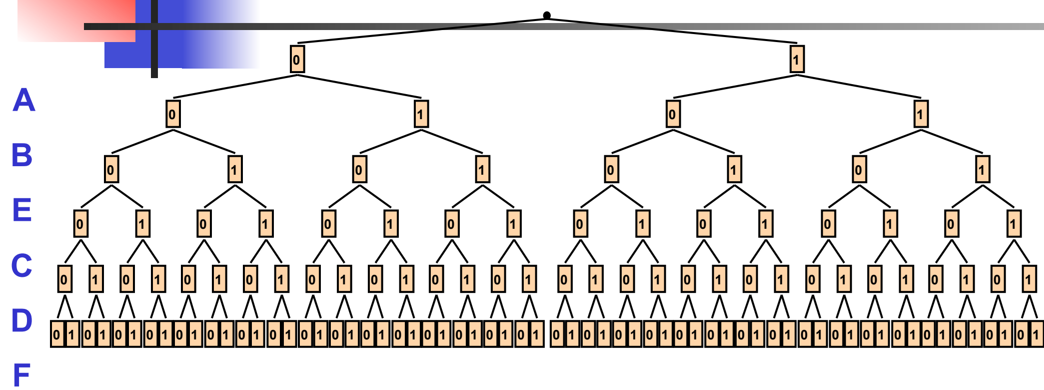
Full AND/OR search tree



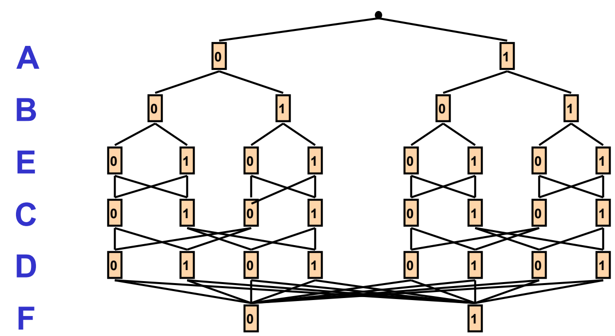
Context minimal AND/OR search graph



All four search spaces

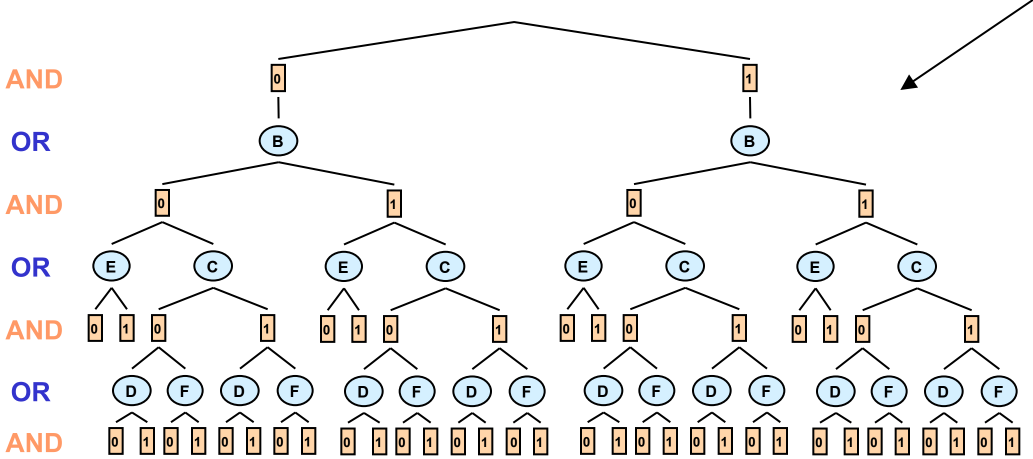


Full OR search tree

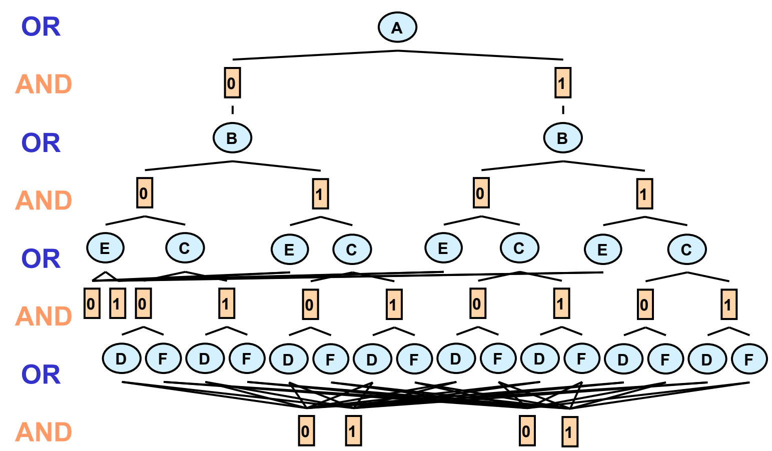


Context minimal OR search graph

Time-space

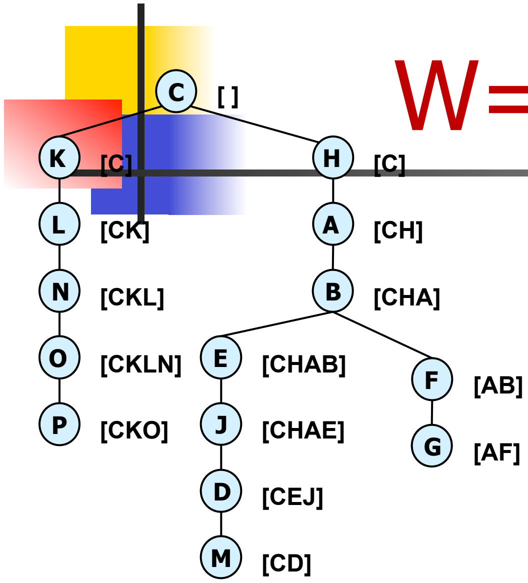


Full AND/OR search tree



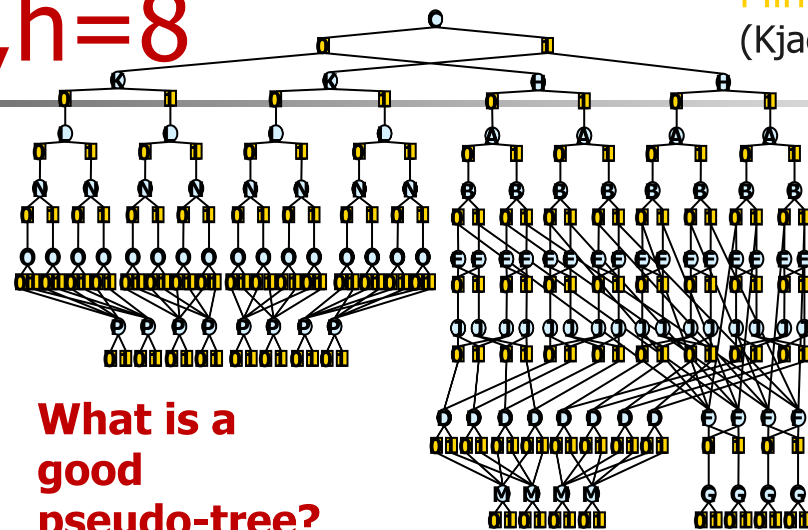
Context minimal AND/OR search graph

The impact of the pseudo-tree

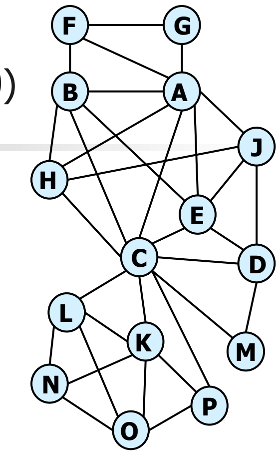


(C K H A B E J L N O D P M F G)

W=4,h=8

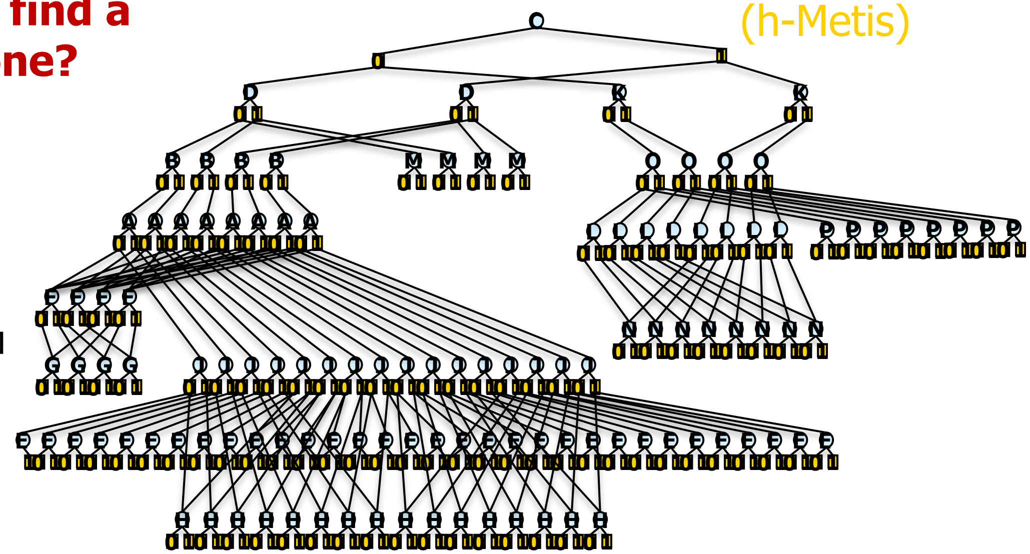


Min-Fill (Kjaerulff90)

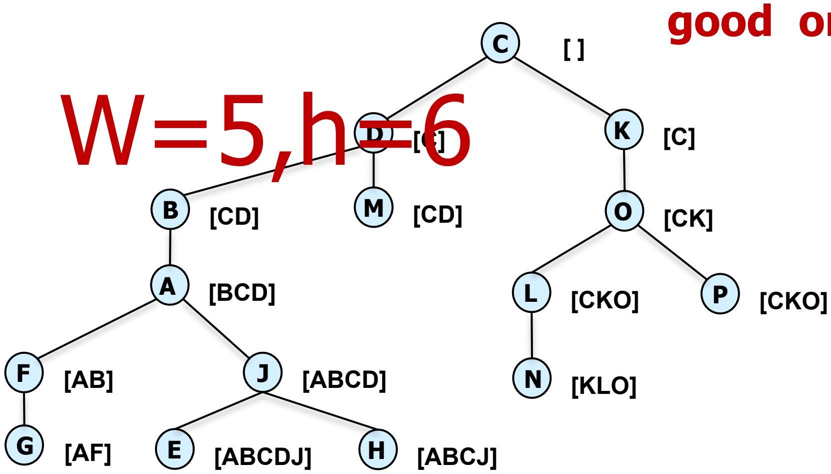


What is a good pseudo-tree? How to find a good one?

Hypergraph Partitioning (h-Metis)



W=5,h=6



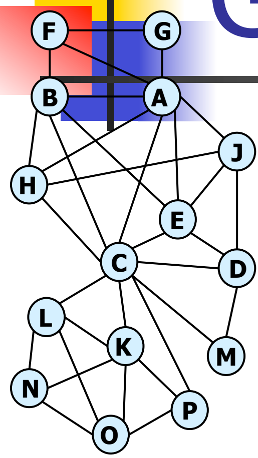
(C D K B A O M L N P J H E F G)

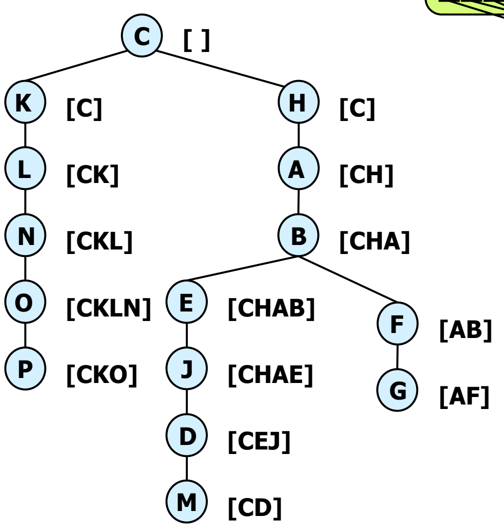
Dead Caches

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

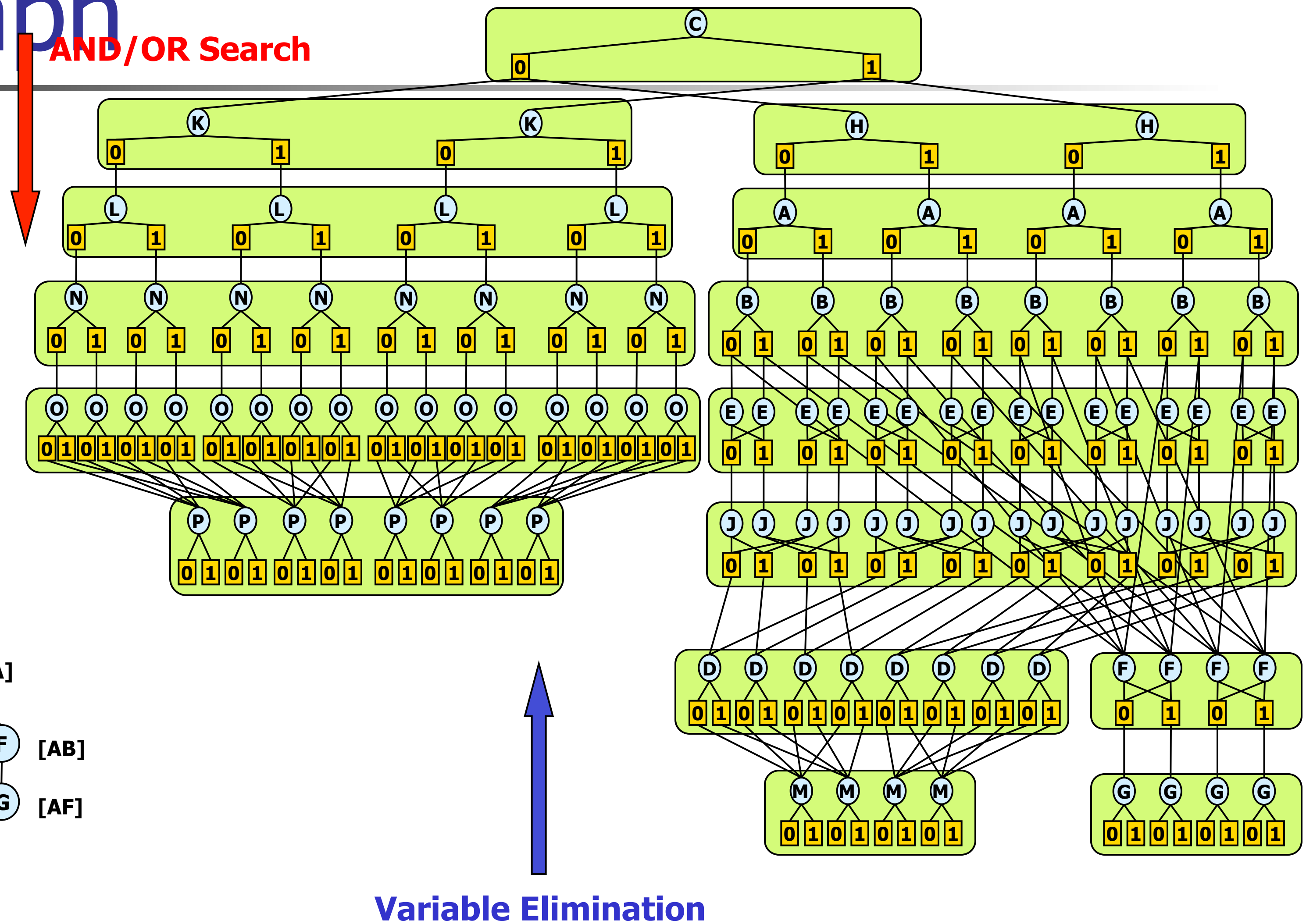
Example 8.1.10 Consider the graphical models and the pseudo-tree in Figure 7.13. The context in the left branch (C, CK, CKL, $CKLN$) are all dead-caches. The only one which is not is CKO of P. As you can see, there are converging arcs into P only along this branch. Indeed if we describe the clusters of the corresponding bucket-tree. we would have just two maximal clusters: $CKLNO$ and $PCKO$ whose separator is CKO, the context of P. □

AND/OR Context Minimal Graph





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



Available code

- http://graphmod.ics.uci.edu/group/Software

Algorithm 2: AO-COUNTING / AO-BELIEF-UPDATING

Input: A constraint network $\mathcal{M} = \langle X, D, C \rangle$, or a belief network $\mathcal{P} = \langle X, D, P \rangle$; a pseudo tree $\mathcal{T}$ rooted at X_1; parents pa_i (OR-context) for every variable X_i; `caching` set to *true* or *false*.
Output: The number of solutions, or the updated belief, $v(X_1)$.

if `caching` == *true* **then** // Initialize cache tables
    Initialize cache tables with entries of "-1"
$v(X_1) \leftarrow 0$; `OPEN` $\leftarrow \{X_1\}$ // Initialize the stack OPEN
while `OPEN` $\neq \varphi$ **do**
    `n` $\leftarrow top($`OPEN`$)$; remove `n` from `OPEN`
    if `caching` == *true* **and** `n` *is OR, labeled* X_i **and** $Cache(asgn(\pi_n)[pa_i]) \neq -1$ **then** // In cache
        $v(\mathtt{n}) \leftarrow Cache(asgn(\pi_n)[pa_i])$ // Retrieve value
        $successors(\mathtt{n}) \leftarrow \varphi$ // No need to expand below
    else // **EXPAND**
        if `n` *is an OR node labeled* X_i **then** // OR-expand
            $successors(\mathtt{n}) \leftarrow \{\langle X_i, x_i \rangle \mid \langle X_i, x_i \rangle$ is consistent with $\pi_n \}$
            $v(\langle X_i, x_i \rangle) \leftarrow 1$, for all $\langle X_i, x_i \rangle \in successors(\mathtt{n})$
            $v(\langle X_i, x_i \rangle) \leftarrow \prod_{f \in B_{\mathcal{T}}(X_i)} f(asgn(\pi_n)[pa_i])$, for all $\langle X_i, x_i \rangle \in successors(\mathtt{n})$ // AO-BU
        if `n` *is an AND node labeled* $\langle X_i, x_i \rangle$ **then** // AND-expand
            $successors(\mathtt{n}) \leftarrow children_{\mathcal{T}}(X_i)$
            $v(X_i) \leftarrow 0$ for all $X_i \in successors(\mathtt{n})$
        Add $successors(\mathtt{n})$ to top of `OPEN`
    while $successors(\mathtt{n}) == \varphi$ **do** // **PROPAGATE**
        if `n` *is an OR node labeled* X_i **then**
            if $X_i == X_1$ **then** // Search is complete
                return $v(\mathtt{n})$
            if `caching` == *true* **then**
                $Cache(asgn(\pi_n)[pa_i]) \leftarrow v(\mathtt{n})$ // Save in cache
            $v(\mathtt{p}) \leftarrow v(\mathtt{p}) * v(\mathtt{c})$
            if $v(\mathtt{p}) == 0$ **then** // Check if p is dead-end
                remove $successors(\mathtt{p})$ from `OPEN`
                $successors(\mathtt{p}) \leftarrow \varphi$
        if `n` *is an AND node labeled* $\langle X_i, x_i \rangle$ **then**
            let `p` be the parent of `n`
            $v(\mathtt{p}) \leftarrow v(\mathtt{p}) + v(\mathtt{n})$;
        remove `n` from $successors(\mathtt{p})$
        `n` $\leftarrow$ `p`

The recursive value rule

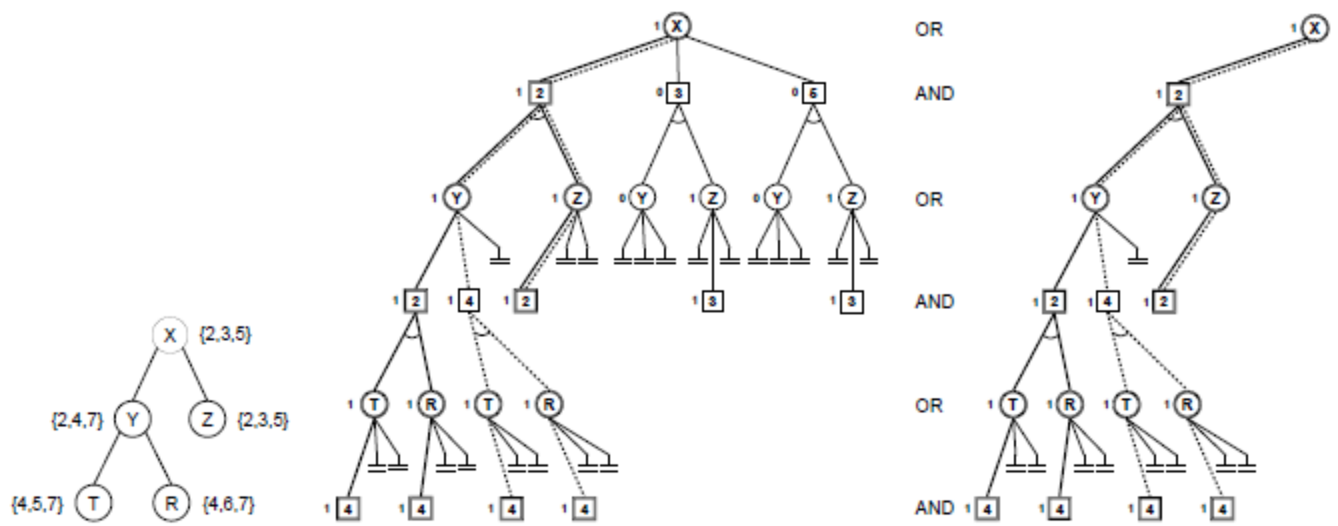
$$v(n) = \bigotimes_{n' \in children(n)} v(n'), \qquad \textit{if } n = \langle X, x \rangle \textit{ is an AND node,}$$
$$v(n) = \Downarrow_{n' \in children(n)} (w_{(n,n')} \bigotimes v(n')), \qquad \textit{if } n = X \textit{ is an OR node.}$$

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.*

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

AND/OR backtrack-free



(a) A constraint tree (b) Search tree (c) Backtrack-free search tree

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

AND/OR CPE (constraint probability evaluation)

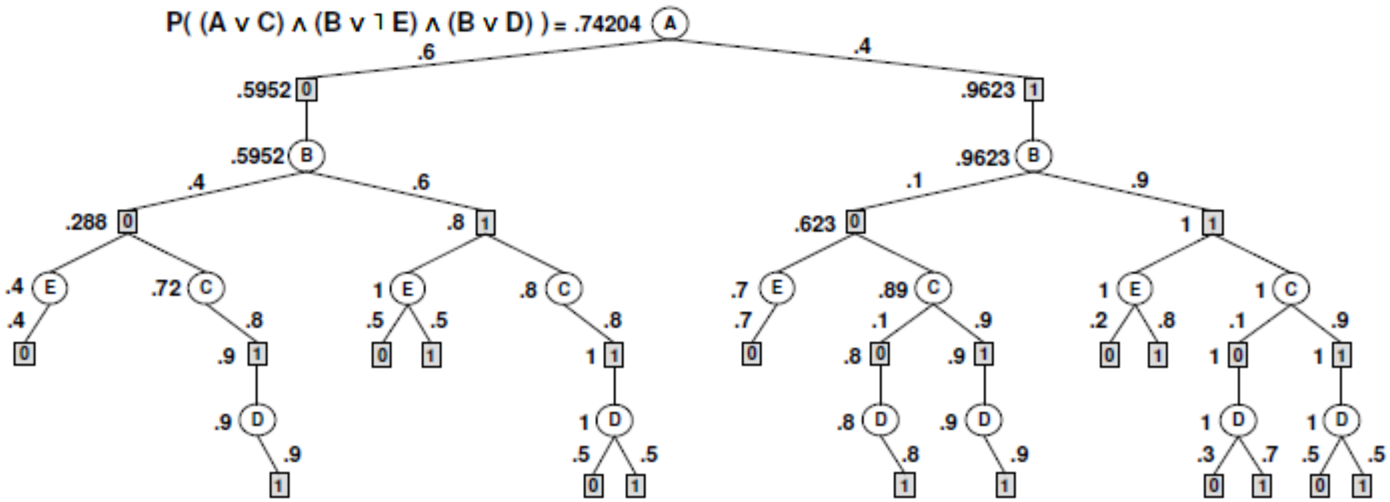
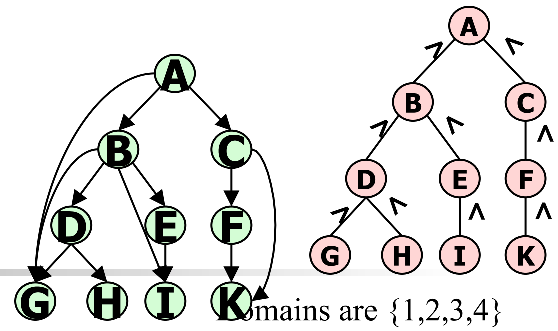
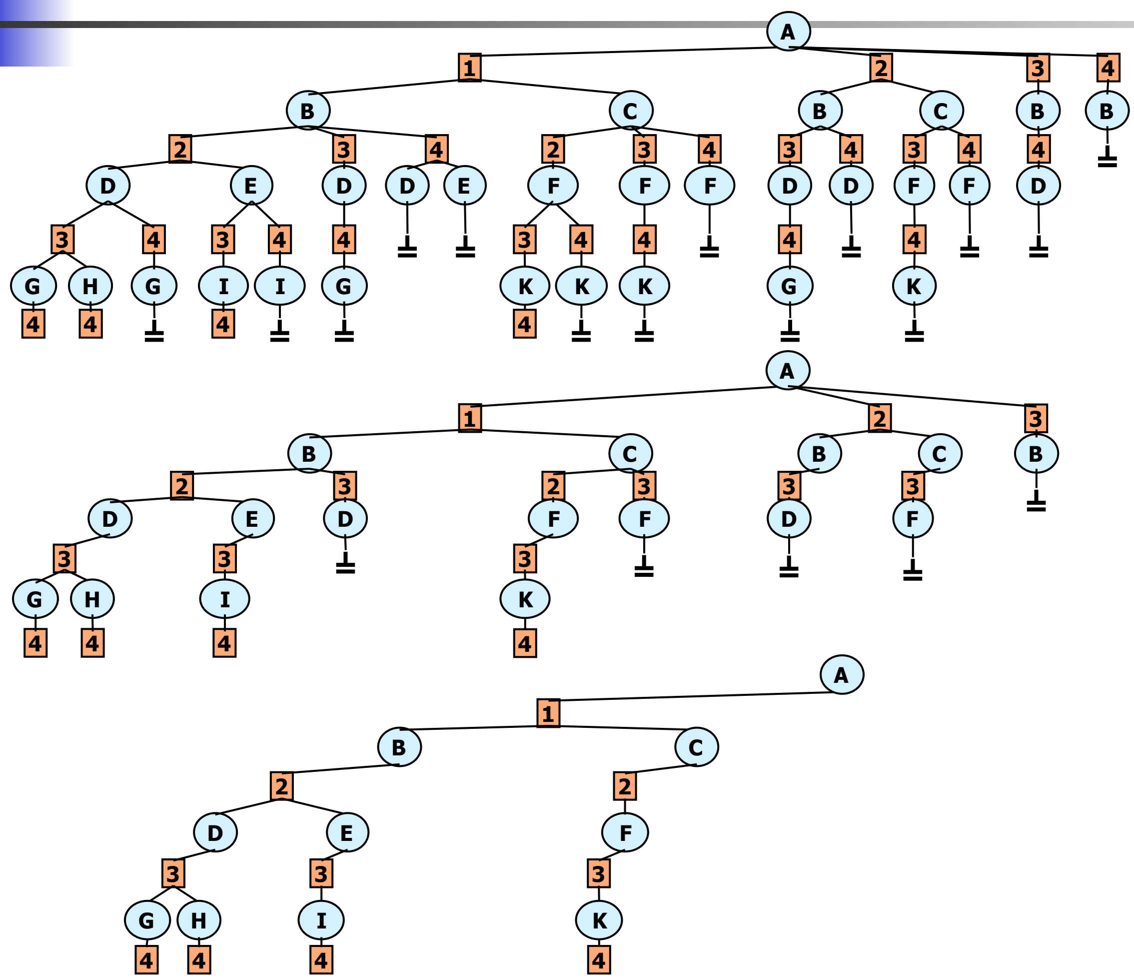


Figure 8.2: Mixed network defined by the query $\varphi = (A \vee C) \wedge (B \vee \neg E) \wedge (B \vee D)$

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 $D \wedge \neg E$. □

The Effect of Constraint Propagation in AND/OR CPE





CONSTRAINTS ONLY

FORWARD CHECKING

MAINTAINING ARC CONSISTENCY

Search for MPE/MAP problem

- Searching the AND/OR space by
  - Branch and bound
  - Best-first

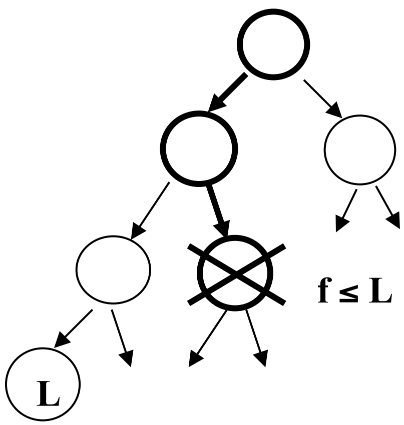
Searching the AND/OR space for MPE/ MAP

Heuristic function **$f(x^p)$** computes a lower bound on the best extension of **x^p** and can be used to guide a heuristic search algorithm. We focus on:

1. DF Branch-and-Bound

Use heuristic function **$f(x^p)$** to prune the depth-first search tree

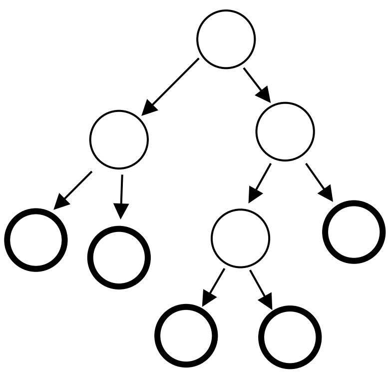
Linear space



2. Best-First Search

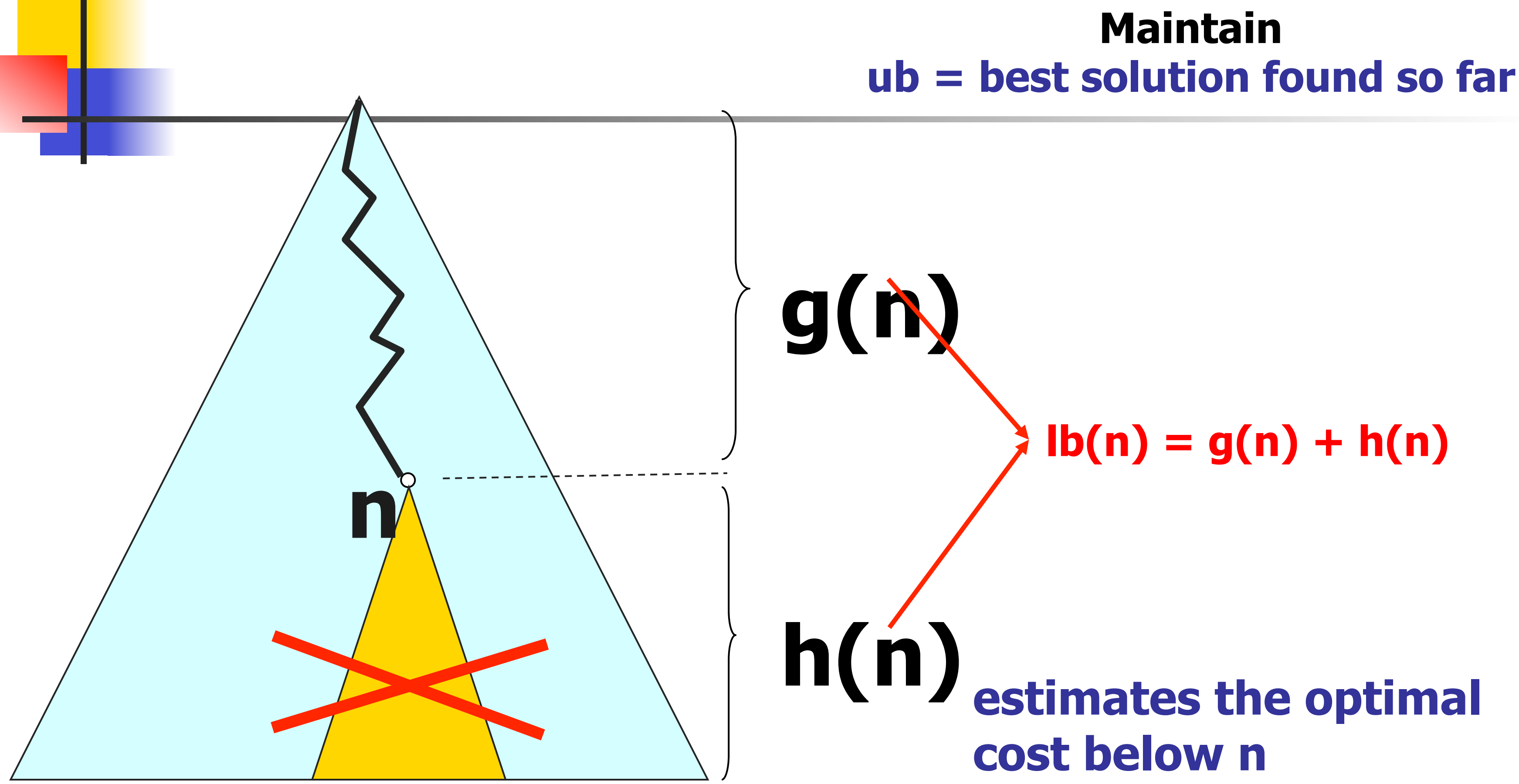
Always expand the node with the highest heuristic value **$f(x^p)$**

Needs lots of memory



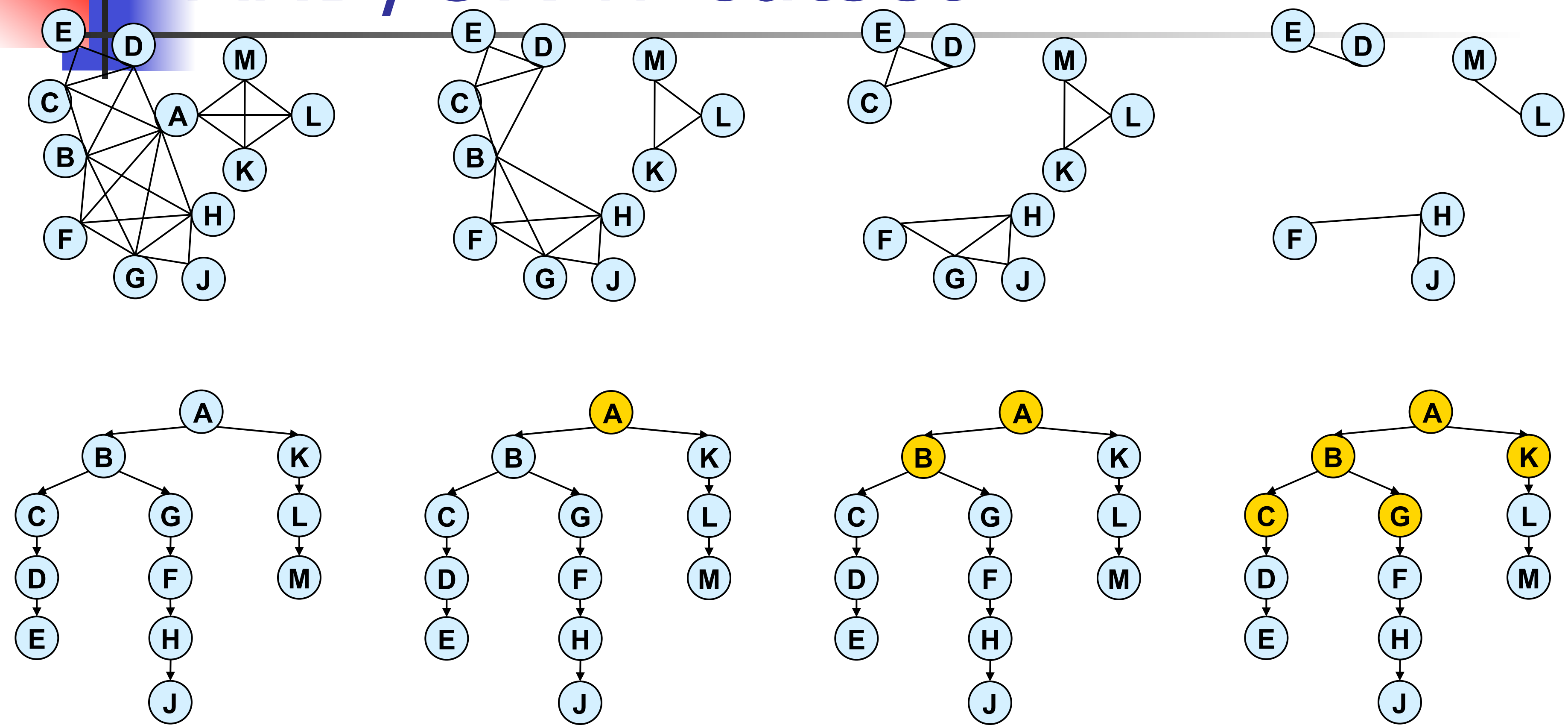
AND/OR Branch-and-Bound (AOBB)

(Marinescu & Dechter, IJCAI'05)



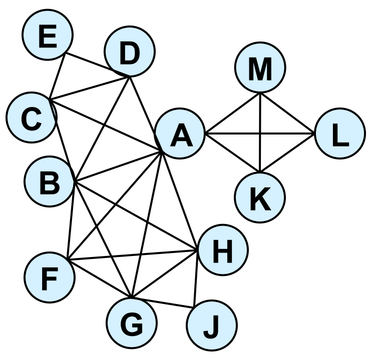
Prune subtree below n if lb(n) ≥ ub

AND/OR w-cutset

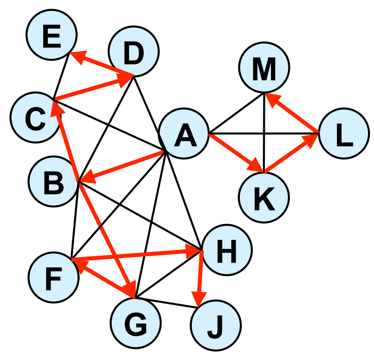


3-cutset 2-cutset 1-cutset

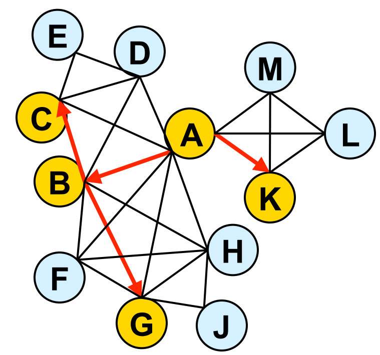
AND/OR w-cutset



grahpical model



pseudo tree



1-cutset tree

w-cutset Trees Over AND/OR space

- Definition:
  - T_w is a w-cutset tree relative to backbone tree T, iff T_w is roots T and when removed, yields tree-width w.

- Theorem:
  - AO(i) time complexity for pseudo-tree T is time O(exp(i+m_i)) and space O(i), m_i is the depth of the T_i tree.

- Better than w-cutset: O(exp(i+c_i)) when c_i is the number of nodes in T_i