Exploiting Tree-Decomposition in Search: The AND/OR Paradigm

Rina Dechter

Bren School of ICS

University of California, Irvine

Collaborators:

Kalev Kask
Radu Marinescu
Robert Mateescu
Vibhav Gogate

Outline

- Background: Search and inference in Graphical models
- AND/OR search trees
  - **Pseudo spanning trees, dfs trees**
- AND/OR search graphs
  - Tree-width and path-width bounds
- Relationship to algorithmic and compilation schemes.

Constraint networks

A *constraint network* is a triple

$R = \langle X, D, C \rangle$ where:

$X = \{X_1, ..., X_n\}$ is a set of variables

$D = \{D_1, ..., D_n\}$ is the set of their domains

$C = \{C_1, ..., C_t\}, C_i = (S_i, R_i)$ are the constraints,

S_i being the *scope* of the *relation* R_i.

The main task:

- Determine if the problem has a solution (an assignment that satisfies all the relations);
- If yes, find one or all of them.

R_1

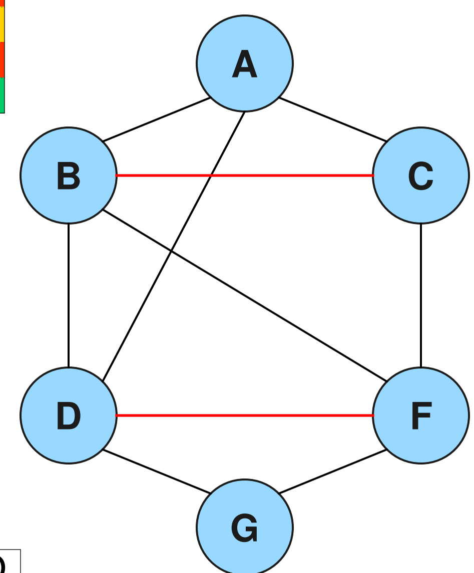
A
1
2
3

R_2

A	B
1	2
1	3
2	1
2	3
3	1
3	2

R_3

A	C
1	2
3	2



R_4

A	B	D
1	2	3
1	3	2
2	1	3
2	3	1
3	1	2
3	2	1

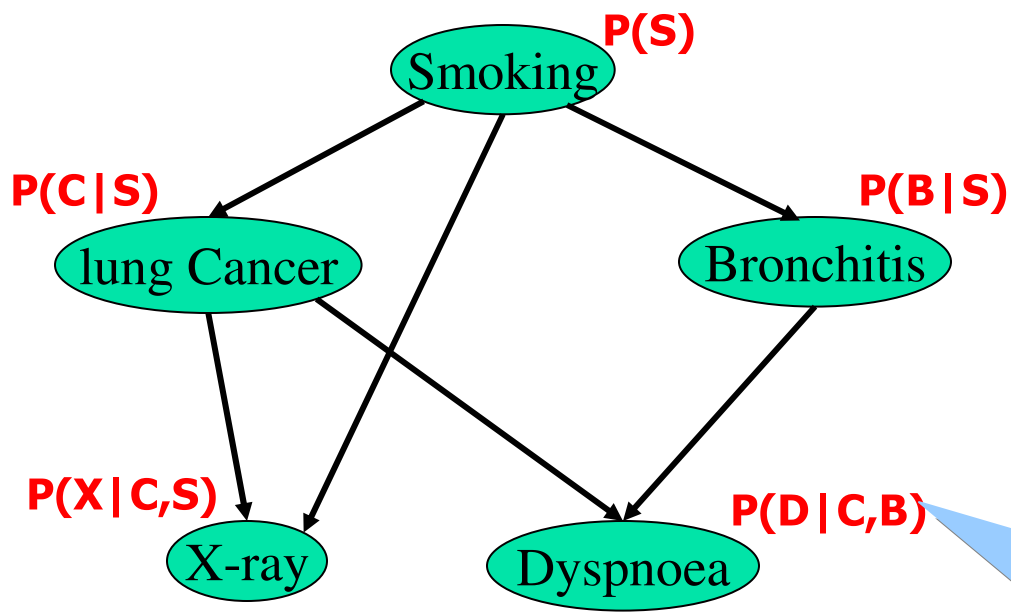
R_5

B	C	F
1	2	3
3	2	1

R_6

D	F	G
1	2	3
2	1	3

Probabilistic Networks



$\mathbf{BN = (G, \Theta)}$

CPD:

C	B	D=0	D=1
0	0	0.1	0.9
0	1	0.7	0.3
1	0	0.8	0.2
1	1	0.9	0.1

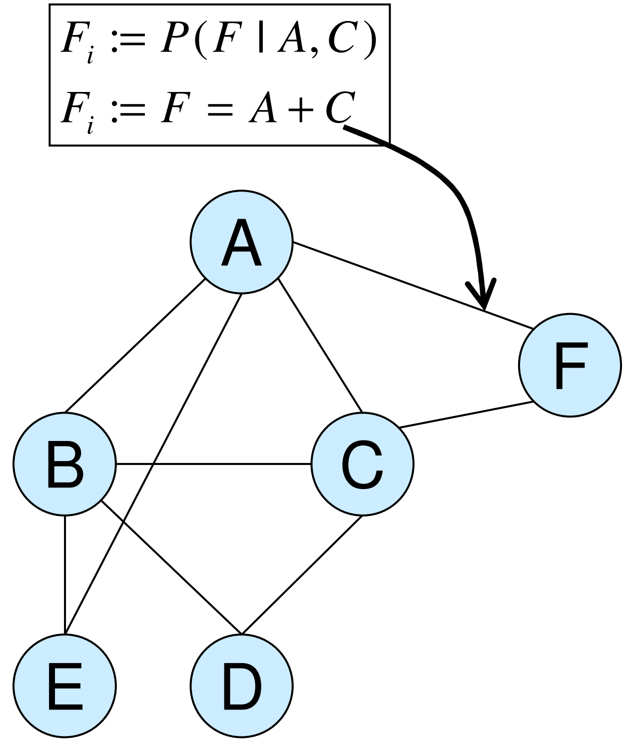
$$P(S, C, B, X, D) = P(S)\, P(C|S)\, P(B|S)\, P(X|C,S)\, P(D|C,B)$$

P(S|d)= ?

MPE: argmax $P(S)\, P(C|S)\, P(B|S)\, P(X|C,S)\, P(D|C,B)$

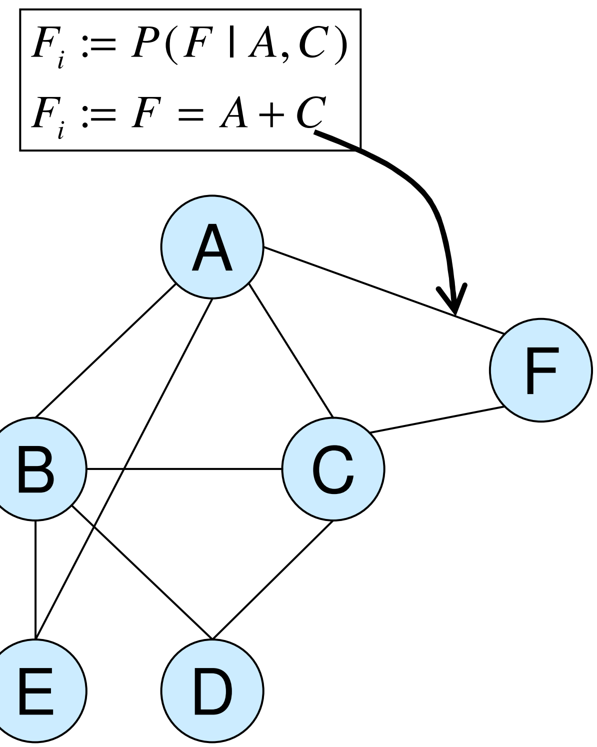
Graphical models

- A graphical model (X,D,C):
  - $X = \{X_1, \ldots X_n\}$ variables
  - $D = \{D_1, \ldots D_n\}$ domains
  - $C = \{F_1, \ldots, F_t\}$ functions (constraints, CPTS, cnfs)



Graphical models

- A graphical model (X,D,C):
  - $X = \{X_1,\ldots X_n\}$ variables
  - $D = \{D_1, \ldots D_n\}$ domains
  - $C = \{F_1,\ldots,F_t\}$ functions (constraints, CPTS, cnfs)

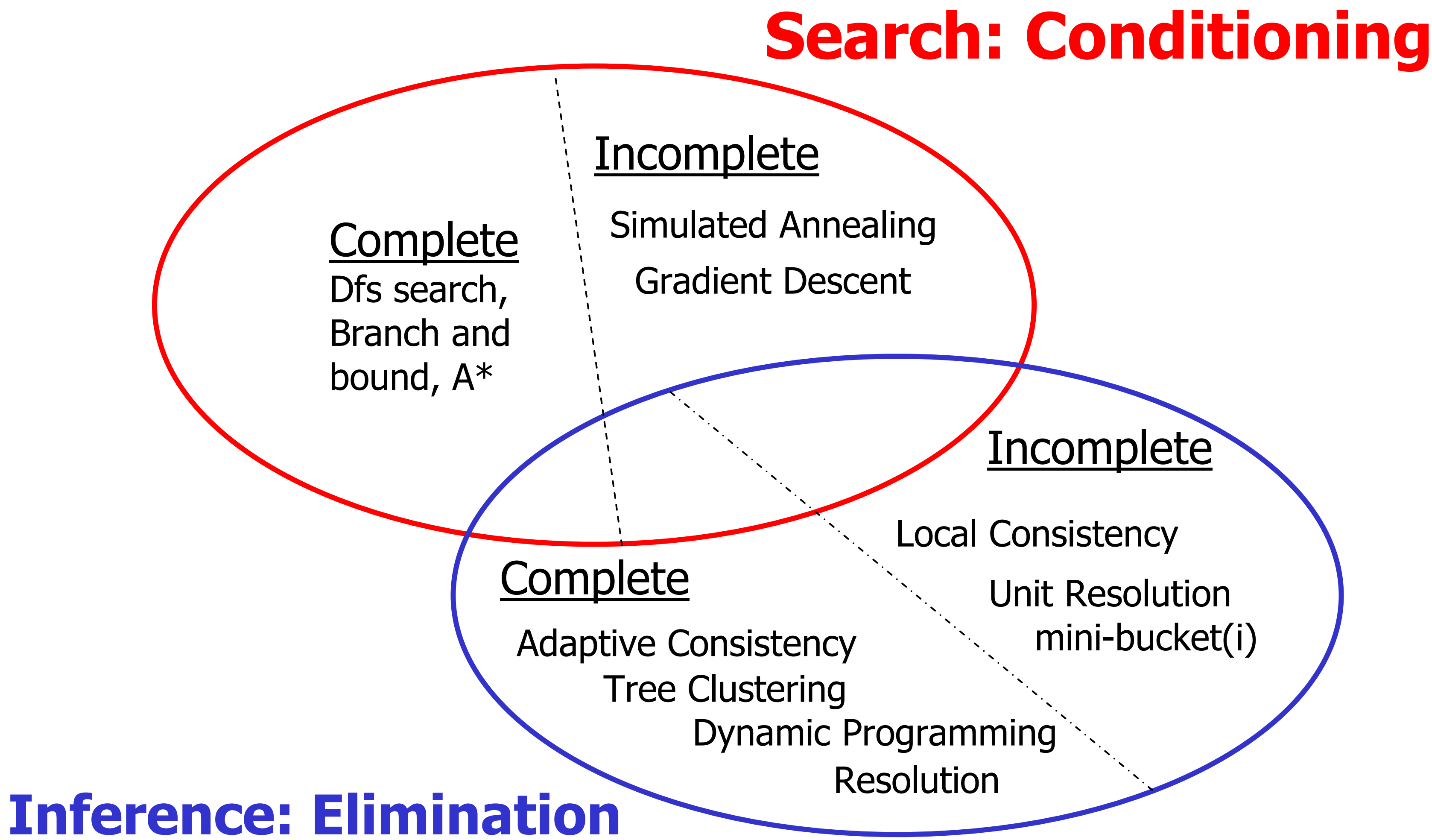


Operators: combine and project:

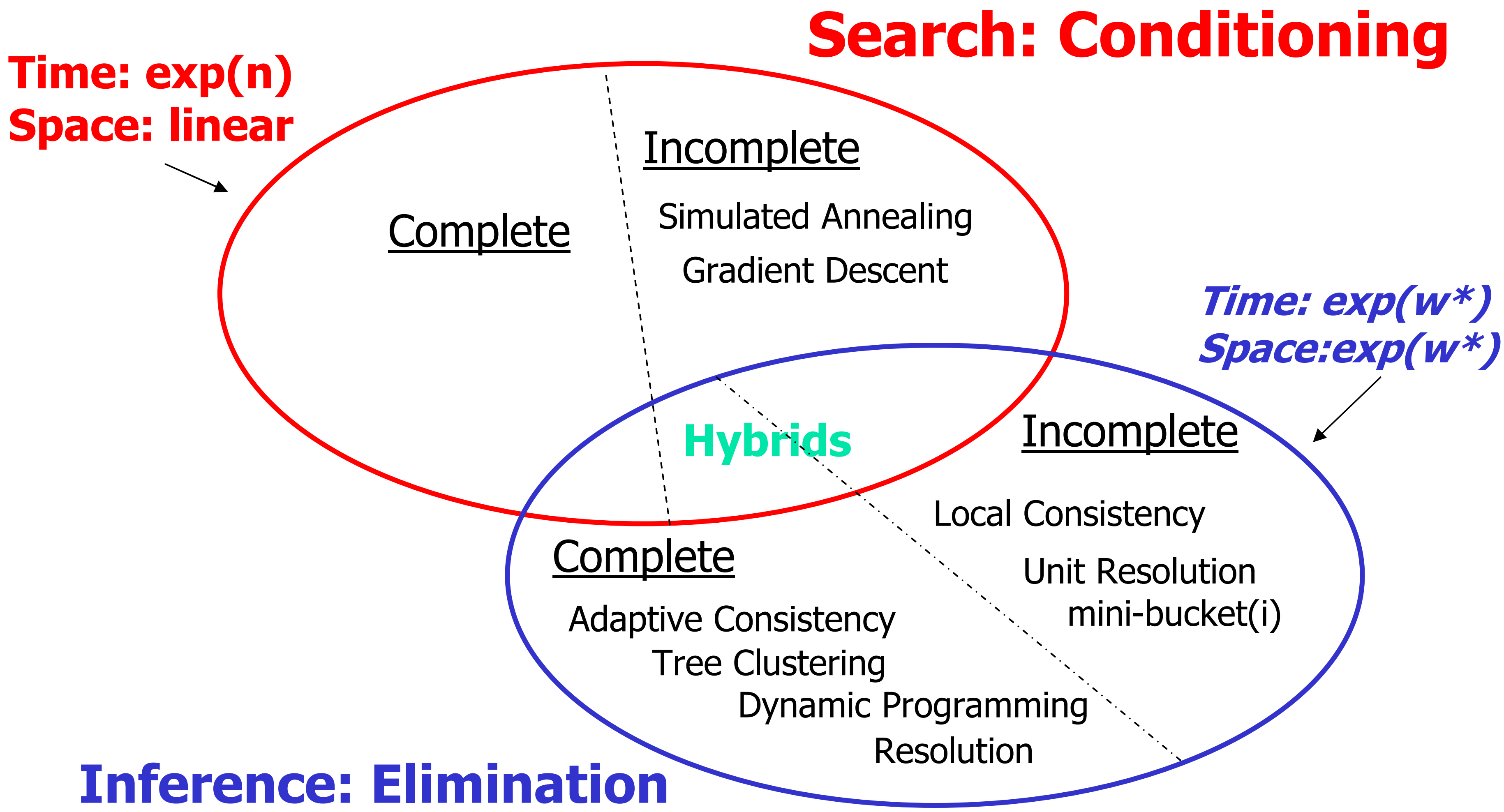
- ❑**MPE: $\max_X \prod_j P_j$**
- ❑**CSP: $\prod_X \times_j C_j$**
- ❑**Max-CSP: $\min_X \Sigma_j F_j$**
- ❑**Belief updating: $\Sigma_{X-y} \prod_j P_i$**

- All these tasks are NP-hard
- → identify special cases
- → approximate

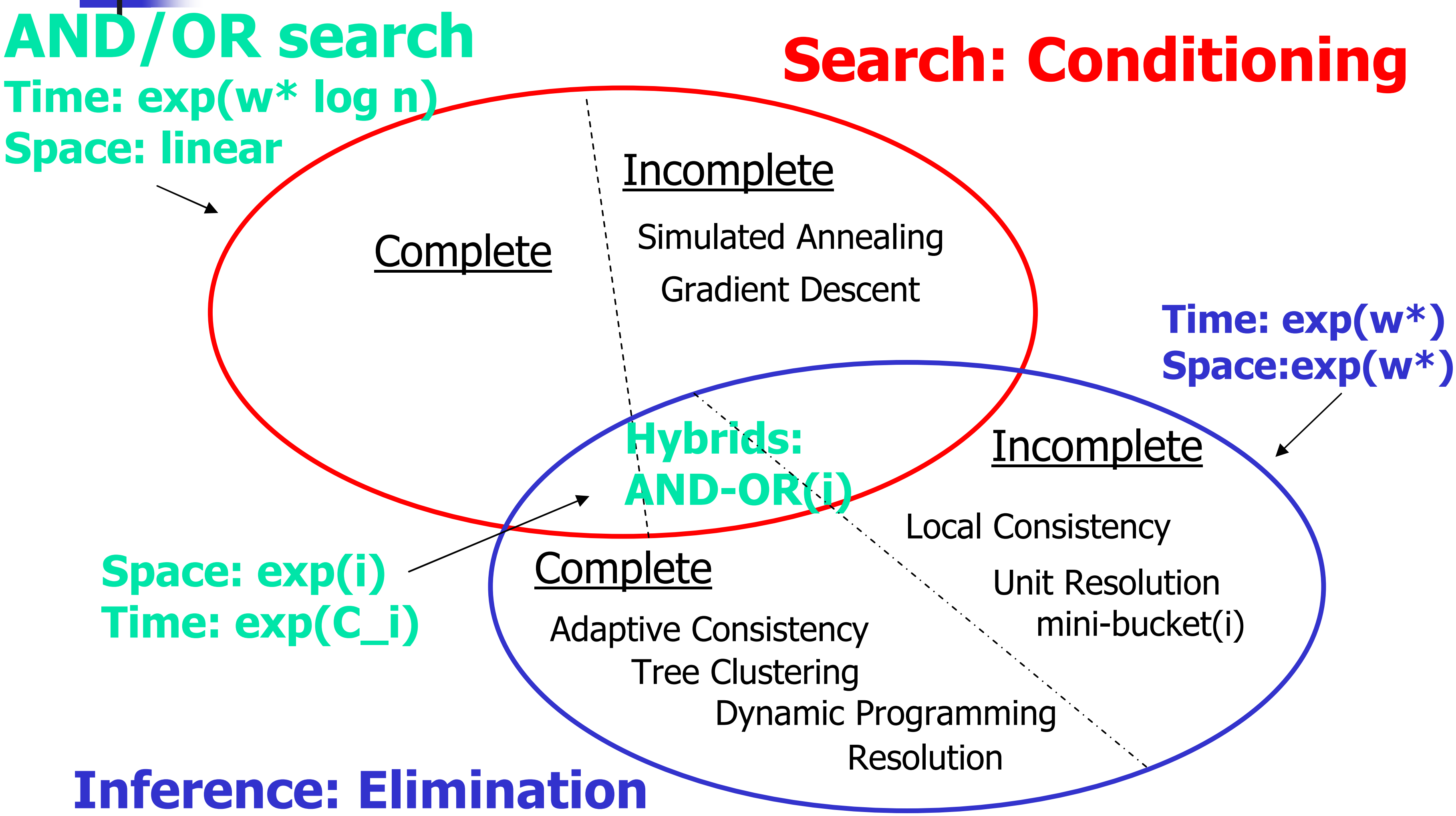
Solution Techniques



Solution Techniques

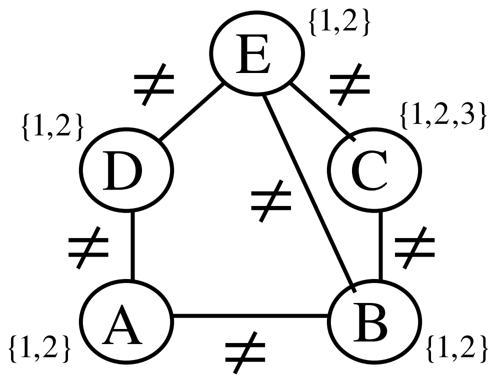


Solution Techniques



Bucket Elimination

Adaptive Consistency (Dechter & Pearl, 1987)



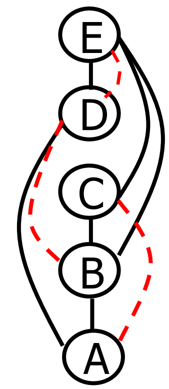
$Bucket(E): \mathrm{E} \neq \mathrm{D},\ \mathrm{E} \neq \mathrm{C},\ \mathrm{E} \neq \mathrm{B}$

$Bucket(D): \mathrm{D} \neq \mathrm{A}\ ||\ R_{DCB}$

$Bucket(C): \mathrm{C} \neq \mathrm{B}\ ||\ R_{ACB}$

$Bucket(B): \mathrm{B} \neq \mathrm{A}\ ||\ R_{AB}$

$Bucket(A): \quad R_{A}$



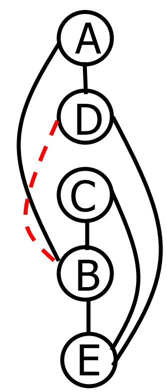
$Bucket(A): \mathrm{A} \neq \mathrm{D},\ \mathrm{A} \neq \mathrm{B}$

$Bucket(D): \mathrm{D} \neq \mathrm{E}\ ||\ R_{DB}$

$Bucket(C): \mathrm{C} \neq \mathrm{B},\ \mathrm{C} \neq \mathrm{E}$

$Bucket(B): \mathrm{B} \neq \mathrm{E}\ ||\ R^{D}{}_{BE},\ R^{C}{}_{BE}$

$Bucket(E): \ ||\ R_{E}$

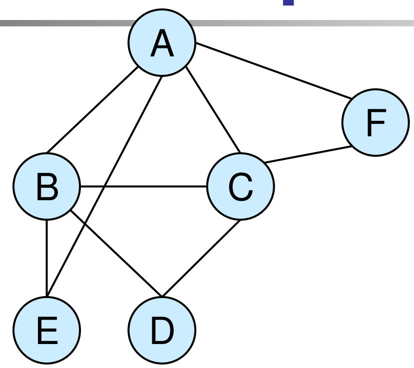


Complexity : $O(n\, exp(w^{*}(d)))$,
w^{*}(d) - ***induced width along ordering d***

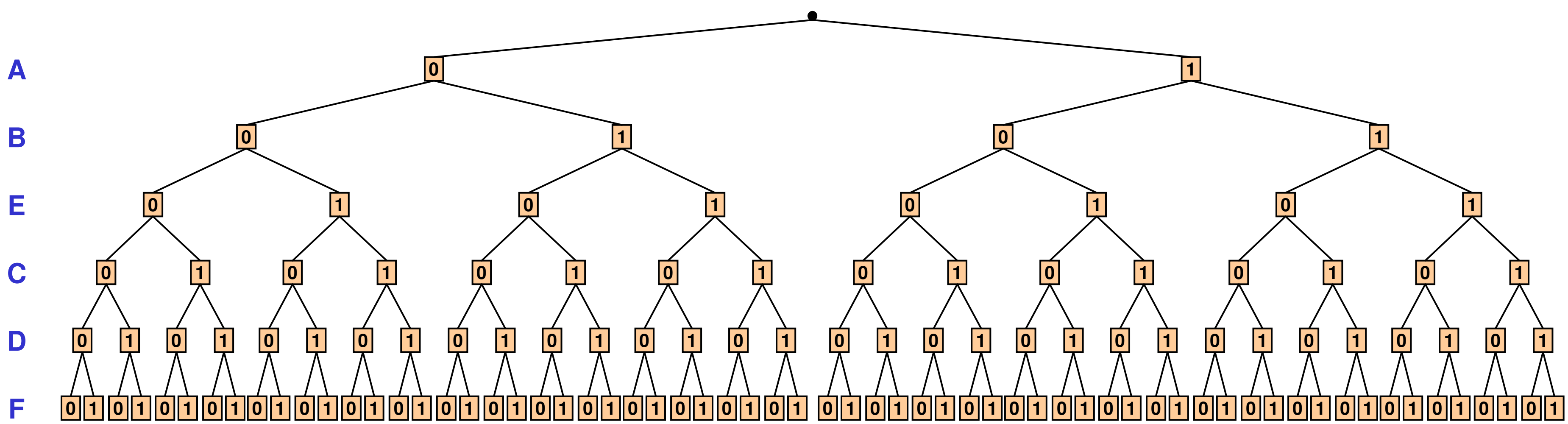
Outline

- Background: Search and inference in Graphical models
- AND/OR search trees
  - **Pseudo spanning trees, dfs trees**
- AND/OR search graphs
  - Tree-width and path-width bounds
- Relationship to algorithmic and compilation schemes.

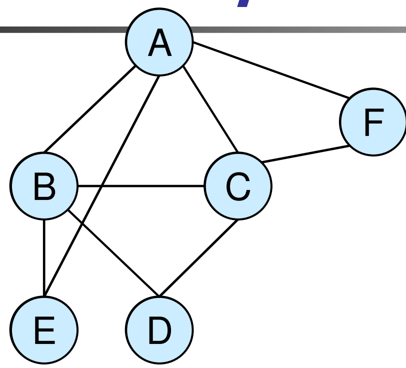
OR search space



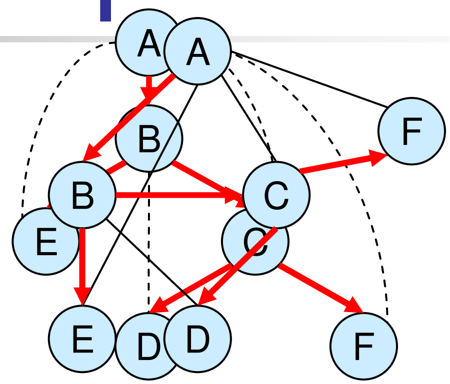
Ordering: A B E C D F



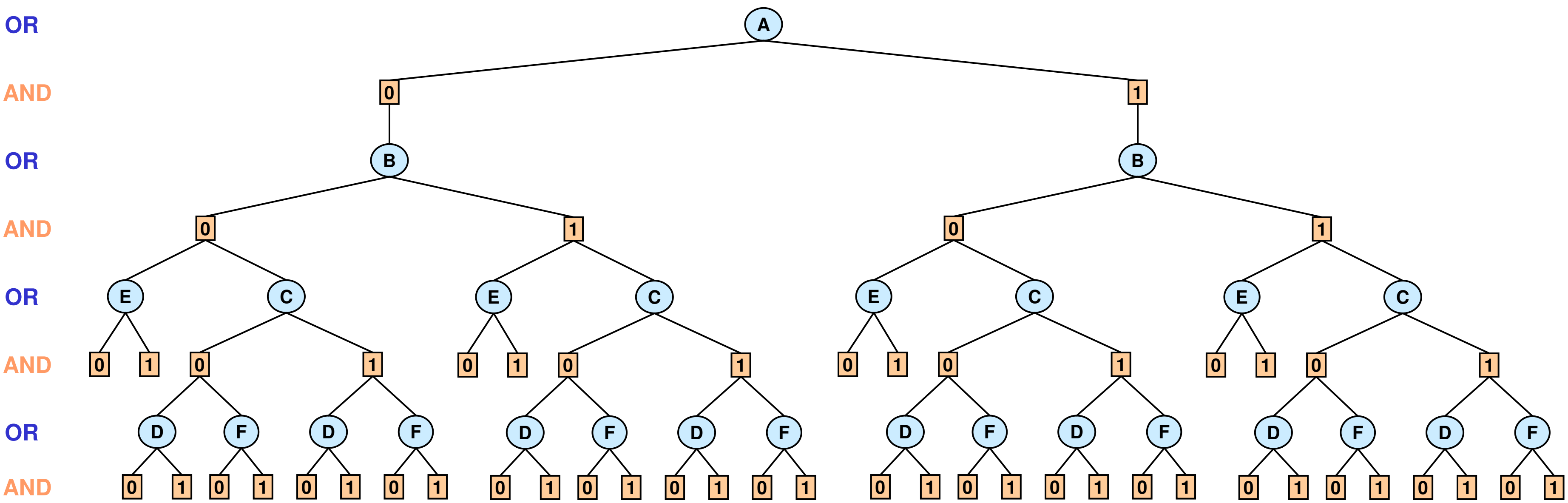
AND/OR search space



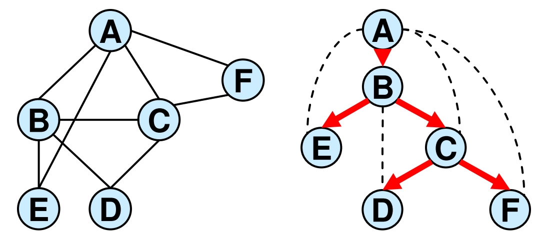
Primal graph

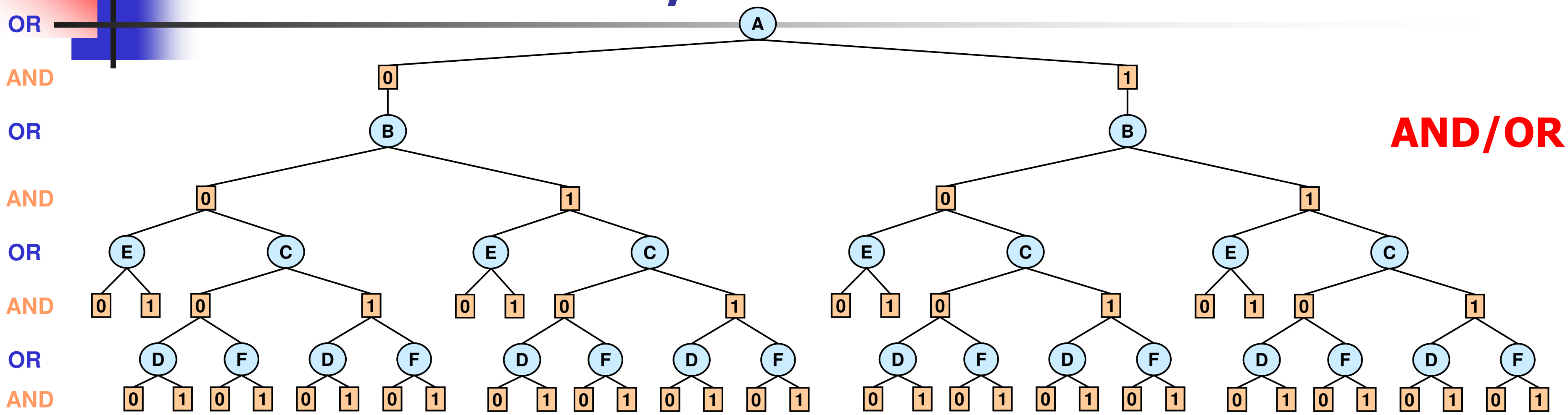


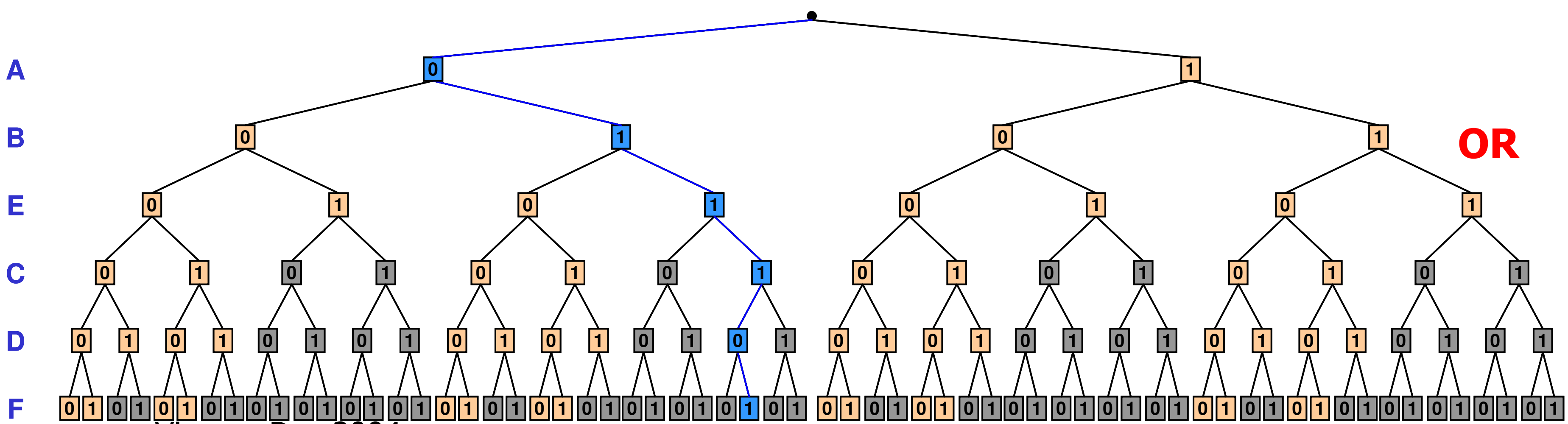
DFS tree



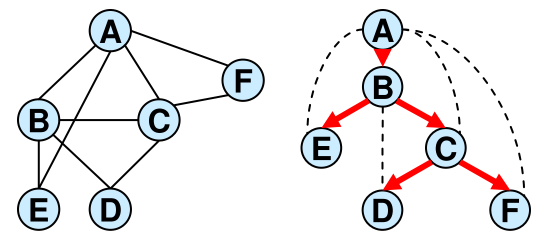
OR vs AND/OR

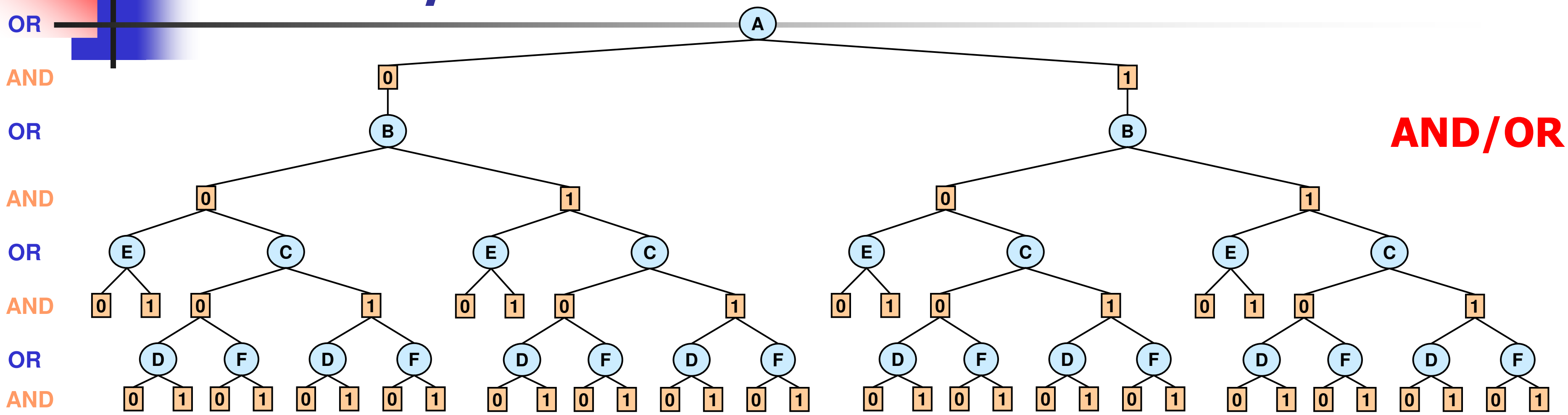




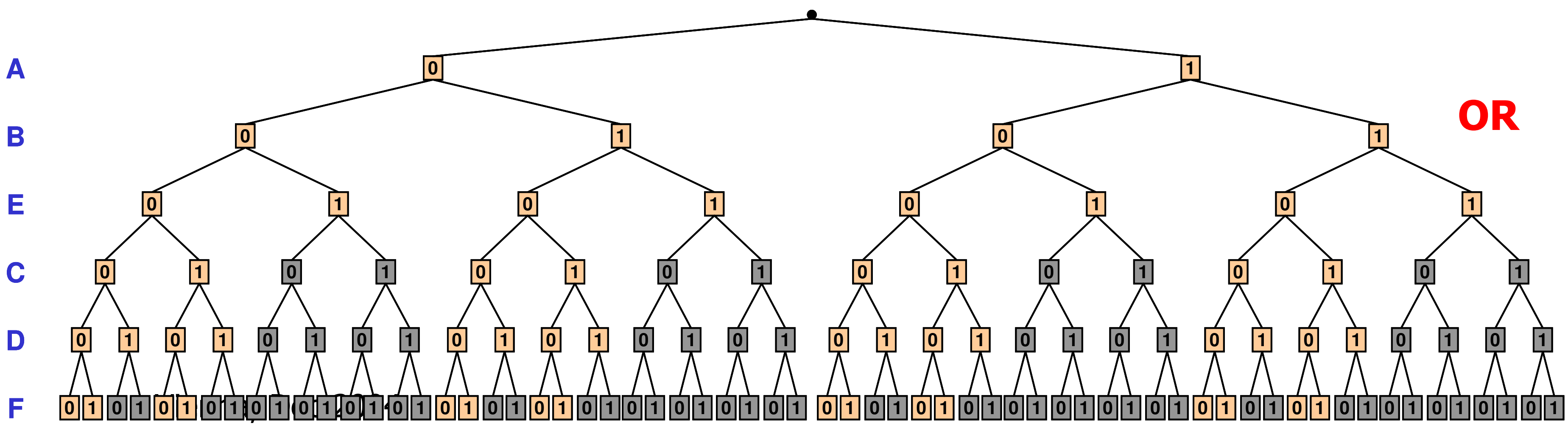


AND/OR vs. OR



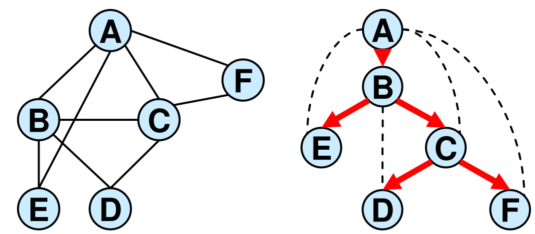


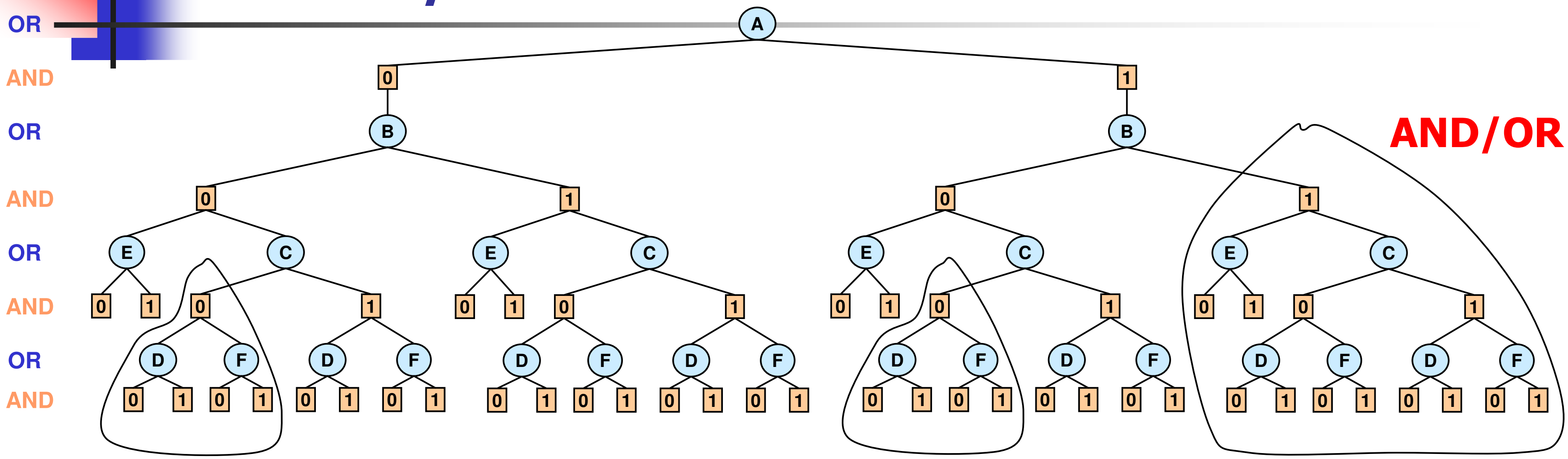
AND/OR size: exp(4), OR size exp(6)

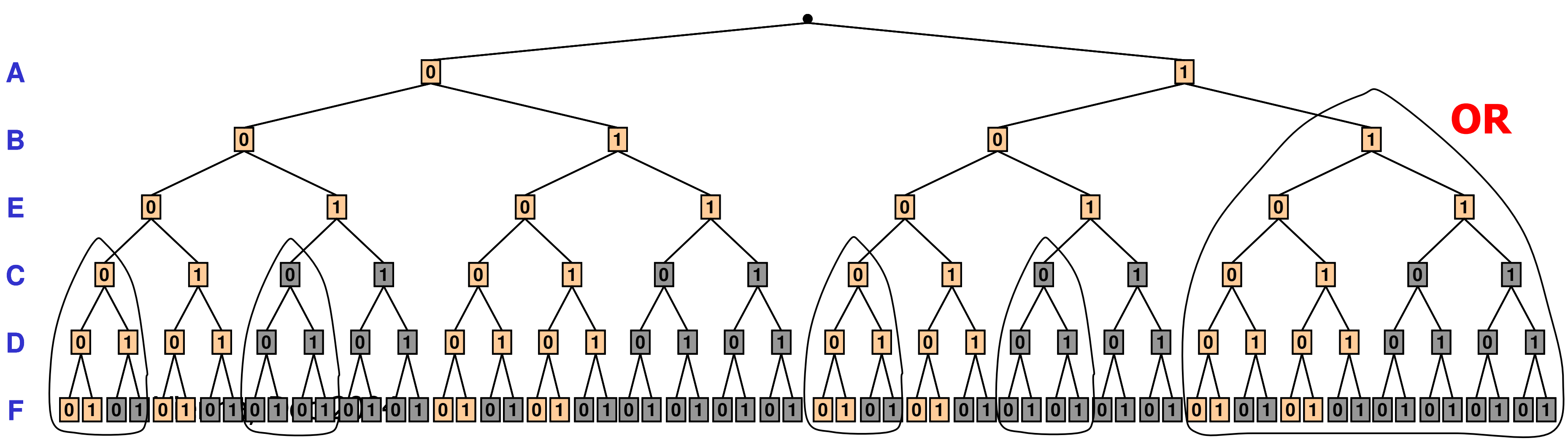


AND/OR vs. OR

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

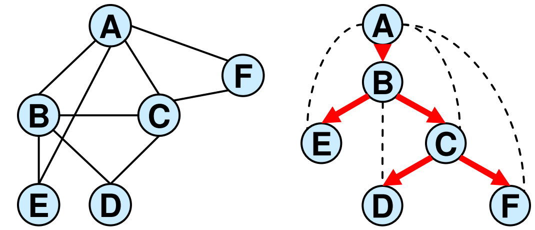


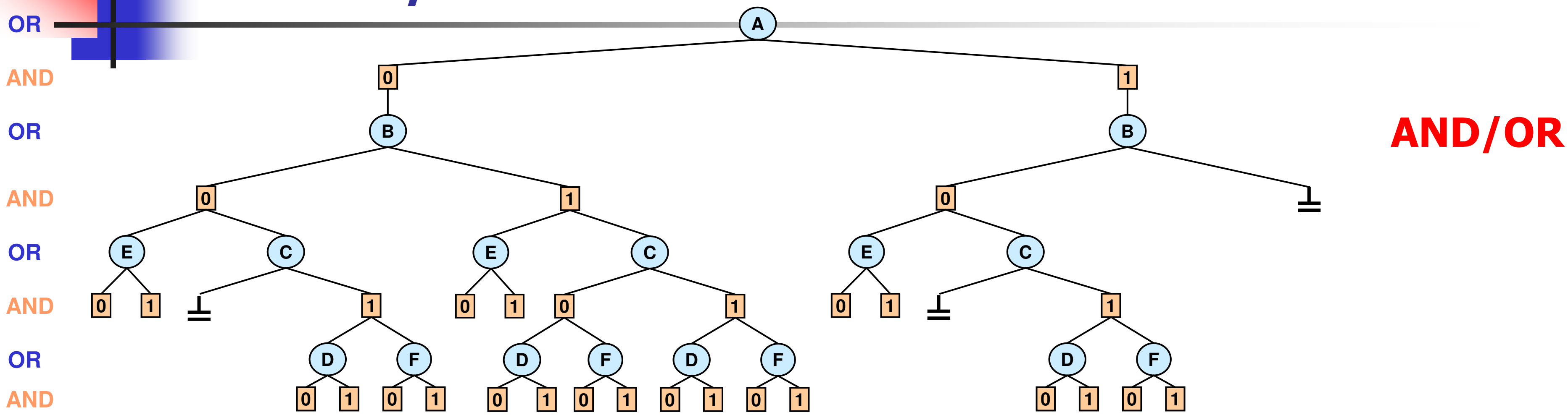


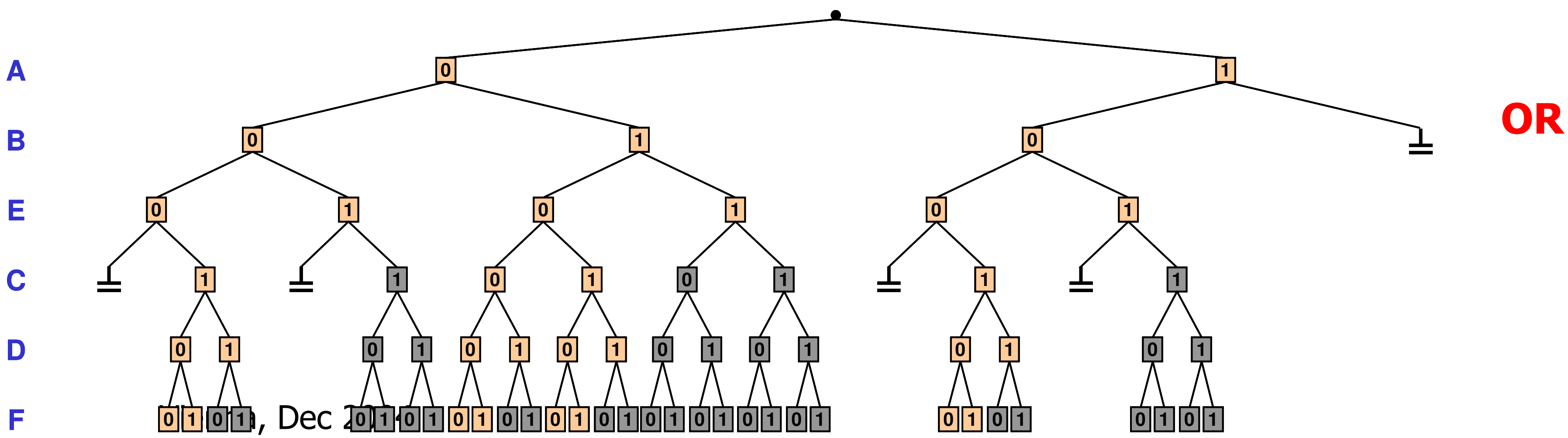


AND/OR vs. OR

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



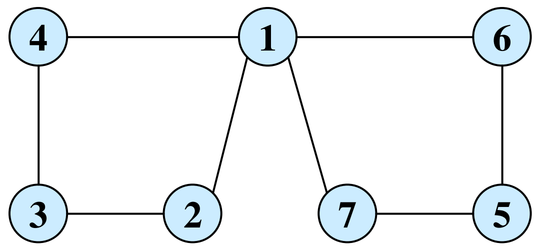




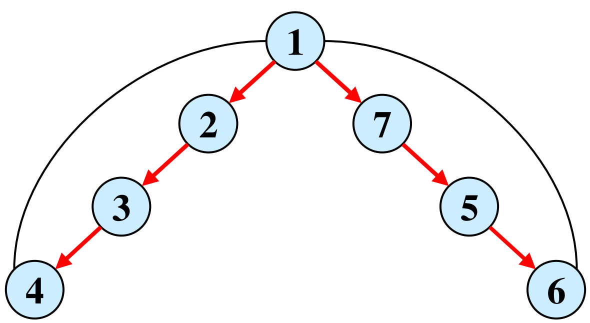
OR space vs. AND/OR space

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

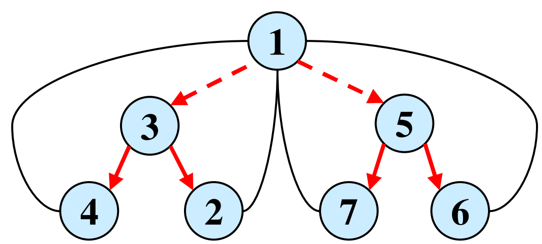
From DFS trees to pseudo-trees (Freuder 85, Bayardo 95)



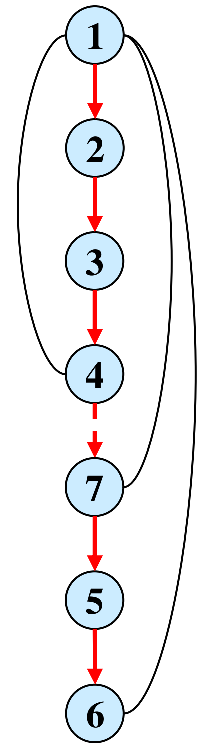
(a) Graph



(b) DFS tree
depth=3

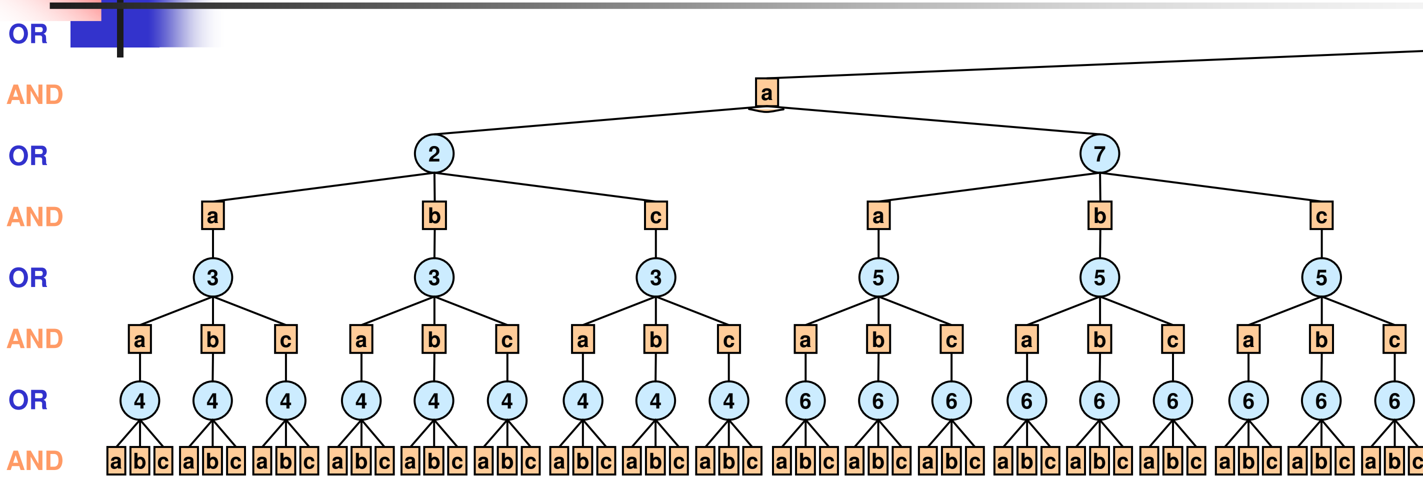


(c) pseudo- tree
depth=2



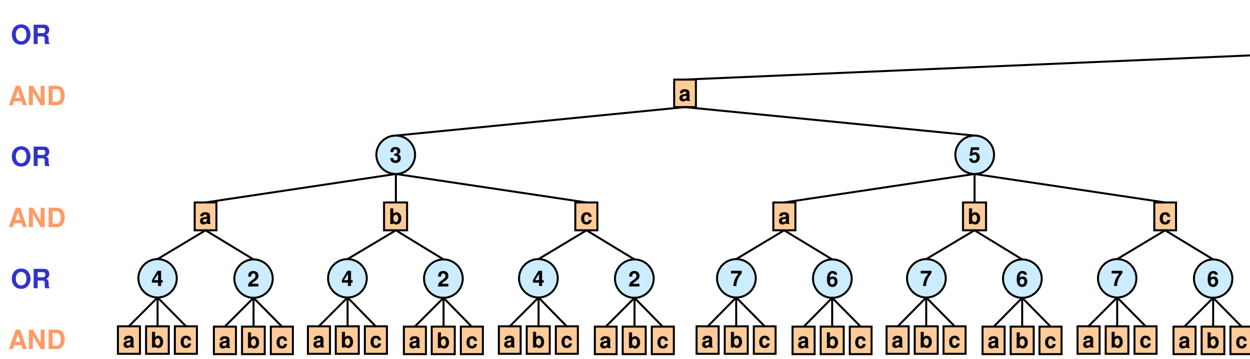
(d) Chain
depth=6

From DFS trees to Pseudo-trees



DFS tree

depth = 3



pseudo- tree

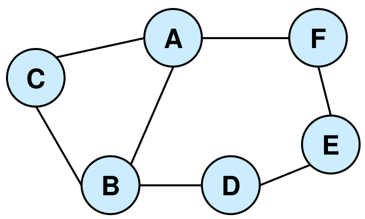
depth = 2

Finding min-depth backbone trees

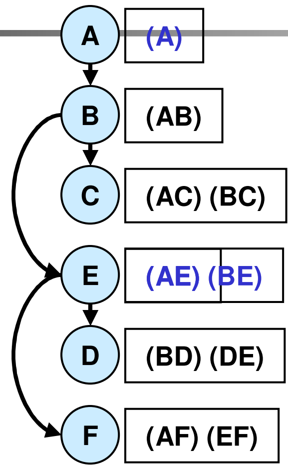
- Finding min depth DFS, or pseudo tree is NP-complete, but:
- Given a tree-decomposition whose tree-width is w*, there exists a pseudo tree T of G whose depth, satisfies (Bayardo and Mirankar, 1996, bodlaender and Gilbert, 91):

$$m <= w^* \log n,$$

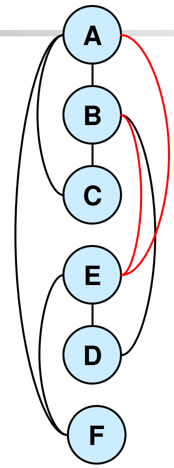
Generating pseudo-trees from Bucket trees



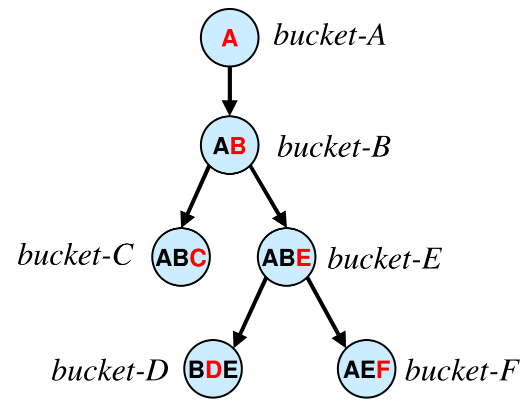
d: A B C E D F



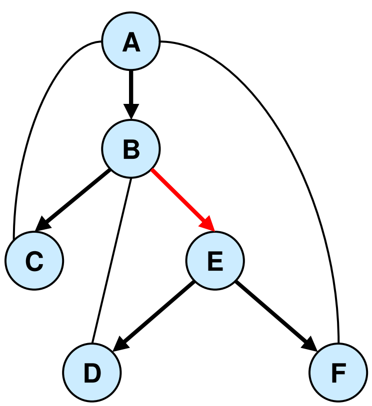
Bucket-tree based on *d*



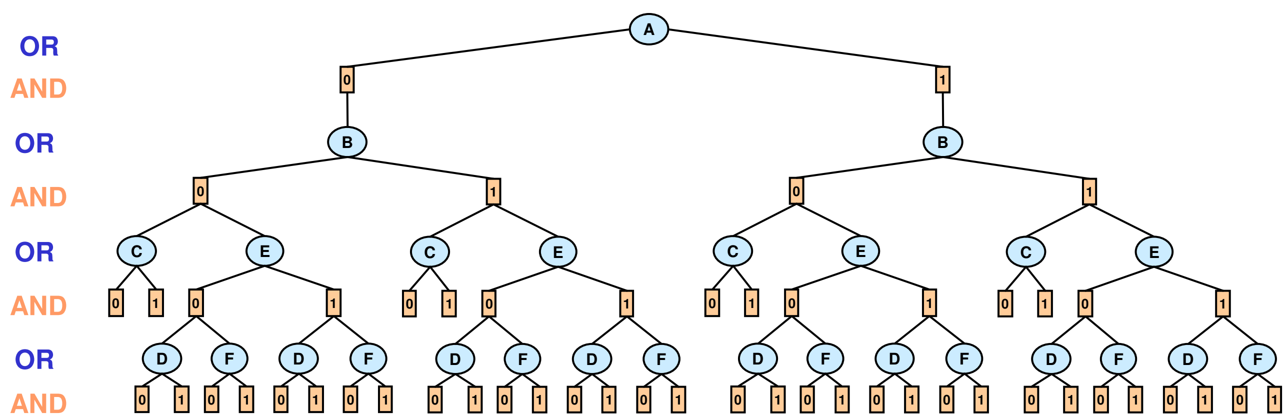
Induced graph



Bucket-tree



Bucket-tree used as pseudo-tree



AND/OR search tree

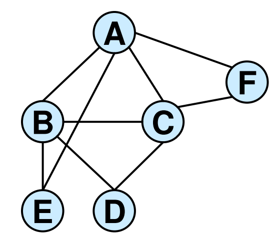
Pseudo Trees vs. DFS Trees

Model (DAG)	w*	Pseudo Tree avg. depth	DFS Tree avg. depth
(N=50, P=2)	9.54	**16.82**	36.03
(N=50, P=3)	16.1	**23.34**	40.6
(N=50, P=4)	20.91	**28.31**	43.19
(N=100, P=2)	18.3	**27.59**	72.36
(N=100, P=3)	30.97	**41.12**	80.47
(N=100, P=4)	40.27	**50.53**	86.54

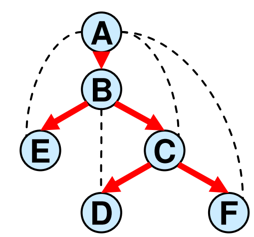
N = number of nodes, P = number of parents. MIN-FILL ordering. 100 instances.

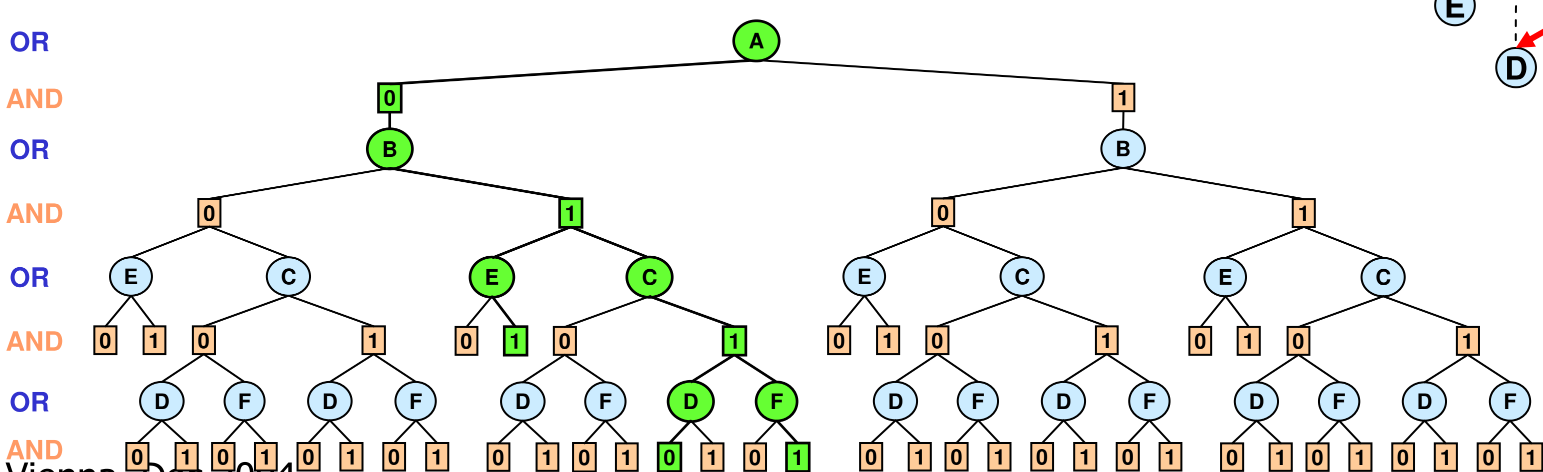
AND/OR search tree for graphical models

- **The AND/OR search tree of R 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**





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 $O(\exp(w^* \log n))$
- Related to: (Freuder 85; Dechter 90, Bayardo et. al. 96, Darwiche 1999, Bacchus 2003)
- When the pseudo-tree is a chain we get an OR space

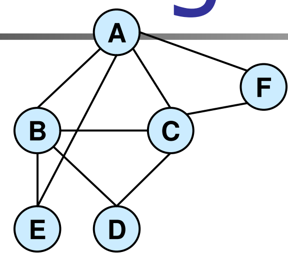
Tasks and value of nodes

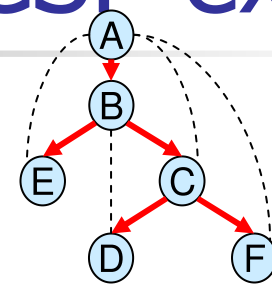
- **V(n) is the value of the tree T(n) for the task:**
  - **Consistency: v(n) is 0 if T(n) inconsistent, 1 othewise.**
  - **Counting: v(n) is number of solutions in T(n)**
  - **Optimization: v(n) is the optimal solution in T(n)**
  - **Belief updating: v(n), probability of evidence in T(n).**
  - **Partition function: v(n) is the total probability in T(n).**

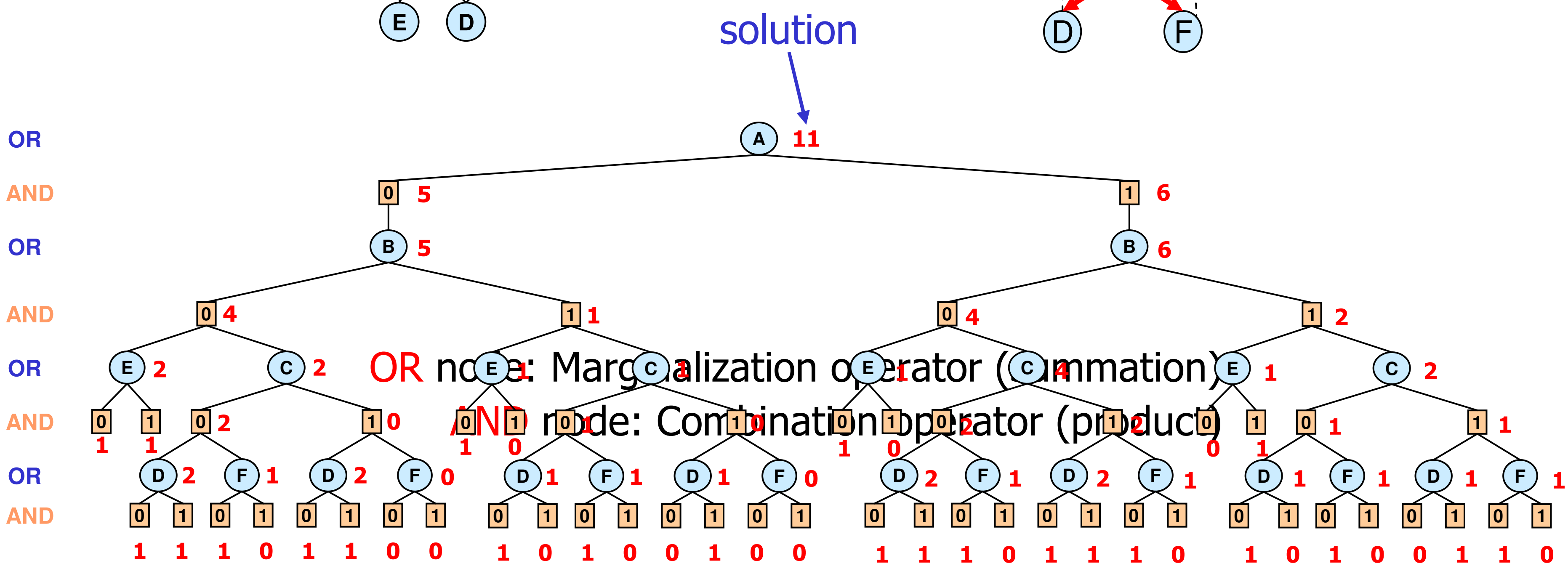
- **Goal: compute the value of the root node recursively using dfs search of the AND/OR tree.**

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

DFS algorithm (#CSP example)







OR node: Marginalization operator (summation)

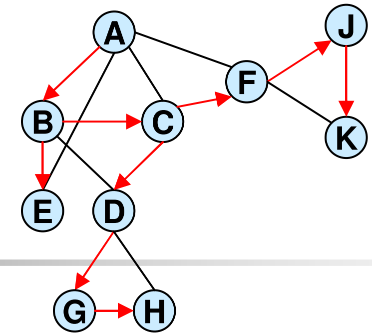
AND node: Combination operator (product)

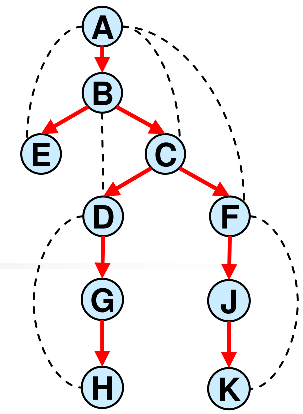
Value of node = number of solutions below it

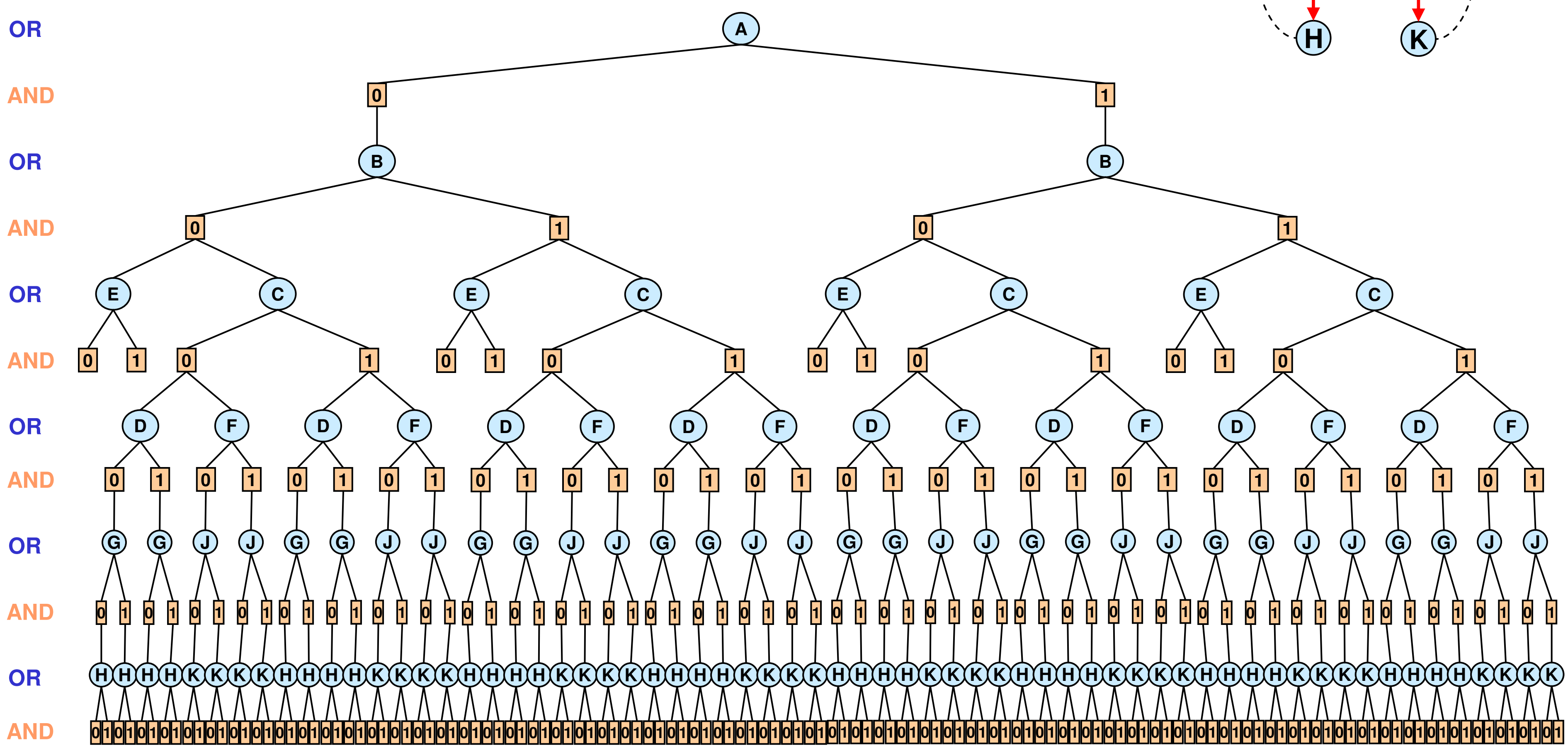
From Search Trees to Search Graphs

- Any two nodes that root identical subtrees/subgraphs can be merged
- Minimal AND/OR search graph: closure under merge of the AND/OR search tree
  - Inconsistent subtrees can be pruned too.
  - Some portions can be collapsed or reduced.

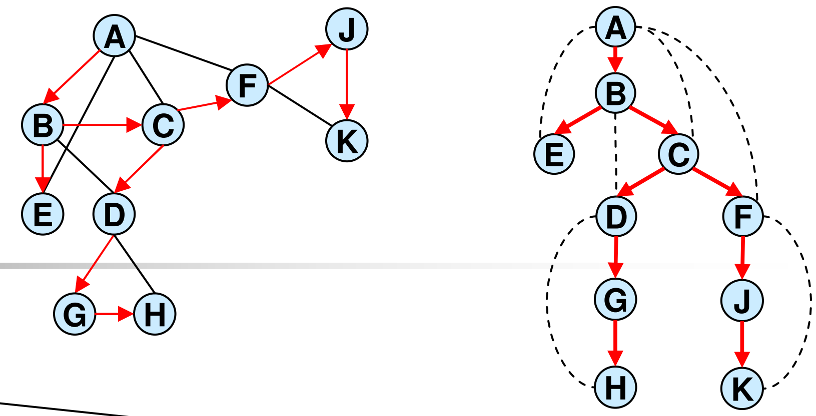
AND/OR Tree

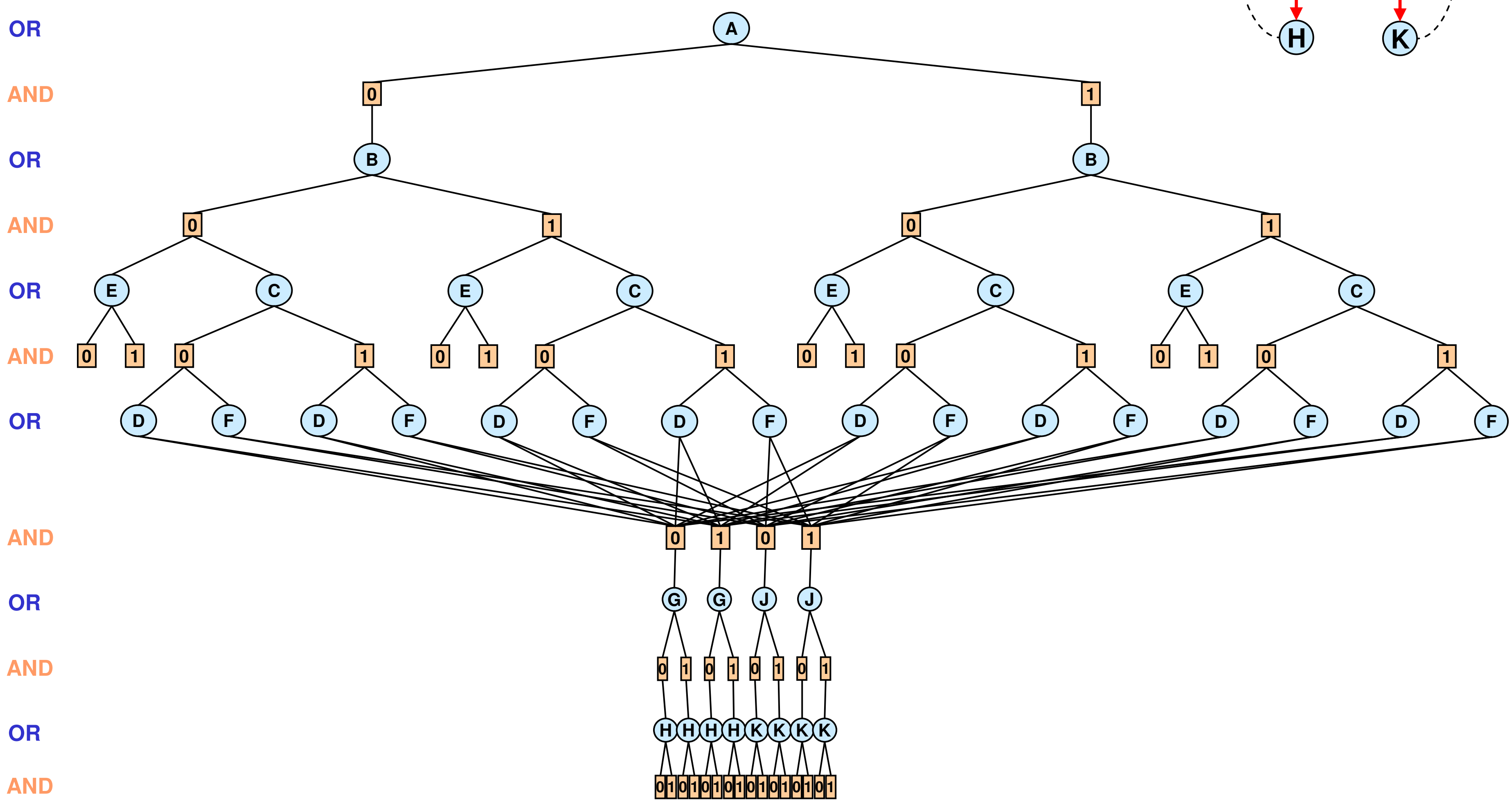

A
B
C
F
J
K
E
D
G
H


A
B
E
C
D
F
G
J
H
K


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

An AND/OR graph

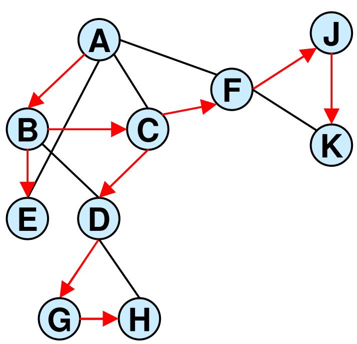


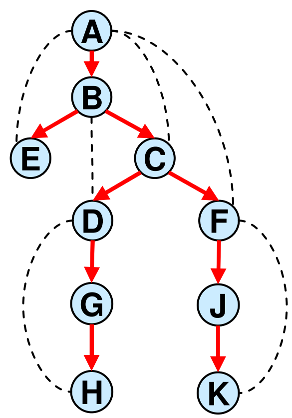


Context based caching

- Caching is possible when context is the same

- context = parent-separator set in induced pseudo-graph
  = current variable +
  parents connected to subtree below





context(B) = {A, B}

context(c) = {A,B,C}

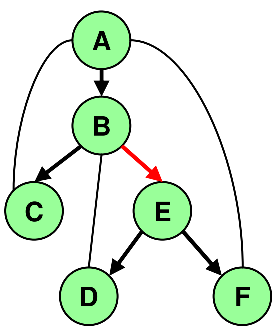
context(D) = {D}

context(F) = {F}

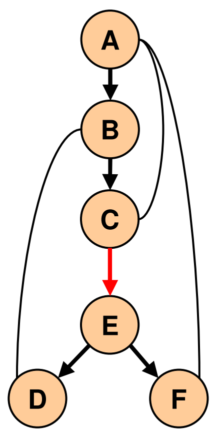
Induced-width of pseudo-trees

The induced-width of a pseudo-tree is its induced-width along a dfs order that includes pseudo arcs.

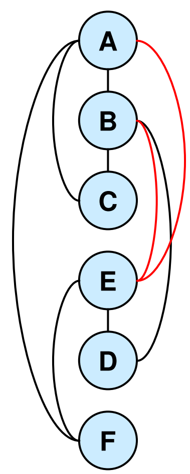
- For pseudo-chains induced-width is path-width (yielding path-decomposition)
- Contexts are bounded by the induced-width of the pseudo tree.
- Min induced-pseudo-width=tree-width

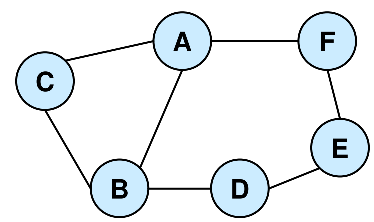


Good pseudo-tree



Bad pseudo-tree





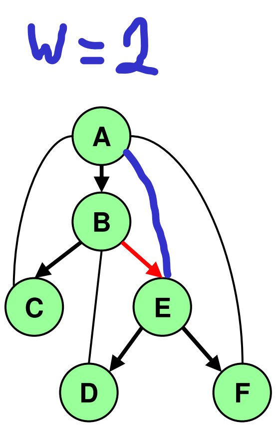
A graph

DFS order of both pseudo trees:
***d* = A B C E D F**

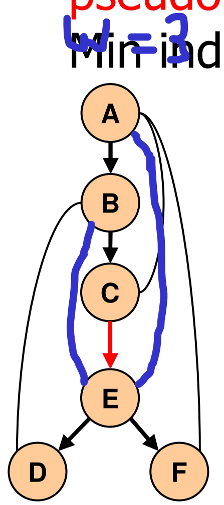
Induced-width of pseudo-trees

The induced-width of a pseudo-tree is its induced-width along a dfs order that includes pseudo arcs.

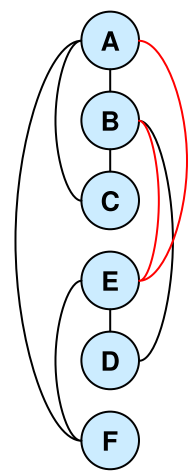
- For pseudo-chains induced-width is path-width (yielding path-decomposition)
- Contexts are bounded by the induced-width of the pseudo tree.
- Min induced-pseudo-width=tree-width

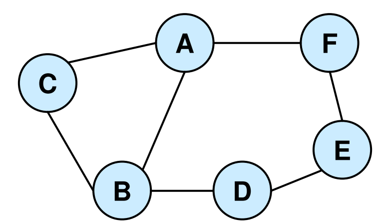


Good pseudo-tree



Bad pseudo-tree





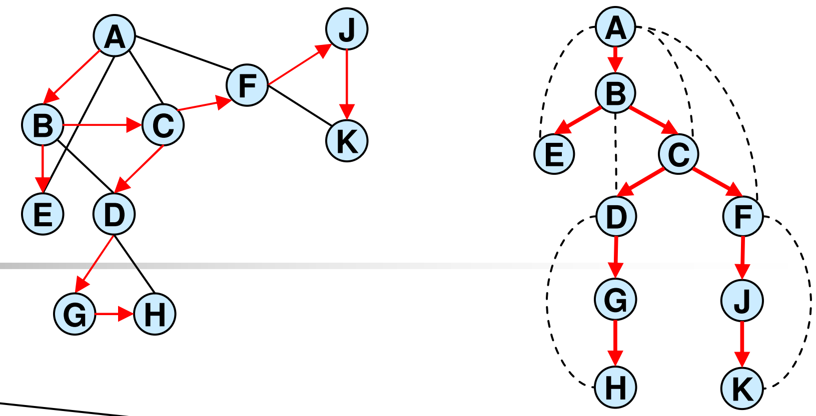
A graph

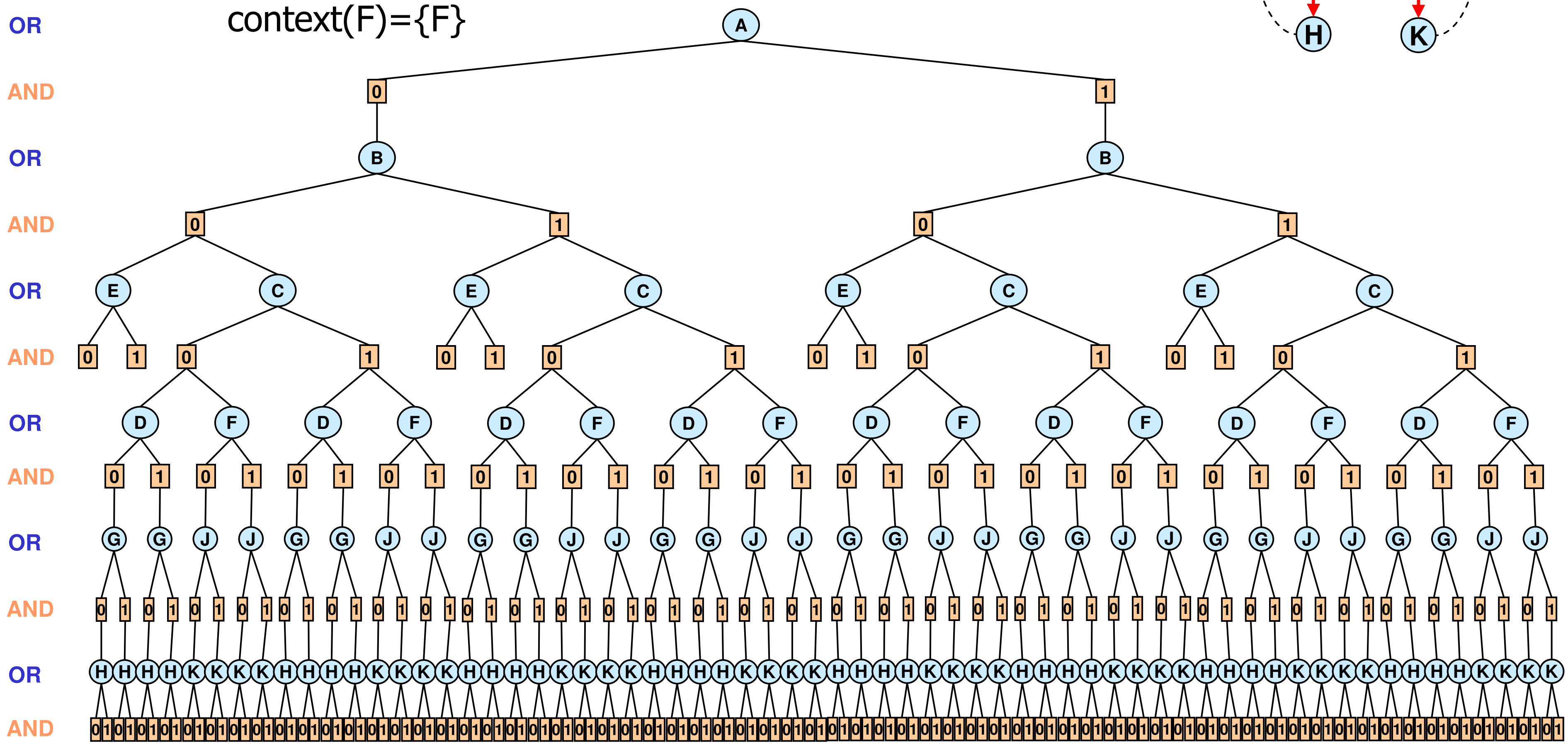
DFS order of both pseudo trees:
d **= A B C E D F**

Caching

context(D)={D}

context(F)={F}

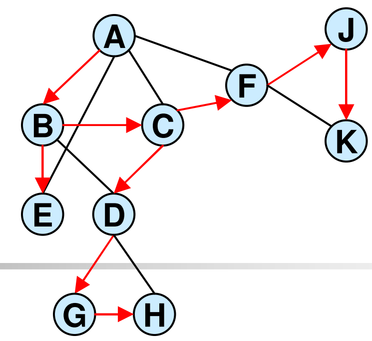


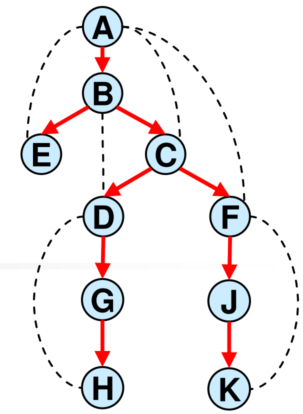


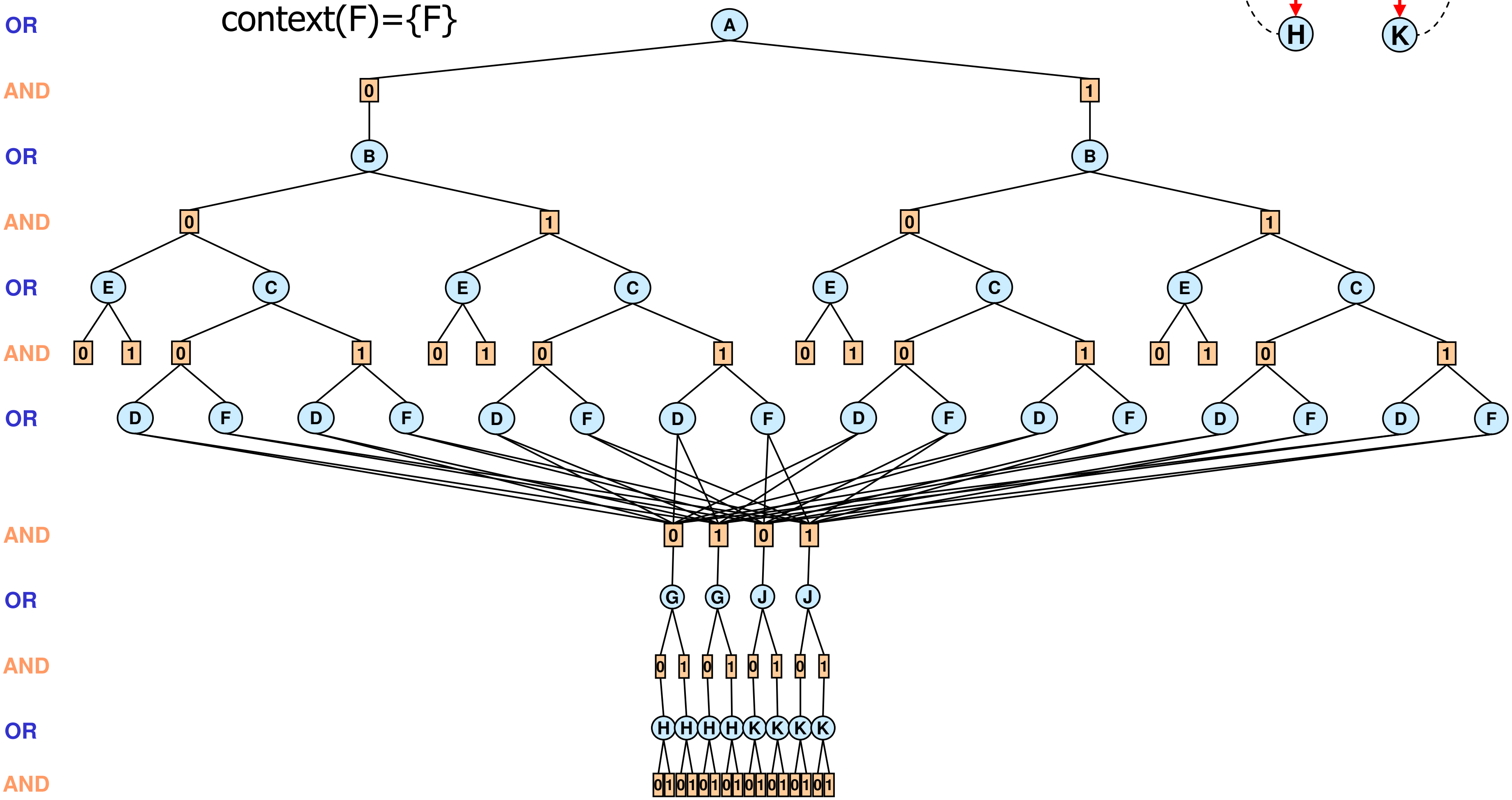
Caching

context(D)={D}

context(F)={F}







Size of minimal AND/OR context graphs

Theorem:

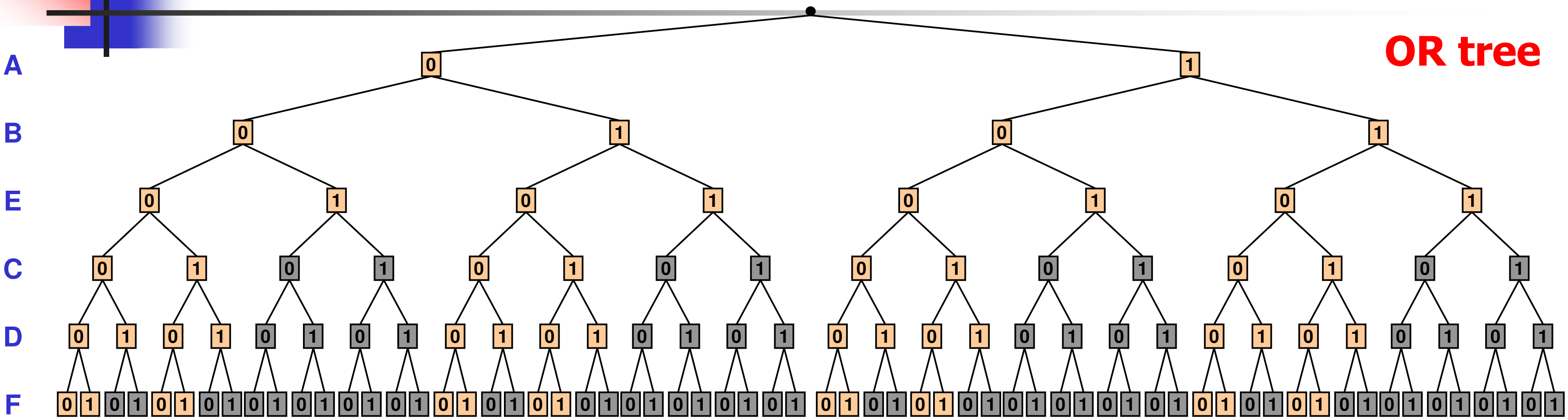
- Minimal AND/OR context graph is bounded exponentially by its pseudo-tree induced-width.

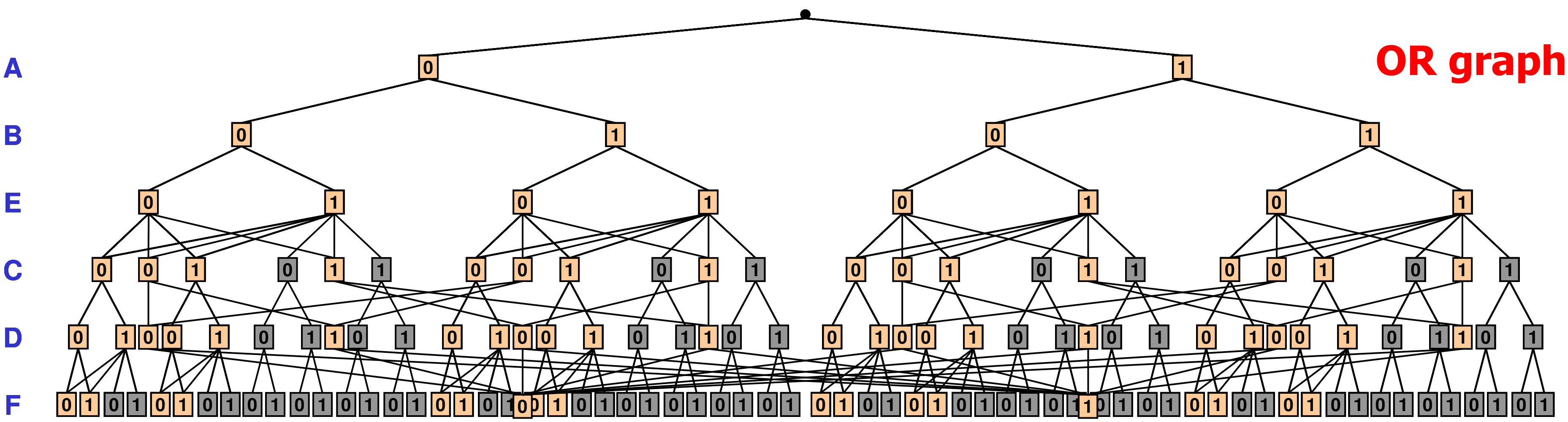
- The tree-width of a pseudo-chain is path-width (pw)

- → Minimal OR search graph is O(exp(pw*)).
- → Minimal AND/OR graph is O(exp(w*))

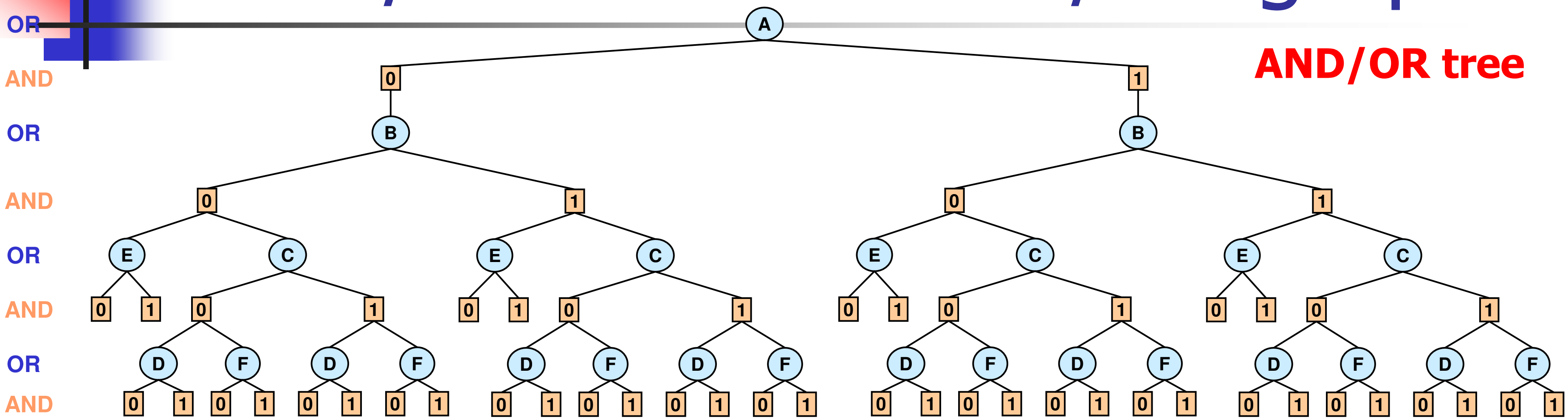
- Always, $w^* \leq pw^*$, but $pw^* <= w^* \log n$

OR tree vs. OR graph

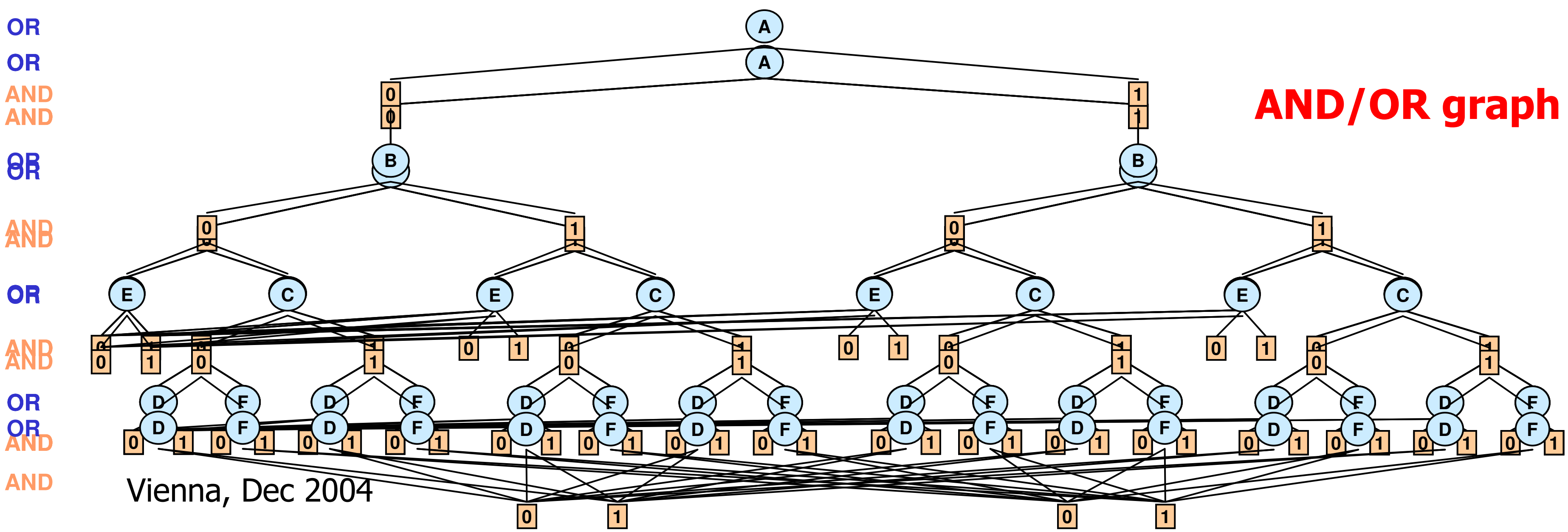




AND/OR tree vs. AND/OR graph

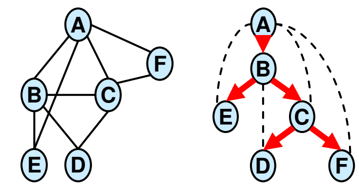


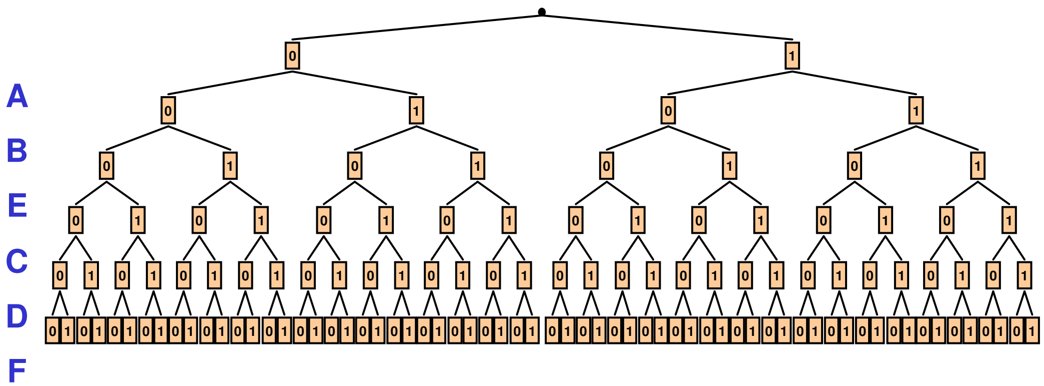
AND/OR tree



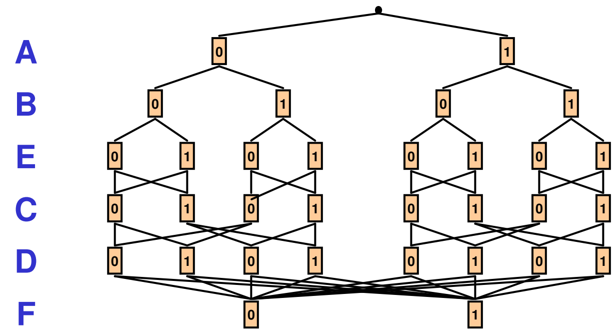
AND/OR graph

All four search spaces

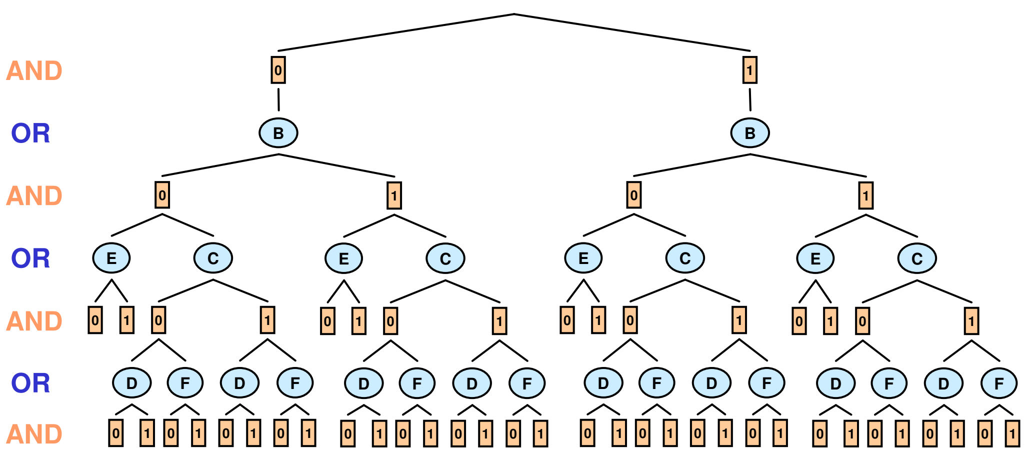




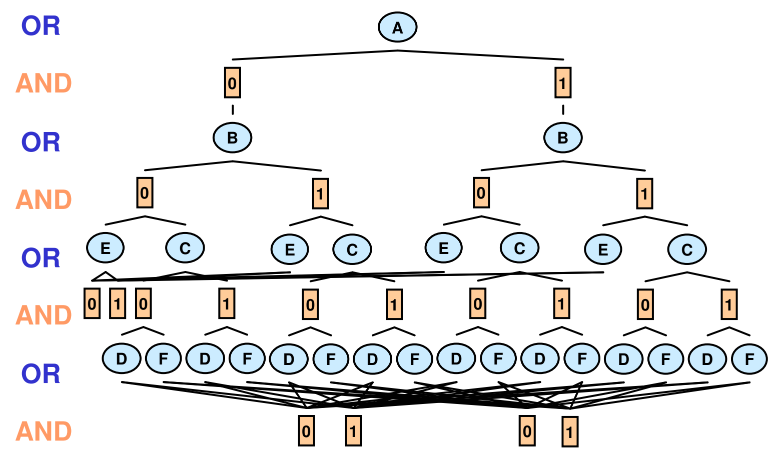
Full OR search tree



Context minimal OR search graph

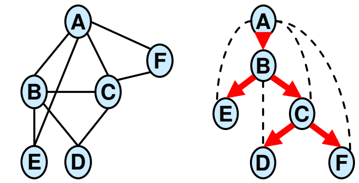


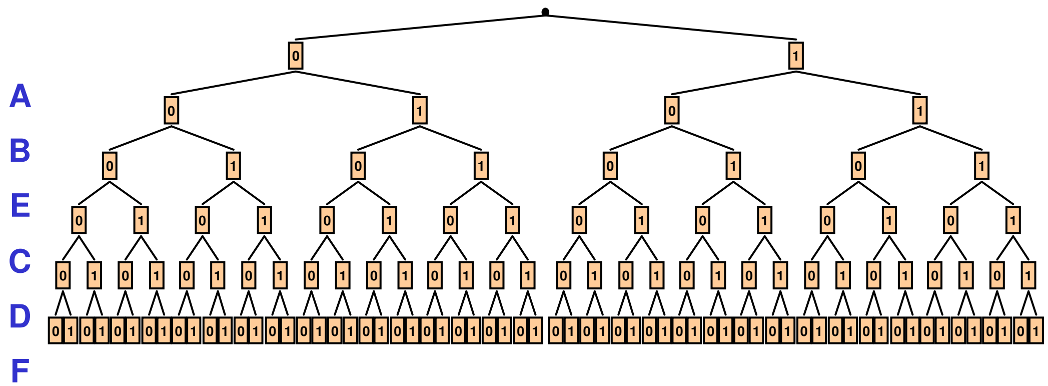
Full AND/OR search tree



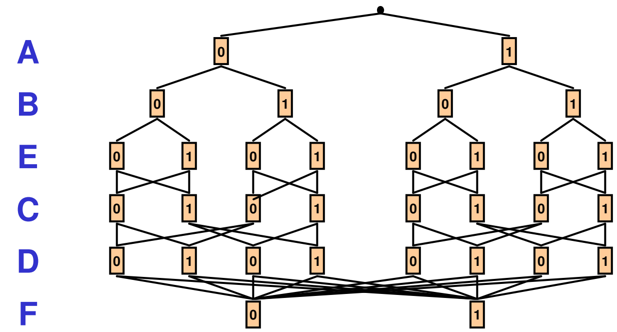
Context minimal AND/OR search graph

All four search spaces

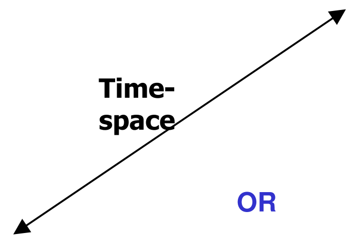


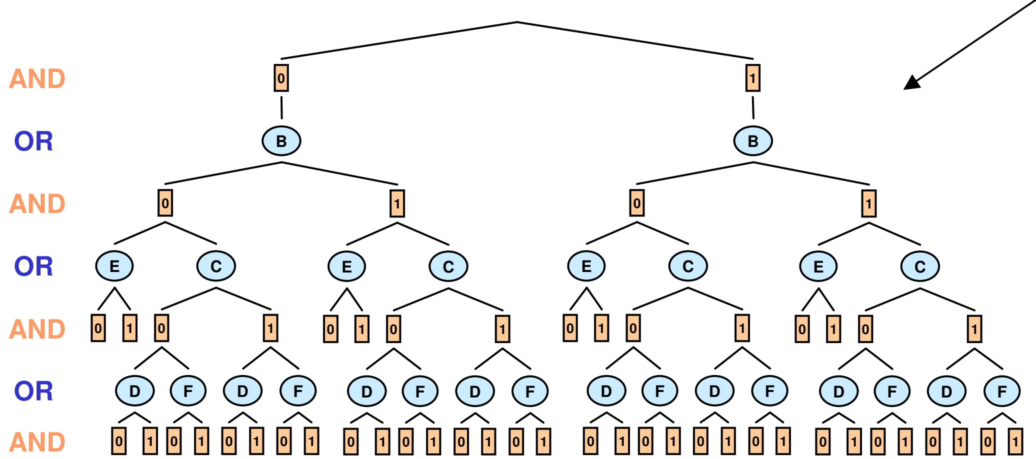


Full OR search tree

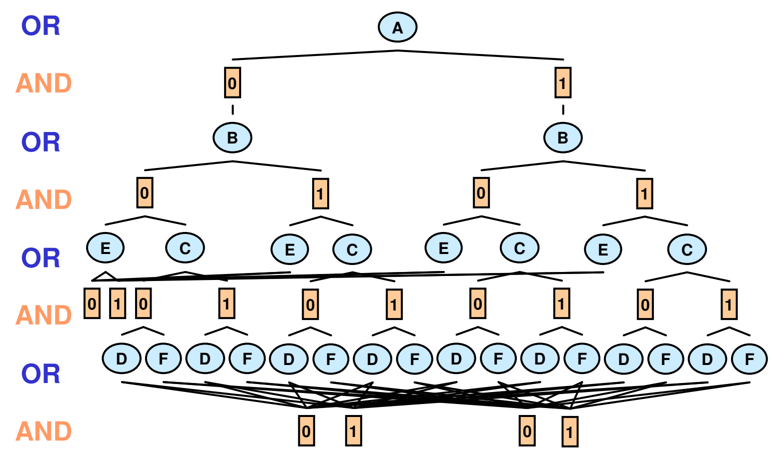


Context minimal OR search graph





Full AND/OR search tree



Context minimal AND/OR search graph

AND/OR vs OR **Graphs**

w*<=pw*<=m*<=w*log n (Bodlaendar et. Al, 1991)

- Theorem: for balanced w-trees

$$pw = m = \log n \bullet tw$$

1) and-orTree-size = orGraph-size

2) $$\frac{orGraph - size}{aorGraph - size} = k^{(w-1)\frac{\log n}{1+\log w} - 2w - 3}$$

Uniqueness of minimal AND/OR graph

- **Theorem:** Given a pseudo-tree, the minimal AND/OR search graph is unique for all graph-models that are consistent with that pseudo tree.

- Related to compilation schemes:
  - Minimal OR – related to OBDDs (McMillan)
  - Minimal AND/OR – related to tree-OBDDs (McMillan 94),
  - AND/OR graphs related to d-DNNF (Darwiche et. Al. 2002)

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)

i=0 → i=w*

Space: O(n)

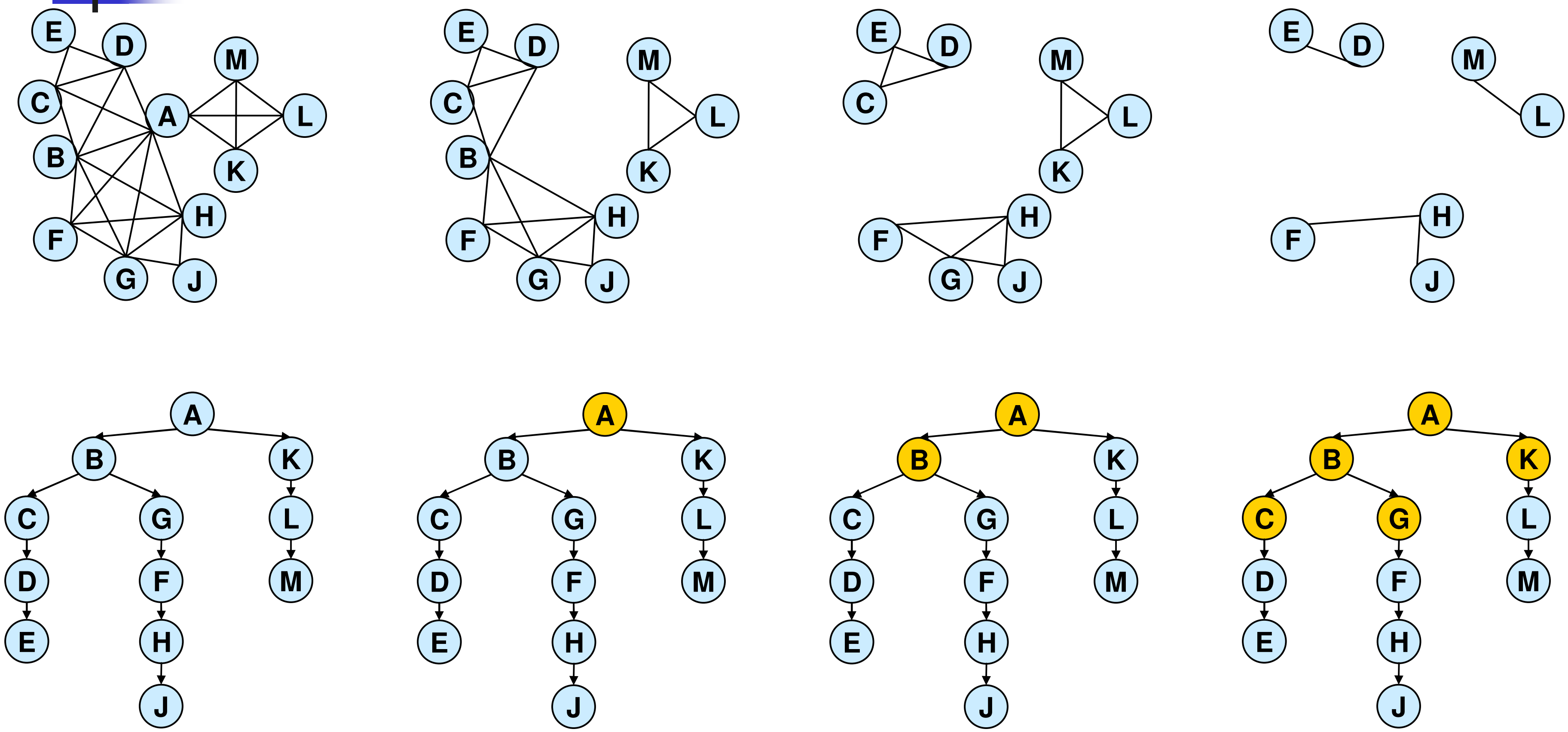
Time: O(exp(w* log n))

Space: O(exp w*)

Time: O(exp w*)

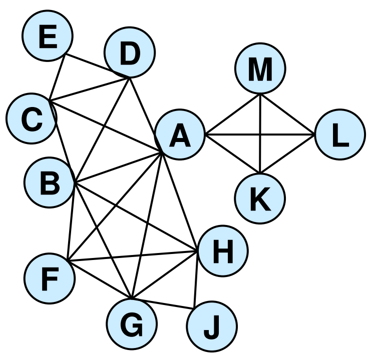
AO(i) time complexity?

AND/OR w-cutset

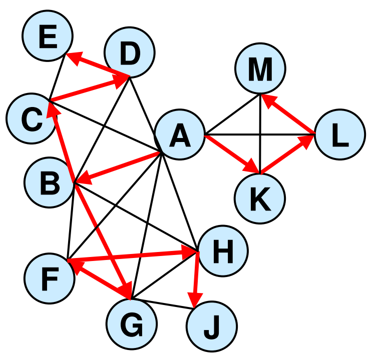


3-cutset 2-cutset 1-cutset

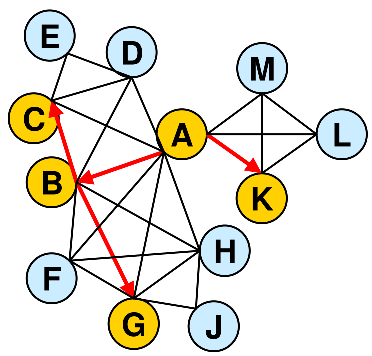
AND/OR w-cutset



grahpical model



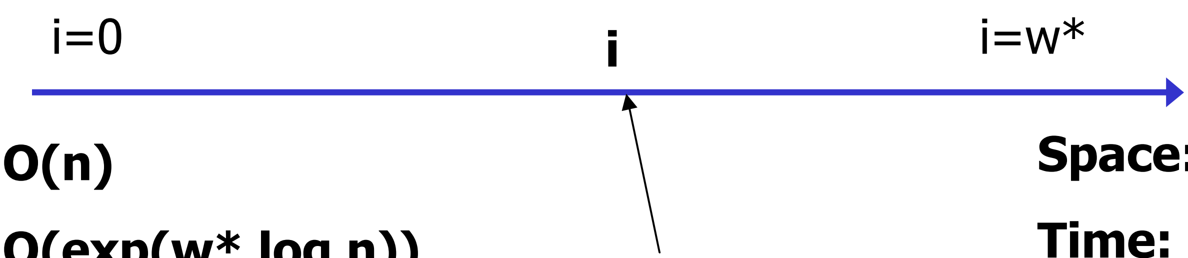
pseudo tree



1-cutset tree

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*)

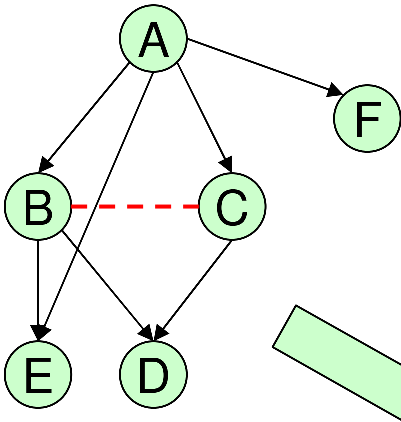
Time: O(exp w*)

Overview

- Introduction and background for graphical models: inference and search
- Inference:
  - Exact: Tree-clustering, variable elimination,
  - Approximate: mini-bucket, belief propagation
- AND/OR search spaces for graphical models
  - mixed networks
- **Empirical evaluation over mixed networks, counting**

Mixed Networks

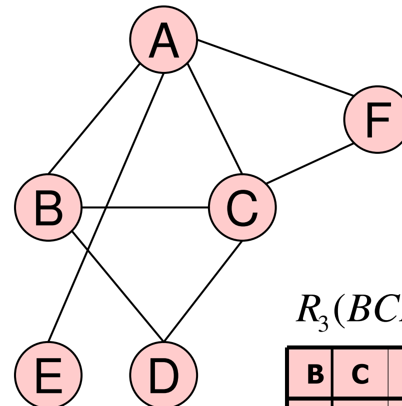
Belief Network



$P(D|B,C)$

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

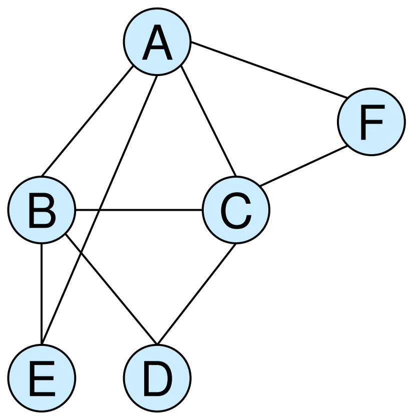
Constraint Network



$R_3(BCD)$

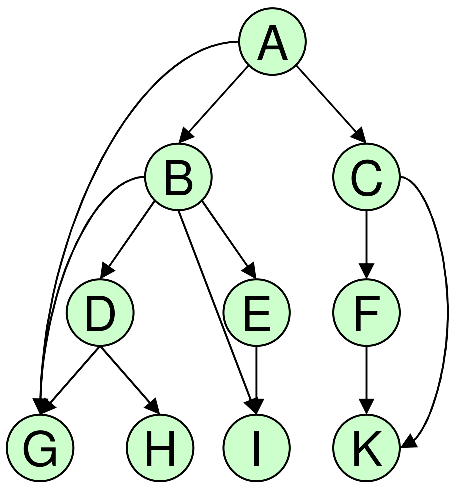
B	C	D
0	0	1
0	1	0
1	1	0

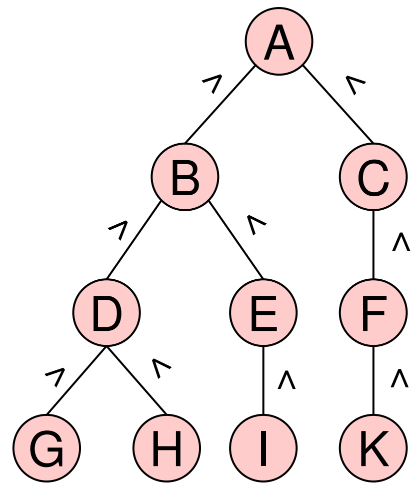
Moral mixed graph



$$P_M(\bar{x}) = \begin{cases} P_B(\bar{x} \mid \bar{x} \in \rho) = \dfrac{P_B(\bar{x})}{P_B(\bar{x} \in \rho)}, \text{if } \bar{x} \in \rho \\ 0, \text{otherwise} \end{cases}$$

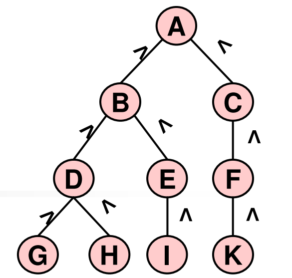
Using look-ahead – example (1)



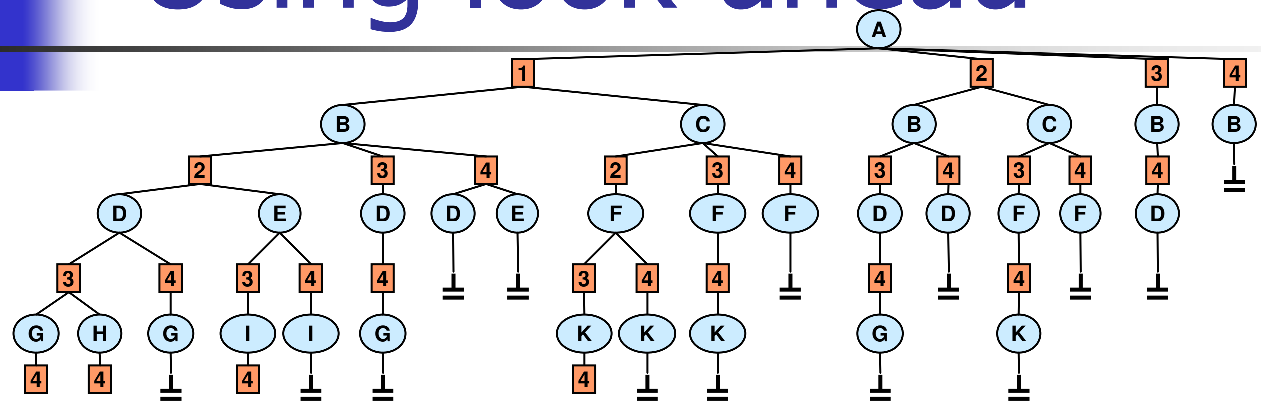


All domains are {1,2,3,4}

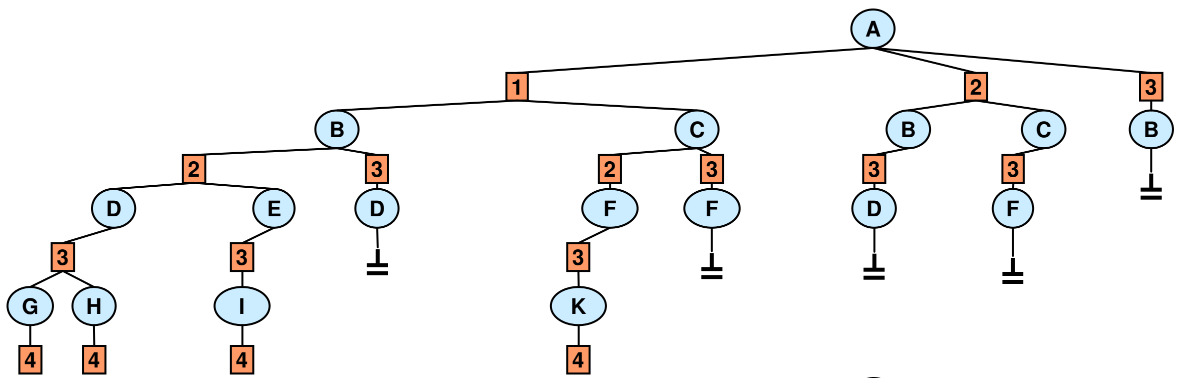
Using look-ahead



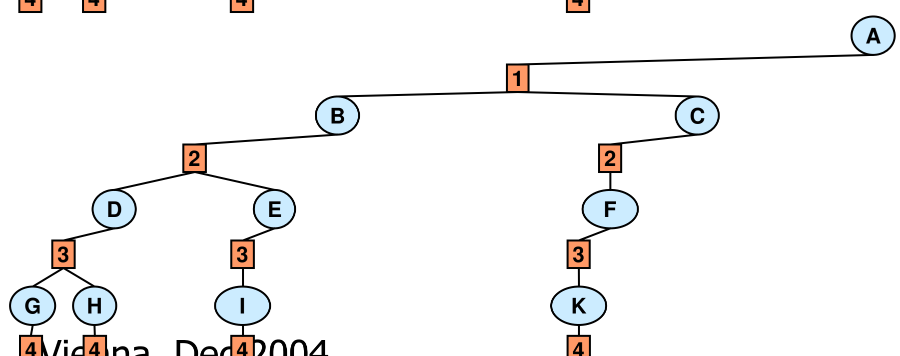
Domains are {1,2,3,4}



CONSTRAINTS ONLY



FORWARD CHECKING



MAINTAINING ARC CONSISTENCY

Experiments – mixed networks

- **Parameters of the mixed networks:**
  - N = number of variables
  - K = number of values per variable

- Belief network - (N,K,R,P):
  - R = number of root nodes
  - P = number of parents
- Measures:
  - Time
  - Number of nodes
  - Number of dead-ends

- Constraint network - (N,K,C,S,t):
  - C = number of constraints
  - S = scope size of the constraints
  - t = tightness (no. of disallowed tuples)
- Algorithms:
  - AO-C constraint checking only
  - AO-FC forward checking
  - AO-RFC relational forward checking

OR vs. AND/OR Spaces

N=25, K=2, R=2, P=2, C=10, S=3, t=30%, 20 instances, w*=9, depth=14				
	Time	Nodes	Dead-ends	Full space
AO-C	0.15	44,895	9,095	152,858
OR-C	11.81	3,147,577	266,215	67,108,862

#CSP N40, K3, C50, P3, 20 inst, w*=13, depth=20
Time (seconds)

	tightness	70%	60%	50\%	40%
	# solutions	**0**	**0**	**46,582**	**147,898,575**
i	**BE**	8.889	8.709	8.531	**8.637**
0	**A/O FC**	**0.110**	**0.454**	**3.129**	32.931
	OR FC	0.113	0.511	14.615	9737.823
3	**A/O FC**	0.111	**0.453**	**3.103**	31.277
	OR FC	0.112	0.509	14.474	9027.365
6	**A/O FC**	0.110	0.454	**3.006**	25.140
	OR FC	0.113	0.508	13.842	7293.472
9	**A/O FC**	0.114	0.453	**2.895**	21.558
	OR FC	0.111	0.509	12.336	5809.917
12	**A/O FC**	0.109	0.458	**2.703**	13.687
	OR FC	0.114	0.496	9.678	2598.778
13	**A/O FC**	0.111	0.457	**2.605**	**11.974**
	OR FC	0.123	0.494	8.703	1170.203

- AND/OR search is orders of magnitudes faster than OR search when problems are loose (consistent)
- AND/OR with full caching equivalent to BE:
  time and space O(exp w*)

Mixed networks results (1)

N=40, K=2, R=2, P=2, C=10, S=4, 20 instances, w*=12, depth=19

tightness	i	Time			Nodes			Dead-ends			#sol
		AO-C	AO-FC	AO-RFC	AO-C	AO-FC	AO-RFC	AO-C	AO-FC	AO-RFC	
80%	0	0.671	0.056	0.022	153,073	4,388	1,066	95,197	3,299	962	1.6E+05
	6	0.479	0.055	0.022	75,397	3,213	936	57,306	3,168	940	
	12	0.103	0.044	0.016	16,579	2,273	683	2,638	1,537	398	
60%	0	2.877	0.791	1.094	774,697	167,921	158,007	239,991	40,069	36,119	7.7E+07
	6	1.409	0.445	0.544	183,286	35,325	31,607	107,362	27,575	24,153	
	12	0.189	0.142	0.149	27,848	9,148	7,357	3,343	3,997	3,048	
40%	0	6.827	4.717	7.427	1,974,952	1,158,544	1,148,044	362,279	162,781	158,968	6.2E+09
	6	2.809	2.219	3.149	346,842	183,895	180,463	150,864	88,822	85,522	
	12	0.255	0.331	0.425	36,262	23,160	22,293	2,825	5,083	4,658	
20%	0	14.181	14.199	21.791	4,282,678	3,703,920	3,702,692	370,314	278,479	277,250	1.1E+11
	6	5.305	6.286	9.061	626,405	519,258	518,029	127,683	98,100	96,872	
	12	0.318	0.543	0.714	44,340	39,550	39,524	1,431	2,647	2,659	
0%	0	23.595	27.129	41.744	7,450,537	7,450,537	7,450,537	0	0	0	1.1E+12
	6	8.325	11.528	16.636	956,965	956,965	956,965	0	0	0	
	12	0.366	0.681	0.884	50,616	50,616	50,616	0	0	0	

- Caching helps more on loose problems
- Constraint propagation helps more on tight problems

Conclusion

- **AND/OR search spaces are a unifying framework for search or compilation applicable to any graphical models.**
- **With caching AND/OR is similar to inference (minimal graphs)**
- **AND/OR time and space bounds are equal to state of the art algorithms**
- **Empirical results**
  - **AND/OR search spaces are always more effective than traditional OR spaces**
  - **AND/OR allows a flexible tradeoff between space and time**
- **Graphical models should always use AND/OR search with embeded inference.**
- **Current work: Hybrid of inference and search: Heuristic generation and Branch and Bound**