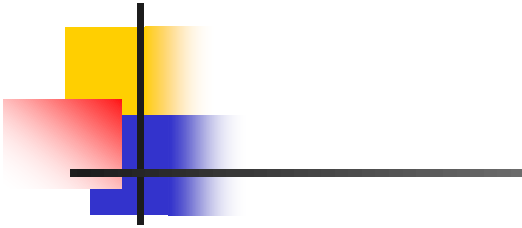


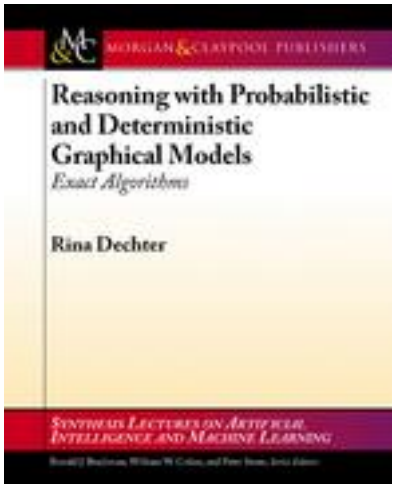
Algorithms for Reasoning with graphical models

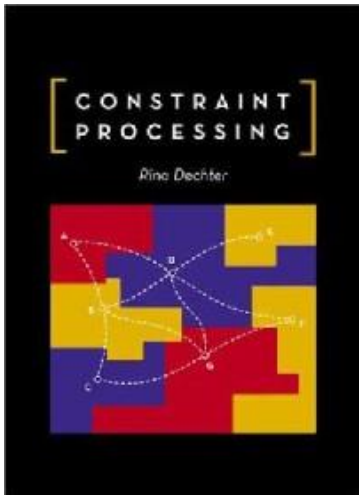
Class2: Constraint Networks

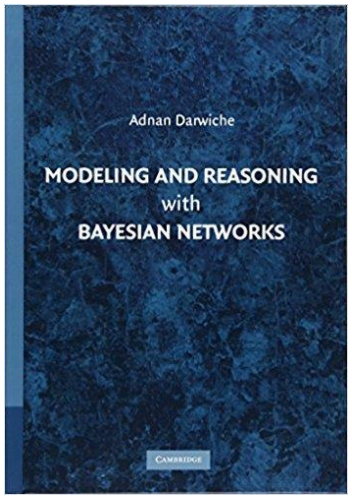
Rina Dechter

Dbook: chapter 2-3,
Constraint book: chapters 2 and 4

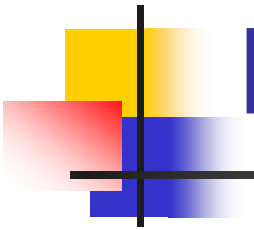
Text Books


Reasoning with Probabilistic and Deterministic Graphical Models
Exact Algorithms
Rina Dechter
Synthesis Lectures on Artificial Intelligence and Machine Learning


CONSTRAINT PROCESSING
Rina Dechter

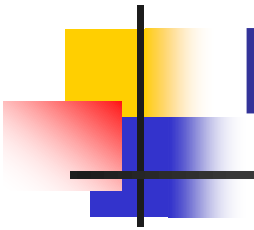

Adnan Darwiche
MODELING AND REASONING with BAYESIAN NETWORKS
CAMBRIDGE

Road Map



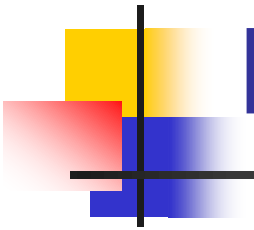
- Graphical models
- Constraint networks Model
- Inference
  - Variable elimination for Constraints
  - Variable elimination for CNFs
  - Variable elimination for Linear Inequalities
  - Constraint propagation
- Search
- Probabilistic Networks

Road Map



- Graphical models
- Constraint networks Model
- Inference
  - Variable elimination for Constraints
  - Variable elimination for CNFs
  - Variable elimination for Linear Inequalities
  - Constraint propagation
- Search
- Probabilistic Networks

Road Map



- Graphical models
- Constraint networks Model
- Inference
  - Variable elimination for Constraints
  - Variable elimination for CNFs
  - Variable elimination for Linear Inequalities
  - Constraint propagation
- Search
- Probabilistic Networks

Sudoku – Approximation: Constraint Propagation

- ***Constraint***
- ***Propagation***
- ***Inference***

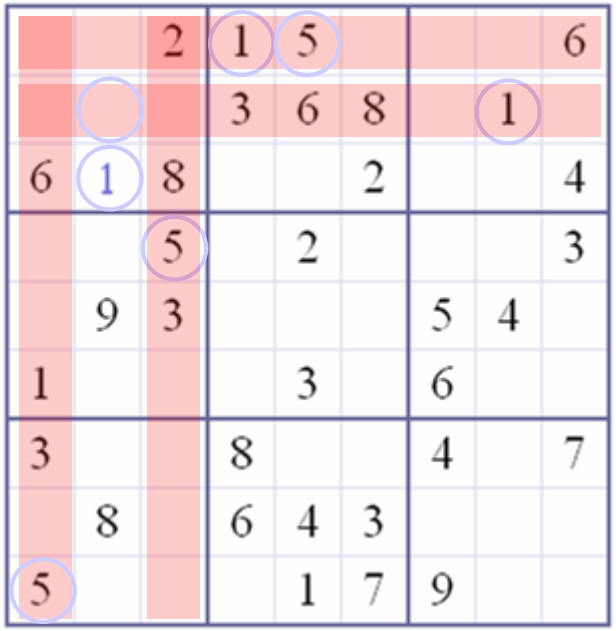
		2	4		6			
8	6	5	1			2		
	1				8	6		9
9				4		8	6	
	4	7				1	9	
	5	8		6				3
4		6	9				7	2 3 ~~2~~ ~~4 6~~
		9			4	5	8	1
			3		2	9		

- ***Variables*: empty slots**
- ***Domains* = {1,2,3,4,5,6,7,8,9}**
- ***Constraints*:**
  - ***27 all-different***

Each row, column and major block must be alldifferent

"Well posed" if it has unique solution: 27 constraints

Sudoku



		2	1	5				6
			3	6	8		1	
6	1	8			2			4
		5		2				3
	9	3				5	4	
1				3		6		
3			8			4		7
	8		6	4	3			
5				1	7	9		

Each row, column and major block must be alldifferent

"Well posed" if it has unique solution

Constraint Networks

A

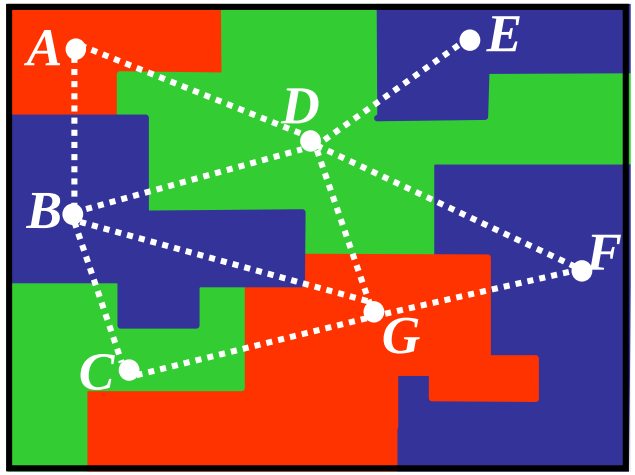
Example: map coloring

Variables - countries (A,B,C,etc.)

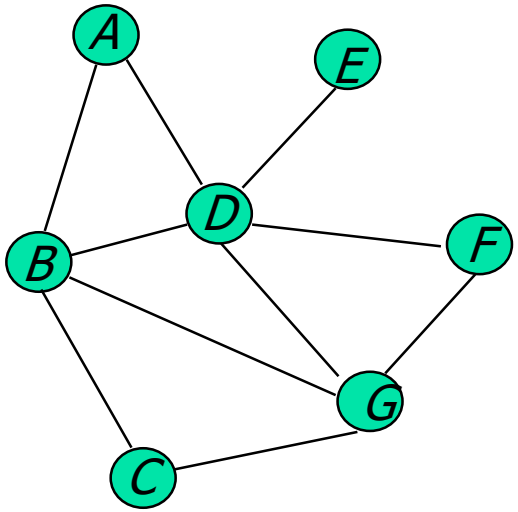
Values - colors (red, green, blue)

Constraints: **A ≠ B, A ≠ D, D ≠ E,** *etc.*

A	B
red	green
red	yellow
green	red
green	yellow
yellow	green
yellow	red



Constraint graph



Constraint Satisfaction Tasks

Example: map coloring

Variables - countries (A,B,C,etc.)

Values - colors (e.g., red, green, yellow)

Constraints:

$A \neq B,\ A \neq D,\ D \neq E,$ *etc.*

A	B	C	D	E…
red	green	red	green	blue
red	blue	green	green	blue
…	…	…	…	green
…	…	…	…	red
red	blue	red	green	red

Are the constraints consistent?

Find a solution, find all solutions

Count all solutions

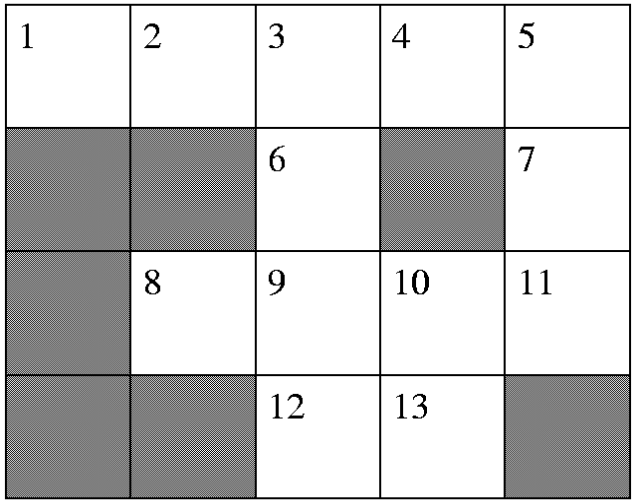
Find a good (optimal) solution

Constraint Network

- A constraint network is: *R=(X,D,C)*
  - **X variables** $X = \{X_1, \ldots, X_n\}$
  - **D domain** $D = \{D_1, \ldots, D_n\}, D_i = \{v_1, \ldots v_k\}$
  - **C constraints** $C = \{C_1, \ldots C_t\}$, $C_i = (S_i, R_i)$
  - ***R* expresses allowed tuples over scopes**
- **A solution** is an assignment to all variables that satisfies all constraints (join of all relations).
- **Tasks:** consistency?, one or all solutions, counting, optimization

Crossword Puzzle

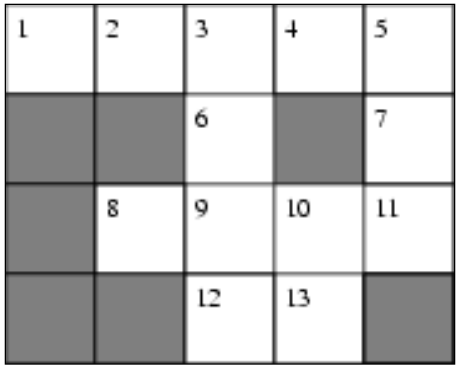
- Variables: x_1, ..., x_{13}
- Domains: letters
- Constraints: words from



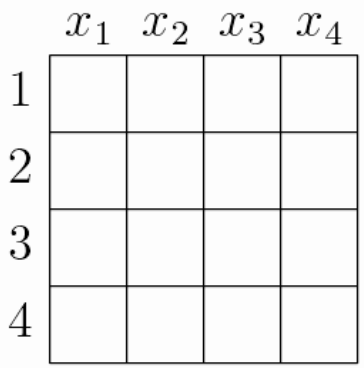
{HOSES, LASER, SHEET, SNAIL, STEER, ALSO, EARN, HIKE, IRON, SAME, EAT, LET, RUN, SUN, TEN, YES, BE, IT, NO, US}

Crossword Puzzle

$$R_{1,2,3,4,5} = \{(H,O,S,E,S),(L,A,S,E,R),(S,H,E,E,T),$$
$$(S,N,A,I,L),(S,T,E,E,R)\}$$
$$R_{3,6,9,12} = \{(H,I,K,E),(I,R,O,N),(K,E,E,T),(E,A,R,N),$$
$$(S,A,M,E)\}$$
$$R_{5,7,11} = \{(R,U,N),(S,U,N),(L,E,T),(Y,E,S),(E,A,T),(T,E,N)\}$$
$$R_{8,9,10,11} = R_{3,6,9,12}$$
$$R_{10,13} = \{(N,O),(B,E),(U,S),(I,T)\}$$
$$R_{12,13} = R_{10,13}$$



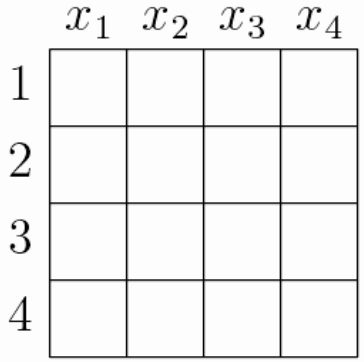
The Queen Problem



(a)

(b)

The Queen Problem



(a)

$$R_{12} = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}$$
$$R_{13} = \{(1,2), (1,4), (2,1), (2,3), (3,2), (3,4), (4,1), (4,3)\}$$
$$R_{14} = \{(1,2), (1,3), (2,1), (2,3), (2,4), (3,1), (3,2), (3,4)$$
$$(4,2), (4,3)\}$$
$$R_{23} = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}$$
$$R_{24} = \{(1,2), (1,4), (2,1), (2,3), (3,2), (3,4), (4,1), (4,3)\}$$
$$R_{34} = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}$$

(b)

The network has four variables, all with domains $D_i = \{1, 2, 3, 4\}$.
(a) The labeled chess board. (b) The constraints between variables.

Varieties of Constraints

Unary constraints involve a single variable,

e.g., SA ≠ green

Binary constraints involve pairs of variables,

e.g., SA ≠ WA

Higher-order constraints involve 3 or more variables,

e.g., cryptarithmetic column constraints

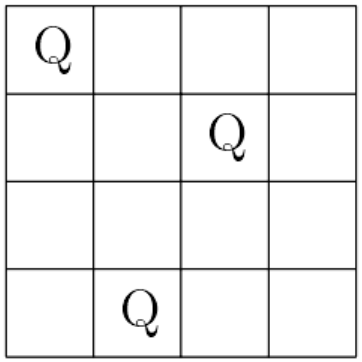
Constraint's Representations

- Relation: allowed tuples

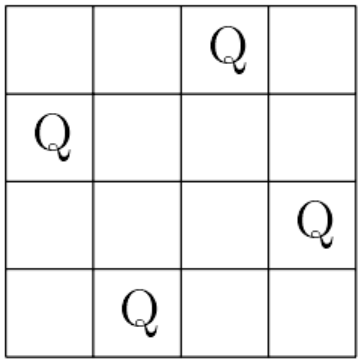
X	Y	Z
1	3	2
2	1	3

- Algebraic expression: $X + Y^2 \leq 10, X \neq Y$
- Propositional formula: $(a \vee b) \rightarrow \neg c$
- Semantics: by a relation

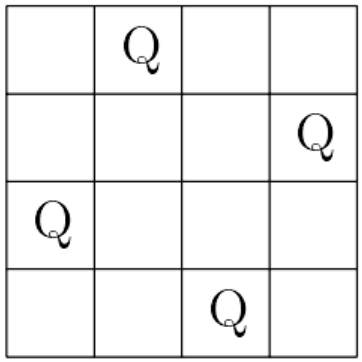
Partial Solutions



(a)



(b)



(c)

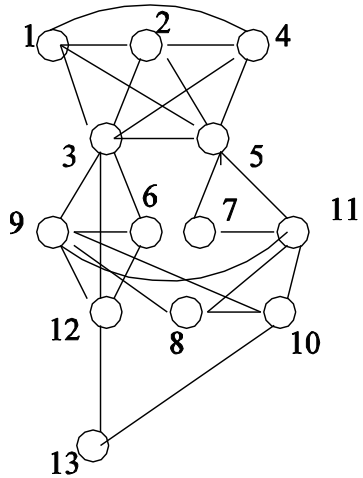
Not all consistent instantiations are part of a solution: (a) A consistent instantiation that is not part of a solution. (b) The placement of the queens corresponding to the solution (2, 4, 1, 3). (c) The placement of the queens corresponding to the solution (3, 1, 4, 2).

Constraint Graphs:
Primal, dual and hypergraphs

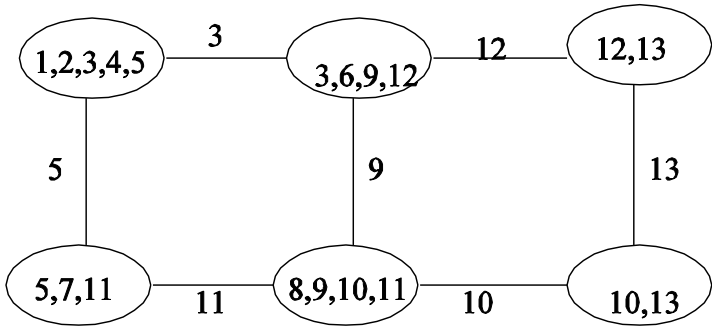
When variables are squares:

1	2	3	4	5
		6		7
	8	9	10	11
		12	13	

A (primal) constraint graph: a node per variable arcs connect constrained variables.
A dual constraint graph: a node per constraint's scope, an arc connect nodes sharing variables =hypergraph



(a)



(b)

Graph Concepts

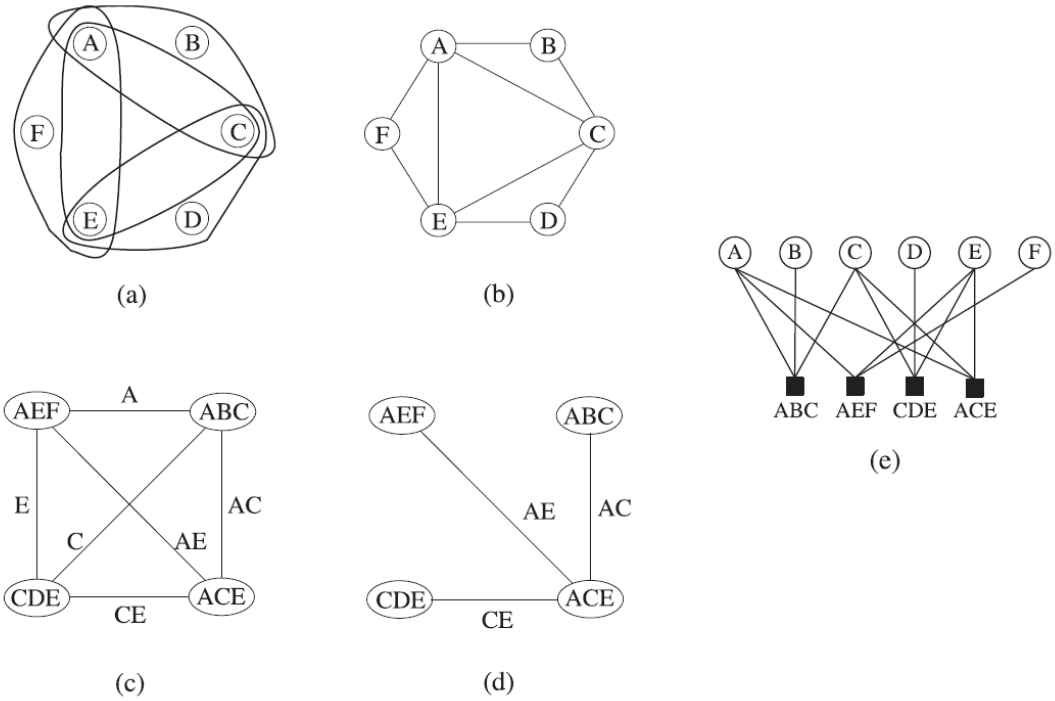
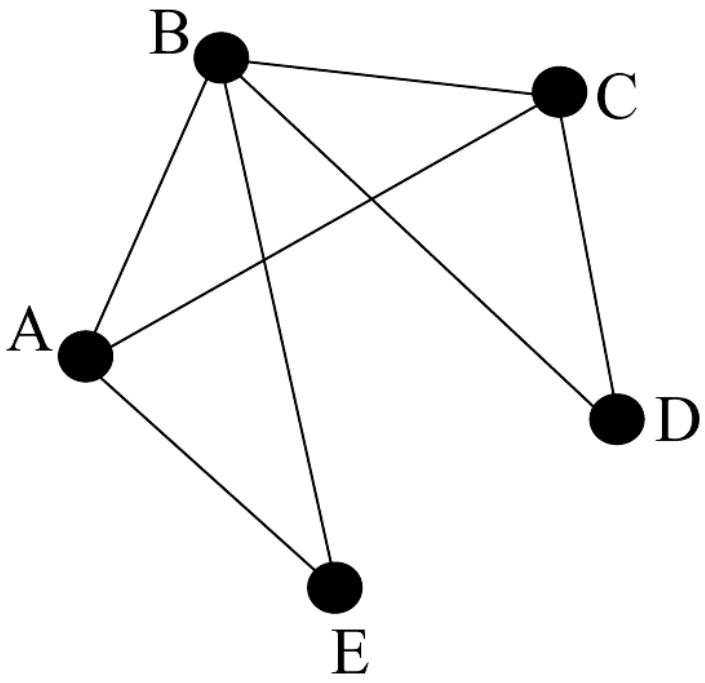


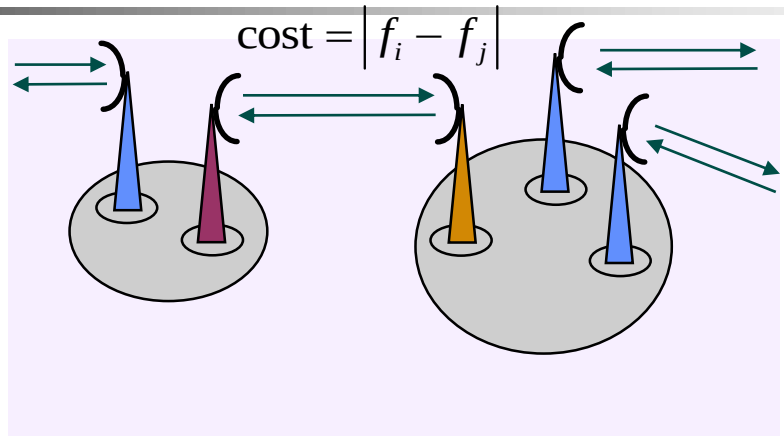
Figure 2.1: (a) Hyper; (b) primal; (c) dual; (d) join-tree of a graphical model having scopes ABC, AEF, CDE and ACE; and (e) the factor graph.

Propositional Satisfiability

φ = {(¬C), (A v B v C), (¬A v B v E), (¬B v C v D)}.



Example: Radio Link Assignment



Given a telecommunication network (where each communication link has various antenas) , assign a frequency to each antenna in such a way that all antennas may operate together without noticeable interference.

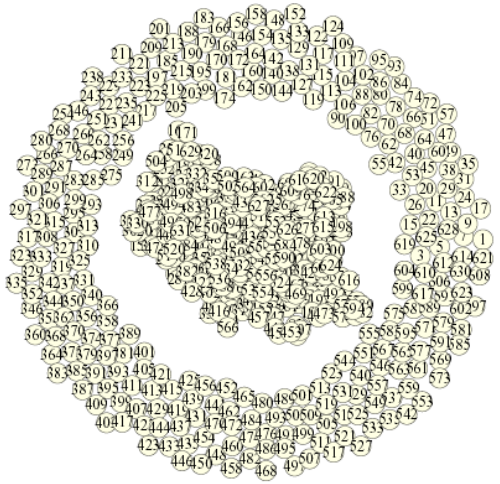
Encoding?

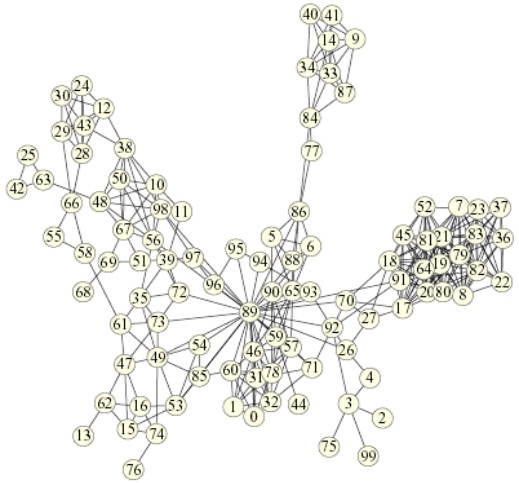
Variables: one for each antenna

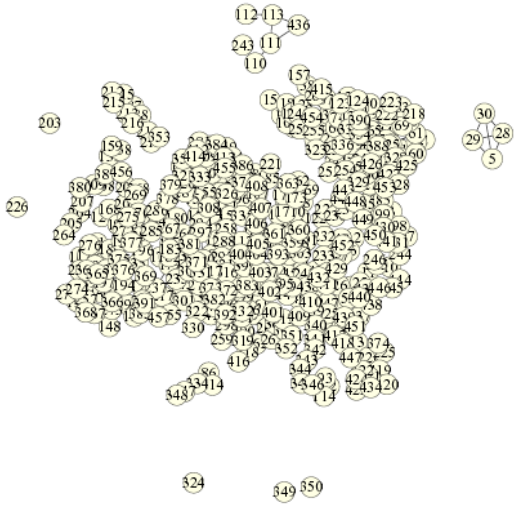
Domains: the set of available frequencies

Constraints: the ones referring to the antennas in the same communication link

Constraint graphs of 3 instances of the Radio frequency assignment problem in CELAR's benchmark







Operations With Relations

- Intersection
- Union
- Difference
- Selection
- Projection
- Join
- Composition

Combination

$f \bowtie g$

- Join :

x_1	x_2
a	a
b	b

$\bowtie$

x_2	x_3
a	a
a	b
b	a

$=$

x_1	x_2	x_3
a	a	a
a	a	b
b	b	a

$f \wedge g$

- Logical AND:

x_1	x_2	f
a	a	true
a	b	false
b	a	false
b	b	true

$\wedge$

x_2	x_3	g
a	a	true
a	b	true
b	a	true
b	b	false

$=$

x_1	x_2	x_3	h
a	a	a	true
a	a	b	true
a	b	a	false
a	b	b	false
b	a	a	false
b	a	b	false
b	b	a	true
b	b	b	false

Example of Selection, Projection and Join

x_1	x_2	x_3
a	b	c
b	b	c
c	b	c
c	b	s

(a) Relation R

x_1	x_2	x_3
b	b	c
c	b	c
c	n	n

(b) Relation R'

x_2	x_3	x_4
a	a	1
b	c	2
b	c	3

(c) Relation R''

x_1	x_2	x_3
b	b	c
c	b	c

(a) $\sigma_{x_3=c}(R')$

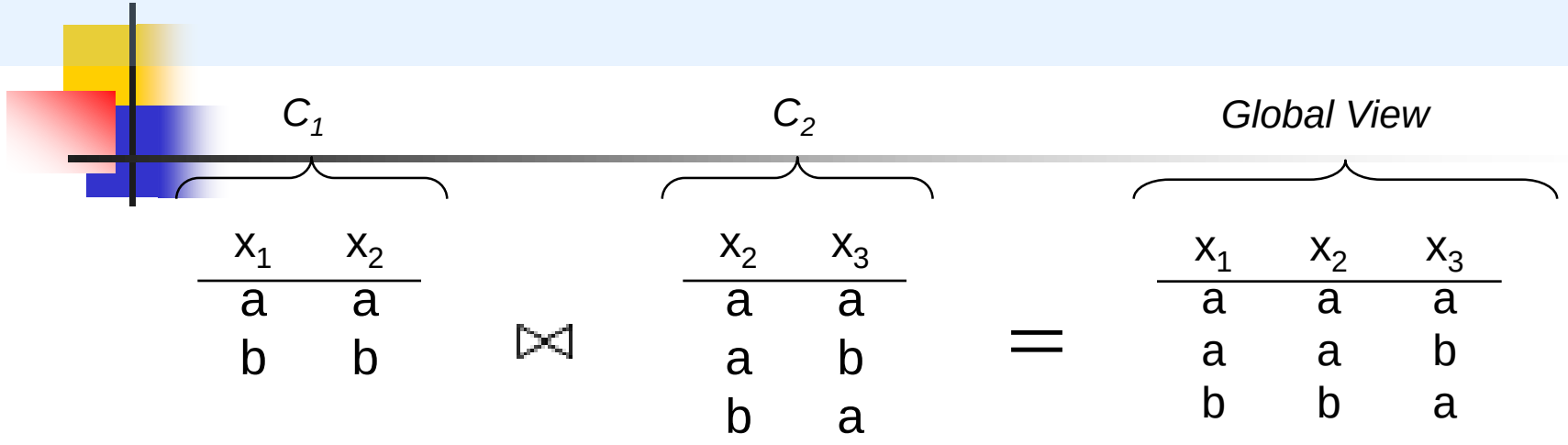
x_2	x_3
b	c
n	n

(b) $\pi_{\{x_2,x_3\}}(R')$

x_1	x_2	x_3	x_4
b	b	c	2
b	b	c	3
c	b	c	2
c	b	c	3

(c) $R' \bowtie R''$

Global View of the Problem



What about counting?

TASK

x_1	x_2	x_3	h
a	a	a	true
a	a	b	true
a	b	a	false
a	b	b	false
b	a	a	false
b	a	b	false
b	b	a	true
b	b	b	false

Number of true tuples

true is 1
false is 0
logical AND?

x_1	x_2	x_3	h
a	a	a	1
a	a	b	1
a	b	a	0
a	b	b	0
b	a	a	0
b	a	b	0
b	b	a	1
b	b	b	0

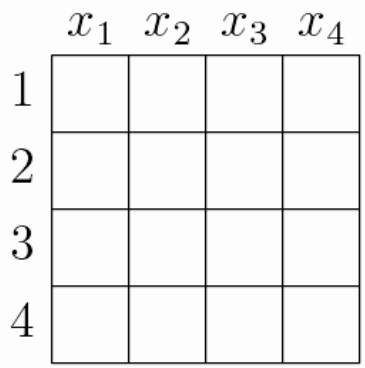
Sum over all the tuples

The minimal network,

An extreme case of re-parameterization

The N-queens Constraint Network

The network has four variables, all with domains Di = {1, 2, 3, 4}.
(a) The labeled chess board. (b) The constraints between variables.



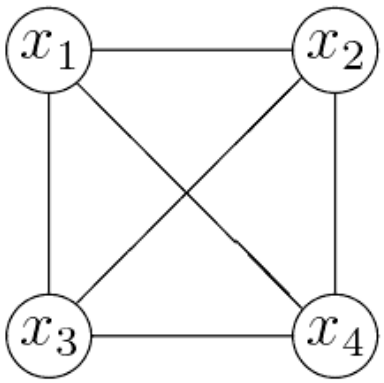
(a)

$$R_{12} = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}$$
$$R_{13} = \{(1,2), (1,4), (2,1), (2,3), (3,2), (3,4), (4,1), (4,3)\}$$
$$R_{14} = \{(1,2), (1,3), (2,1), (2,3), (2,4), (3,1), (3,2), (3,4)$$
$$(4,2), (4,3)\}$$
$$R_{23} = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}$$
$$R_{24} = \{(1,2), (1,4), (2,1), (2,3), (3,2), (3,4), (4,1), (4,3)\}$$
$$R_{34} = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}$$

(b)

Solutions are: (2,4,1,3) (3,1,4,2)

Figure 2.11: The 4-queens constraint network: (a) The constraint graph. (b) The minimal binary constraints. (c) The minimal unary constraints (the domains).



(a)

$$M_{12} = \{(2,4), (3,1)\}$$
$$M_{13} = \{(2,1), (3,4)\}$$
$$M_{14} = \{(2,3), (3,2)\}$$
$$M_{23} = \{(1,4), (4,1)\}$$
$$M_{24} = \{(1,2), (4,3)\}$$
$$M_{34} = \{(1,3), (4,2)\}$$

(b)

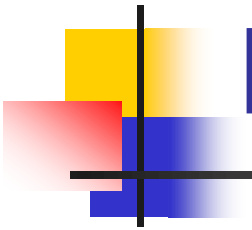
$$D_1 = \{1,3\}$$
$$D_2 = \{1,4\}$$
$$D_3 = \{1,4\}$$
$$D_4 = \{1,3\}$$

(c)

Solutions are: (2,4,1,3) (3,1,4,2)

The Minimal Network

- The minimal network is perfectly explicit for binary and unary constraints:
  - Every pair of values permitted by the minimal constraint is in a solution.

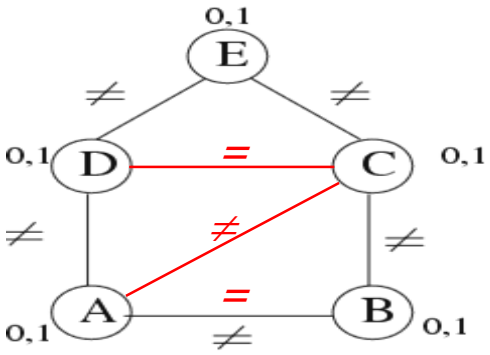


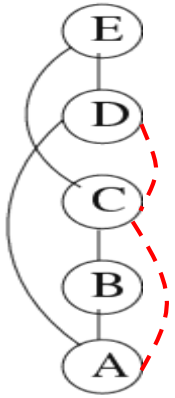
Road Map

- Graphical models
- Constraint networks Model
- Inference
  - Variable elimination for Constraints
  - Variable elimination for CNFs
  - Variable elimination for Linear Inequalities
  - Constraint propagation
- Search
- Probabilistic Networks

Bucket Elimination

Adaptive Consistency (Dechter & Pearl, 1987)





Bucket E: $E \neq D,\ E \neq C$

Bucket D: $D \neq A$ → $\boldsymbol{D = C}$

Bucket C: $C \neq B$ → $\boldsymbol{A \neq C}$

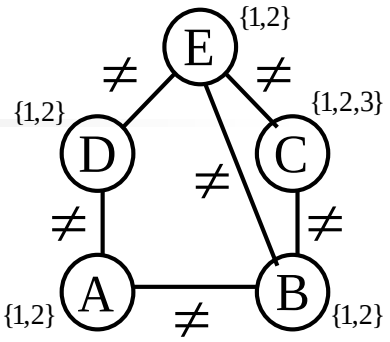
Bucket B: $B \neq A$ → $\boldsymbol{B = A}$

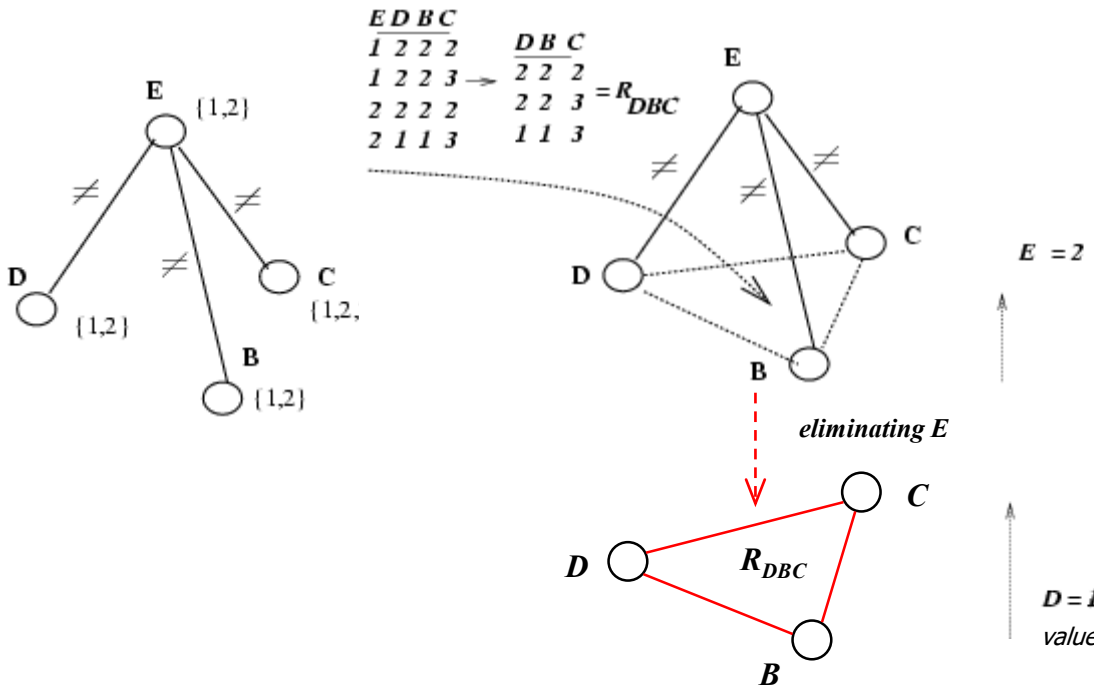
Bucket A: → ***contradiction***

Complexity: $O(n \exp(w^*))$

w^* - ***induced width***

The Idea of Elimination

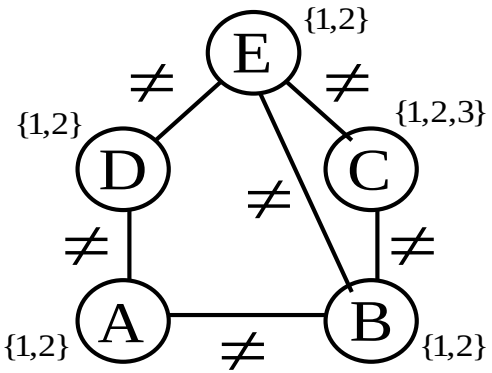




$$R_{DBC} = \prod_{DBC} R_{ED} \bowtie R_{EB} \bowtie R_{EC}$$

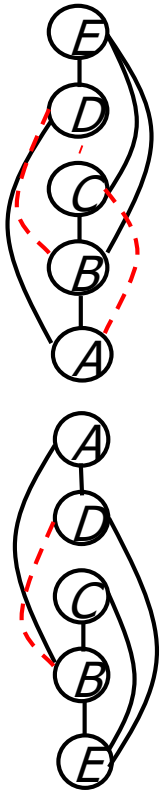
Eliminate variable E $\Leftrightarrow$ join and project

Bucket-Elimination



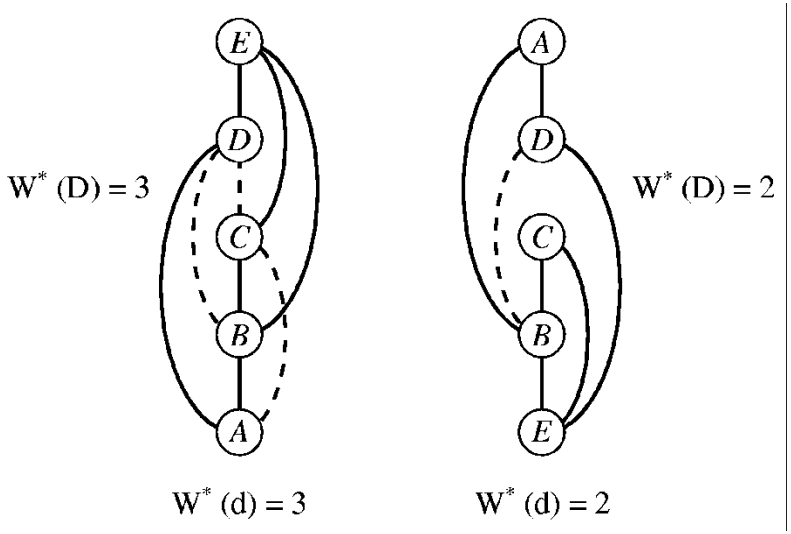
$Bucket(E)$: E ≠ D, E ≠ C, E ≠ B
$Bucket(D)$: D ≠ A || R_{DCB}
$Bucket(C)$: C ≠ B || R_{ACB}
$Bucket(B)$: B ≠ A || R_{AB}
$Bucket(A)$: R_A

$Bucket(A)$: A ≠ D, A ≠ B
$Bucket(D)$: D ≠ E || R_{DB}
$Bucket(C)$: C ≠ B, C ≠ E
$Bucket(B)$: B ≠ E || R^D_{BE}, R^C_{BE}
$Bucket(E)$: || R_E



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

The Induced-Width



- Width along *d*, w(d):
  - max # of previous parents
- Induced width w*(d):
  - The width in the ordered *induced graph*
- Induced-width w*:
  - Smallest induced-width over all orderings
- Finding w*
  - NP-complete *(Arnborg, 1985) but greedy heuristics (min-fill).*

Adaptive-Consistency, Bucket-Elimination

Initialize: partition constraints into $bucket_1, \ldots, bucket_n$

For *i=n* down to *1* along d (process in reverse order)

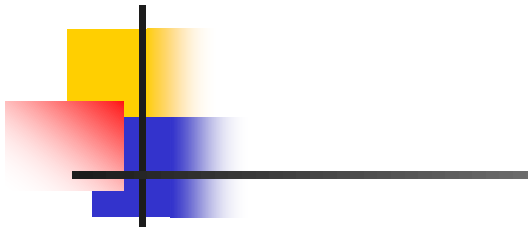
for all relations $R_1, \ldots, R_m \in bucket_i$ **do**

(join all relations and "project-out" X_i)

$$R_{new} \leftarrow \prod_{(-X_i)} (\bowtie_j R_j)$$

If R_{new} *is not empty, add it to* $bucket_k, k < i,$
where k is the largest variable index in R_{new}
Else *problem is unsatisfiable*

Return *the set of all relations (old and new) in the buckets*

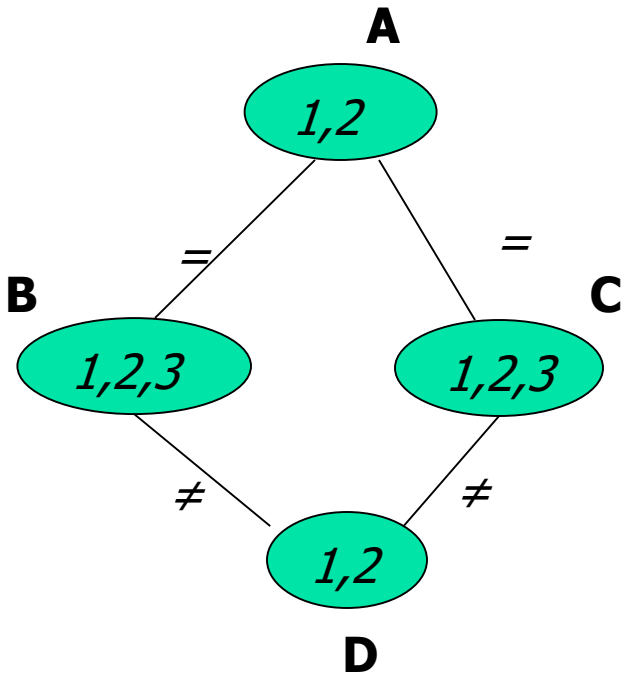


Algorithms for Reasoning with graphical models

Class3

Rina Dechter

Example: deadends, backtrack-freeness



Assign values in the order
D,B,C,A before and after adaptive-consistence

Order A,B,C,D, order A,B,D,C

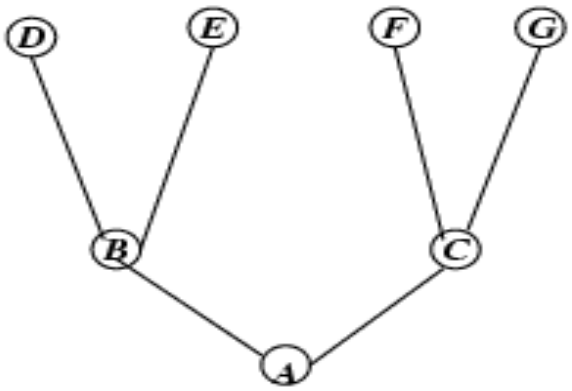
Properties of Adaptive-Consistency

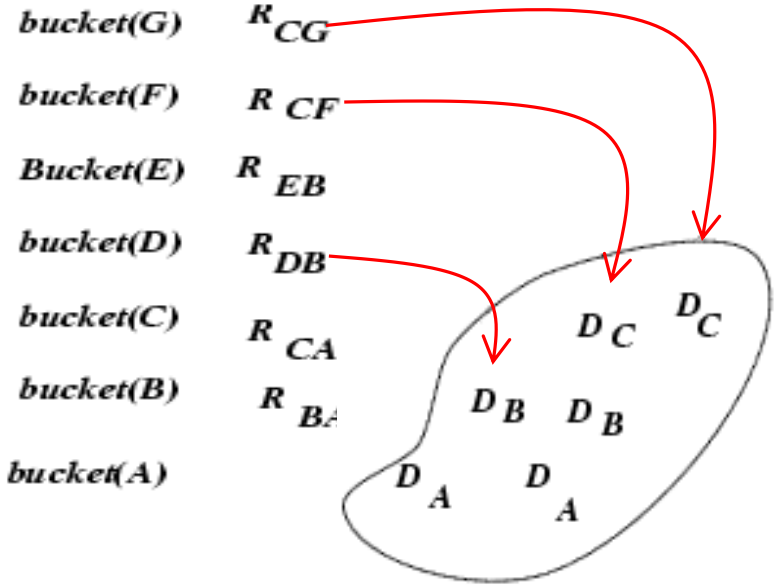
- Adaptive-consistency generates a constraint network that is backtrack-free (can be solved without dead-ends).
- The time and space complexity of adaptive-consistency along ordering *d* is time and memory exponential in w*(d) .
- Therefore, problems having bounded induced width are tractable (solved in polynomial time).
  - *trees* (w*=1),
  - *series-parallel networks* (w*=2),
  - and in general *k-trees* (w*=k).

Solving Trees

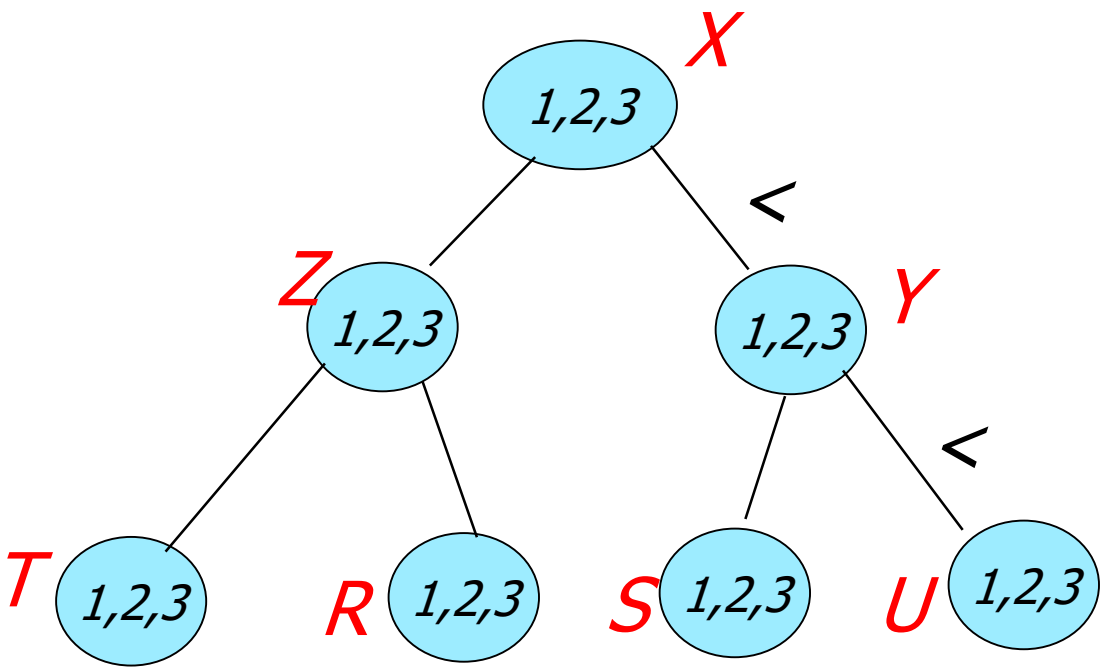
(Mackworth and Freuder, 1985)

Adaptive consistency is linear for trees and equivalent to enforcing directional arc-consistency (recording only unary constraints)

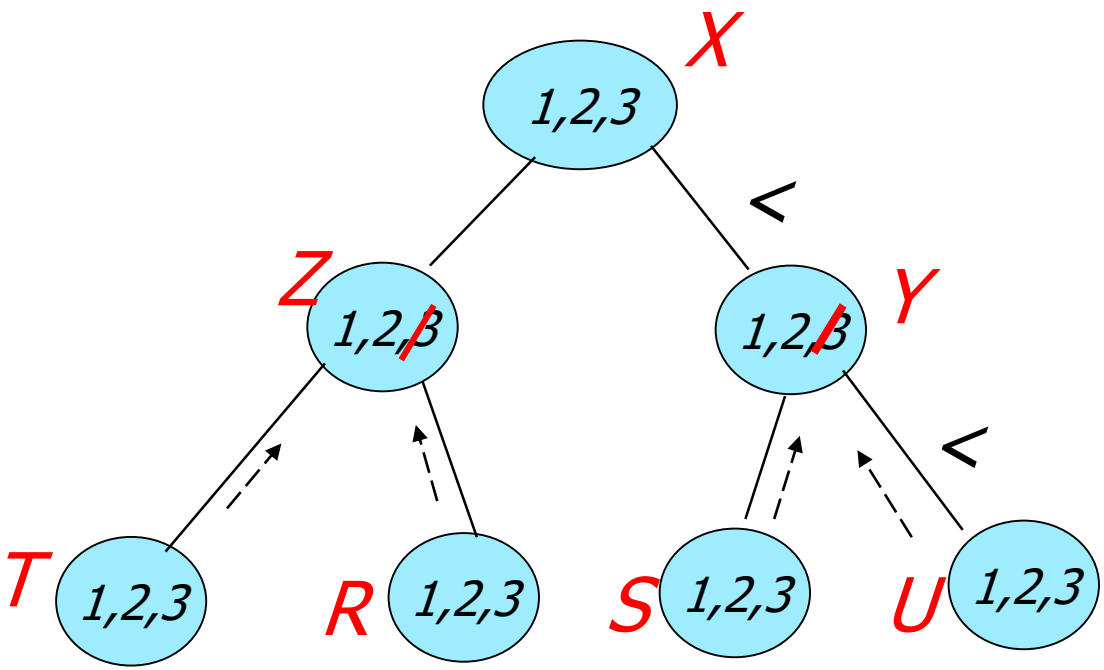




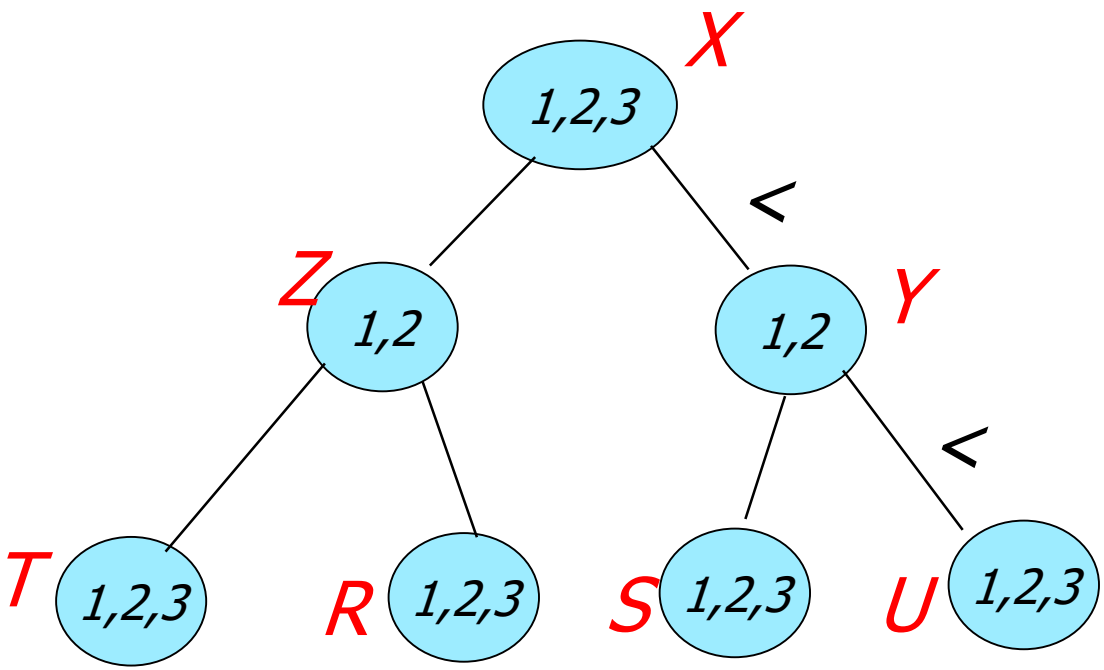
Tree Solving is Easy



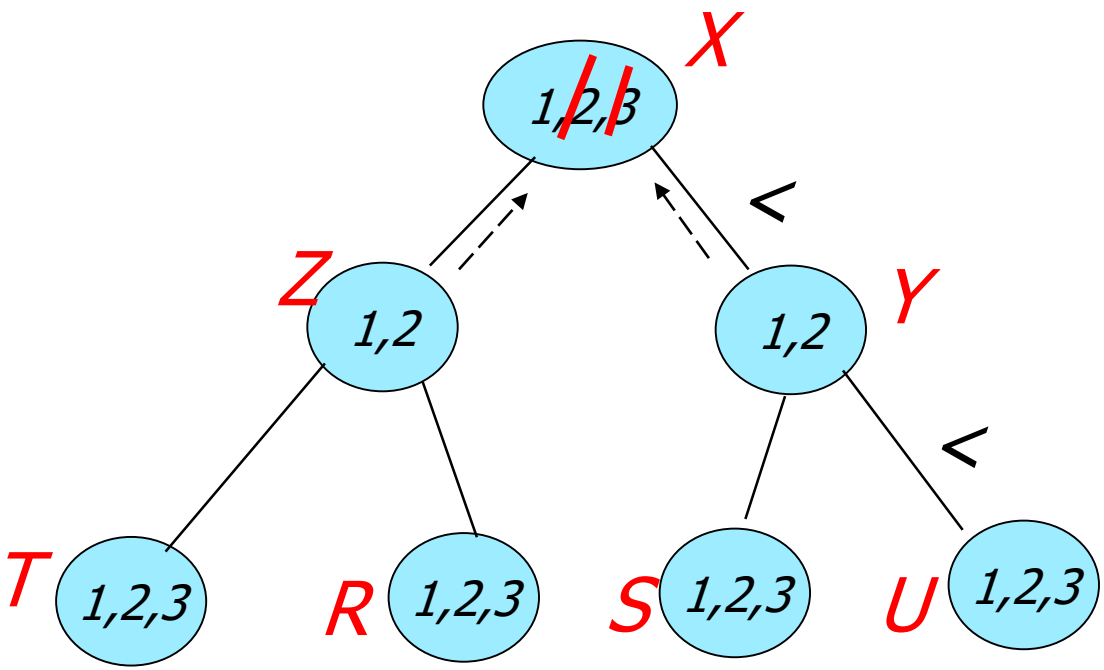
Tree Solving is Easy



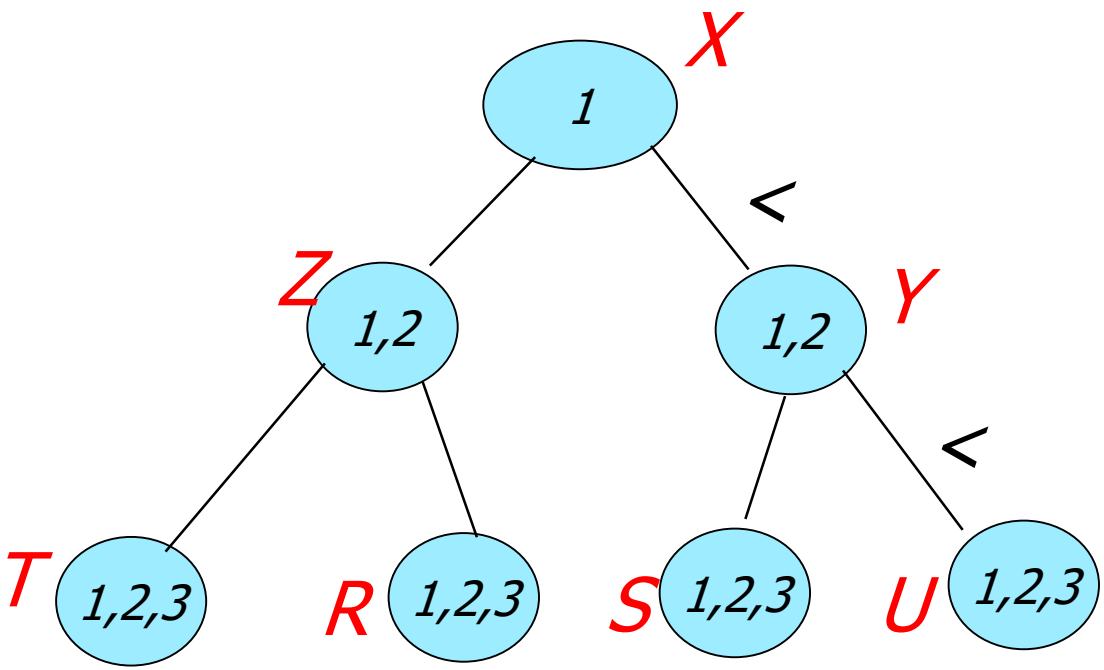
Tree Solving is Easy



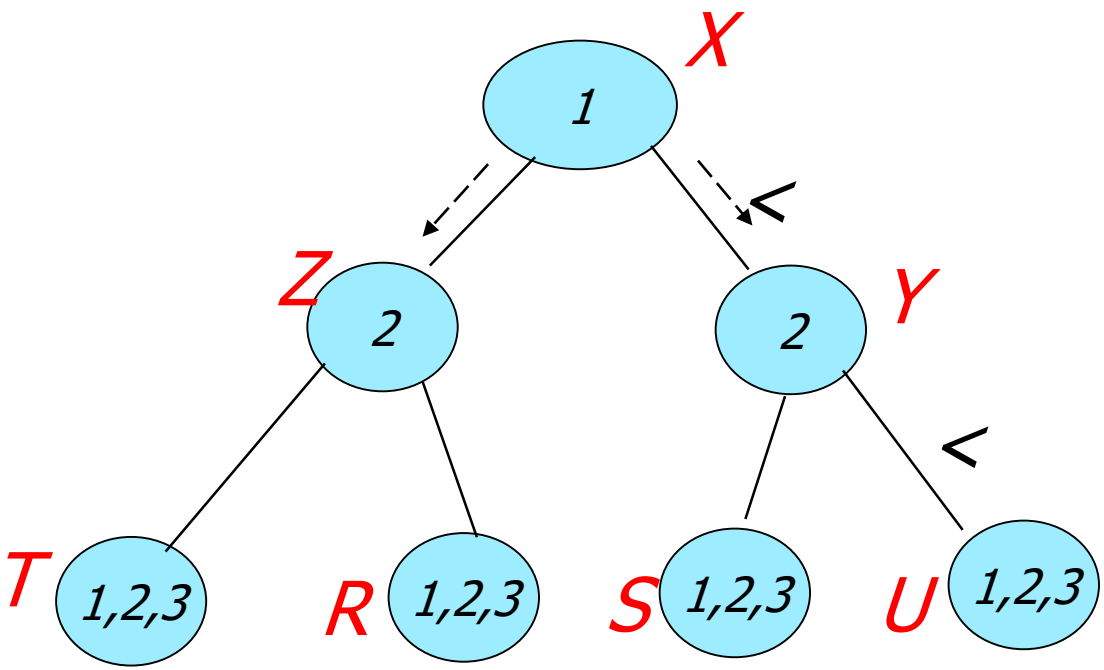
Tree Solving is Easy



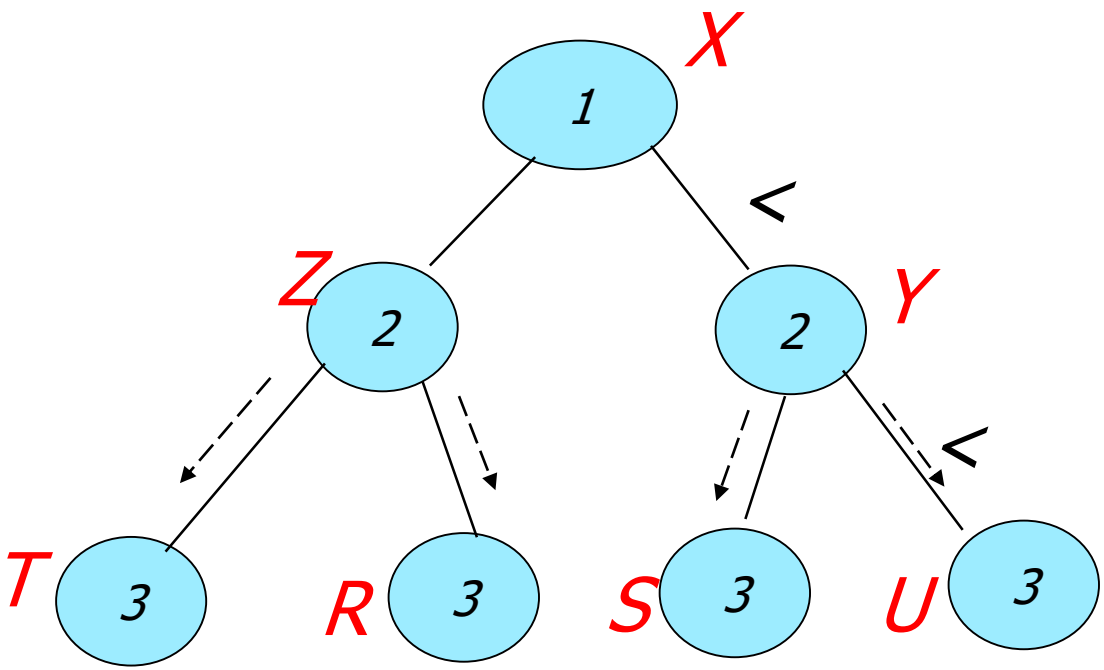
Tree Solving is Easy



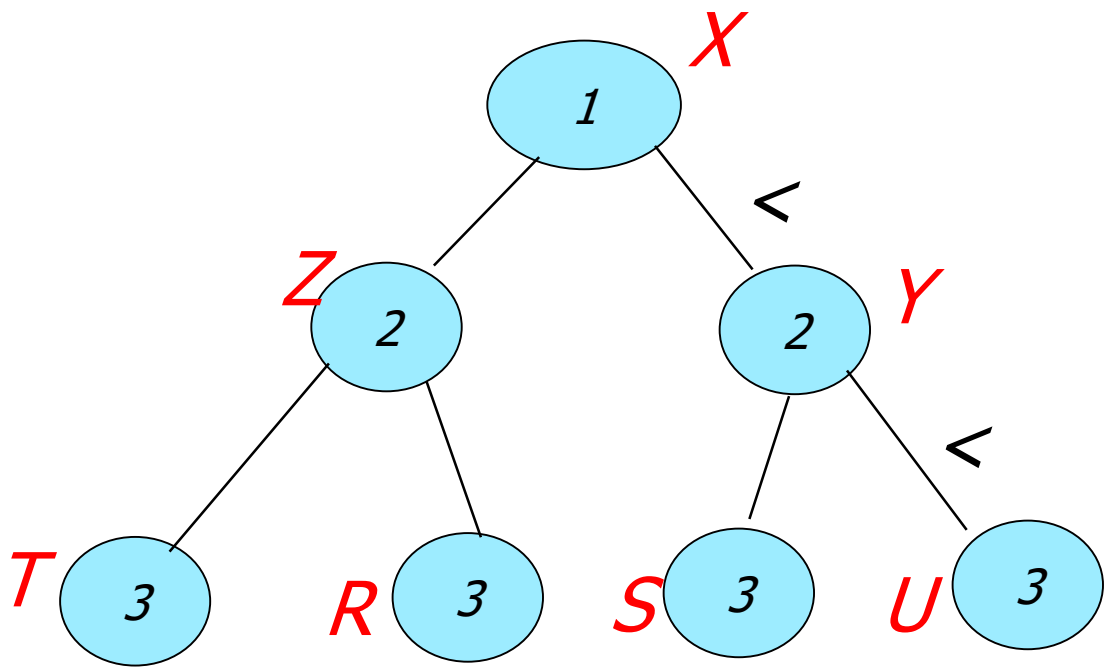
Tree Solving is Easy



Tree Solving is Easy



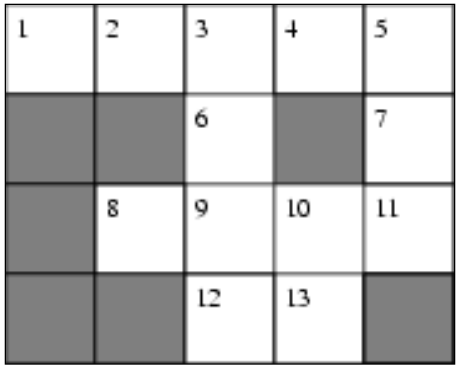
Tree Solving is Easy



Adaptive-consistency is linear time because induced-width is 1
(Constraint propagation Solves trees in linear time)

Example: crossword puzzle

$$R_{1,2,3,4,5} = \{(H,O,S,E,S),(L,A,S,E,R),(S,H,E,E,T),$$
$$(S,N,A,I,L),(S,T,E,E,R)\}$$
$$R_{3,6,9,12} = \{(H,I,K,E),(A,R,O,N),(K,E,E,T),(E,A,R,N),$$
$$(S,A,M,E)\}$$
$$R_{5,7,11} = \{(R,U,N),(S,U,N),(L,E,T),(Y,E,S),(E,A,T),(T,E,N)\}$$
$$R_{8,9,10,11} = R_{3,6,9,12}$$
$$R_{10,13} = \{(N,O),(B,E),(U,S),(I,T)\}$$
$$R_{12,13} = R_{10,13}$$



Adaptive-Consistency on the Crossword Puzzle

$$R_{1,2,3,4,5} = \{(H,O,S,E,S),(L,A,S,E,R),(S,H,E,E,T),(S,N,A,I,L),(S,T,E,E,R)\}$$
$$R_{3,6,9,12} = \{(H,I,K,E),(A,R,O,N),(K,E,E,T),(E,A,R,N),(S,A,M,E)\}$$
$$R_{5,7,11} = \{(R,U,N),(S,U,N),(L,E,T),(Y,E,S),(E,A,T),(T,E,N)\}$$
$$R_{8,9,10,11} = R_{3,6,9,12}$$
$$R_{10,13} = \{(N,O),(B,E),(U,S),(I,T)\}$$
$$R_{12,13} = R_{10,13}$$

1	2	3	4	5
		6		7
	8	9	10	11
		12	13	

$bucket(x_1)$

$bucket(x_2)$

$bucket(x_3)$

$bucket(x_4)$

$bucket(x_5)$

$bucket(x_6)$

$bucket(x_7)$

$bucket(x_8)$

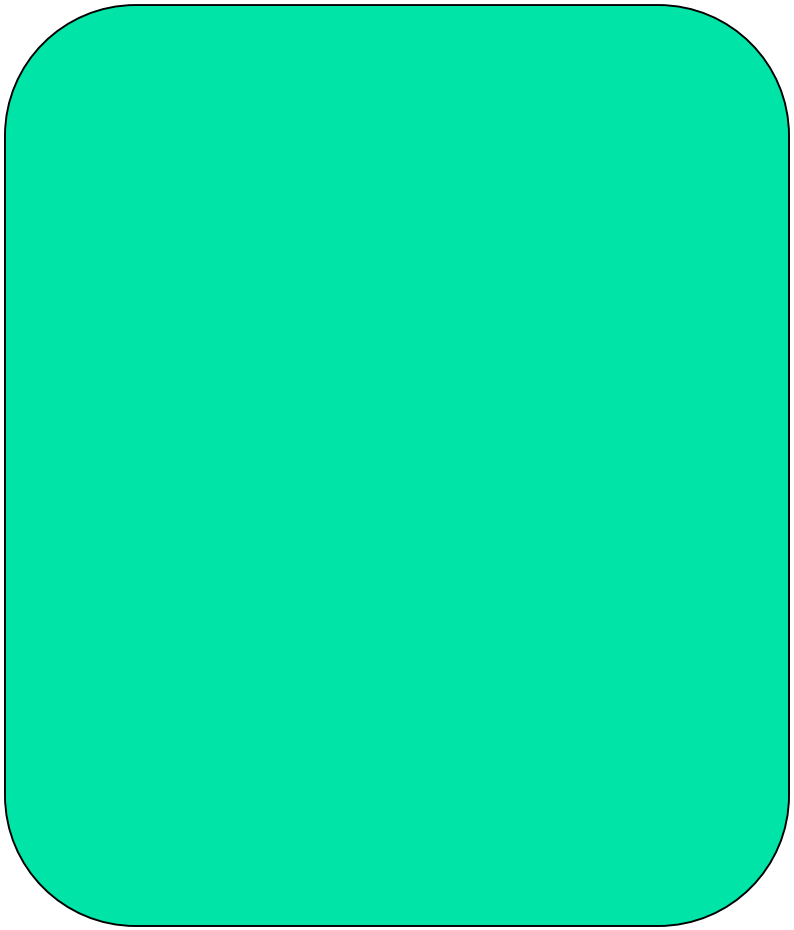
$bucket(x_9)$

$bucket(x_{10})$

$bucket(x_{11})$

$bucket(x_{12})$

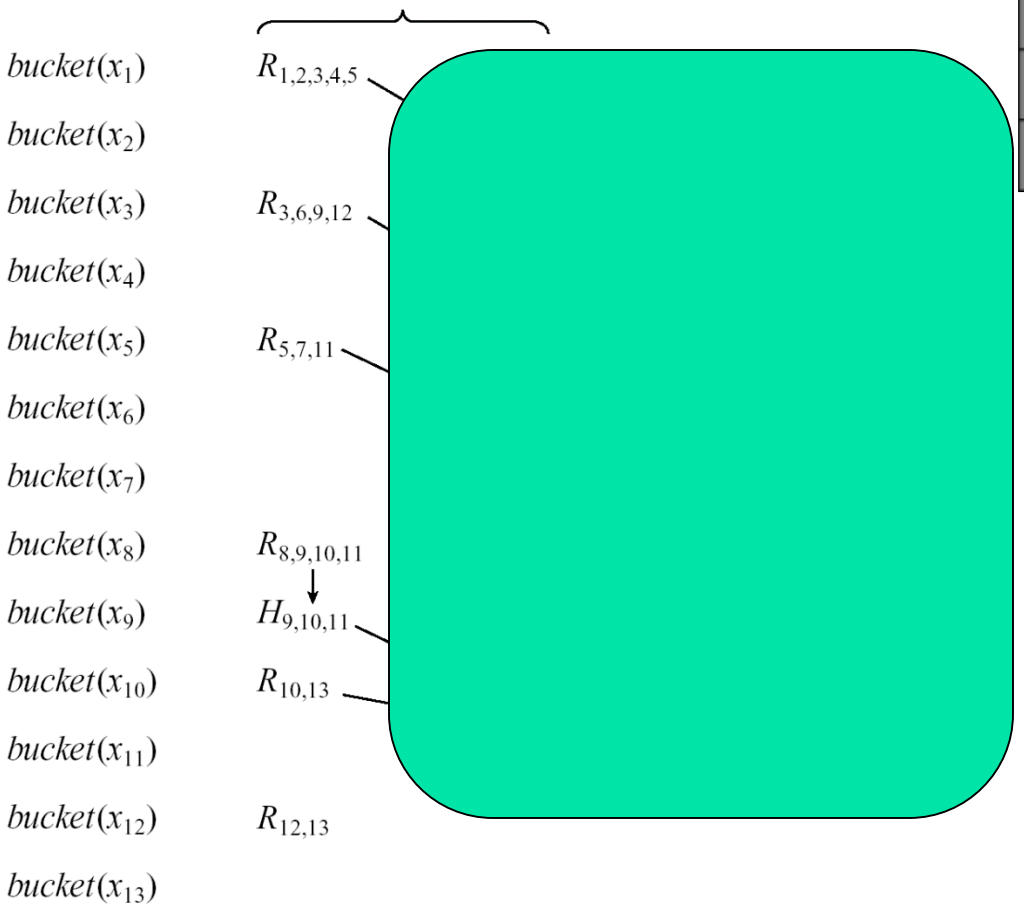
$bucket(x_{13})$



Adaptive-Consistency on the Crossword Puzzle

$R_{1,2,3,4,5} = \{(H,O,S,E,S),(L,A,S,E,R),(S,H,E,E,T),$
$(S,N,A,I,L),(S,T,E,E,R)\}$
$R_{3,6,9,12} = \{(H,I,K,E),(A,R,O,N),(K,E,E,T),(E,A,R,N),$
$(S,A,M,E)\}$
$R_{5,7,11} = \{(R,U,N),(S,U,N),(L,E,T),(Y,E,S),(E,A,T),(T,E,N)\}$
$R_{8,9,10,11} = R_{3,6,9,12}$
$R_{10,13} = \{(N,O),(B,E),(U,S),(I,T)\}$
$R_{12,13} = R_{10,13}$

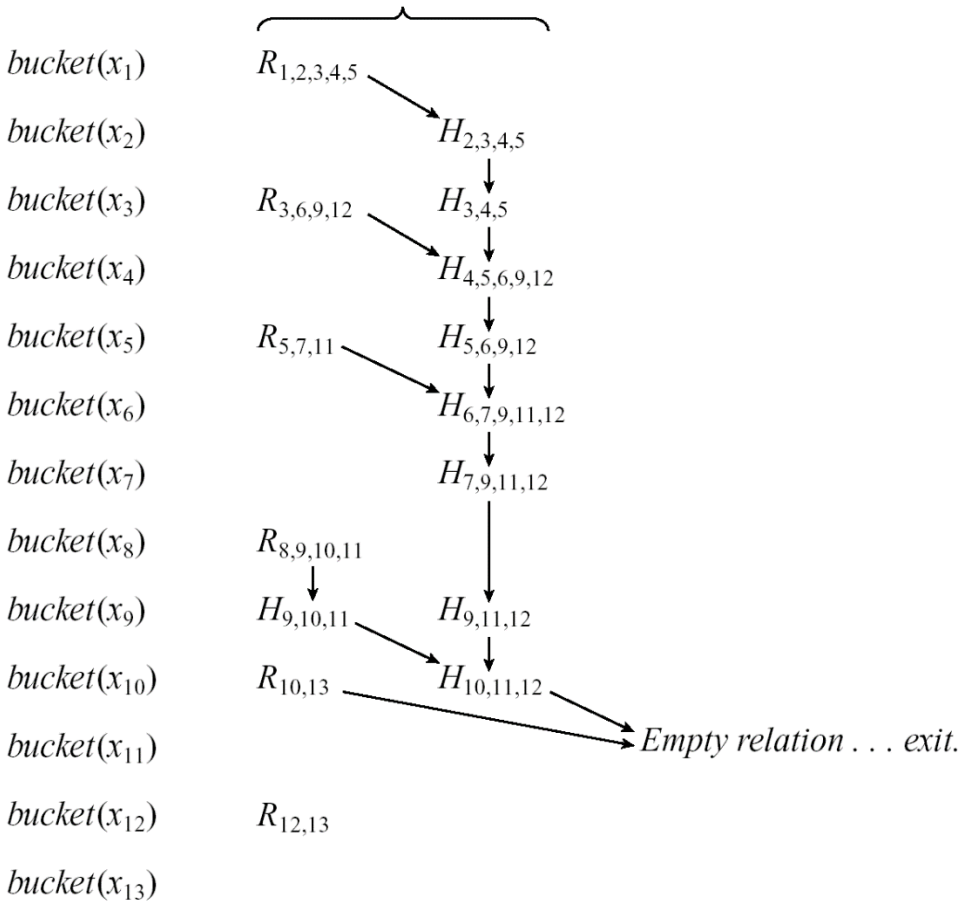
1	2	3	4	5
		6		7
	8	9	10	11
		12	13	



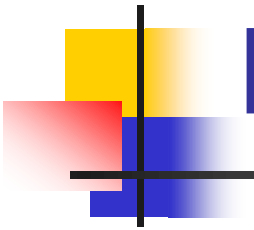
Adaptive-Consistency on the Crossword Puzzle

$R_{1,2,3,4,5} = \{(H,O,S,E,S),(L,A,S,E,R),(S,H,E,E,T),$
$(S,N,A,I,L),(S,T,E,E,R)\}$
$R_{3,6,9,12} = \{(H,I,K,E),(A,R,O,N),(K,E,E,T),(E,A,R,N),$
$(S,A,M,E)\}$
$R_{5,7,11} = \{(R,U,N),(S,U,N),(L,E,T),(Y,E,S),(E,A,T),(T,E,N)\}$
$R_{8,9,10,11} = R_{3,6,9,12}$
$R_{10,13} = \{(N,O),(B,E),(U,S),(I,T)\}$
$R_{12,13} = R_{10,13}$

1	2	3	4	5
		6		7
	8	9	10	11
		12	13	



Road Map



- Graphical models
- Constraint networks Model
- Inference
  - Variable elimination for Constraints
  - Variable elimination for CNFs
  - Variable elimination for Linear Inequalities
  - Constraint propagation
- Search
- Probabilistic Networks

Gausian and Boolean Propagation, Resolution

- Linear inequalities

$$x + y + z \leq 15, z \geq 13 \Rightarrow$$

$$x \leq 2, y \leq 2$$

- Boolean constraint propagation, unit resolution

$$(A \vee B \vee \neg C), (\neg B) \Rightarrow$$

$$(A \vee \neg C)$$

The Effect of Resolution on Its Graph

(~C) (AVBVC) (~AvBvE)(~B,C,E)

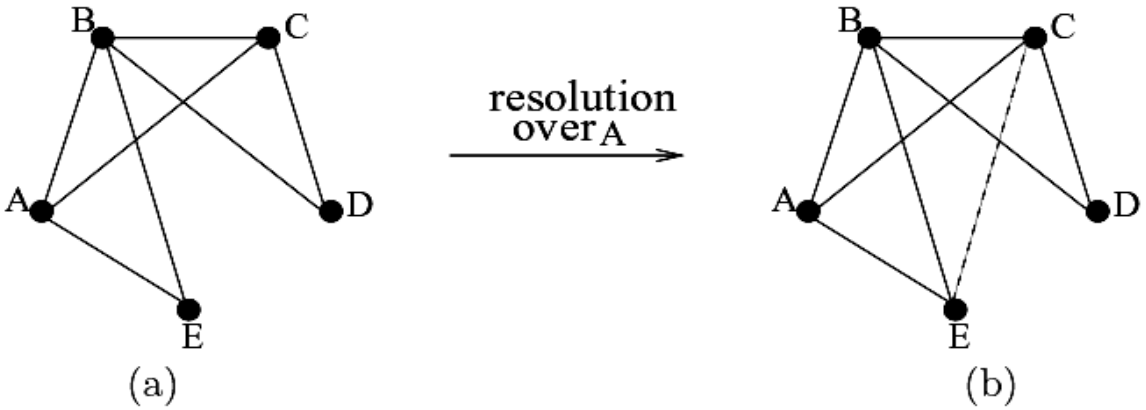
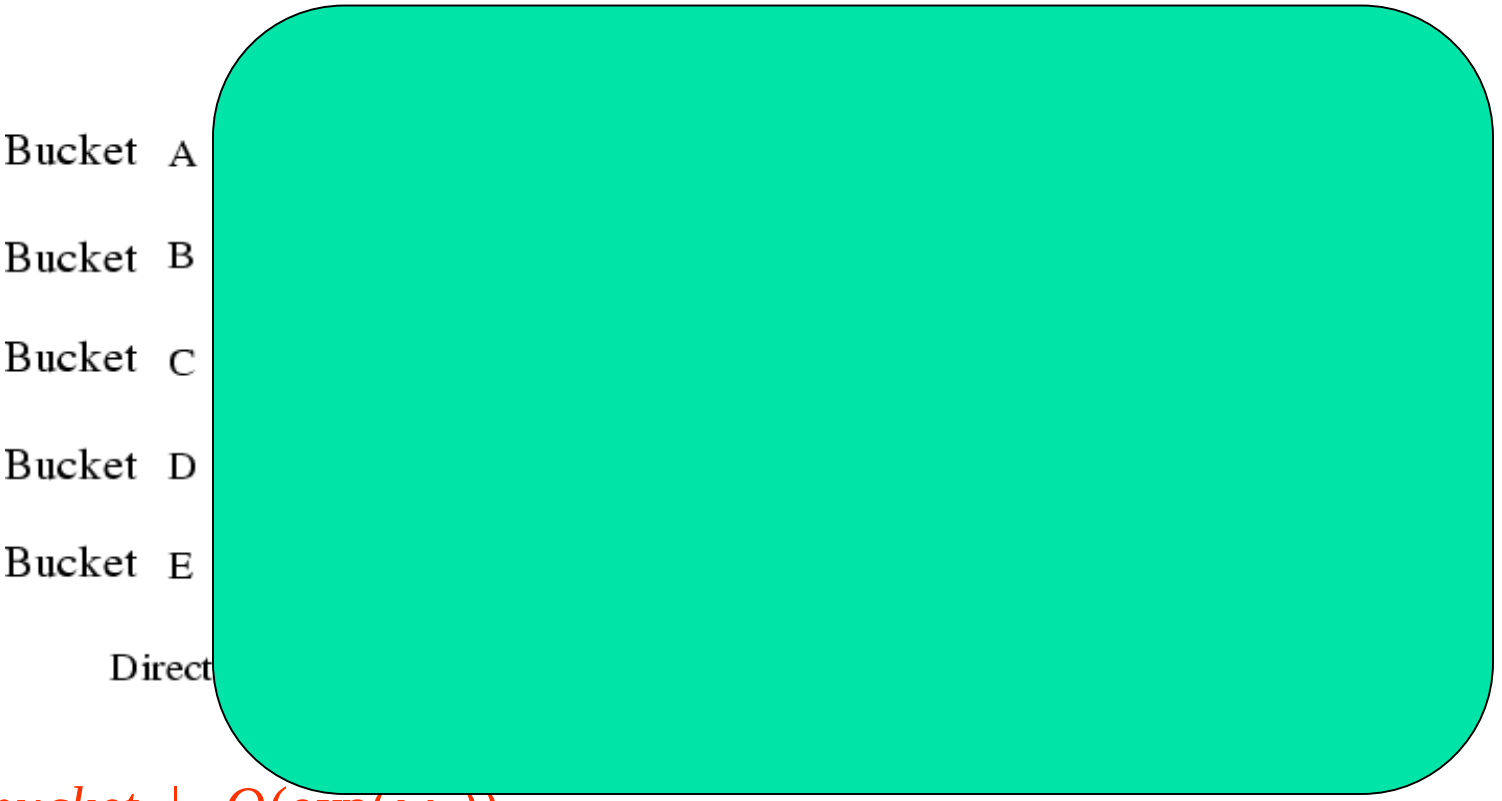


Figure 4.19: (a) The interaction graph of theory $\varphi_1 = \{(\neg C), (A \vee B \vee C), (\neg A \vee B \vee E), (\neg B \vee C \vee D)\}$, and (b) the effect of resolution over A on that graph.

Directional Resolution ⇔ Adaptive Consistency

(~C) (AVBVC) (~AvBvE)(~B,C,E)

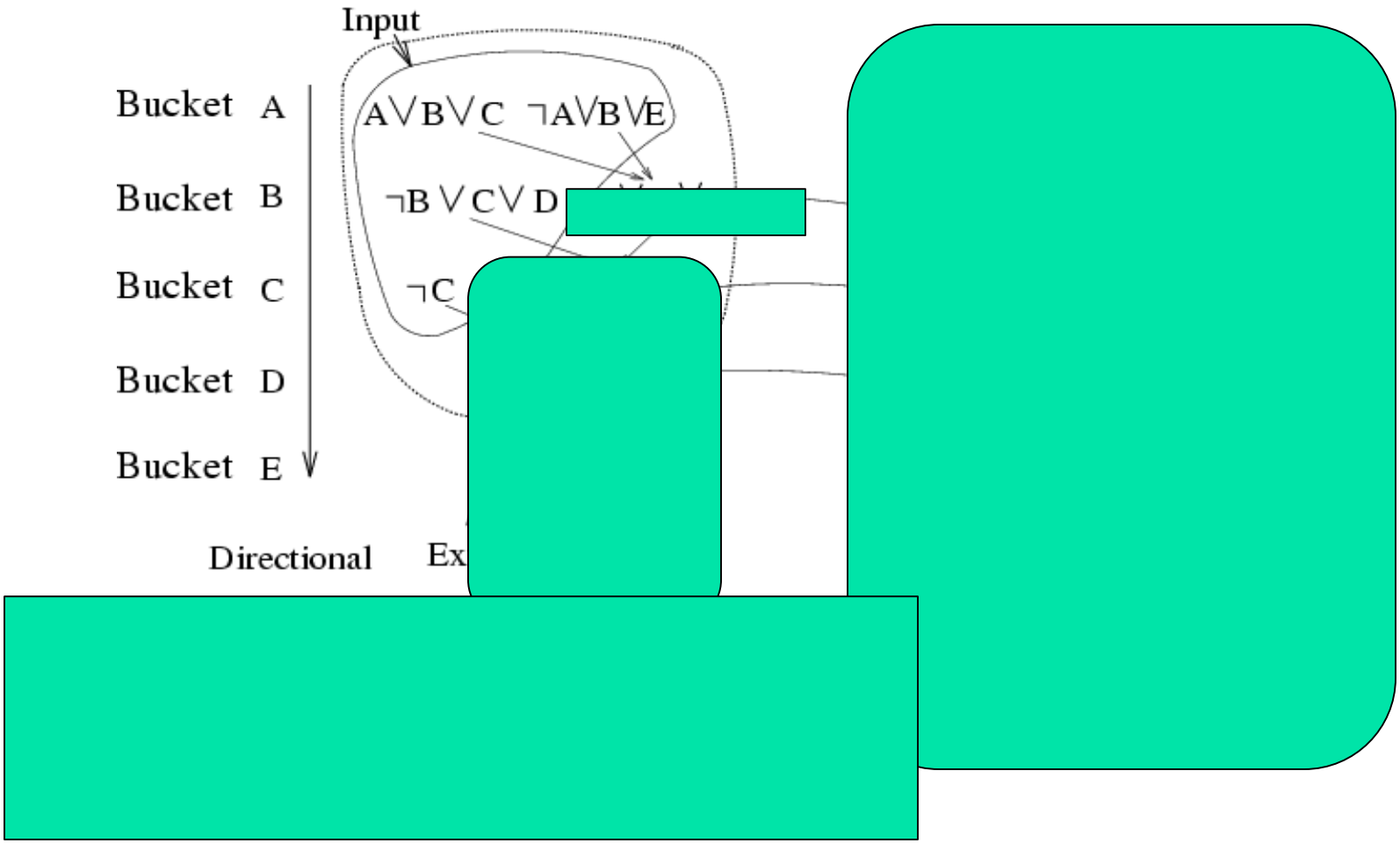


$|bucket_i| = O(\exp(w^*))$

DR time and space : $O(n \exp(w^*))$

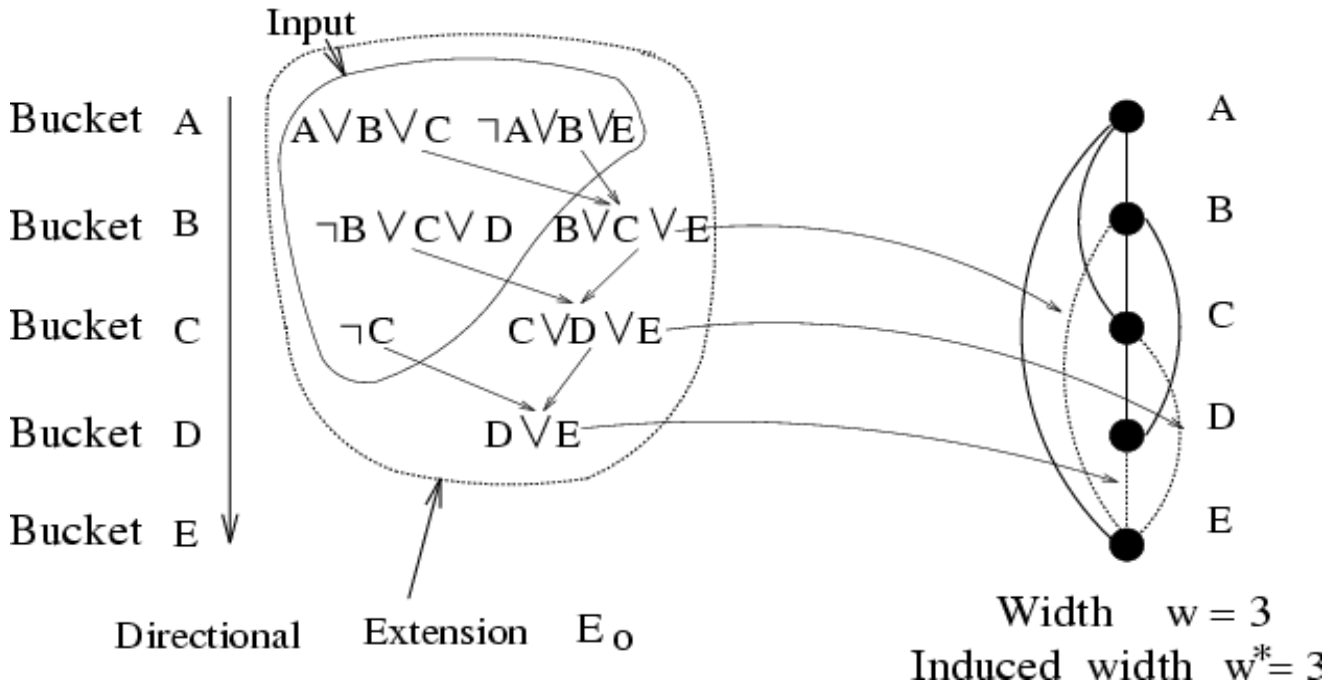
Directional Resolution ⇔ Adaptive Consistency

(~C) (AVBVC) (~AvBvE)(~B,C,E)



Directional Resolution ⇔ Adaptive Consistency

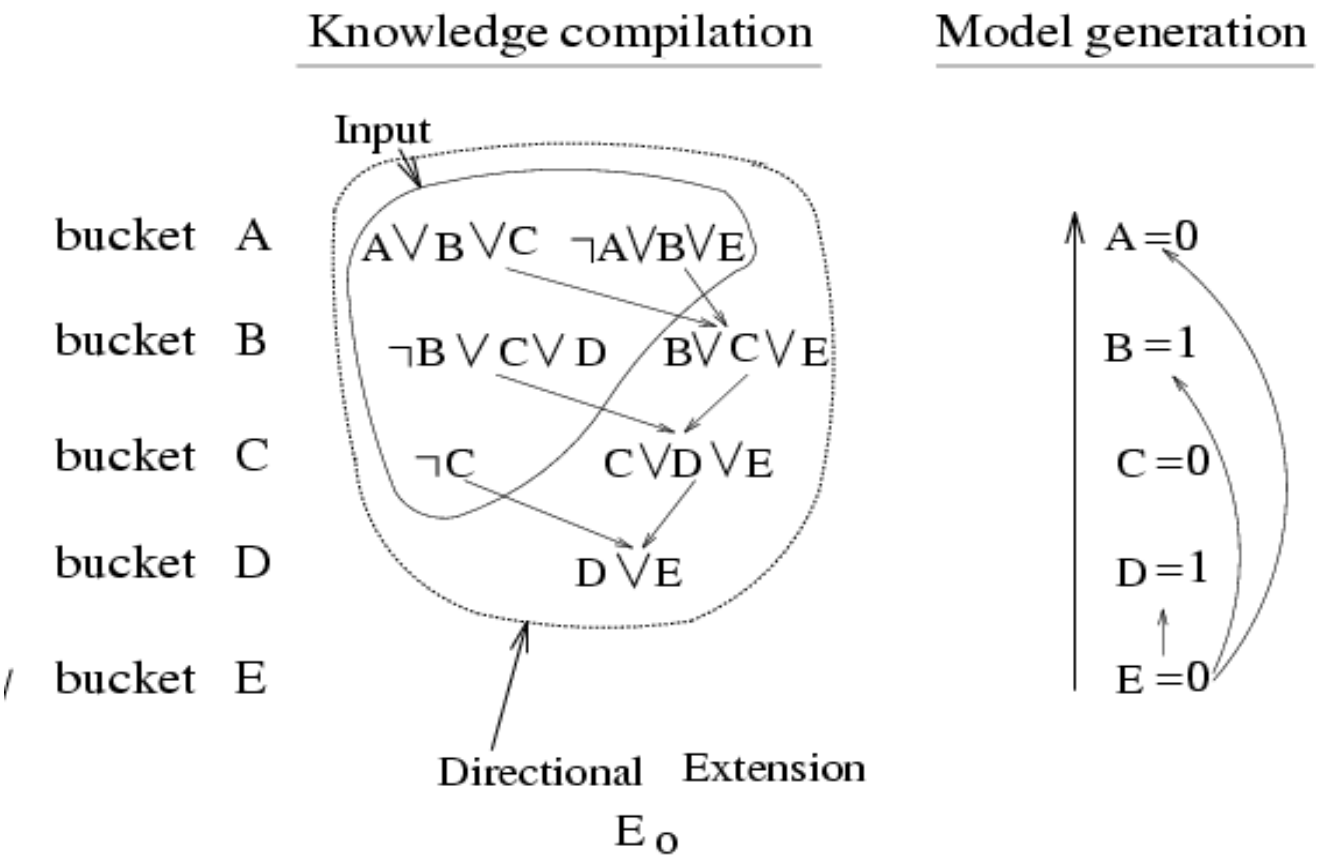
(~C) (AVBVC) (~AvBvE)(~B,C,E)



$|bucket_i| = O(\exp(w^*))$

DR time and space : $O(n \exp(w^*))$

Directional Resolution ⇔ Adaptive Consistency



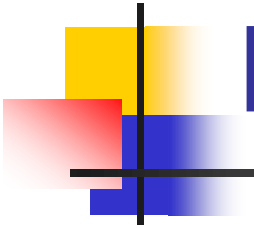
Directional Resolution

DIRECTIONAL-RESOLUTION

Input: A *CNF* theory φ, an ordering $d = Q_1, \ldots, Q_n$ of its variables.

Output: A decision of whether φ is satisfiable. If it is, a theory $E_d(\varphi)$, equivalent to φ, else an empty directional extension.

1. **Initialize:** generate an ordered partition of clauses into buckets. $bucket_1, \ldots, bucket_n$, where $bucket_i$ contains all clauses whose highest literal is Q_i.
2. **for** $i \leftarrow n$ **downto** 1 process $bucket_i$:
3. **if** there is a unit clause **then** (the instantiation step)
   apply unit-resolution in $bucket_i$ and place the resolvents in their right buckets.
   if the empty clause was generated, theory is not satisfiable.
4. **else** resolve each pair $\{(\alpha \vee Q_i), (\beta \vee \neg Q_i)\} \subseteq bucket_i$.
   if $\gamma = \alpha \vee \beta$ is empty, return $E_d(\varphi) = \{\}$, theory is not satisfiable
   else determine the index of γ and add it to the appropriate bucket.
5. **return** $E_d(\varphi) \leftarrow \bigcup_i bucket_i$



History

- 1960 – resolution-based Davis-Putnam algorithm
- 1962 – resolution step replaced by conditioning (Davis, Logemann and Loveland, 1962) to avoid memory explosion, resulting into a backtracking search algorithm known as Davis-Putnam (DP), or DPLL procedure.
- The dependency on induced width was not known in 1960.
- 1994 – Directional Resolution (DR), a rediscovery of the original Davis-Putnam, identification of tractable classes (Dechter and Rish, 1994).

Properties of DR

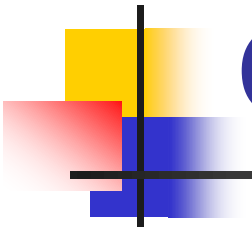
Lemma 3.2.6 *Given a theory φ and an ordering $d = (Q_1, ..., Q_n)$, if Q_i has at most k parents in the induced graph along d, then the bucket of Q_i in $E_d(\varphi)$ contains no more than 3^{k+1} clauses.*

Proof: Given a clause α in the bucket of Q_i, there are three possibilities for each parent P of Q_i: either P appears in α, $\neg P$ appears in α, or neither of them appears in α. Since Q_i also appears in α, either positively or negatively, the number of possible clauses in a bucket is no more than $2 \cdot 3^k < 3^{k+1}$. ■

Theorem 3.2.7 *(complexity of DR)*
Given a theory φ and an ordering of its variables d, the time complexity of algorithm DR along d is $O(n \cdot 9^{w_d^*})$, and $E_d(\varphi)$ contains at most $n \cdot 3^{w_d^*+1}$ clauses, where w_d^* is the induced width of φ's interaction graph along d. □

Road Map

- Graphical models
- Constraint networks Model
- Inference
  - Variable elimination for Constraints
  - Variable elimination for CNFs
  - Greedy search for induced-width orderings
  - Variable elimination for Linear Inequalities
- Constraint propagation
- Search
- Probabilistic Networks



Graph Properties

- Finding a minimum induced-width ordering is hard (NP-complete, lots of literature). So we have approximation greedy schemes.

Greedy Algorithms for Induced-Width

- Min-width ordering
- Min-induced-width ordering
- Max-cardinality ordering
- Min-fill ordering
- Chordal graphs
- Hypergraph partitionings

(Project: present papers on induced-width, run algorithms for induced-width on new benchmarks…)

Min-width Ordering

MIN-WIDTH (MW)

input: a graph $G = (V, E)$, $V = \{v_1, ..., v_n\}$

output: A min-width ordering of the nodes $d = (v_1, ..., v_n)$.

1. **for** $j = n$ to 1 by -1 do
2. $r \leftarrow$ a node in G with smallest degree.
3. put r in position j and $G \leftarrow G - r$.
   (Delete from V node r and from E all its adjacent edges)
4. **endfor**

Proposition: *algorithm min-width finds a min-width ordering of a graph*
Complexity:?
O(e)

Greedy Orderings Heuristics

min-induced-width (miw)

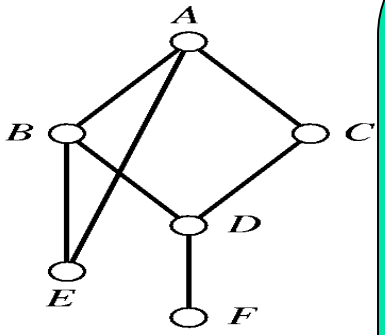
input: a graph G = (V;E), V = {v1; :::; vn}
output: A miw ordering of the nodes d = (v1; :::; vn).
1. for j = n to 1 by -1 do
2. r ← a node in V with smallest degree.
3. put r in position j.
4. connect r's neighbors: E ← E union {(vi; vj)| (vi; r) in E; (vj ; r) 2 in E},
5. remove r from the resulting graph: V ←V - {r}.

Theorem: *A graph is a tree iff it has both width and induced-width of 1.*

min-fill (min-fill)

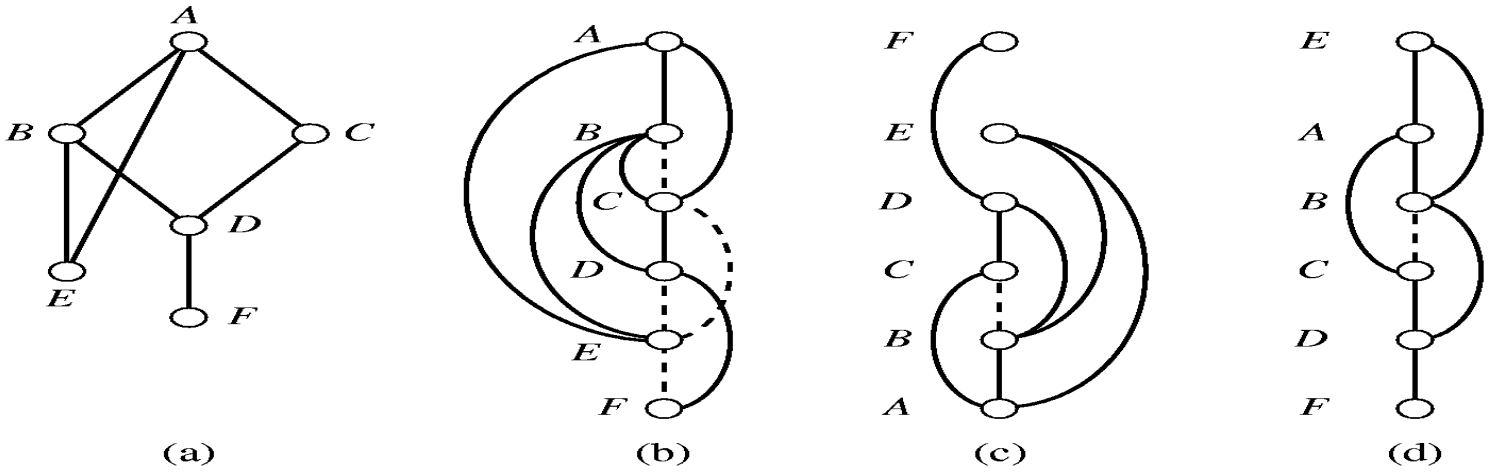
input: a graph G = (V;E), V = {v1; :::; vn}
output: An ordering of the nodes d = (v1; :::; vn).
1. for j = n to 1 by -1 do
2. r ←a node in V with smallest fill edges for his parents.
3. put r in position j.
4. connect r's neighbors: E ←E union {(vi; vj)| (vi; r) 2 E; (vj ; r) in E},
5. remove r from the resulting graph: V ←V –{r}.

Example



(a)

Example



Chordal Graphs; Max-Cardinality Ordering

- A graph is chordal if every cycle of length at least 4 has a chord
- Finding w* over chordal graph is easy using the max-cardinality ordering
- If G* is an induced graph of it is chordal
- K-trees are special chordal graphs.
- Finding the max-clique in chordal graphs is easy (just enumerate all cliques in a max-cardinality ordering

Max-Cardinality Ordering

MAX-CARDINALITY (MC)

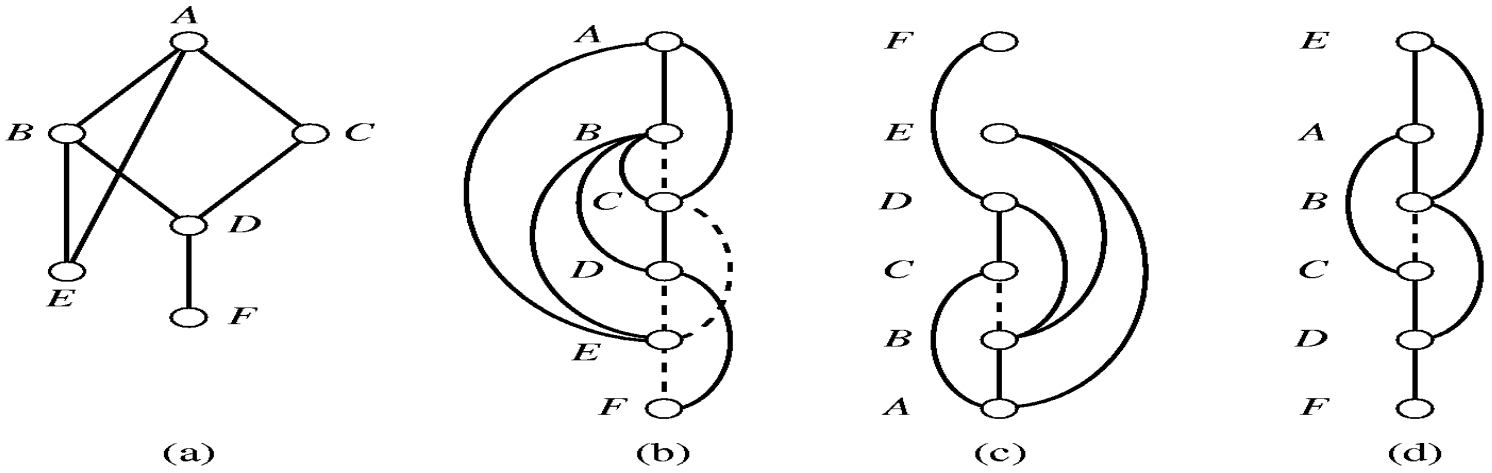
input: a graph $G = (V, E)$, $V = \{v_1, ..., v_n\}$

output: An ordering of the nodes $d = (v_1, ..., v_n)$.

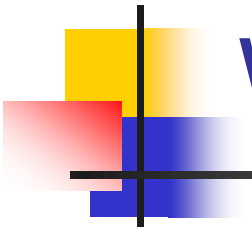
1. Place an arbitrary node in position 0.
2. **for** $j = 1$ to n do
3. $r \leftarrow$ a node in G that is connected to a largest subset of nodes in positions 1 to $j - 1$, breaking ties arbitrarily.
4. **endfor**

Example

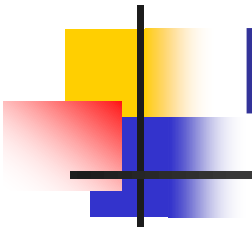
We see again that *G* in the Figure (a) is not chordal since the parents of *A* are not connected in the max-cardinality ordering in Figure (d). If we connect *B* and *C*, the resulting induced graph is chordal.



Which Greedy Algorithm is Best?



- MinFill, prefers a node who add the least number of fill-in arcs.
- Empirically, fill-in is the best among the greedy algorithms (MW,MIW,MF,MC)
- Complexity of greedy orderings?
- MW is O(?), MIW: O(?) MF (?) MC is O(mn)



Road Map

- Graphical models
- Constraint networks Model
- Inference
  - Variable elimination for Constraints
  - Variable elimination for CNFs
  - Greedy search for induced-width orderings
  - Variable elimination for Linear Inequalities
- Constraint propagation
- Search
- Probabilistic Networks

Linear Inequalities

$$(3x_i + 2x_j \leq 3) \wedge (-4x_i + 5x_j \leq 1)$$

Definition 3.3.1 (Linear elimination) *Let* $\alpha = \sum_{i=1}^{(r-1)} a_i x_i + a_r x_r \leq c$, *and* $\beta = \sum_{i=1}^{(r-1)} b_i x_i + b_r x_r \leq d$. *Then* $elim_r(\alpha, \beta)$ *is applicable only if* a_r *and* b_r *have opposite signs, in which case* $elim_r(\alpha, \beta) = \sum_{i=1}^{r-1}(-a_i \frac{b_r}{a_r} + b_i)x_i \leq -\frac{b_r}{a_r}c + d$. *If* a_r *and* b_r *have the same sign the elimination implicitly generates the universal constraint.*

Linear Inequalities: Fourier Elimination

DIRECTIONAL-LINEAR-ELIMINATION (φ, d)

Input: A *set of linear inequalities* φ, an ordering $d = x_1, \ldots, x_n$.

Output: A decision of whether φ is satisfiable. If it is, a backtrack-free theory $E_d(\varphi)$.

1. **Initialize:** Partition inequalities into ordered buckets.
2. **for** $i \leftarrow n$ **downto** 1 **do**
3. **if** x_i has one value in its domain **then**
. substitute the value into each inequality in the bucket and put the resulting inequality in the right bucket.
4. **else,for each** pair $\{\alpha, \beta\} \subseteq bucket_i$, compute $\gamma = elim_i(\alpha, \beta)$
if γ has no solutions, return $E_d(\varphi) = \{\}$, "inconsistency"
else add γ to the appropriate lower bucket.
5. **return** $E_d(\varphi) \leftarrow \bigcup_i bucket_i$

Directional linear elimination, DLE : generates a backtrack-free representation

Theorem 4.8.3 *Given a set of linear inequalities φ, algorithm DLE (Fourier elimination) decides the consistency of φ over the Rationals and the Reals, and it generates an equivalent backtrack-free representation.* □

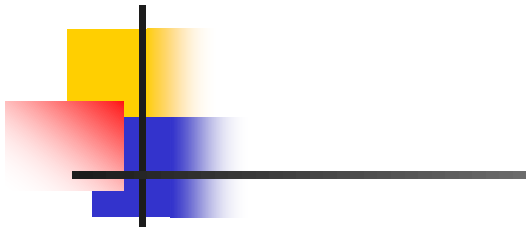
Example

$$
\begin{aligned}
bucket_4: &\quad 5x_4 + 3x_2 - x_1 \le 5,\ x_4 + x_1 \le 2,\ -x_4 \le 0,\\
bucket_3: &\quad x_3 \le 5,\ x_1 + x_2 - x_3 \le -10\\
bucket_2: &\quad x_1 + 2x_2 \le 0.\\
bucket_1: &
\end{aligned}
$$

Figure 4.23: initial buckets

$$
\begin{aligned}
bucket_4: &\quad 5x_4 + 3x_2 - x_1 \le 5,\ x_4 + x_1 \le 2,\ -x_4 \le 0,\\
bucket_3: &\quad x_3 \le 5,\ x_1 + x_2 - x_3 \le -10\\
bucket_2: &\quad x_1 + 2x_2 \le 0 \ ||\ 3x_2 - x_1 \le 5, x_1 + x_2 \le -5\\
bucket_1: &\quad ||\ x_1 \le 2.
\end{aligned}
$$

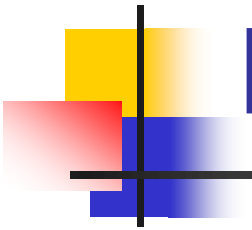
Figure 4.24: final buckets



Algorithms for Reasoning with graphical models

Class4

Rina Dechter



Road Map

- Graphical models
- Constraint networks Model
- Inference
  - Variable elimination for Constraints
  - Variable elimination for CNFs
  - Variable elimination for Linear Inequalities
  - Constraint propagation (ch 2 Dechter2)
- Search
- Probabilistic Networks

Sudoku – Approximation: Constraint Propagation

- ***Constraint***
- ***Propagation***
- ***Inference***

		2	4		6			
8	6	5	1			2		
	1				8	6		9
9				4		8	6	
	4	7				1	9	
	5	8		6				3
4		6	9				7	2 3 ~~2~~ ~~4 6~~
		9			4	5	8	1
			3		2	9		

- ***Variables*: empty slots**
- ***Domains* = {1,2,3,4,5,6,7,8,9}**
- ***Constraints*:**
  - ***27 all-different***

Each row, column and major block must be alldifferent

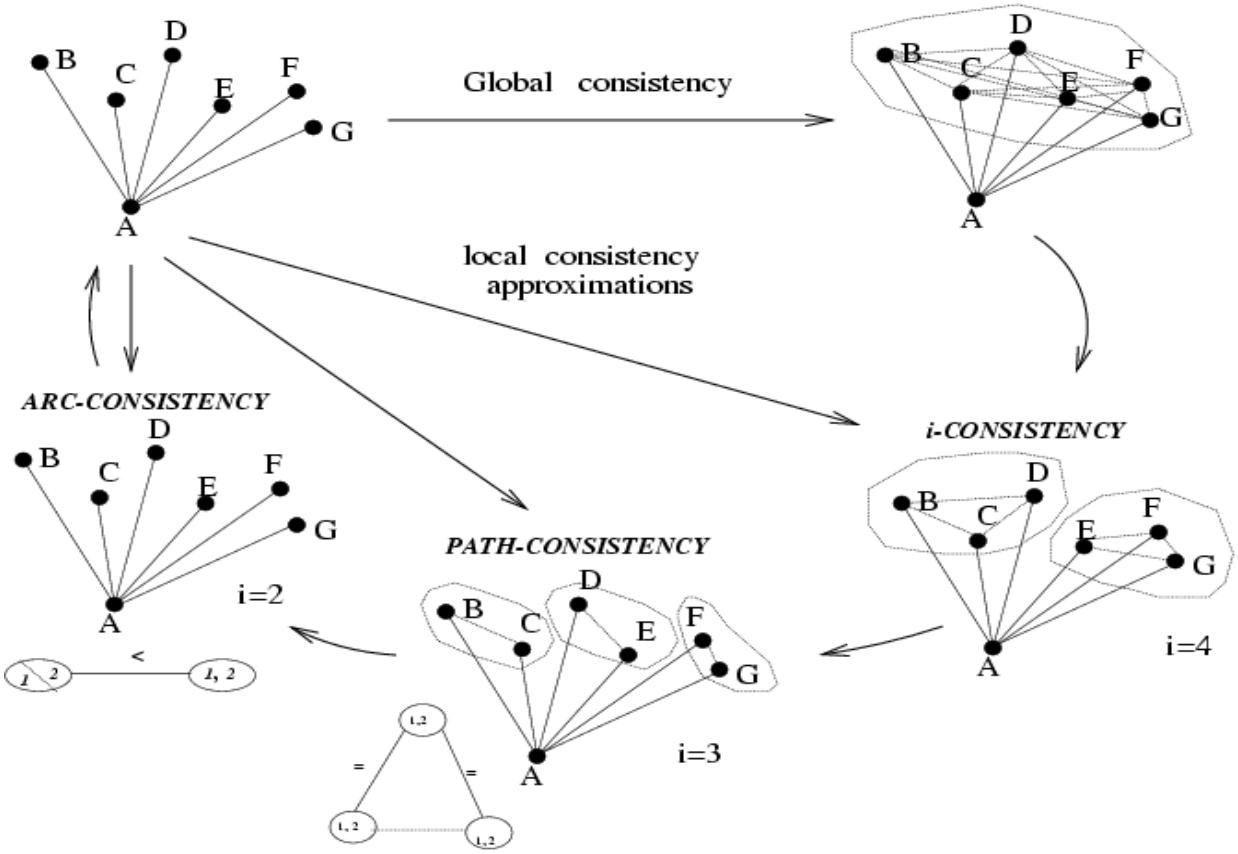
"Well posed" if it has unique solution: 27 constraints

Approximating Inference: Local Constraint Propagation

- **Problem:** bucket-elimination/tree-clustering algorithms are intractable when *induced width* is large

- **Approximation:** bound the size of recorded dependencies, i.e. perform ***local constraint propagation (local inference)***

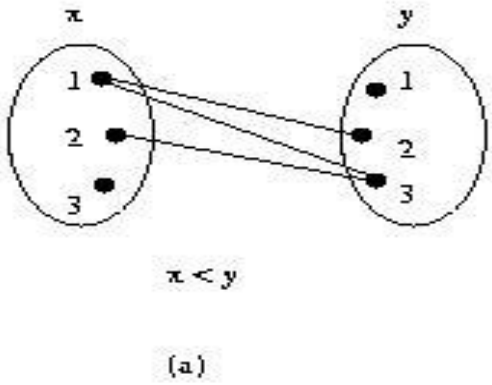
From Global to Local Consistency

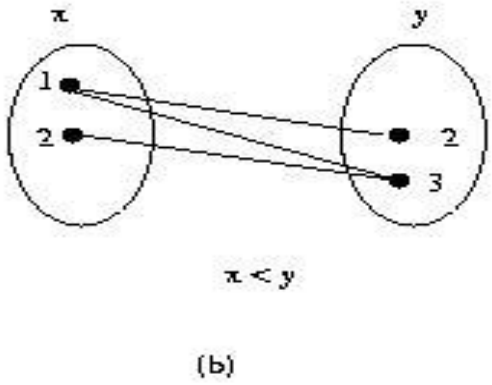


Arc-Consistency

A binary constraint R(X,Y) is arc-consistent w.r.t. X is every value In x's domain has a match in y's domain.

$$R_X = \{1,2,3\},\ R_Y = \{1,2,3\},\ \text{constraint}\ \ X < Y$$





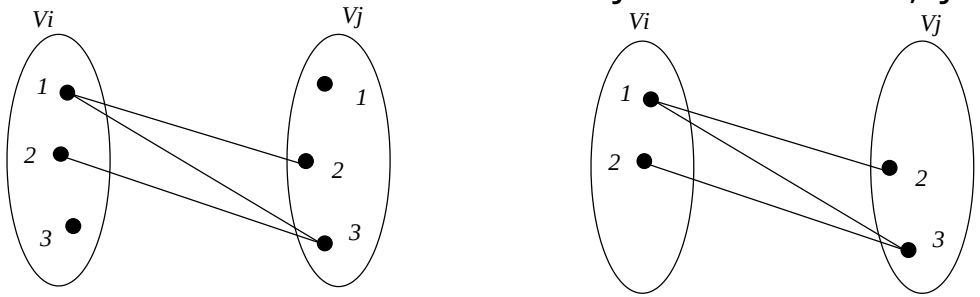
Only domains are reduced:

$$R_X \leftarrow \prod_X R_{XY} \bowtie D_Y$$

Arc-Consistency

Definition: Given a constraint graph **G**,

- A variable V_i is arc-consistent relative to V_j iff for every value $a \in D_{Vi}$ there exists a value $b \in D_{Vj}$ | $(a, b) \in R_{Vi,Vj}$.



- The constraint $R_{Vi,Vj}$ is arc-consistent iff
  - V_i is arc-consistent relative to V_j and
  - V_j is arc-consistent relative to V_i.
- A binary CSP is arc-consistent iff every constraint (or sub-graph of size 2) is arc-consistent

Arc-consistency

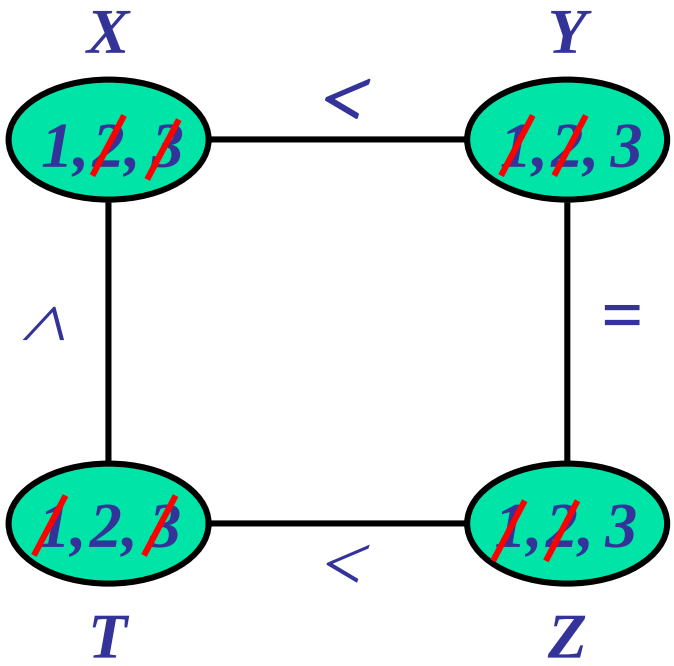
$1 \leq X, Y, Z, T \leq 3$

$X < Y$

$Y = Z$

$T < Z$

$X \leq T$



Arc-consistency

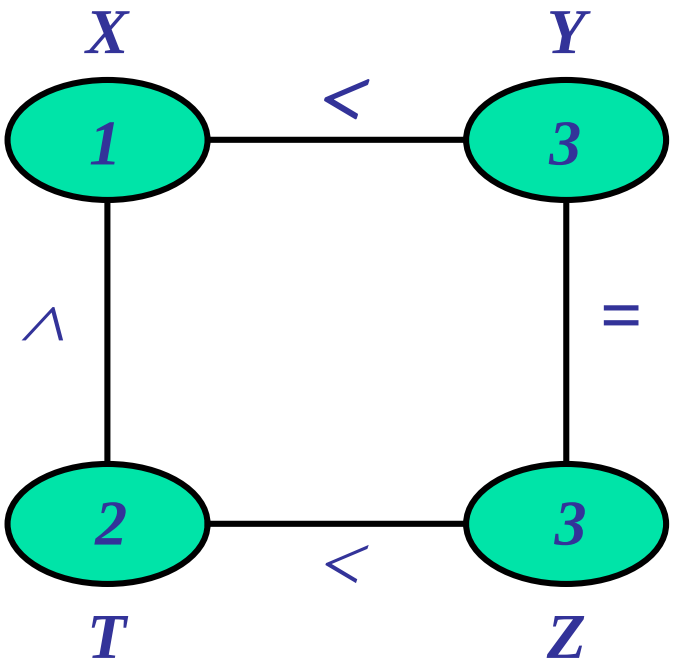
$1 \le X, Y, Z, T \le 3$

$X < Y$

$Y = Z$

$T < Z$

$X \le T$



$$R_X \leftarrow \prod_X R_{XY} \bowtie D_Y$$

AC-1

AC-1($\mathcal{R}$)

input: a network of constraints $\mathcal{R} = (X, D, C)$
output: $\mathcal{R}'$ which is the loosest arc-consistent network equivalent to $\mathcal{R}$

1. **repeat**
2. **for** every pair $\{x_i, x_j\}$ that participates in a constraint
3. Revise($(x_i), x_j$) (or $D_i \leftarrow D_i \cap \pi_i(R_{ij} \bowtie D_j)$)
4. Revise($(x_j), x_i$) (or $D_j \leftarrow D_j \cap \pi_j(R_{ij} \bowtie D_i)$)
5. **endfor**
6. **until** no domain is changed

Figure 3.4: Arc-consistency-1 (AC-1)

- Complexity (Mackworth and Freuder, 1986):
- *e* = number of arcs, *n* variables, *k* values
- (*ek^2*, each loop, *nk* number of loops), best-case = *ek*
- Arc-consistency is: $\Omega(ek^2)$
- Complexity of AC-1: O(enk^3)

AC-3

AC-3($\mathcal{R}$)
input: a network of constraints $\mathcal{R} = (X, D, C)$
output: $\mathcal{R}'$ which is the largest arc-consistent network equivalent to $\mathcal{R}$
1. **for** every pair $\{x_i, x_j\}$ that participates in a constraint $R_{ij} \in \mathcal{R}$
2. $queue \leftarrow queue \cup \{(x_i, x_j), (x_j, x_i)\}$
3. **endfor**
4. **while** $queue \neq \{\}$
5. select and delete (x_i, x_j) from $queue$
6. $Revise((x_i), x_j)$
7. **if** $Revise((x_i), x_j)$ causes a change in D_i
8. **then** $queue \leftarrow queue \cup \{(x_k, x_i), i \neq k\}$
9. **endif**
10. **endwhile**

Figure 3.5: Arc-consistency-3 (AC-3)

- Complexity: $O(ek^3)$
- Best case O(ek), since each arc may be processed in O(2k)

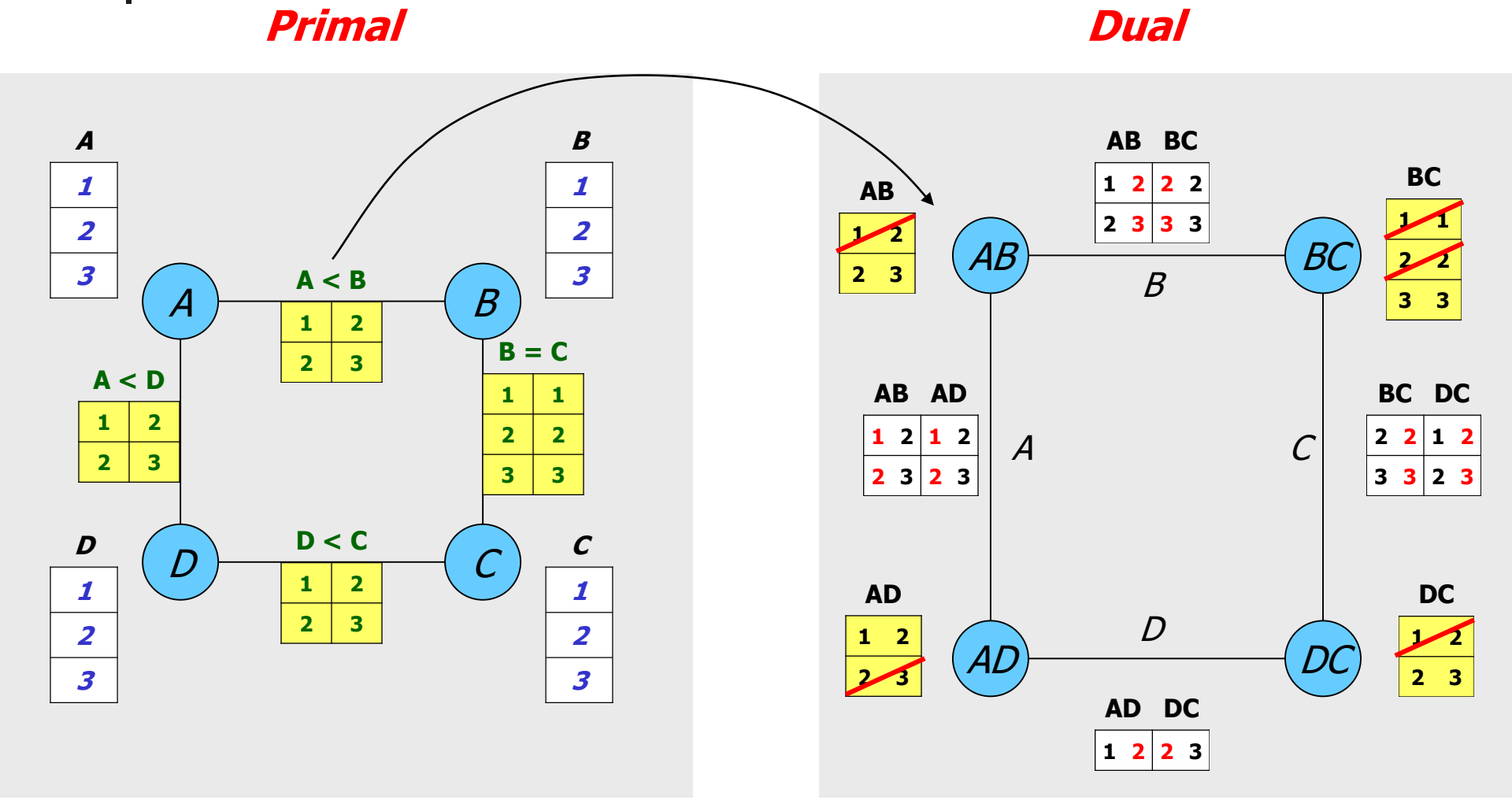
Arc-Consistency Algorithms

- AC-1: brute-force, distributed $O(nek^3)$
- AC-3, queue-based $O(ek^3)$
- AC-4, context-based, optimal $O(ek^2)$
- AC-5,6,7,…. Good in special cases
- Important: applied at every node of search

n=number of variables, e=#constraints, k=domain size

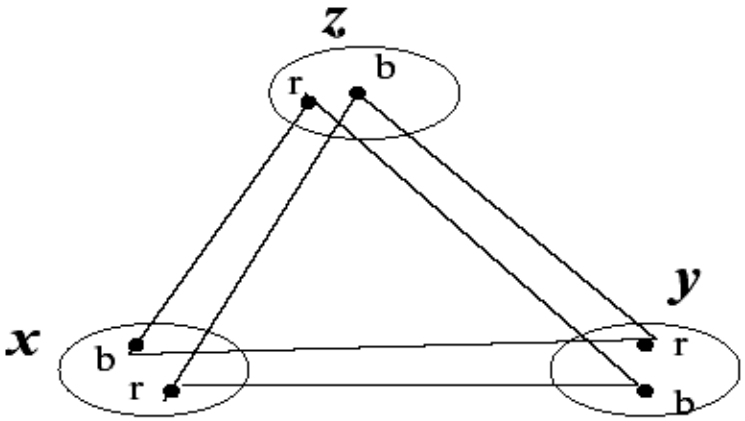
Mackworth and Freuder (1977,1983), Mohr and Anderson, (1985)…

Relational Distributed Arc-Consistency

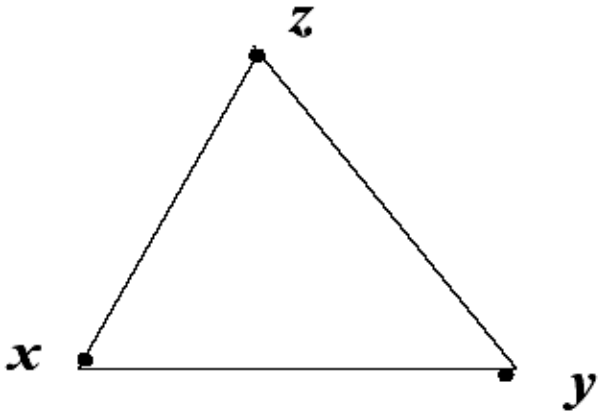


Path-Consistency

- A pair (x, y) is path-consistent relative to Z, if every consistent assignment (x, y) has a consistent extension to z.



(a)



(b)

Figure 3.8: (a) The matching diagram of a 2-value graph coloring problem. (b) Graphical picture of path-consistency using the matching diagram.

Example: Path-Consistency

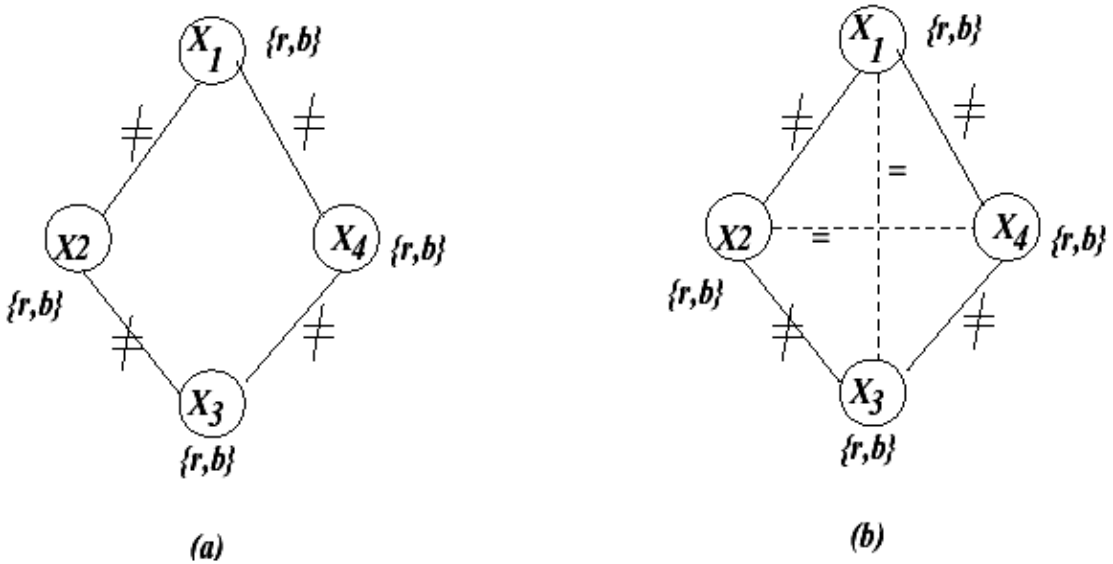
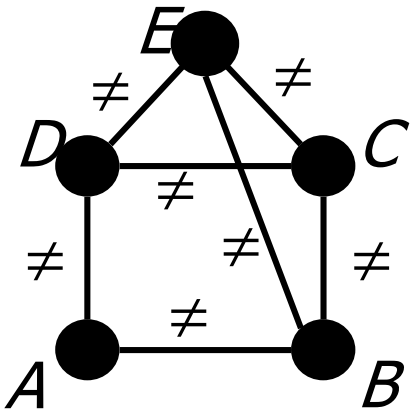


Figure 3.12: A graph-coloring graph (a) before path-consistency (b) after path-consistency

Directional i-Consistency



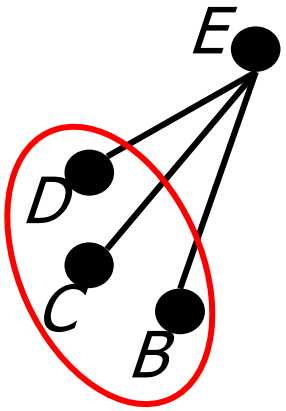
E: $E \neq D, E \neq C, E \neq B$

D: $D \neq C, D \neq A$

C: $C \neq B$

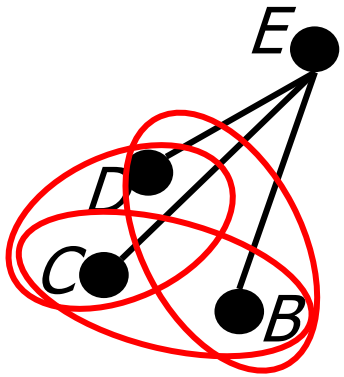
B: $A \neq B$

A:



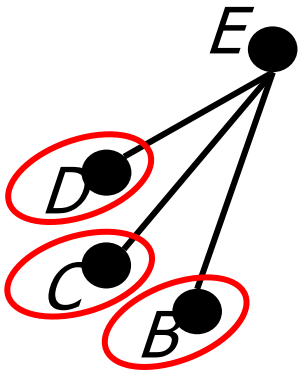
Adaptive

R_{DCB}



d-path

R_{DC}, R_{DB}
R_{CB}



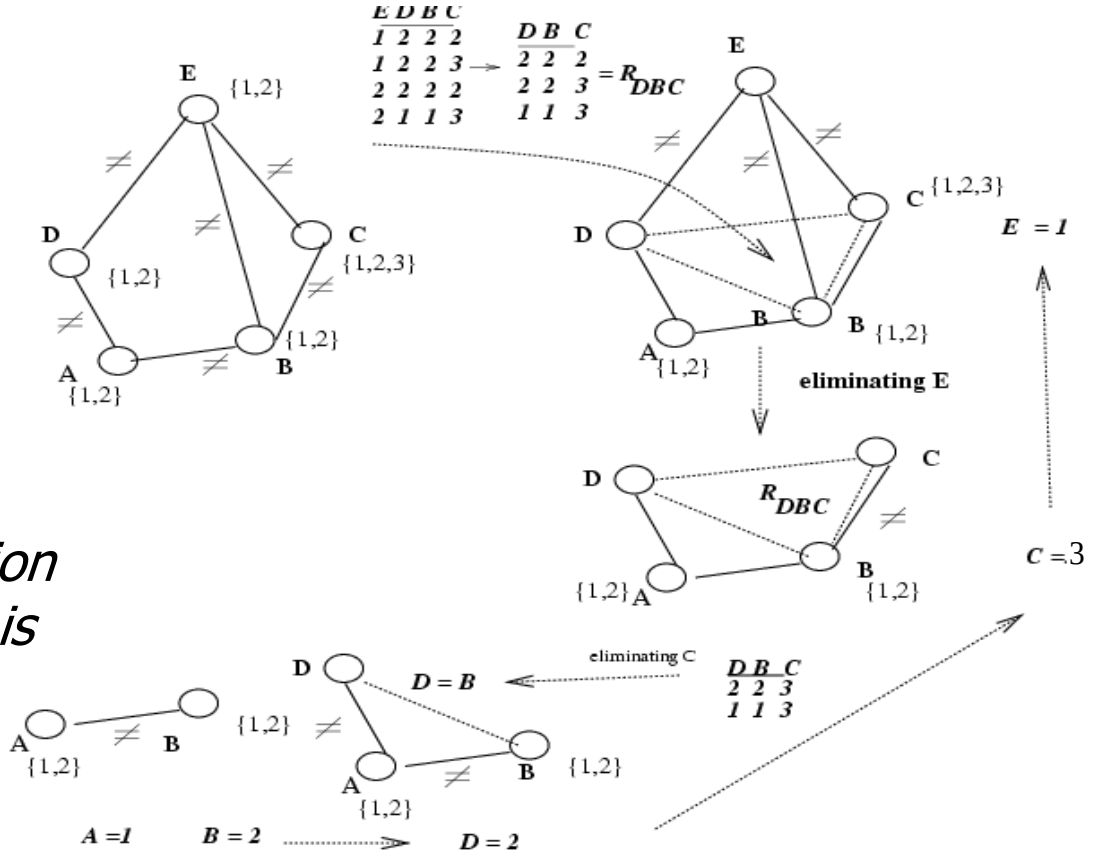
d-arc

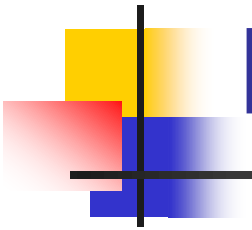
R_D
R_C
R_D

Variable Elimination

Eliminate variables one by one: "constraint propagation"

Solution generation after elimination is backtrack-free





Road Map

- Graphical models
- Constraint networks Model
- Inference
  - Variable elimination:
  - Tree-clustering
  - Constraint propagation
- Search
- Probabilistic Networks