Structural Decomposition Methods and Islands of Tractability for NP-hard Problems

Georg Gottlob, Gianluigi Greco, and Francesco Scarcello

IJCAI-13

UNIVERSITY OF OXFORD

Campus di Arcavacata
UNIVERSITÀ DELLA CALABRIA

Outline of PART I

Introduction to Decomposition Methods

Tree Decompositions

Applications of Tree Decompositions

Outline of PART II

Beyond Tree Decompositions

Applications to Databases and CSPs

Structural and Consistency Properties

Outline of Part III

Applications to Optimization Problems

Application: Nash Equilibria

Application: Coalitional Games

Application: Combinatorial Auctions

Appendix: Beyond Hypertree Width

Outline of PART I

Introduction to Decomposition Methods

Tree Decompositions

Applications of Tree Decompositions

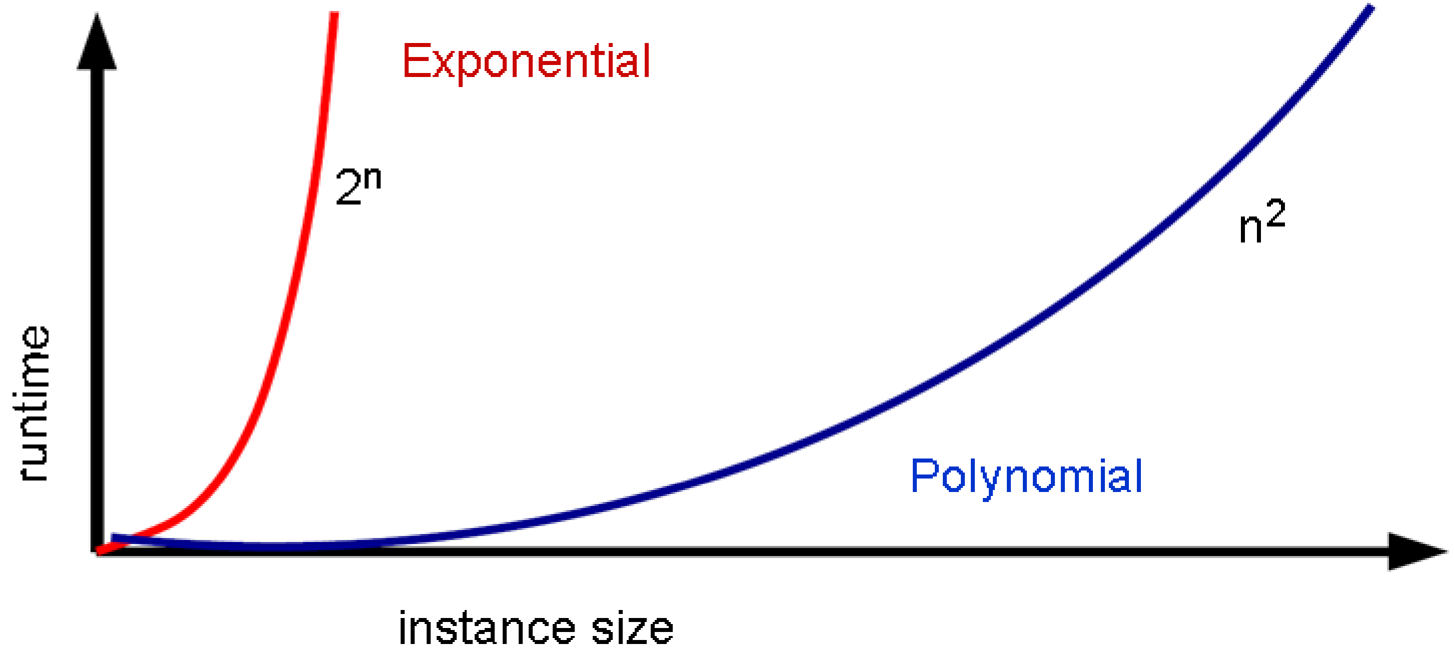
Inherent Problem Complexity

- Problems *decidable* or *undecidable*.
- We concentrate on decidable problems here.
- A problem is as complex as the best possible algorithm which solves it.

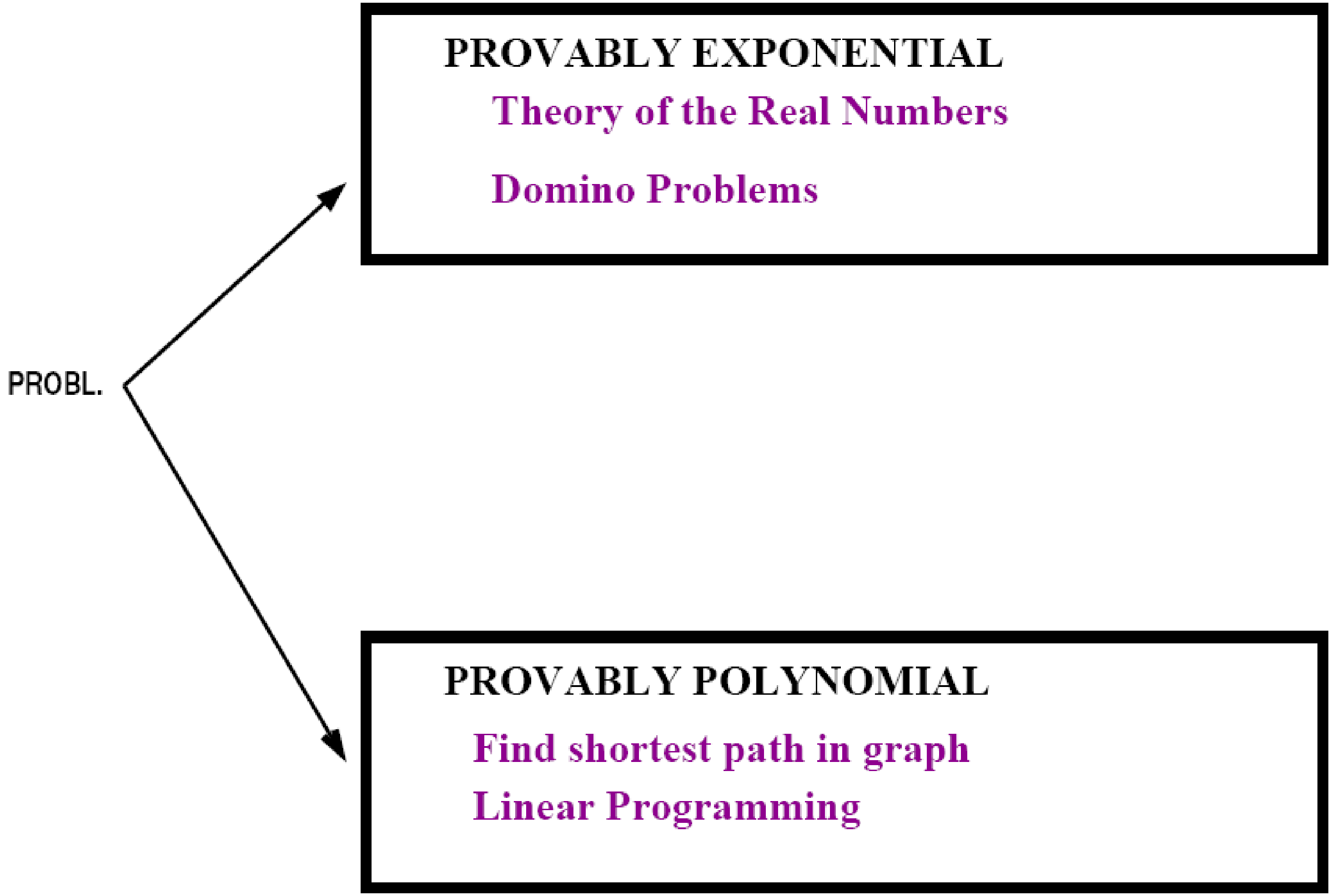
Inherent Problem Complexity

- Problems *decidable* or *undecidable*.
- We concentrate on decidable problems here.
- A problem is as complex as the best possible algorithm which solves it.

Number of steps it takes for input of size n



Time Complexity



PROBL.

PROVABLY EXPONENTIAL

Theory of the Real Numbers

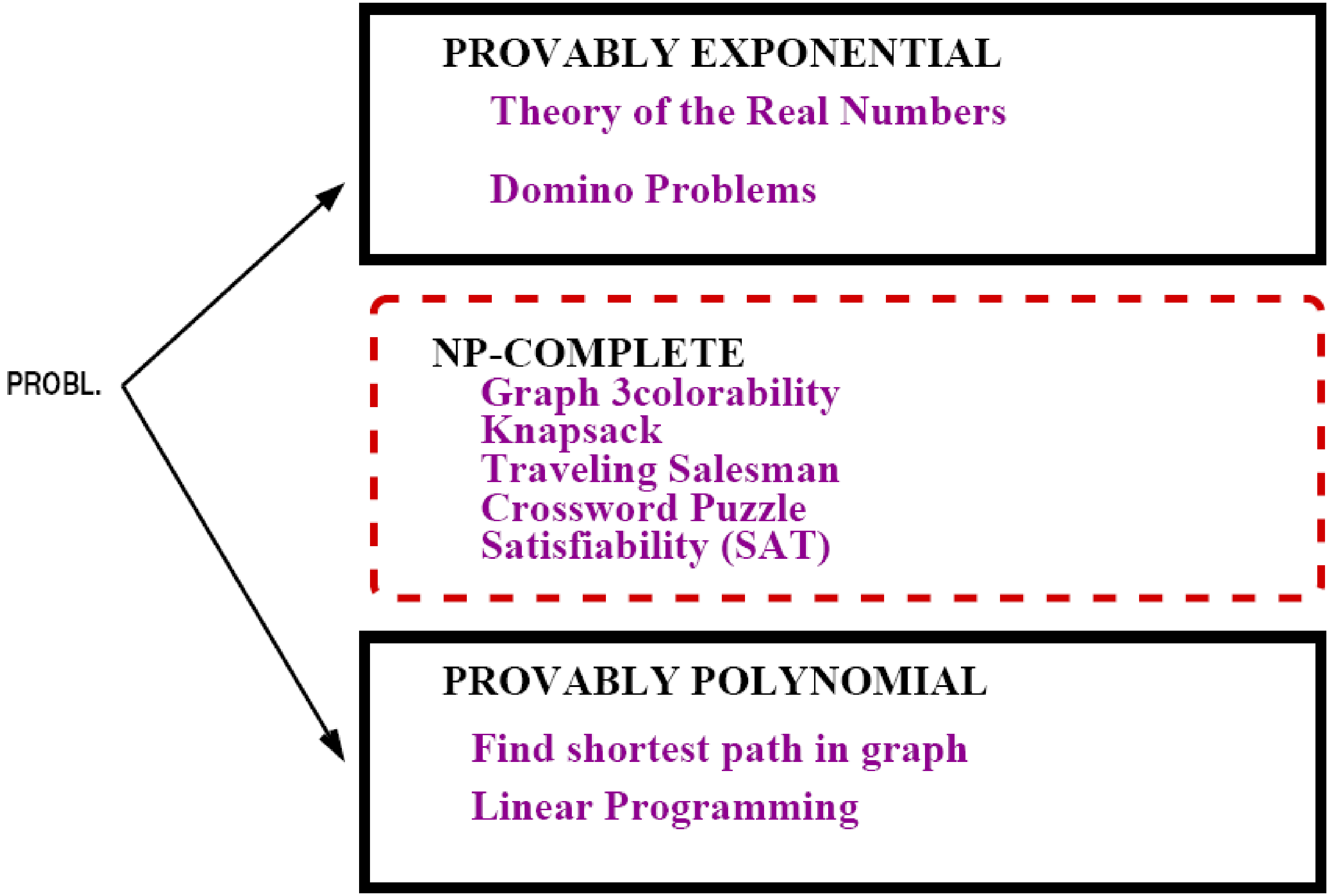
Domino Problems

PROVABLY POLYNOMIAL

Find shortest path in graph

Linear Programming

Time Complexity



PROBL.

PROVABLY EXPONENTIAL

Theory of the Real Numbers

Domino Problems

NP-COMPLETE

Graph 3colorability

Knapsack

Traveling Salesman

Crossword Puzzle

Satisfiability (SAT)

PROVABLY POLYNOMIAL

Find shortest path in graph

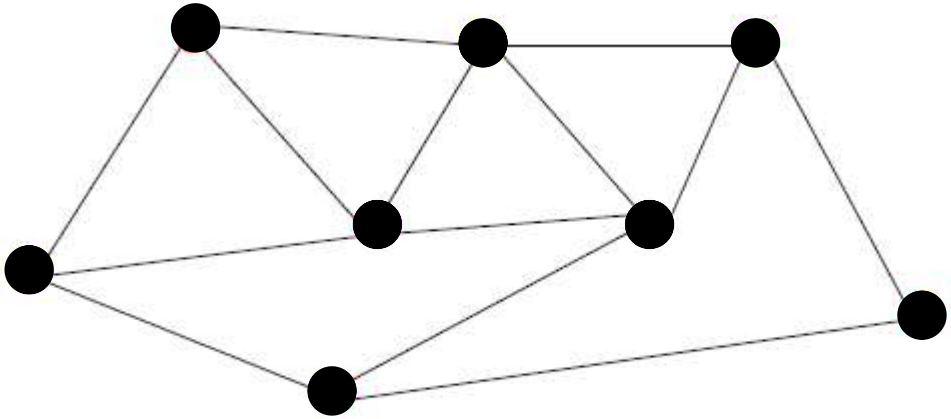
Linear Programming

Graph Three-colorability

Instance: A graph G.

Question: Is G 3-colorable?

Examples of instances:

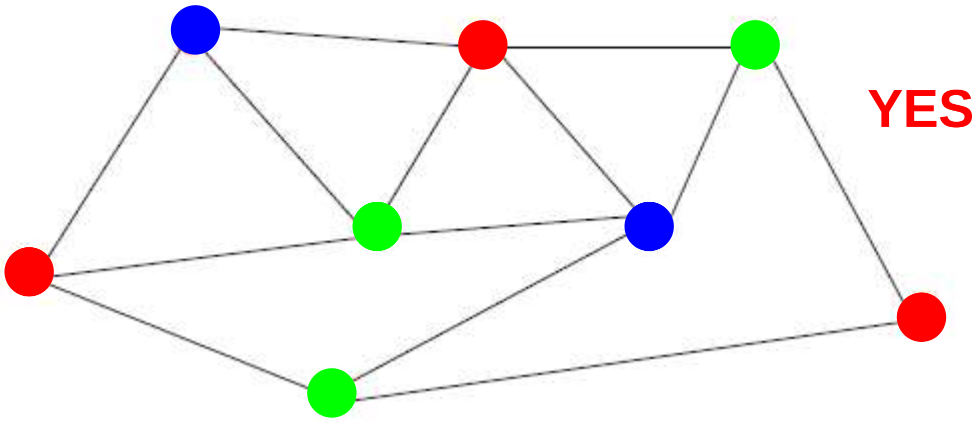


Graph Three-colorability

Instance: A graph G.

Question: Is G 3-colorable?

Examples of instances:



YES!

Approaches for Solving Hard Problems

- NP-complete problems often occur in practice.
- They must be solved by acceptable methods.
- Three approaches:
  - Randomized local search
  - Approximation
  - Identification of easy (=polynomial) subclasses.

Approaches for Solving Hard Problems

- NP-complete problems often occur in practice.
- They must be solved by acceptable methods.
- Three approaches:
  - Randomized local search
  - Approximation
  - Identification of easy (=polynomial) subclasses.

Identification of Polynomial Subclasses

- High complexity often arises in “rare” worst case instances
- Worst case instances exhibit intricate structures
- In practice, many input instances have *simple* structures
- Therefore, our goal is to
  - Define polynomially solvable subclasses (possibly, the largest ones)
  - Prove that membership testing is tractable for these classes
  - Develop efficient algorithms for instances in these classes

Graph and Hypergraph Decompositions

- The evil in Computer science is hidden in (vicious) cycles.

- We need to get them under control!

- Decompositions: Tree-Decomposition, path decompositions, hypertree decompositions,…

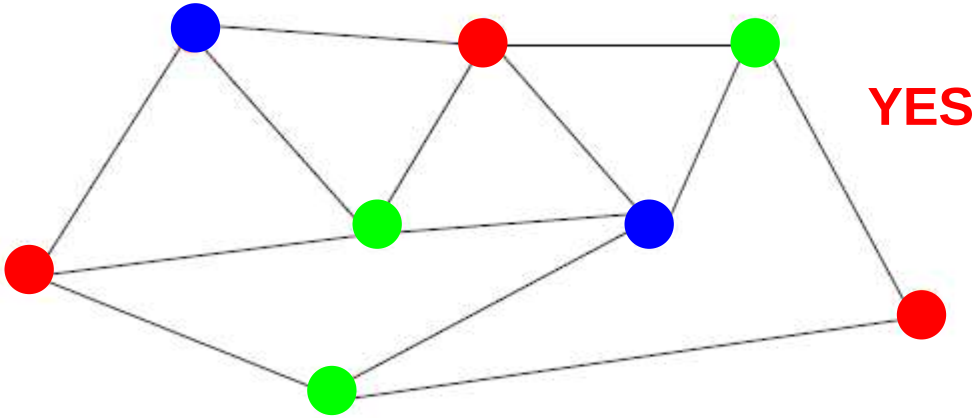
  Exploit bounded degree of cyclicity.

Graph Three-colorability

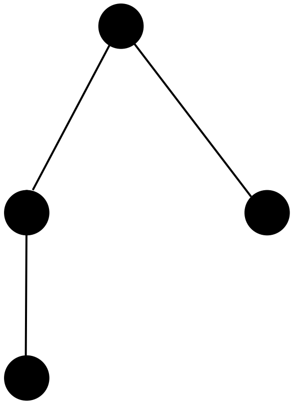
Instance: A graph G.

Question: Is G 3-colorable?

Examples of instances:



YES!



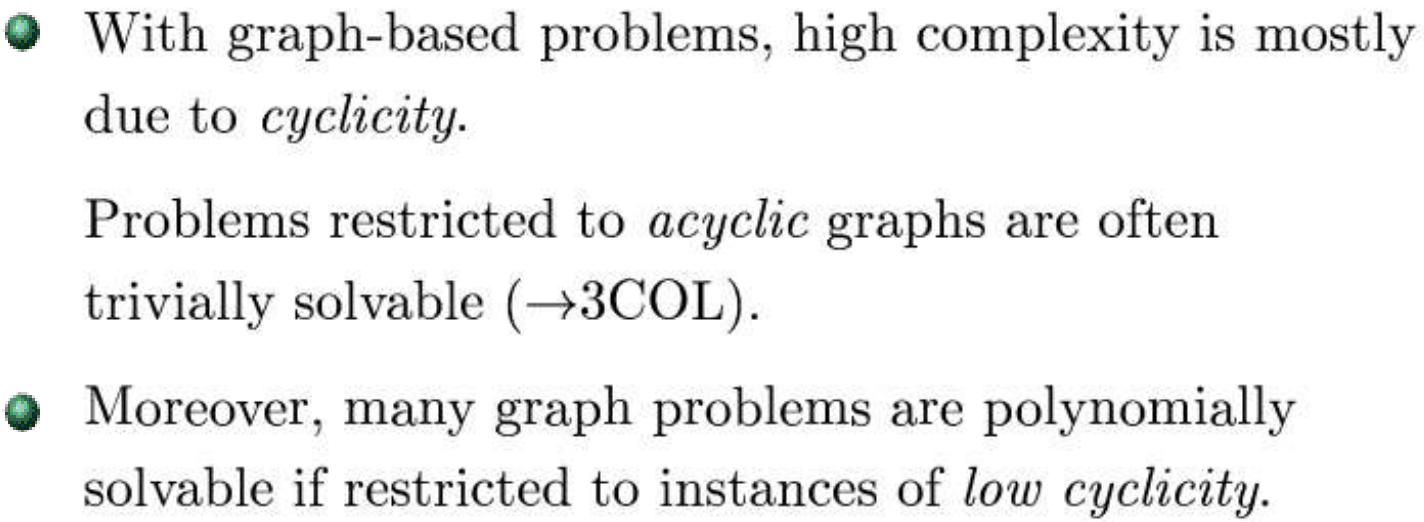
Problems with a Graph Structure

- With graph-based problems, high complexity is mostly due to *cyclicity*.

  Problems restricted to *acyclic* graphs are often trivially solvable (→3COL).

- Moreover, many graph problems are polynomially solvable if restricted to instances of *low cyclicity*.

Problems with a Graph Structure



- With graph-based problems, high complexity is mostly due to *cyclicity*.

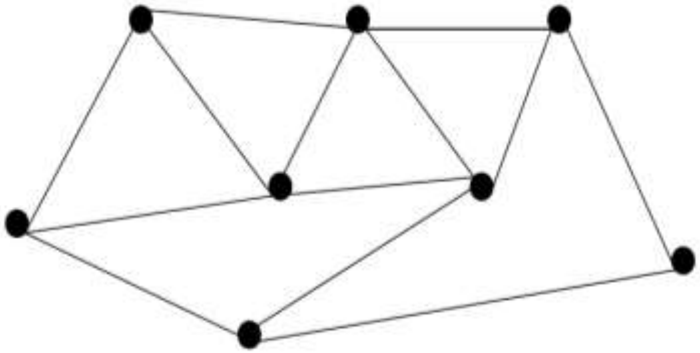
  Problems restricted to *acyclic* graphs are often trivially solvable (→3COL).

- Moreover, many graph problems are polynomially solvable if restricted to instances of *low cyclicity*.

How can we measure the degree of cyclicity?

How much “cyclicity” in this graph?

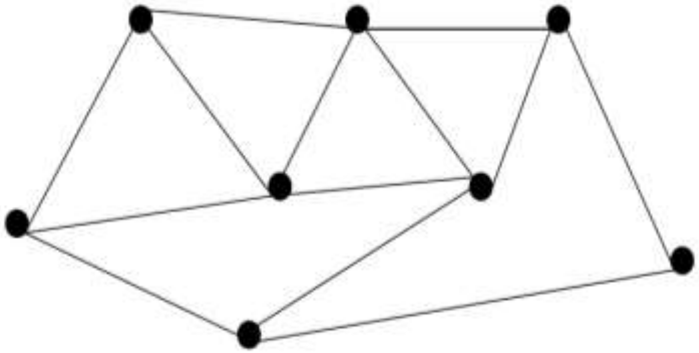
- Suggest a measure of distance from an acyclic graph



Three Early Approaches

(1) **Feedback vertex set**

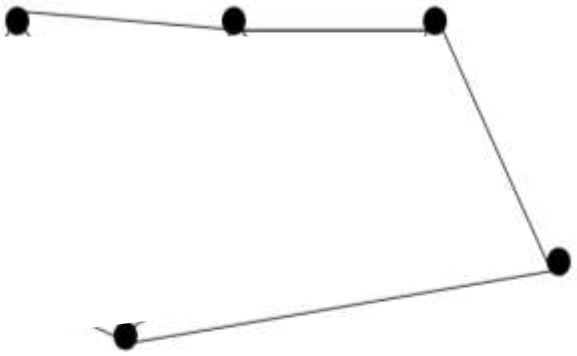
Set of vertices whose deletion makes the graph acyclic



The feedback vertex number

1 Feedback vertex number

Min. number of vertices I need to eliminate to make the graph acyclic

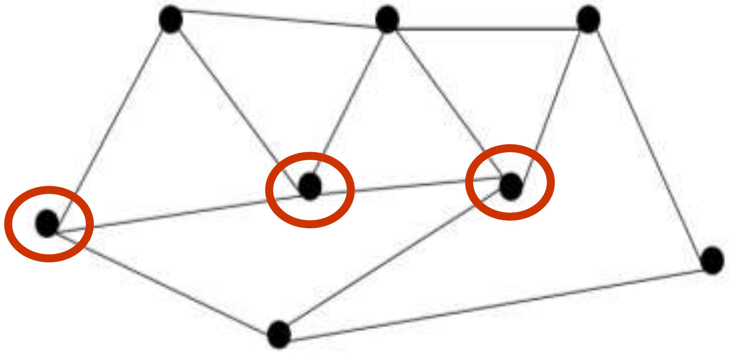


fwn(G)=3

FVN: Properties

[1] **Feedback vertex number**

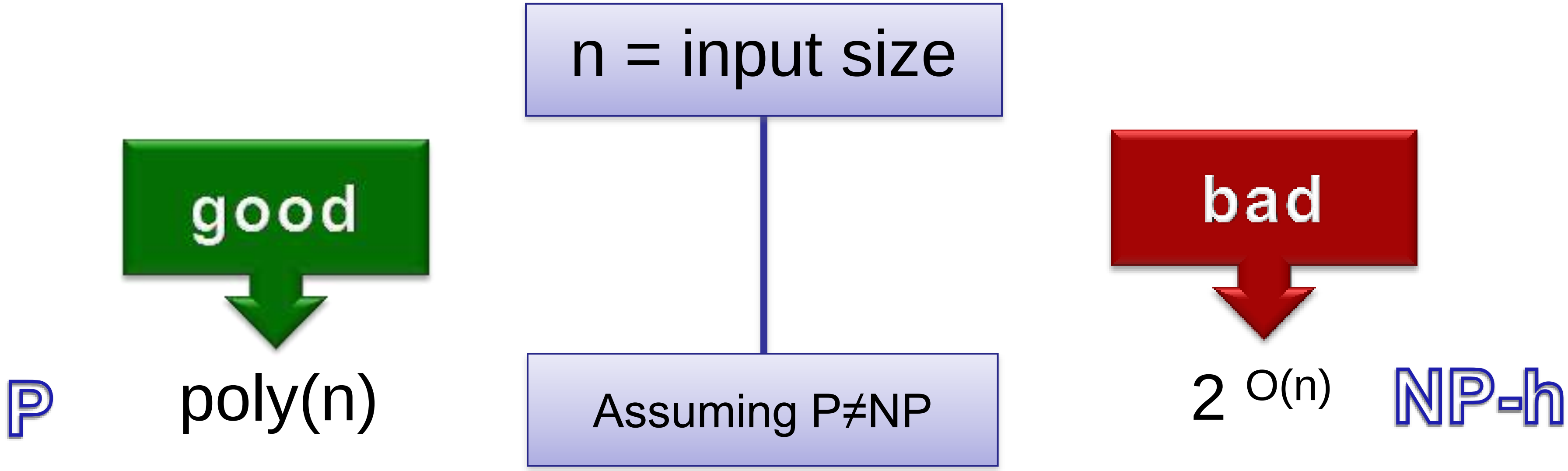
Min. number of vertices I need to eliminate to make the graph acyclic



fwn(G)=3

- *Is this really a good measure for the "degree of acyclicity" ?*
- **Pro:** For fixed k we can check efficiently whether *fwn(G) ≤ k*
  - What does it mean *efficiently* when parameter *k* is fixed?

Classical Computational Complexity



n = input size

good

bad

P poly(n)

Assuming P≠NP

$2^{O(n)}$ NP-h

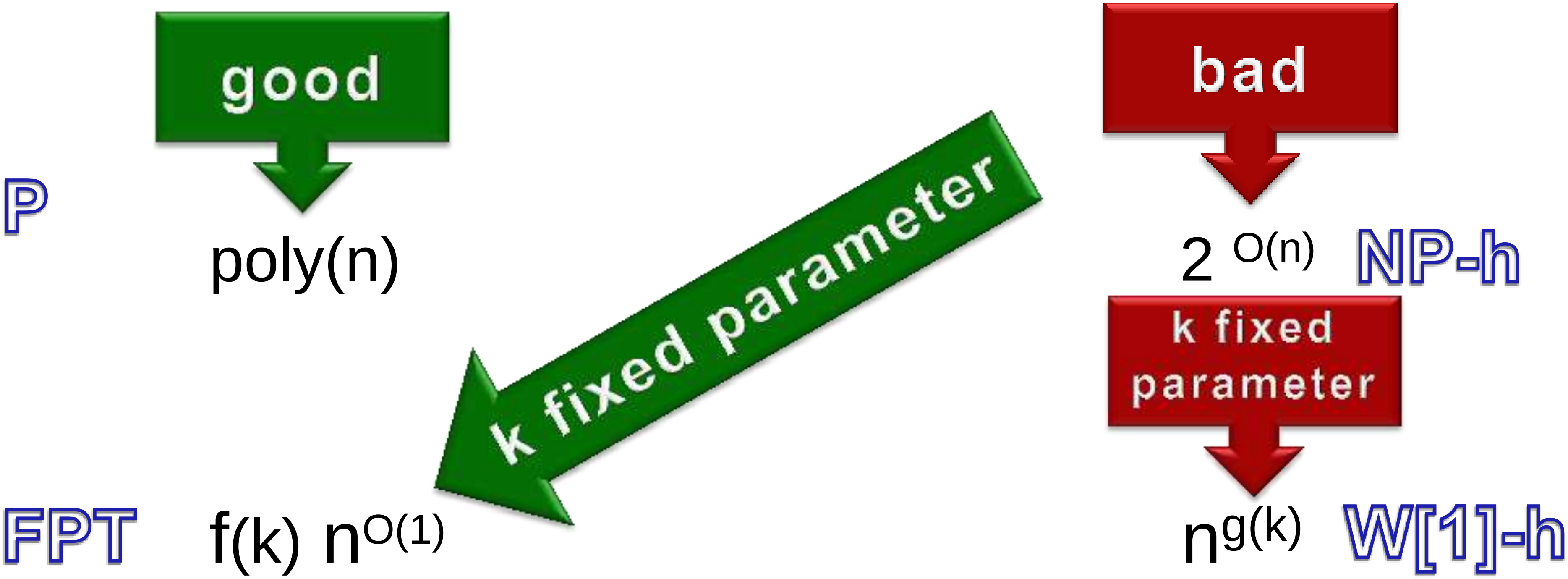
But…

- In many problems there exists some part of the input that are quite small in practical applications
- Natural parameters
- Many NP-hard problems become easy if we fix such parameters (or we assume they are below some fixed threshold)
- Positive examples: k-vertex cover, k-feedback vertex set, k-clique, …
- Negative examples: k-coloring, k-CNF, …

Parameterized Complexity

- Initiated by Downey and Fellows, late ‘80s

n = input size



good

P

poly(n)

k fixed parameter

FPT

$f(k)\ n^{O(1)}$

bad

$2^{O(n)}$

NP-h

k fixed parameter

$n^{g(k)}$

W[1]-h

Typical assumption: FPT ≠ W[1]

W[1]-hard problems: k-clique

- k-clique is hard w.r.t. fixed parameter complexity!

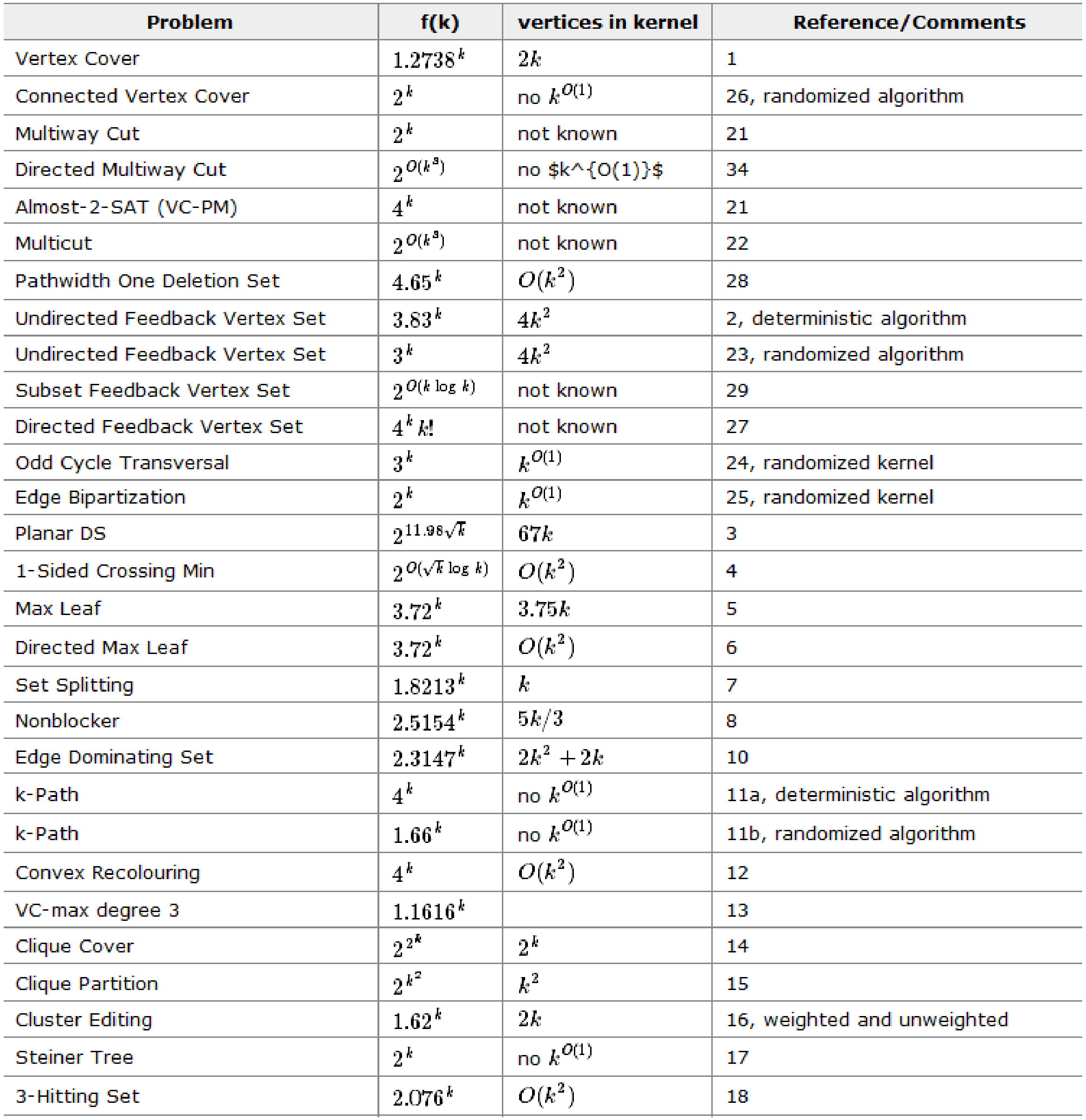
INPUT: A graph *G=(V,E)*

PARAMETER: Natural number *k*

- Does *G* have a clique over *k* vertices?

FPT races

- http://fpt.wikidot.com/



Problem	f(k)	vertices in kernel	Reference/Comments
Vertex Cover	1.2738^k	$2k$	1
Connected Vertex Cover	2^k	no $k^{O(1)}$	26, randomized algorithm
Multiway Cut	2^k	not known	21
Directed Multiway Cut	$2^{O(k^3)}$	no \$k^{O(1)}\$	34
Almost-2-SAT (VC-PM)	4^k	not known	21
Multicut	$2^{O(k^3)}$	not known	22
Pathwidth One Deletion Set	4.65^k	$O(k^2)$	28
Undirected Feedback Vertex Set	3.83^k	$4k^2$	2, deterministic algorithm
Undirected Feedback Vertex Set	3^k	$4k^2$	23, randomized algorithm
Subset Feedback Vertex Set	$2^{O(k \log k)}$	not known	29
Directed Feedback Vertex Set	$4^k k!$	not known	27
Odd Cycle Transversal	3^k	$k^{O(1)}$	24, randomized kernel
Edge Bipartization	2^k	$k^{O(1)}$	25, randomized kernel
Planar DS	$2^{11.98\sqrt{k}}$	$67k$	3
1-Sided Crossing Min	$2^{O(\sqrt{k} \log k)}$	$O(k^2)$	4
Max Leaf	3.72^k	$3.75k$	5
Directed Max Leaf	3.72^k	$O(k^2)$	6
Set Splitting	1.8213^k	k	7
Nonblocker	2.5154^k	$5k/3$	8
Edge Dominating Set	2.3147^k	$2k^2 + 2k$	10
k-Path	4^k	no $k^{O(1)}$	11a, deterministic algorithm
k-Path	1.66^k	no $k^{O(1)}$	11b, randomized algorithm
Convex Recolouring	4^k	$O(k^2)$	12
VC-max degree 3	1.1616^k		13
Clique Cover	2^{2^k}	2^k	14
Clique Partition	2^{k^2}	k^2	15
Cluster Editing	1.62^k	$2k$	16, weighted and unweighted
Steiner Tree	2^k	no $k^{O(1)}$	17
3-Hitting Set	2.076^k	$O(k^2)$	18

FPT Tractability of Feedback Vertex Set

INPUT: A graph $G=(V,E)$

PARAMETER: Natural number k

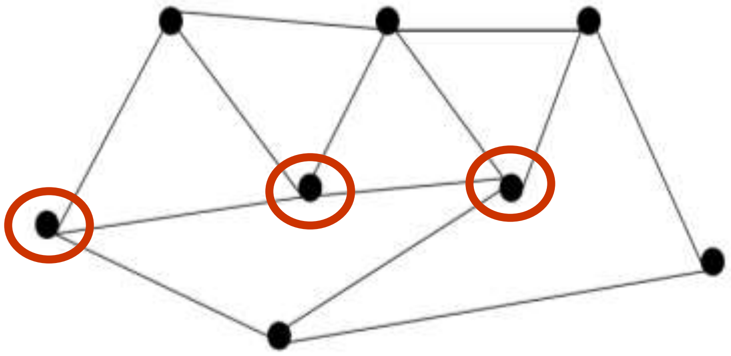
- Does G has a feedback vertex set of k vertices?

- Naïve algorithm: $O(n^{k+1})$ *Not good!*
- Solvable in $O((2k+1)^k n^2)$ [Downey and Fellows ‘92]
- A practical randomized algorithm runs in time: $O(4^k kn)$ [Becker et al 2000]

Feedback Vertex Set: troubles

(1) **Feedback vertex number**

Min. number of vertices I need to eliminate to make the graph acyclic

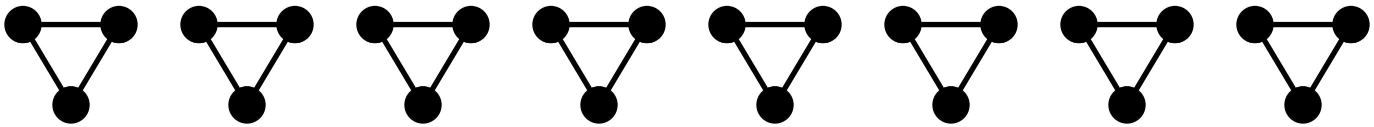


fwn(G)=3

Is this really a good measure for the "degree of acyclicity" ?

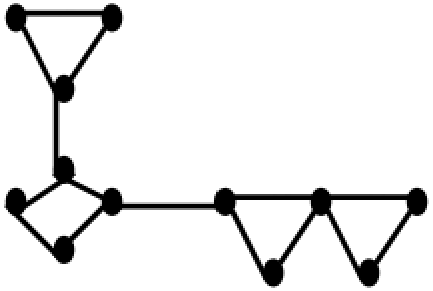
Pro: For fixed k we can check in quadratic time if fwn(G)=k (FPT) .

Con: Very simple graphs can have large FVN:

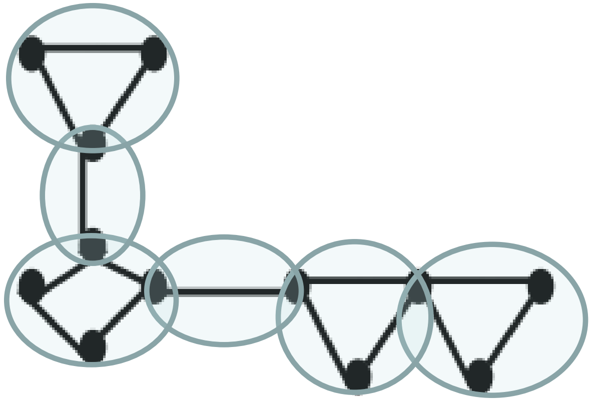


Feedback edge number

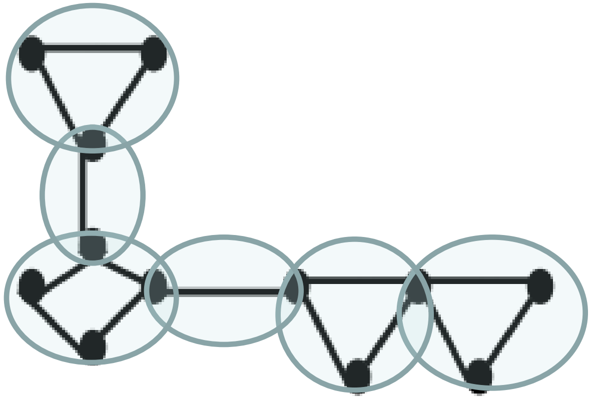
(2) Feedback edge number → same problem.



Any idea for further techniques?



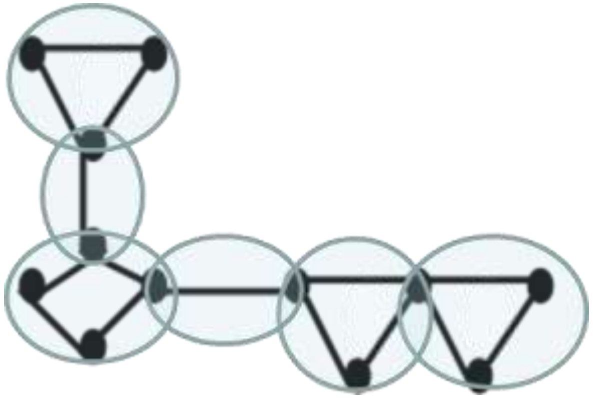
Yes! A tree of clusters (subproblems)



- Well known graph properties:
  - A biconnected component is a maximal subgraph that remains connected after deleting any single vertex
  - In any graph, its biconnected components form a tree

Biconnected width

[3] **Maximum size of biconnected components**

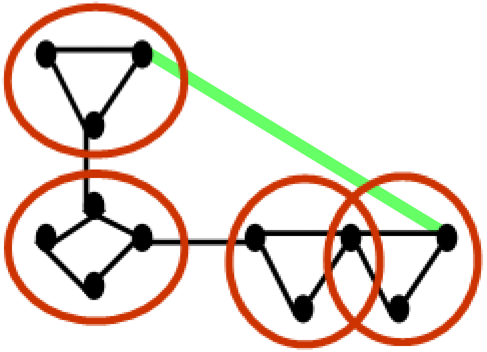


bcw(G)=4

Pro: Actually bcw(G) can be computed in linear time

Drawbacks of BiComp

[3] **Maximum size of biconnected components**



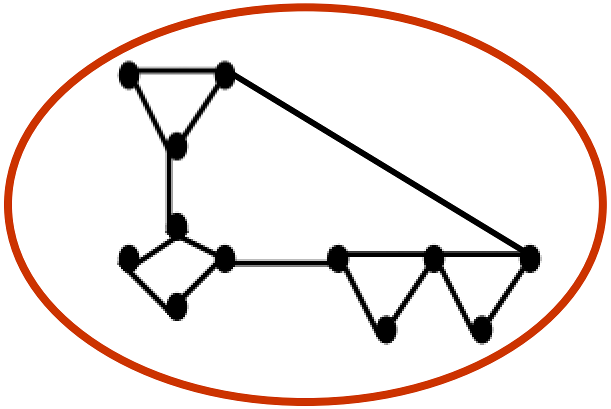
bcw(G)=4

Pro: Actually bcw(G) can be computed in linear time

Con: Adding a single edge may have tremendous effects to bcw(G)

Drawbacks of BiComp

[3] **Maximum size of biconnected components**



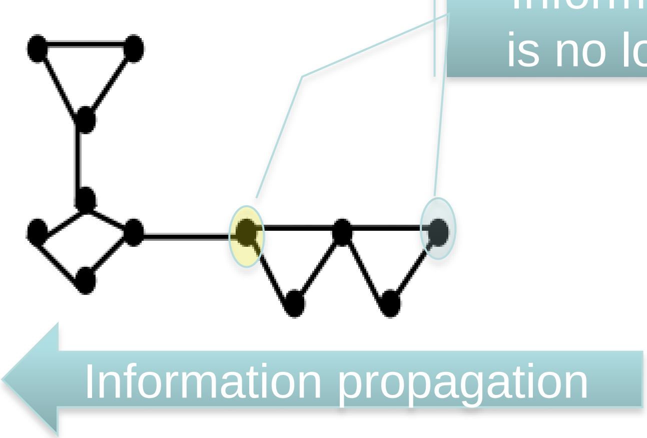
bcw(G)=~~4~~ 12

Pro: Actually bcw(G) can be computed in linear time

Con: Adding a single edge may have tremendous effects to bcw(G)

Can we do better?

- Hint:
  - why should clusters of vertices be of this limited kind?
- Use arbitrary (possibly small) sets of vertices!
  - How can we arrange them in some tree-shape?
  - What is the key property of tree-like structures (in most applications)?

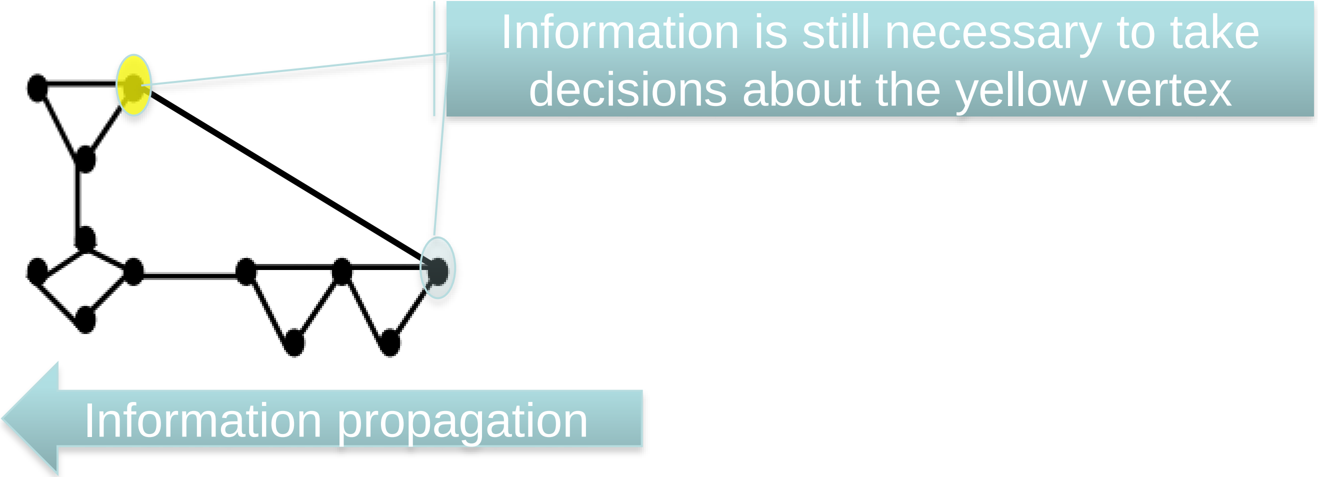


Information on the rightmost vertex is no longer necessary (quite often)

Information propagation

Can we do better?

- Hint:
  - why should clusters of vertices be of this limited kind?
- Use arbitrary (possibly small) sets of vertices!
  - How can we arrange them in some tree-shape?
  - What is the key property of tree-like structures, in applications?



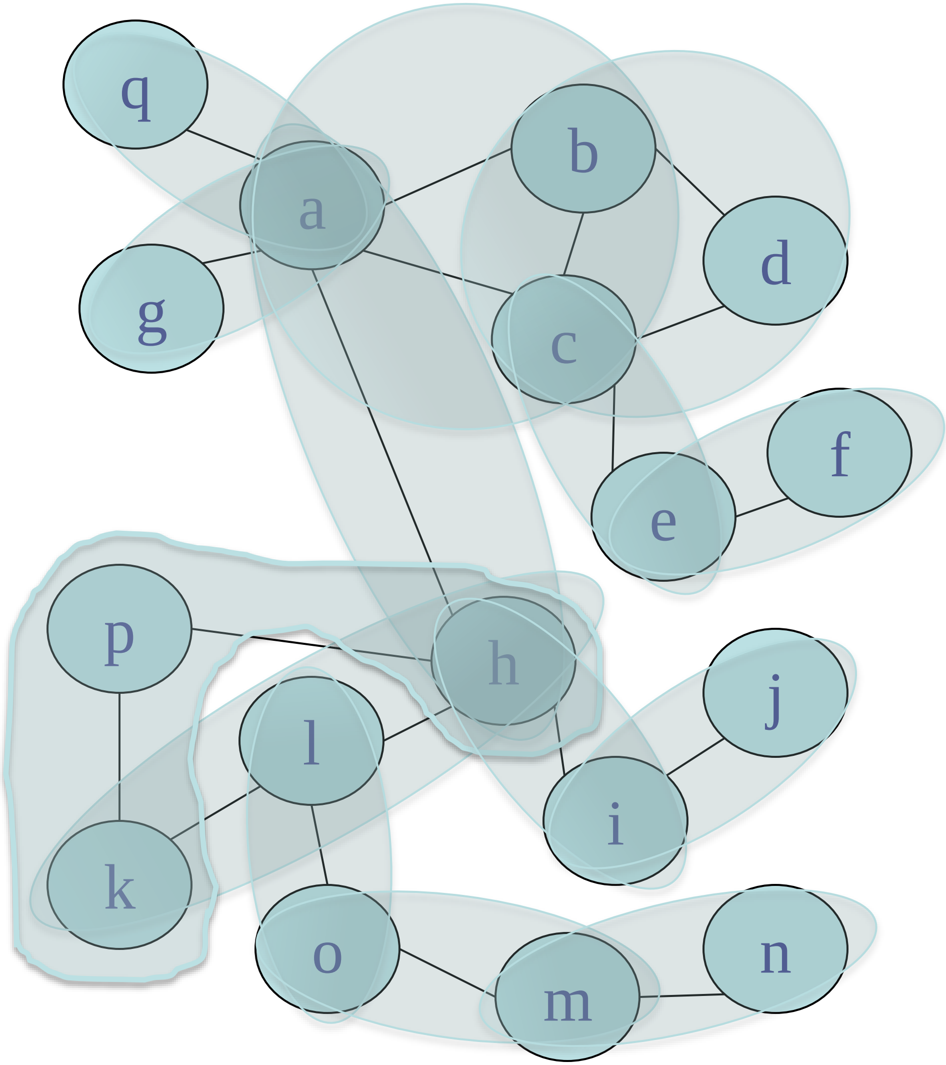
Information is still necessary to take decisions about the yellow vertex

Information propagation

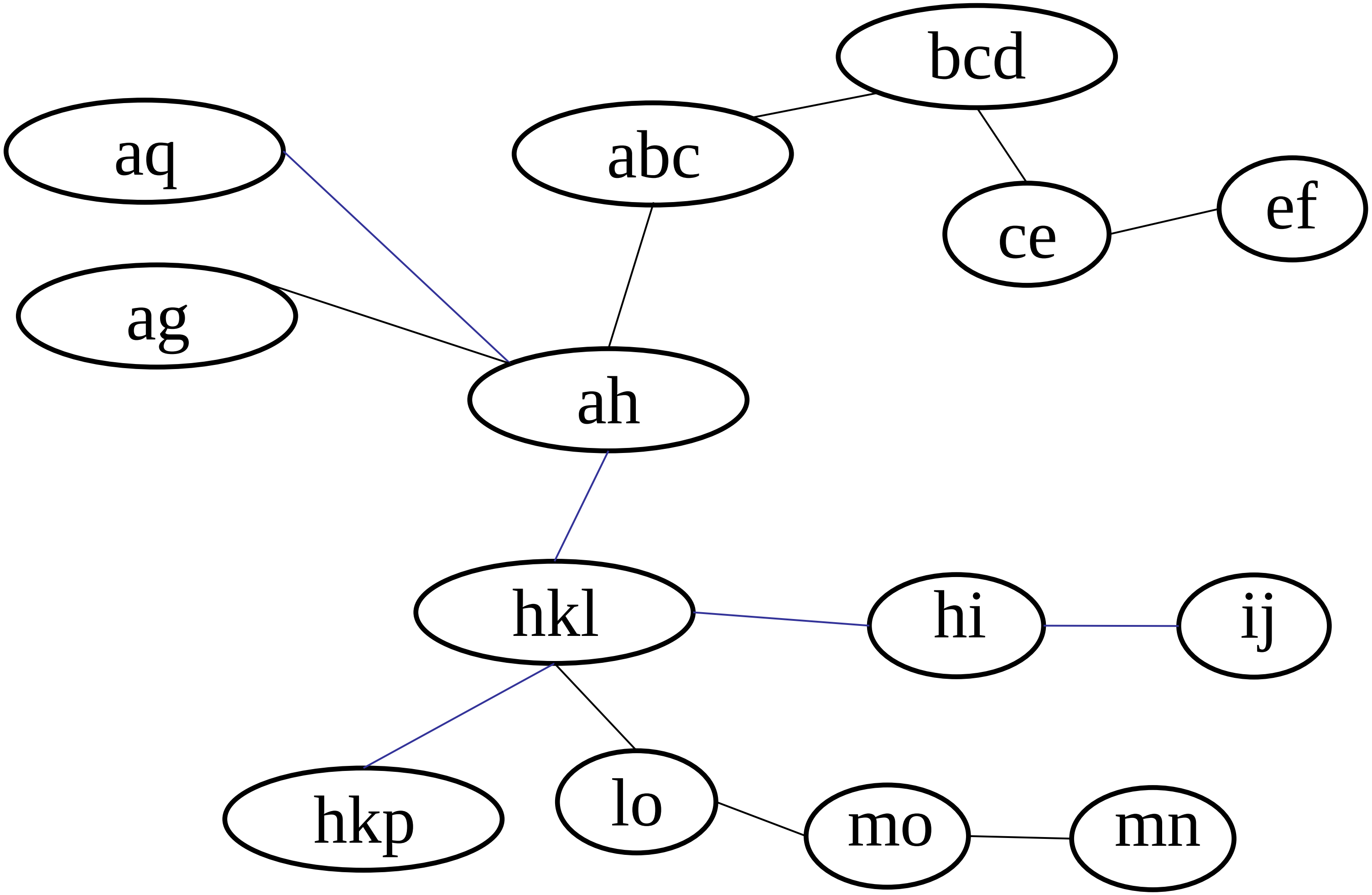
Outline of PART I

- Introduction to Decomposition Methods
- **Tree Decompositions**
- Applications of Tree Decompositions

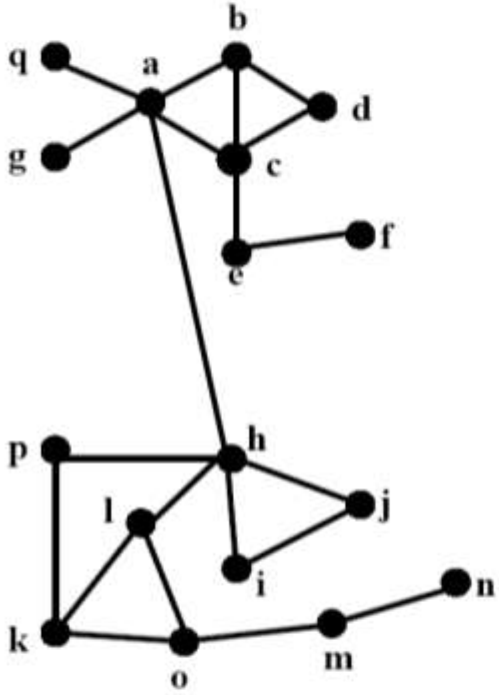
Tree Decompositions [Robertson & Seymour ‘86]



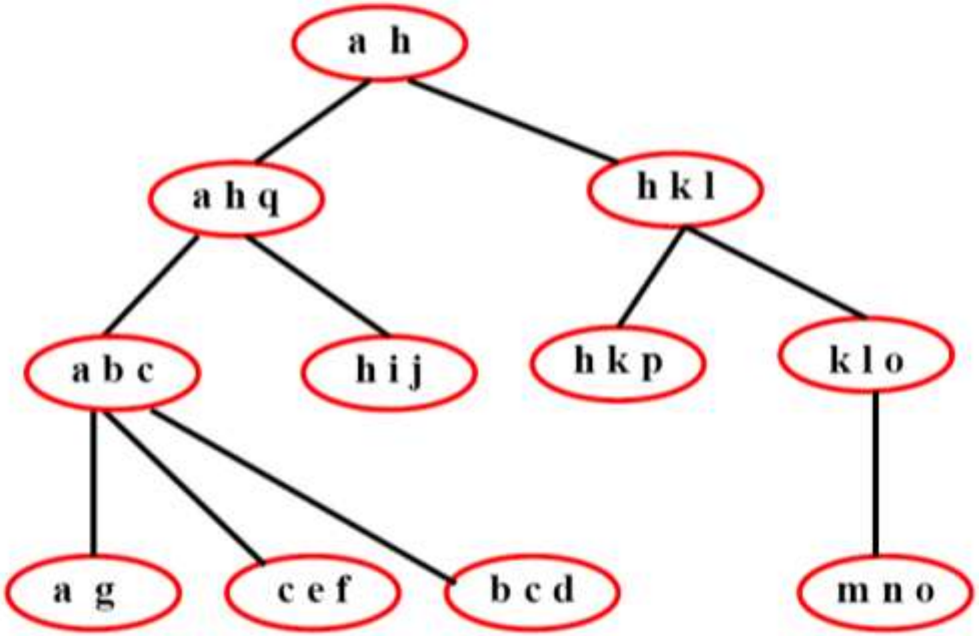
Tree Decompositions [Robertson & Seymour '86]



Tree Decompositions [Robertson & Seymour ‘86]

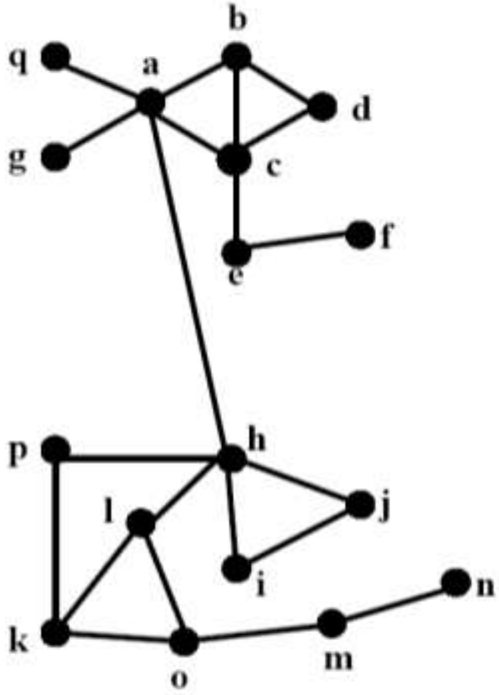


Graph G

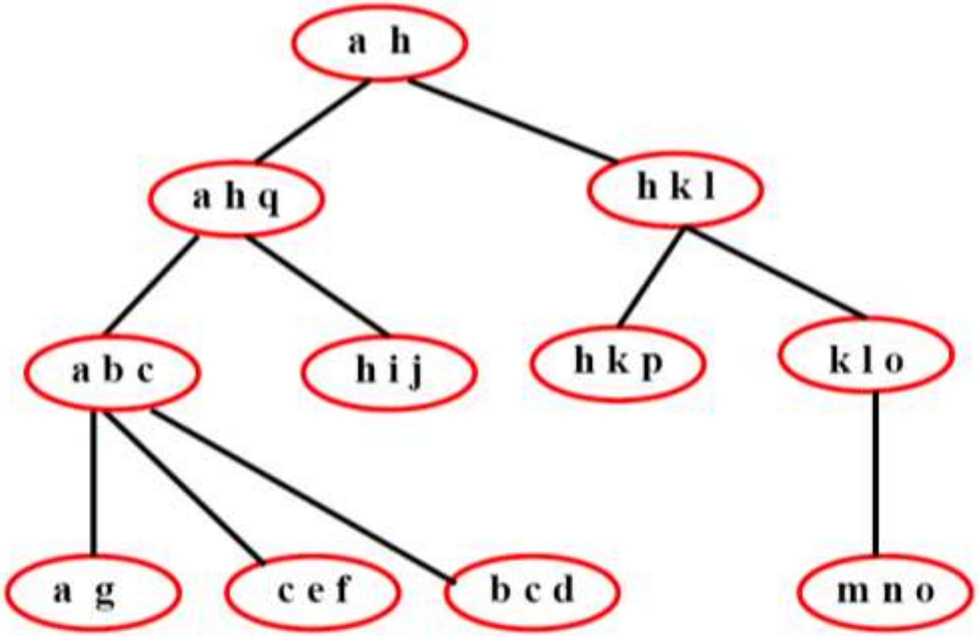


Tree decomposition of width 2 of G

Tree Decompositions [Robertson & Seymour ‘86]



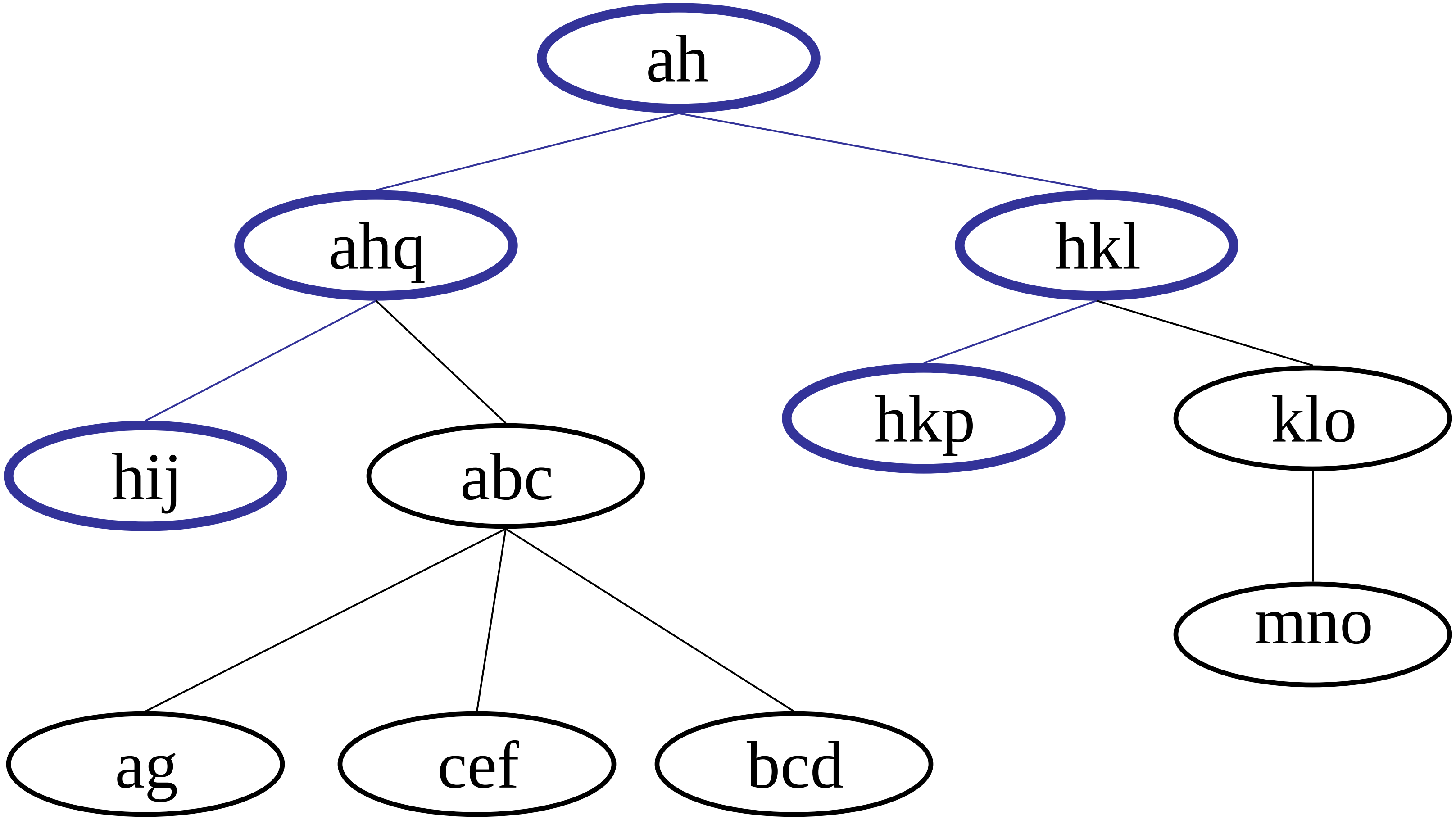
Graph G



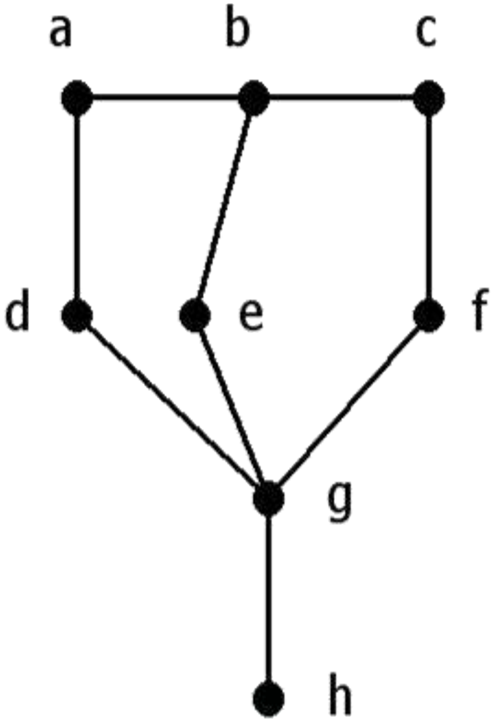
Tree decomposition of width 2 of G

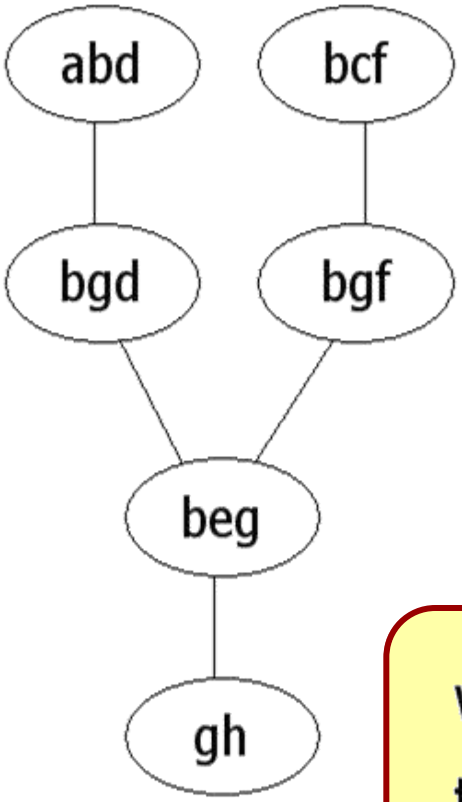
- Every edge realized in some bag
- Connectedness condition

Connectedness condition for *h*



Tree Decompositions and Treewidth

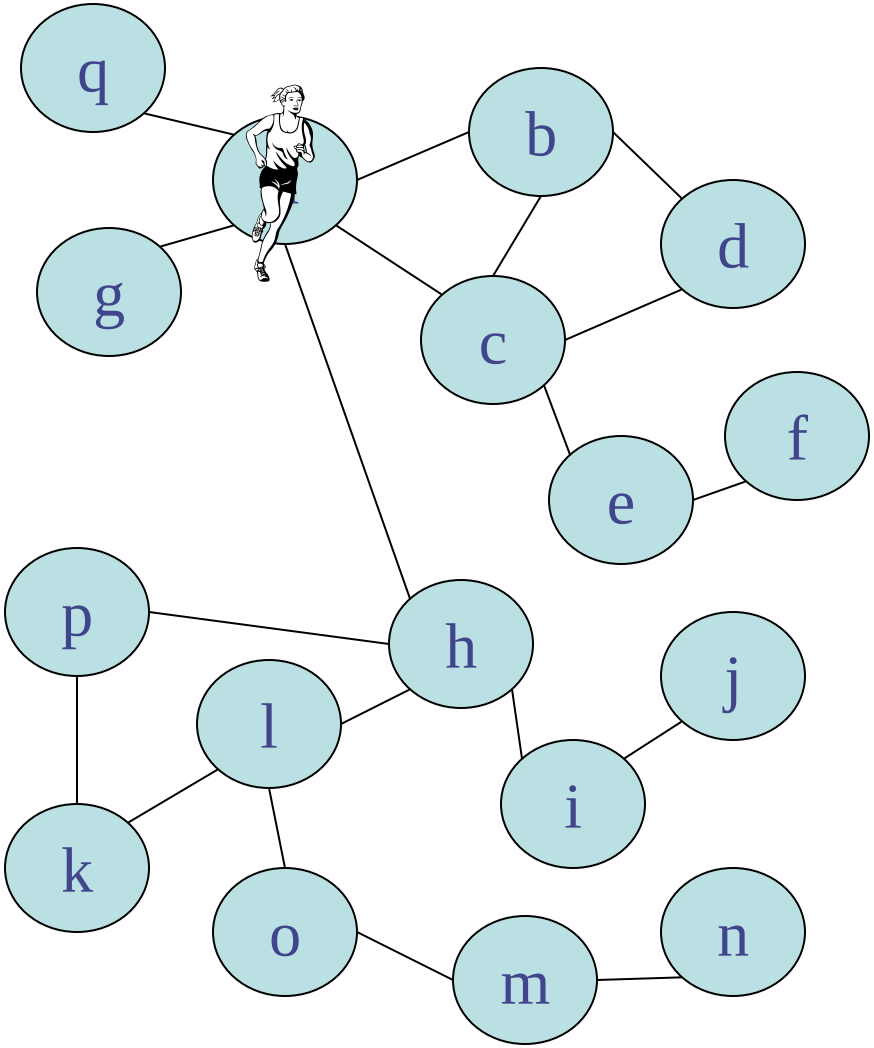




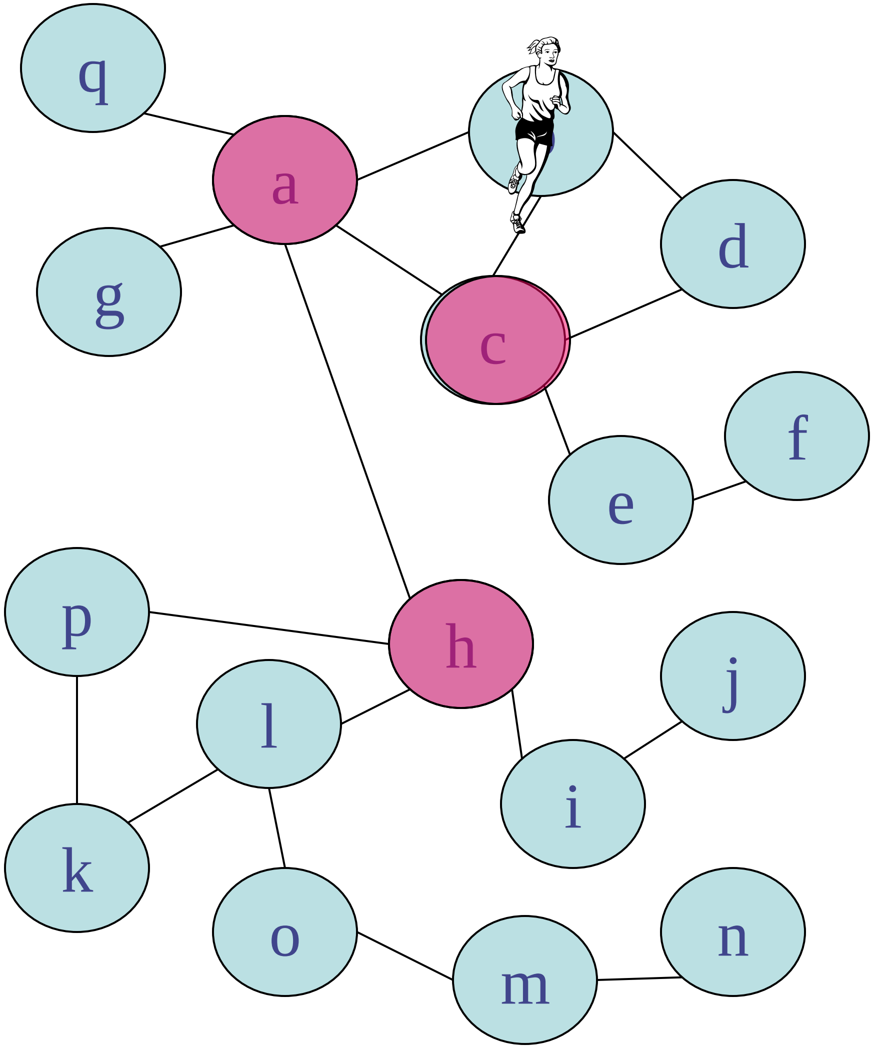
$$\text{width}(T,X_i) = \max |X_i| - 1$$

$$\text{tw}(G) = \min \text{width}(T,X_i)$$

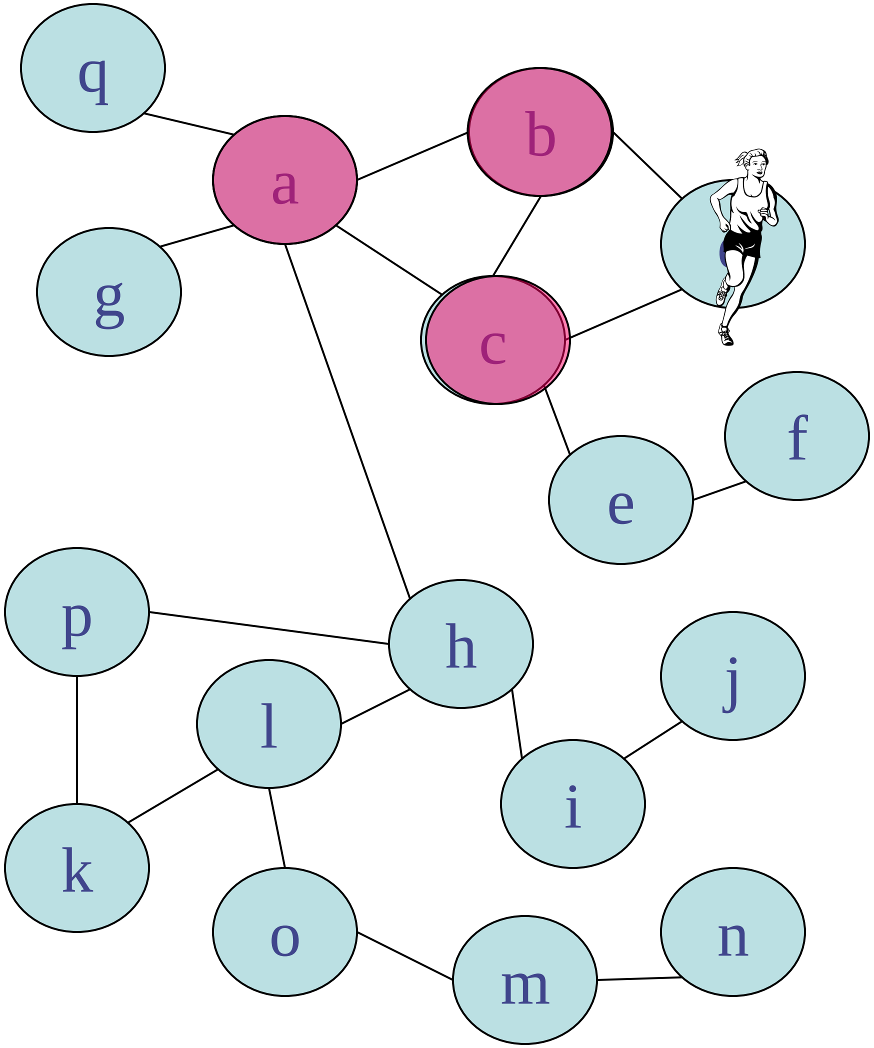
Playing the Robber & Cops Game



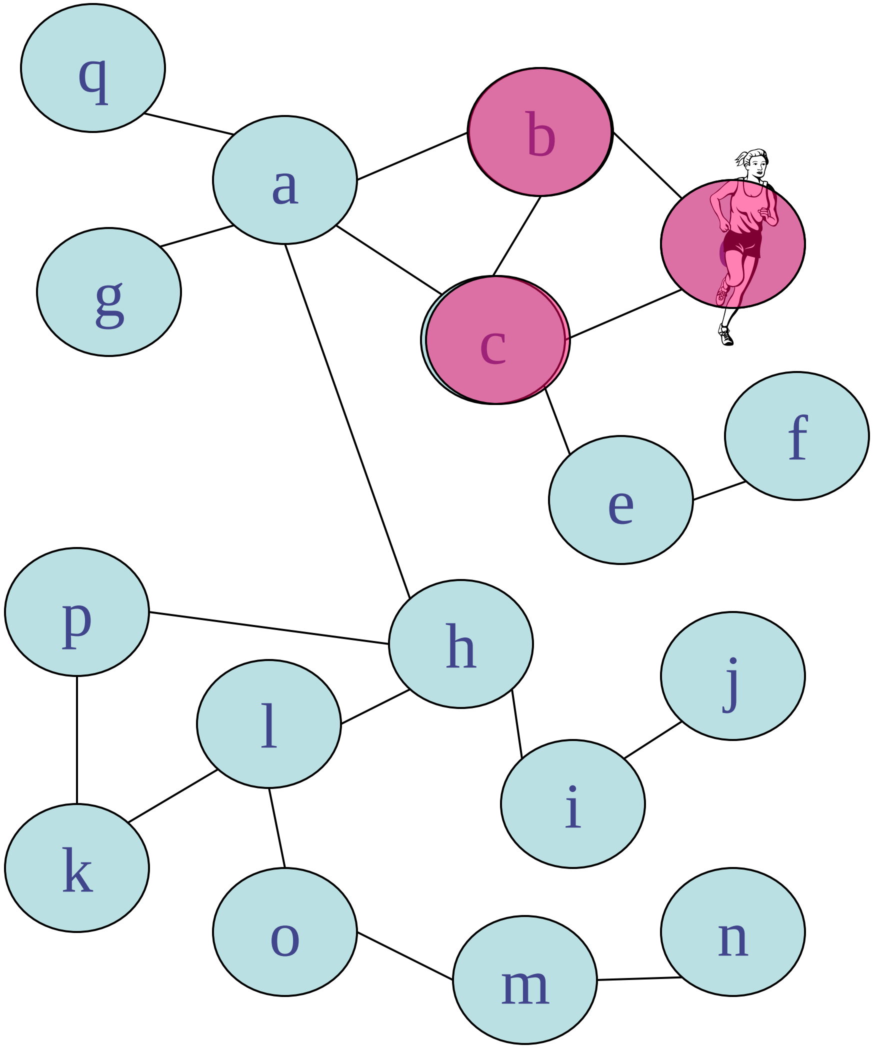
Playing the Robber & Cops Game



Playing the Robber & Cops Game



Playing the Robber & Cops Game



Properties of Treewidth

- tw(acyclic graph)=1
- tw(cycle) = 2
- tw(G+v) $\leq$ tw(G)+1
- tw(G+e) $\leq$ tw(G)+1
- tw(K_n) = n-1
- tw is fixed-parameter tractable (parameter: treewidth)

Outline of PART I

- Introduction to Decomposition Methods
- Tree Decompositions
- **Applications of Tree Decompositions**

Use of Tree Decompositions

1. **Prove Tractability of bounded-width instances**

   a) Genuine tractability: $O(n^{f(w)})$-bounds

   b) Fixed-Parameter tractability: $f(w) * O(n^k)$

2. **Tool for proving general tractability**

   a) Prove tractability for both large & small width

   b) Prove all yes-instances to have small width

Use of Tree Decompositions

1. **Prove Tractability of bounded-width instances**

   a) Genuine tractability: $O(n^{f(w)})$-bounds

   constraint satisfaction = conjunctive database queries

   b) Fixed-Parameter tractability: $f(w)*O(n^k)$

   multicut problem

2. **Tool for proving general tractability**

   a) Prove tractability for both large & small width

   finding even cycles in graphs – ESO over graphs

   b) Prove all yes-instances to have small width

   the Partner Unit Problem

Use of Tree Decompositions

1. **Prove Tractability of bounded-width instances**

   a) Genuine tractability: O($n^{f(w)}$)-bounds

   In PART II

   b) Fixed-Parameter tractability: f(w)$*$O(n^k)

2. **Tool for proving general tractability**

   a) Prove tractability for both large & small width

   b) Prove all yes-instances to have small width

Use of Tree Decompositions

1. **Prove Tractability of bounded-width instances**
   a) Genuine tractability: $O(n^{f(w)})$-bounds

   b) Fixed-Parameter tractability: $f(w) * O(n^k)$

2. **Tool for proving general tractability**
   a) Prove tractability for both large & small width

   b) Prove all yes-instances to have small width

An important Metatheorem

Courcelle's Theorem [1987]

Let P be a problem on graphs that can be formulated in **Monadic Second Order Logic** (MSO).

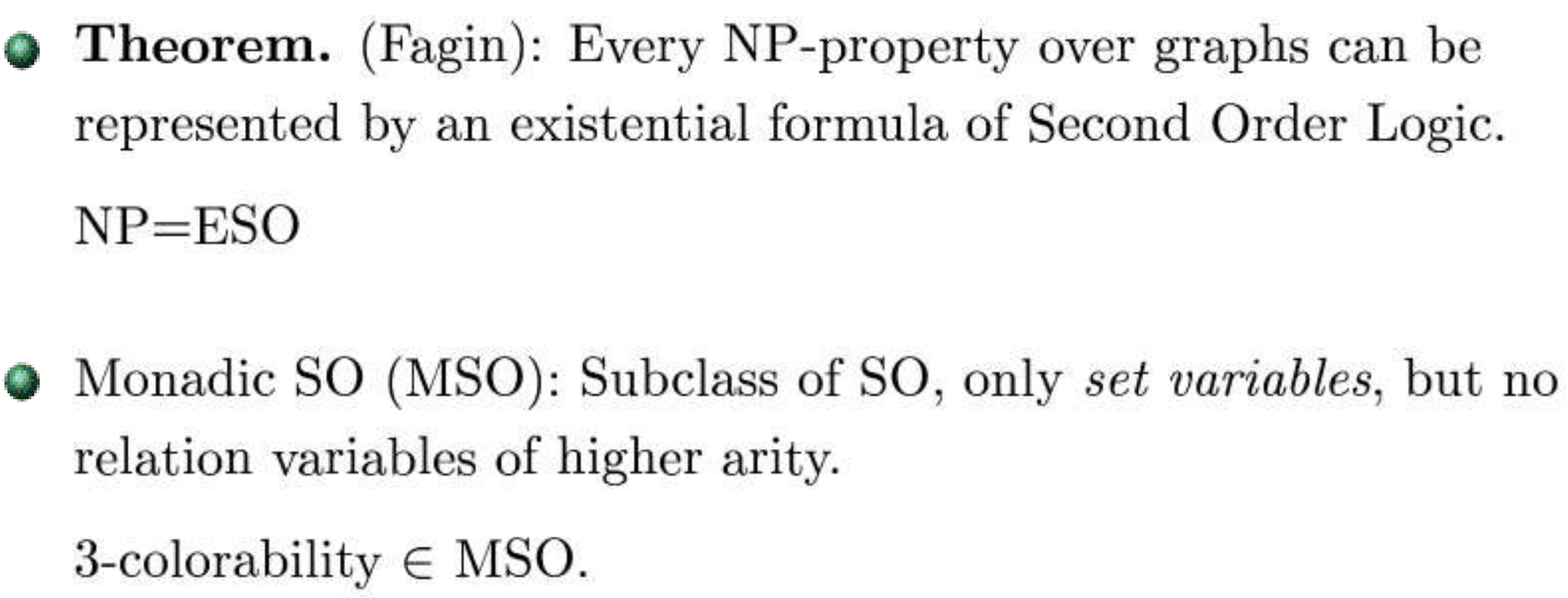
Then P can be solved in liner time on graphs of bounded treewidth

An important Metatheorem

Courcelle's Theorem [1987]

Let P be a problem on graphs that can be formulated in **Monadic Second Order Logic** (MSO).

Then P can be solved in liner time on graphs of bounded treewidth



- **Theorem.** (Fagin): Every NP-property over graphs can be represented by an existential formula of Second Order Logic.

  NP=ESO

- Monadic SO (MSO): Subclass of SO, only *set variables*, but no relation variables of higher arity.

  3-colorability $\in$ MSO.

Three Colorability in MSO

$$
\begin{array}{lll}
(\exists R, G, B) & [& (\forall x\, (R(x) \vee G(x) \vee B(x))) \\
 & \wedge & (\forall x (R(x) \Rightarrow (\neg G(x) \wedge \neg B(x)))) \\
 & \wedge & \ldots \\
 & \wedge & \ldots \\
 & \wedge & (\forall x, y (E(x,y) \Rightarrow (R(x) \Rightarrow (G(x) \vee B(y))))) \\
 & \wedge & (\forall x, y (E(x,y) \Rightarrow (G(x) \Rightarrow (R(x) \vee B(y))))) \\
 & \wedge & (\forall x, y (E(x,y) \Rightarrow (B(x) \Rightarrow (R(x) \vee G(y)))))\,]
\end{array}
$$

Master Theorems for Treewidth

Courcelle's Theorem: Problems expressible in MSO_2 are solvable in linear time on structures of bounded treewidth

…and in LOGSPACE [Elberfeld, Jacoby,Tantau]

Example – Graph Coloring

$$\exists P\ \forall x \forall y :\ (E(x,y) \rightarrow (P(x) \neq P(y))$$

Master Theorems for Treewidth

Arnborg, Lagergren, Seese '91:

Optimization version of Courcelle's Theorem:

Finding an optimal set P such that $G \models \Phi(P)$ is FP-linear over inputs G of bounded treewidth.

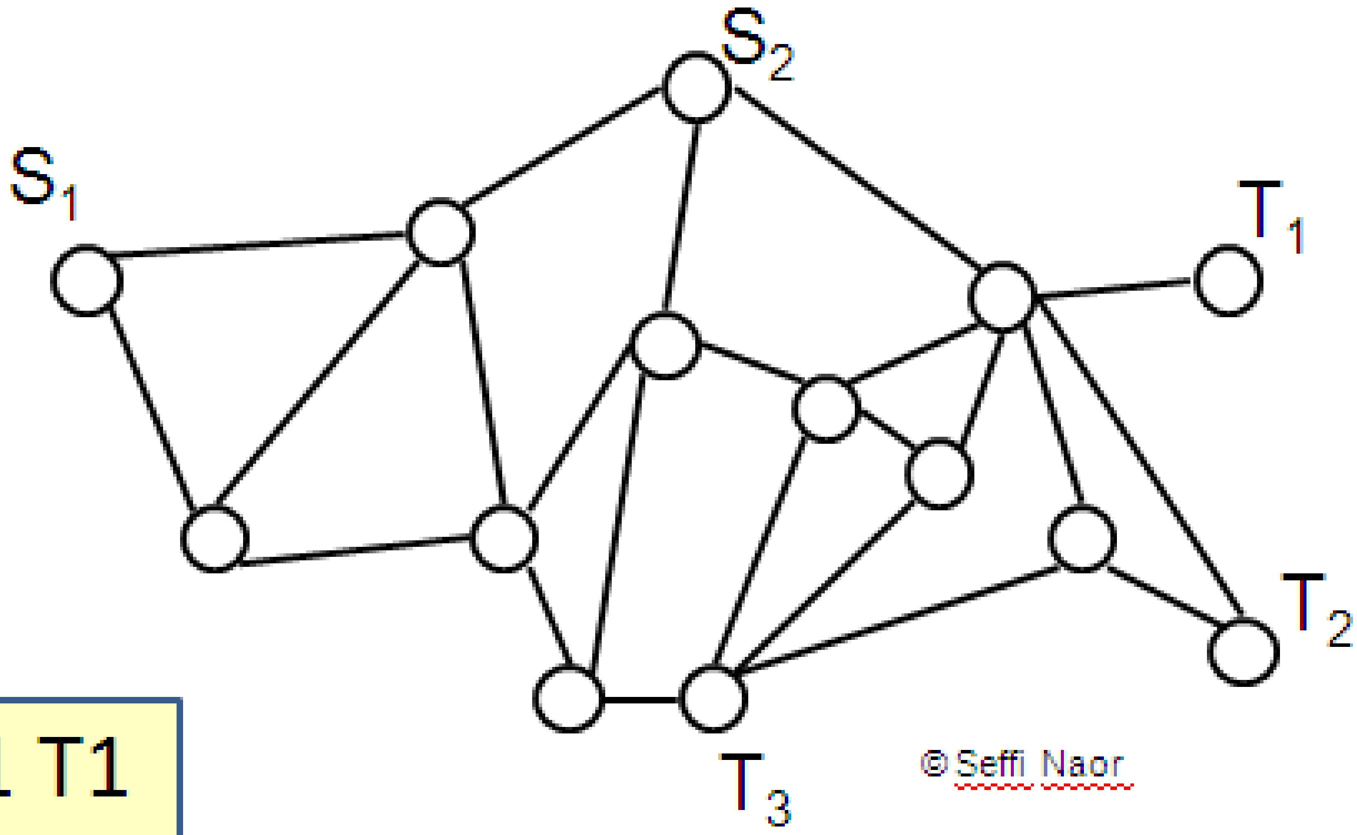
Example:

Given a graph G=(V,E)

Find a ***smallest*** P such that

$$\forall x \forall y : (E(x,y) \rightarrow (P(x) \not\equiv P(y))$$

Unrestricted Vertex Multicut Problems



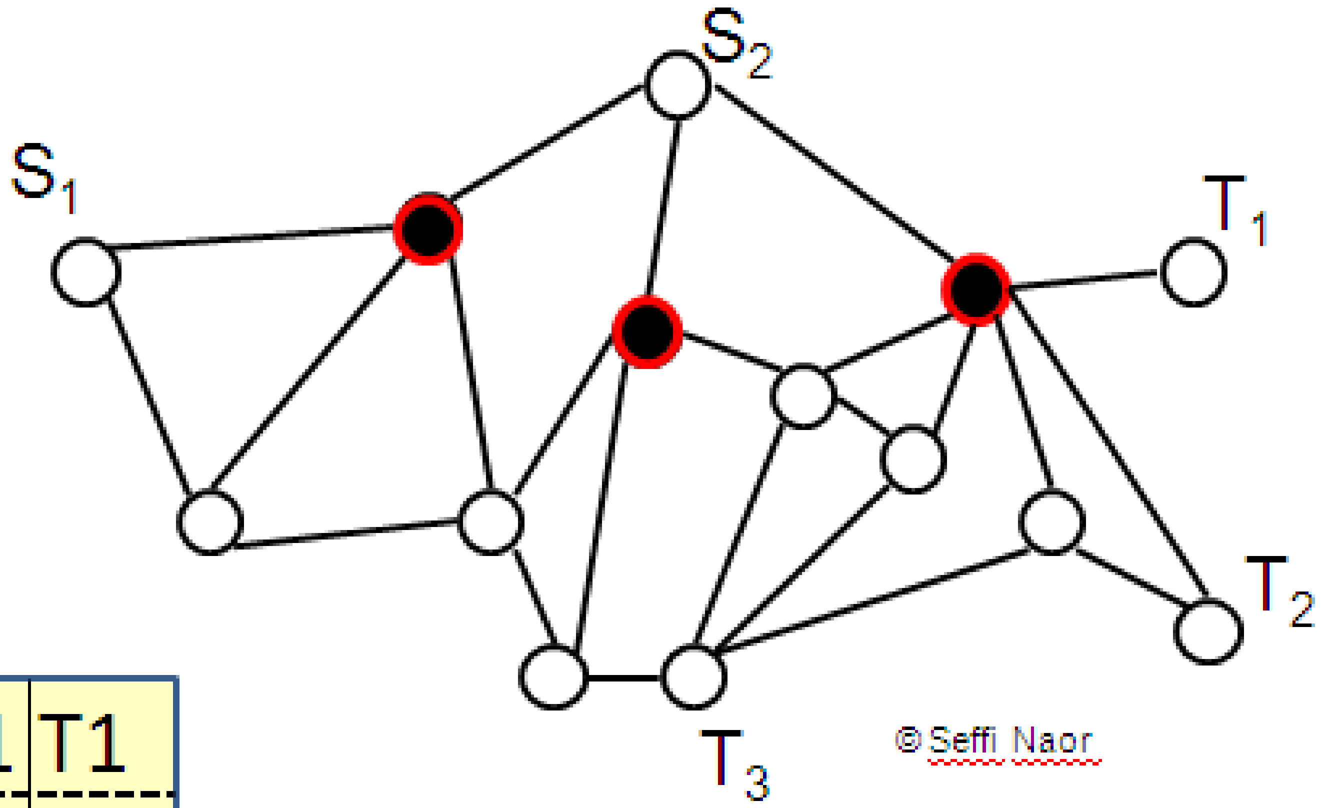
© Seffi Naor

H:

S1	T1
S2	T3
S2	T2

Find minimum-cardinality vertex set separating Si from Tj for each tuple <Si,Tj> in relation H

Unrestricted Vertex Multicut Problems



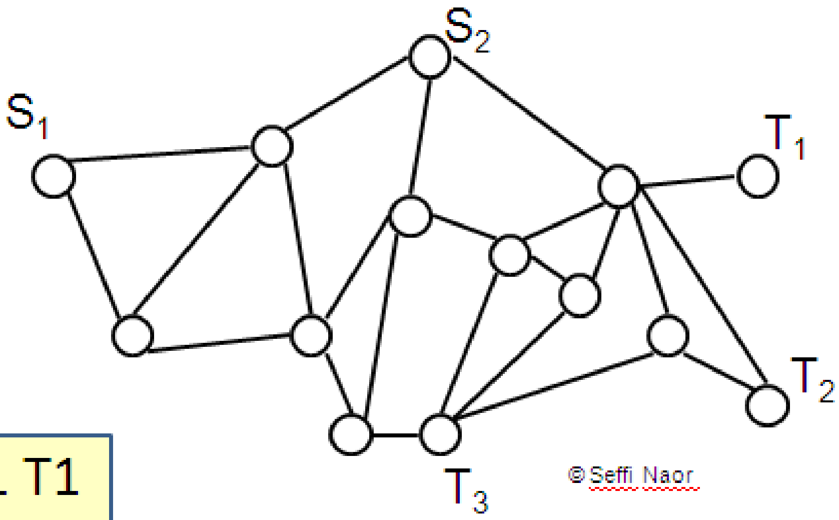
© Seffi Naor

H:

S1	T1
S2	T3
S2	T2

Unrestricted Vertex Multicut Problems

Results



H:

S1 T1
S2 T3
S2 T2

[Guo et al. 06] UVMC FPT if |S|, |C| and tree-width fixed

[G. & Tien Lee] UVMC FPT if overall structure has bounded tw.

using master theorem by Arnborg, Lagergren and Seese.

Unrestricted Vertex Multicut Problems

PROOF

Definition 8. On structures $\mathcal{A} = (V, E, H)$ as above, let *connects*(S, x, y) be defined as follows:

$$S(x) \wedge S(y) \wedge \forall P \Big(\big(P(x) \wedge \neg P(y)\big) \rightarrow \big(\exists v \exists w \, (S(v) \wedge S(w) \wedge P(v) \wedge \neg P(w) \wedge E(v, w))\big) \Big).$$

$$uvmc(X) \quad \equiv \quad \forall x \, \forall y \left(H(x, y) \rightarrow \forall S \big(\text{connects}(S, x, y) \rightarrow \exists v (X(v) \wedge S(v))\big) \right)$$

Minimize *X* in *uvmc*

X intersects each set that connects x and y

Use of Tree Decompositions

1. **Prove Tractability of bounded-width instances**

   a) Genuine tractability: $O(n^{f(w)})$-bounds

   b) Fixed-Parameter tractability: $f(w) * O(n^k)$

2. **Tool for proving general tractability**

   a) Prove tractability for both large & small width

   b) Prove all yes-instances to have small width

The Generalized Even Cycle Problem

INPUT: A graph G, a constant k.

QUESTION: Decide whether G has a cycle of length 0 (mod k)

In the past century, this was an open problem for a long time.

Carsten Thomassen in 1988 proved it polynomial for *all graphs* using treewidth as a tool.

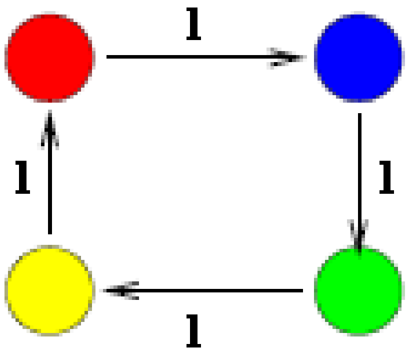
Proof Idea

Small Treewidth (≤c)

"cycle of length 0 (mod k)" can be expressed un MSO

example

k=4

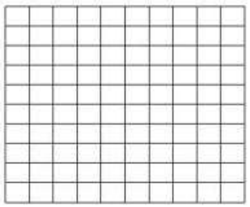


→Courcelle's Theorem

(but was not known then…)

Large Treewidth (>c)

∀k ∃c: each graph G with tw(G)>c contains a subdivision of the f(k)-grid. [for suitable f]



∀n>f(k), each subdivision of f(k)-grid contains a cycle of length 0 (mod k).

Long Term Research Programme

Determine the complexity of SO fragments over finite structures.

Finite structures: words (strings), graphs, relational databases

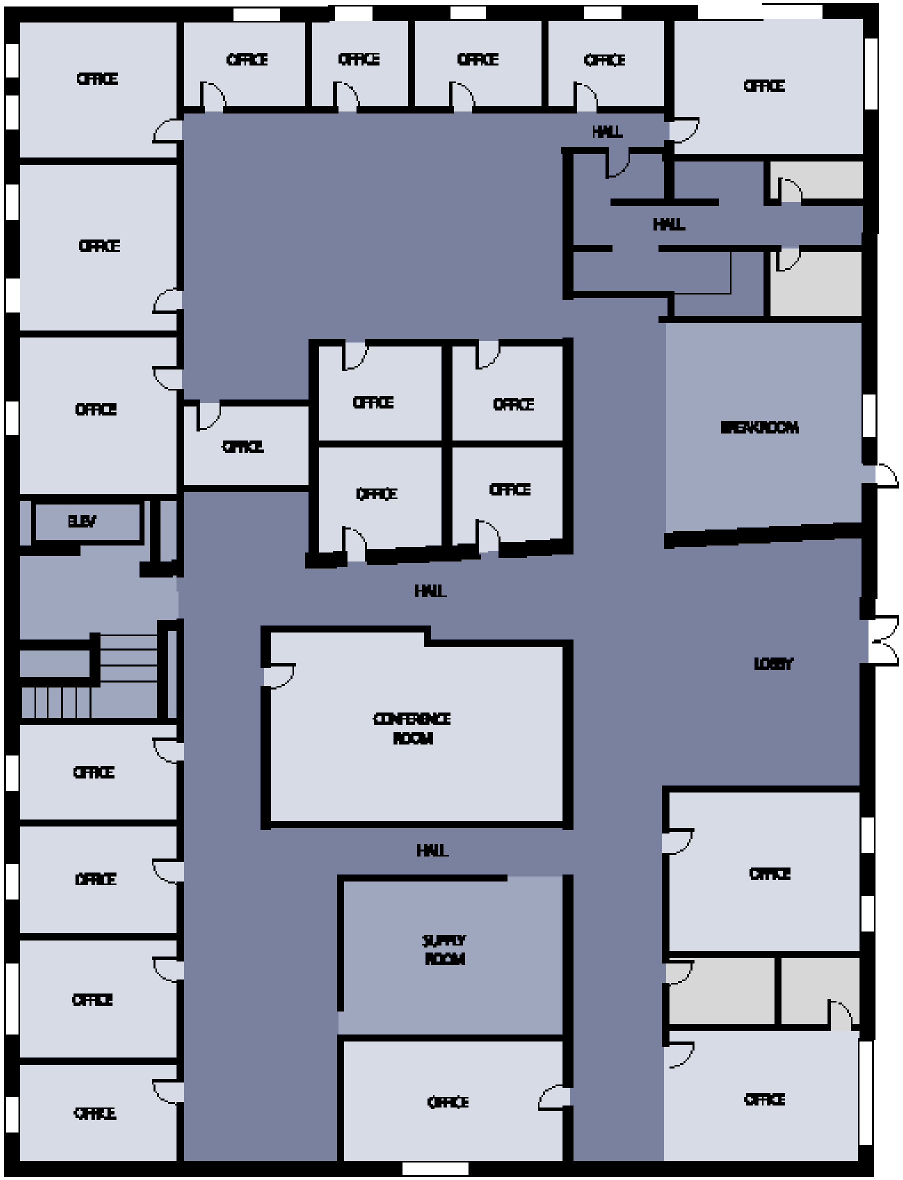
Known: SO=PH; ESO = NP

Which SO-fragments can be evaluated in polynomial time?

Which SO-fragments express regular languages on strings ?

More modestly: What about prefix classes?

A “simple” Facility Placement Problem



Every room should be equipped with a computer.

If a printer is not present in a room, then one should be available in an adjacent room.

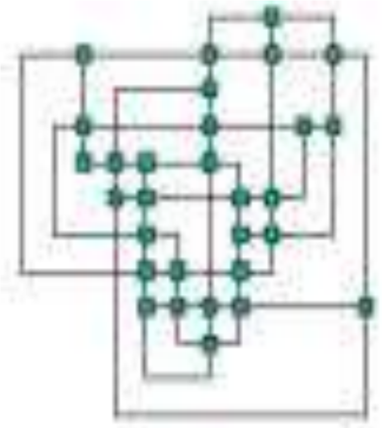
No room with a printer should be a meeting room.

Every room is at most 5 rooms distant from a meeting room.

[…]

Simplest Form

Given an office layout as a graph, decide whether the facility placement constraints are satisfiable.



$\exists P \exists M \ldots \; \forall x \exists y \; ((P(x) \vee E(x,y) \;\&\; P(y)) \;\&\; \ldots$

Observe that this is an E_1^*ae formula

This leads to the questions:

Are formulas of the type E_1^*ae or even E*ae polynomially verifiable over graphs?

What about other fragments of ESO or SO?

Simplest Form

This motivates the following question:

Can formulas in classes such as $E_2(ae_2)$ or even ESO(e*ae*) be evaluated in polynomial time over strings ?

More generally:

Which ESO-fragments admit polynomial-time model checking over strings ?

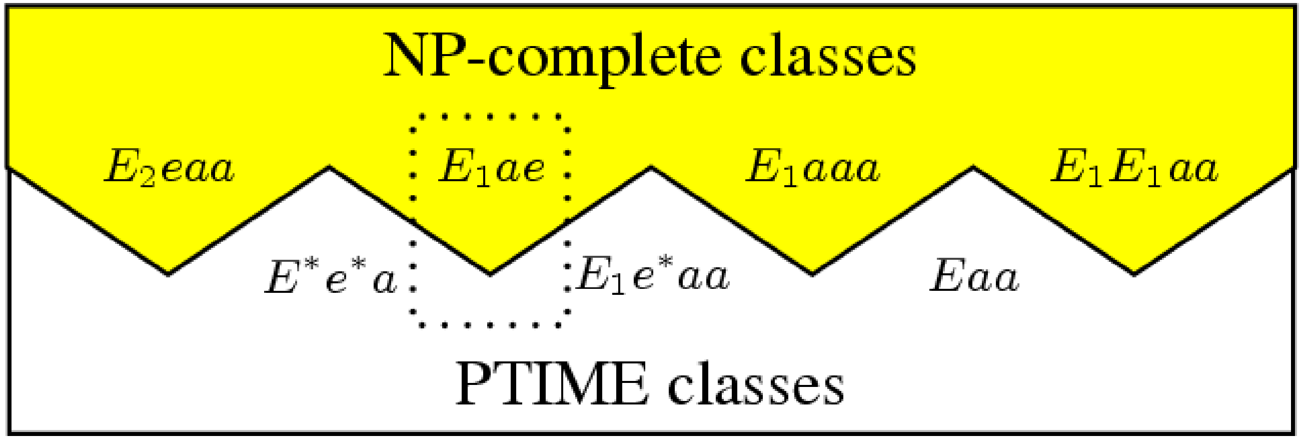
A similar, even more important question can be asked for graphs and general finite structures:

Which ESO-fragments admit polynomial-time model checking over graphs or arbitrary finite structures?

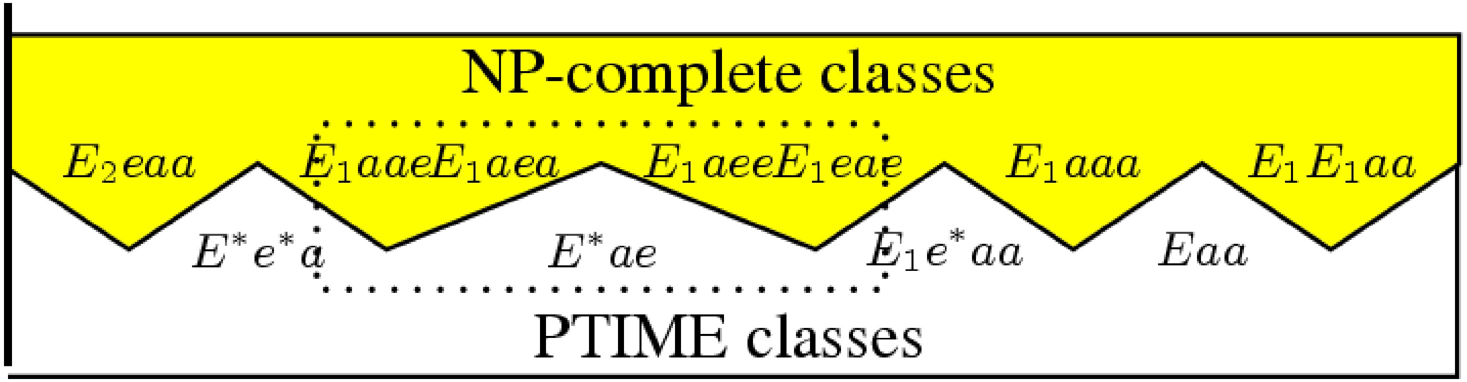
Complexity of ESO Prefix Classes

[G.,Kolaitis, Schwentick 2000]

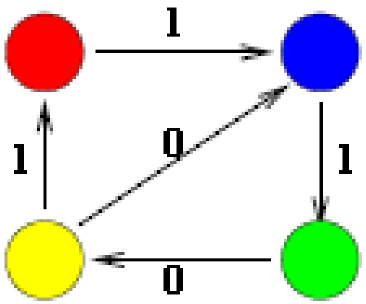
Directed graphs (or undirected graphs with self-loops):



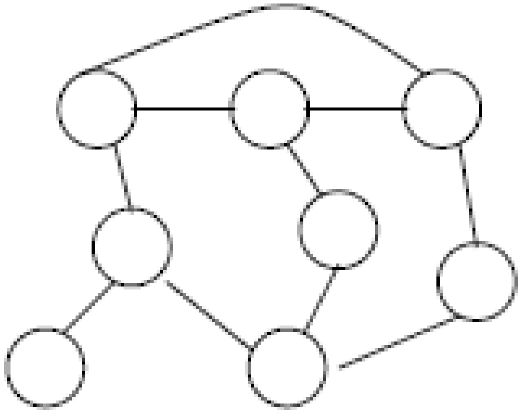
Undirected graphs w/o self-loops:



The Saturation Problem

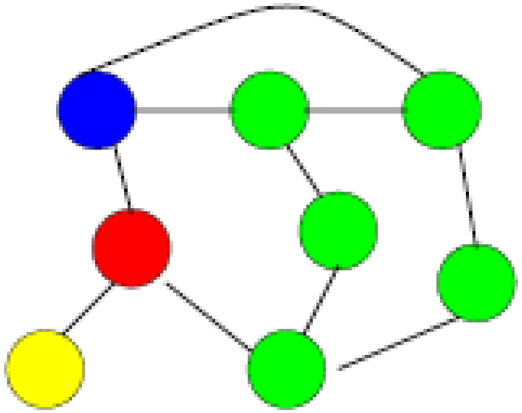


Pattern graph P1



Graph G

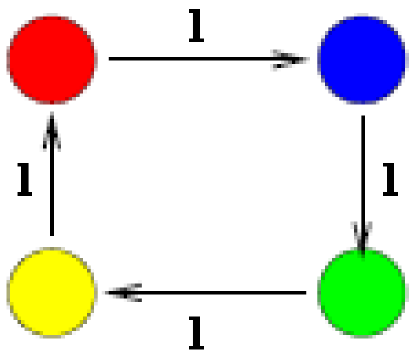
Saturation of G via P1:



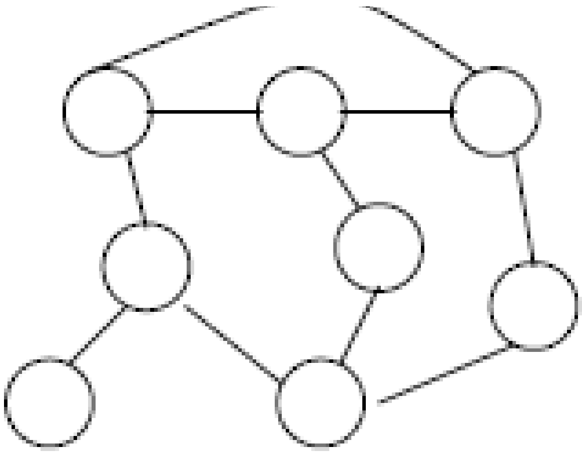
In PTIME!

The Saturation Problem

Relating $E_1^* ae$ to the Saturation Problem



Pattern graph P2

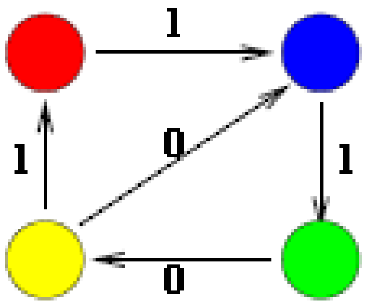


Graph G

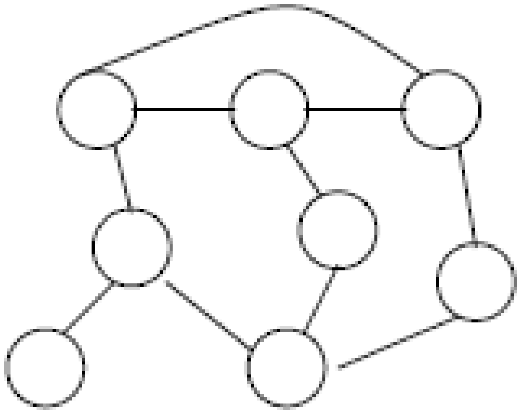
Saturation of G via P2 impossible!

No cycle of length 0 (mod 4) in G.

The Saturation Problem

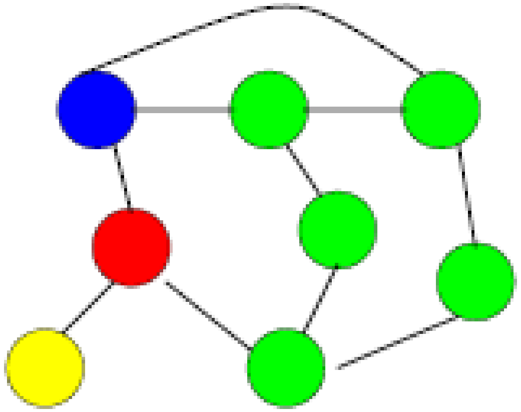


Pattern graph P1



Graph G

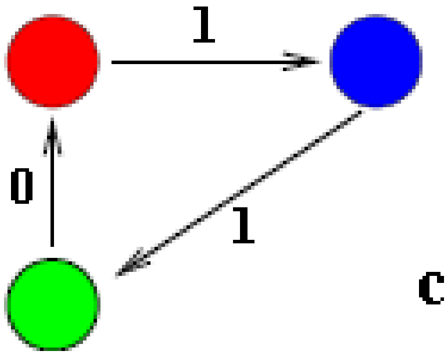
Saturation of G via P1:



The Saturation Problem

Relating E_1^*ae to the Saturation Problem

$$\begin{aligned}\exists \quad & P_1, P_2 \forall x \exists y \\ & [(E(x,y) \wedge P_1(x) \wedge P_2(x) \wedge P_1(y) \wedge \neg P_2(y)) \vee \\ & (E(x,y) \wedge P_1(x) \wedge \neg P_2(x) \wedge \neg P_1(y) \wedge \neg P_2(y)) \vee \\ & (\neg E(x,y) \wedge \neg P_1(x) \wedge \neg P_2(x) \wedge P_1(y) \wedge P_2(y))]\end{aligned}$$



corresponding pattern graph

Use of Tree Decompositions

1. **Prove Tractability of bounded-width instances**
   a) Genuine tractability: $O(n^{f(w)})$-bounds

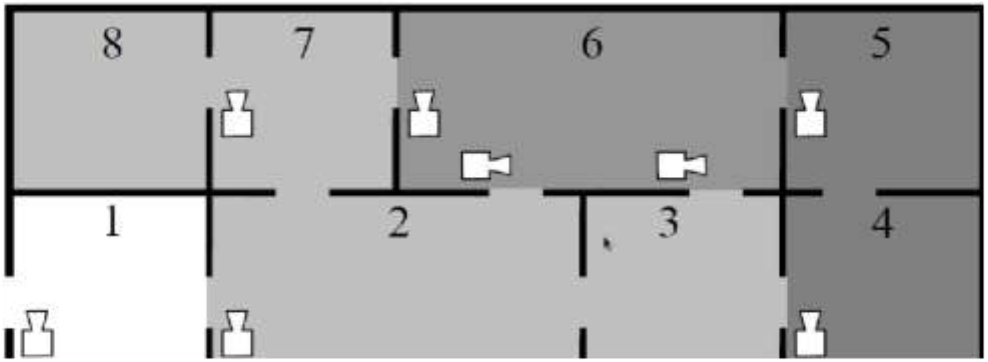
   b) Fixed-Parameter tractability: $f(w)*O(n^k)$

2. **Tool for proving general tractability**
   a) Prove tractability for both large & small width

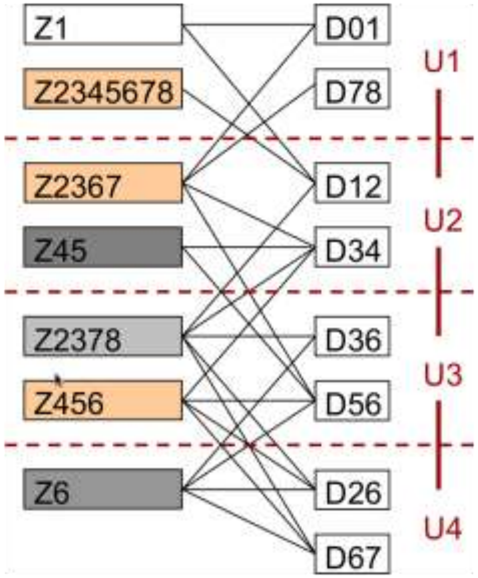
   b) Prove all yes-instances to have small width

Partner Units Scenario

- Track People in Buildings
- Sensors on Doors, Rooms Grouped into Zones



Partner Units Solution



- Assigning Sensors and Zones to Control Units
- Respect Adjacency Constraints

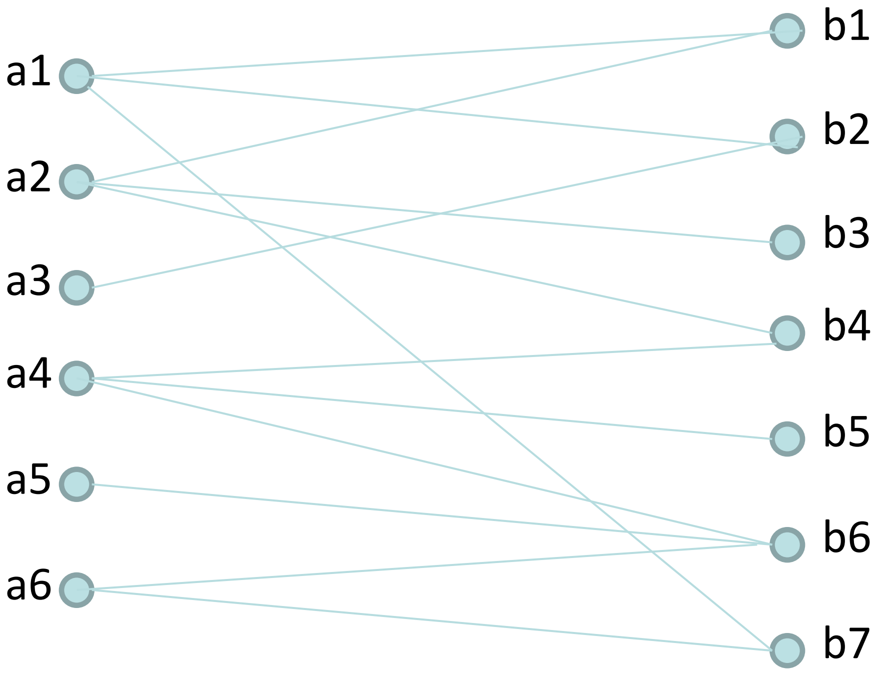
The Partner-Unit Problem

Bipartite graph G=(V,E) V=Va∪Vb;

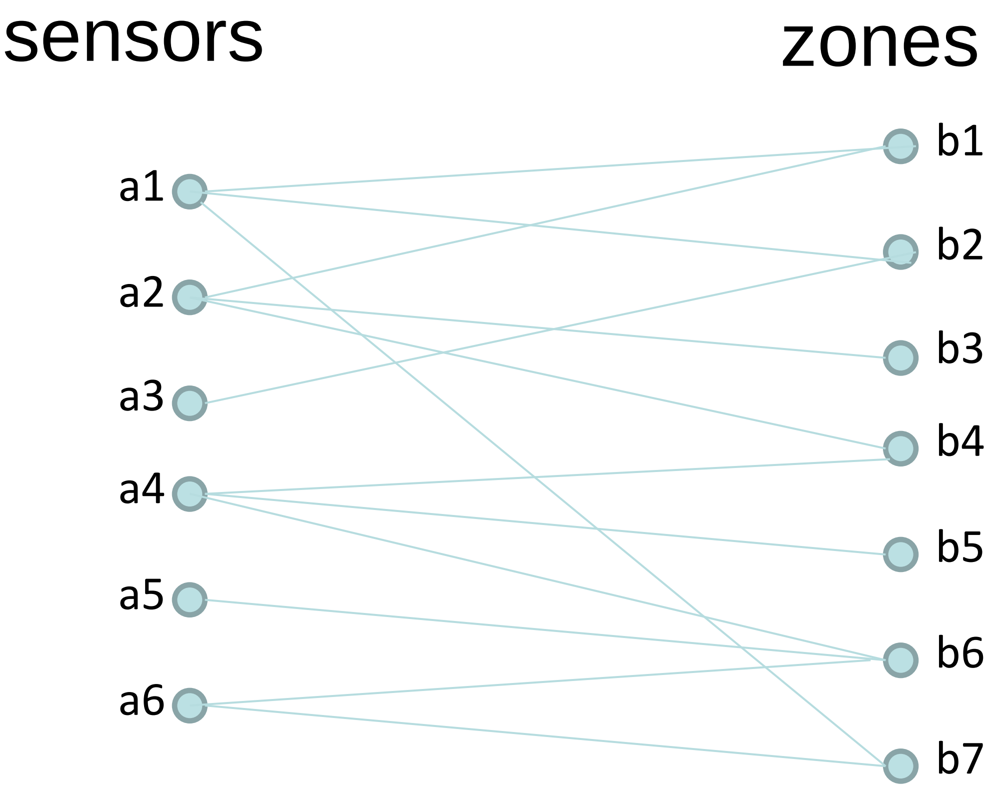
Va= {a1,…,ar},

Vb={b1,…,bs},

E: edges btw. Va and Vb



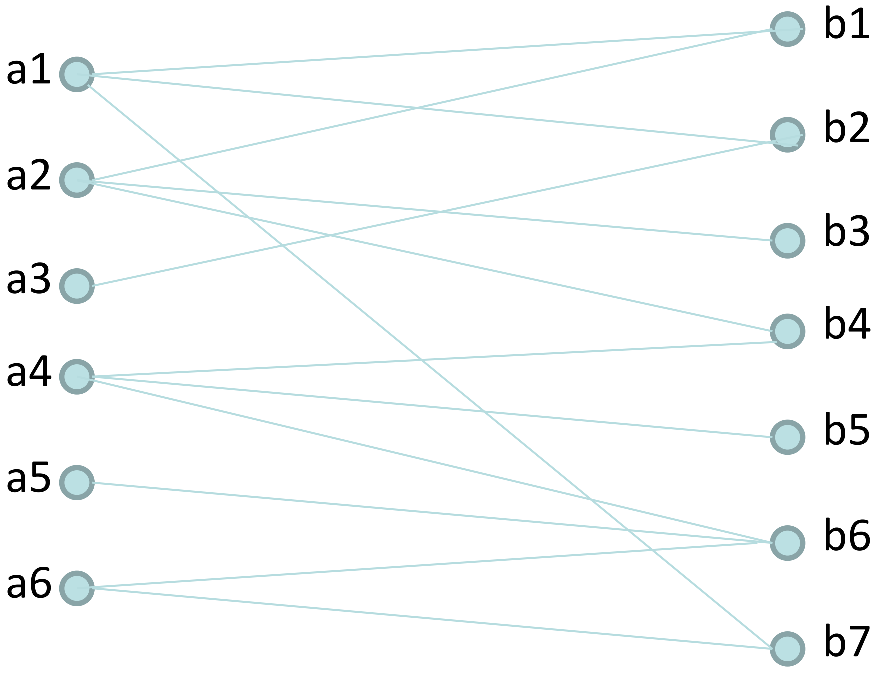
The Partner-Unit Problem



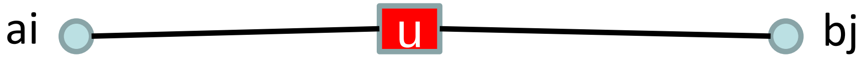
Replace connections by connections to units

The Partner-Unit Problem

Bipartite graph G=(V,E) V=Va∪Vb; Va= {a1,...,ar}, Vb={b1,...,bs}, E: edges btw. Va and Vb



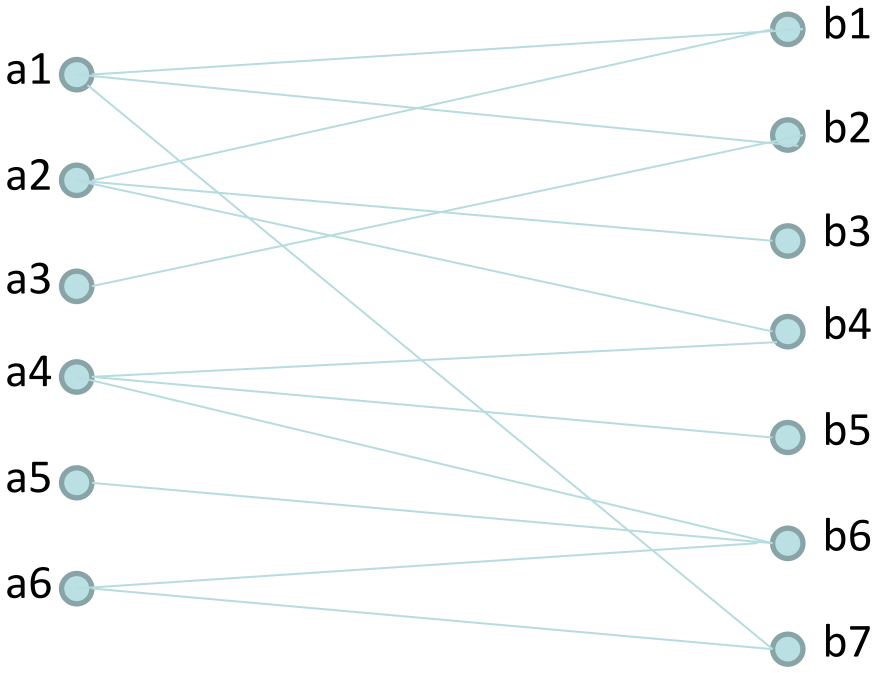
Replace connections by connections to units



OR

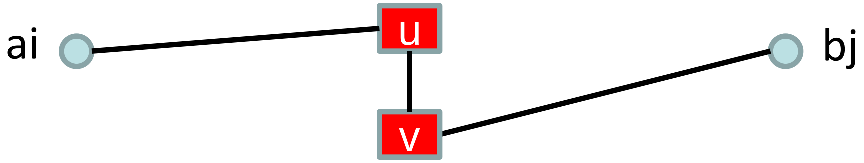
The Partner-Unit Problem

Bipartite graph G=(V,E) V=Va∪Vb; Va= {a1,...,ar}, Vb={b1,...,bs}, E: edges btw. Va and Vb

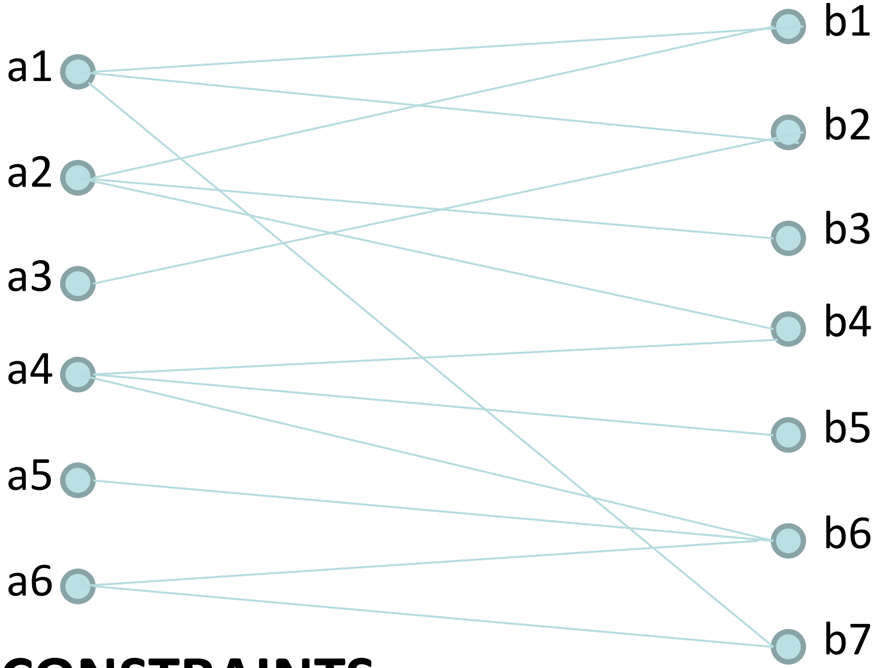


Replace connections by connections to units

OR

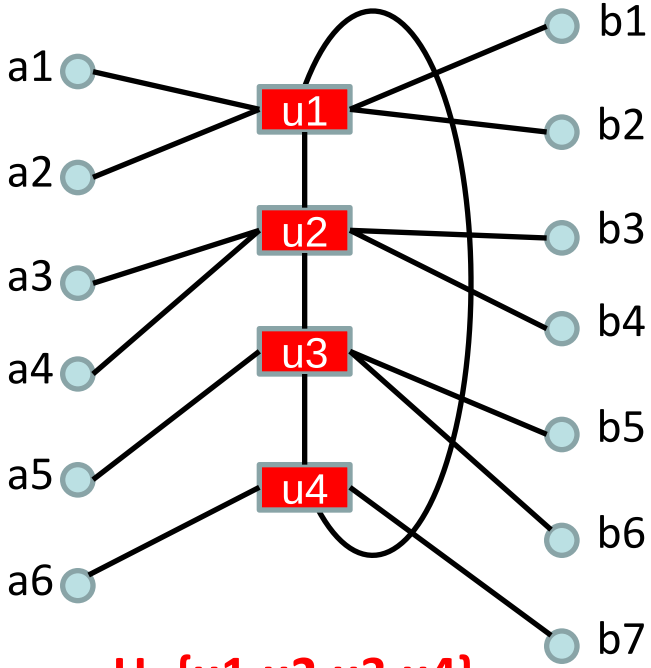


The Partner-Unit Problem



G

G*



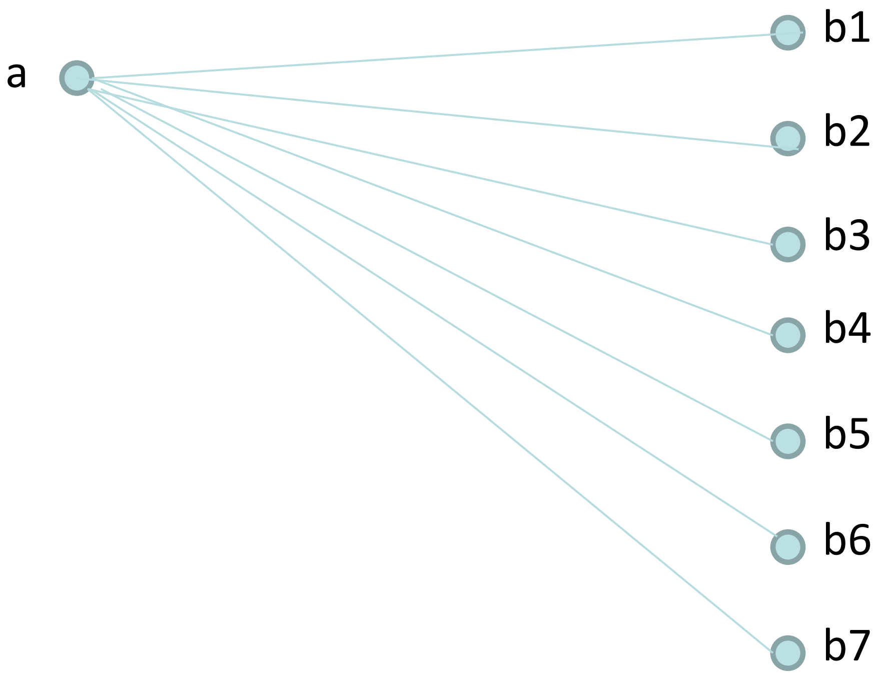
U={u1,u2,u3,u4}

CONSTRAINTS:

- **Each ai or bi is connected to exactly 1 unit.**
- **Each unit connected to:**
  - **at most 2 other units,**
  - **at most 2 elements from Va,**
  - **at most 2 elements from Vb,**
- **If ai connected to bj in G, then dist(ai,bi)≤3 in G***

A No-Instance of Partner-Unit

Assume one node a is connected to 7 nodes b1,...,b7 in G. Then instance G is unsolvable.



Thus, no vertex can have more than 6 neighbours in G.

The PU Problem(s)

- **PU DECISION PROBLEM (PUDP):**

  Given G, is there a G* satisfying the constraints?

  (Number of units irrelevant.)

- **PU SEARCH PROBLEM (PUSP)**

  Given G, find a suitable G* whenever possible.

- **PU OPTIMIZATION PROBLEM (PUOP)**

  Given G, find a suitable G* with minimum number of units |U| (whenever possible).

PUDP

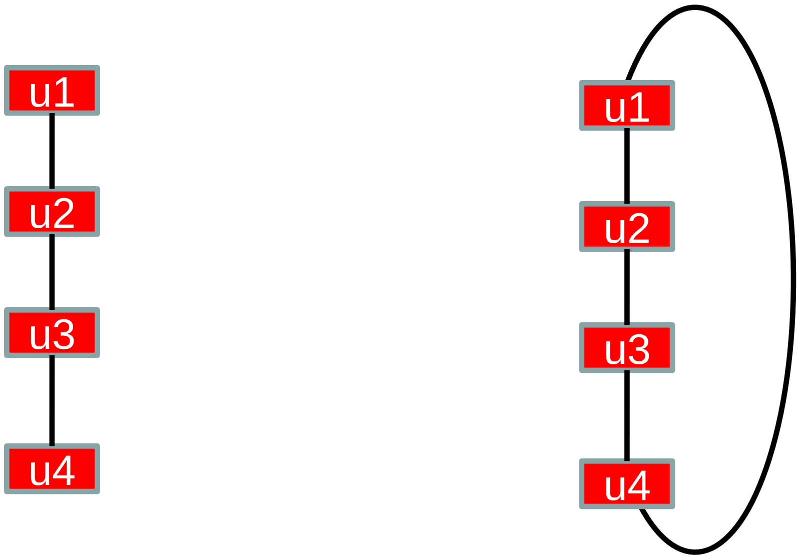
ASSUMPTION: G is connected.

Note: This assumption can be made wlog, because the PUDP can be otherwise decomposed into a conjunction of independent PUDPs, one for each component.

Lemma 1: If G is connected and solvable, then there exists a solution G* in which the unit-graph UG=G*[U] is connected.

Topology of the Unit-Graph

Lemma 2: If G is connected and solvable, then there exists a solution G* whose unit graph is a cycle.



Note: We still don’t know |U|, but we may just try all cycles of length max(|Va|,|Vb|)/2 to length |Va|+|Vb|.
There are only linearly many! (Guessable in logspace)

Result

Theorem:

Assume G is solvable through solution G* with |U|=n and having
unit function f . Then:

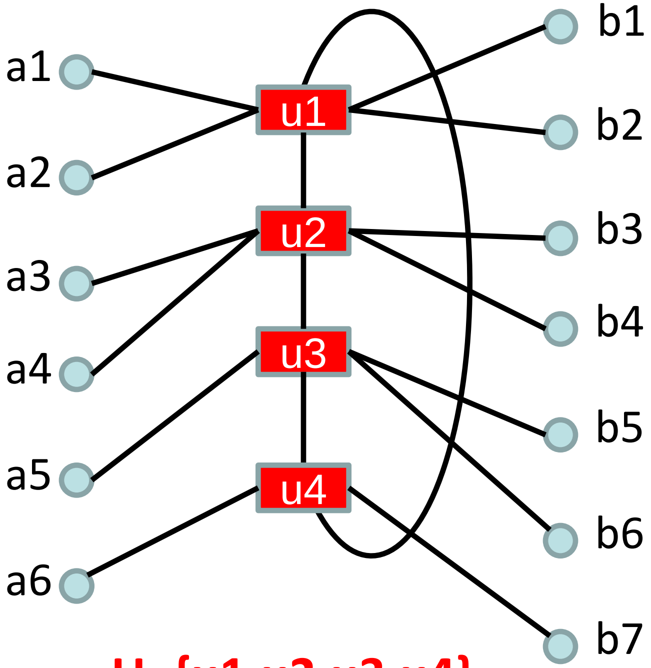
(1) $pw(G) \leq 11$

(2) $tw(G) \leq 5$

(3) There is a path decomposition T=(W,A)
that can be locally check to witnss PUDP solution G*

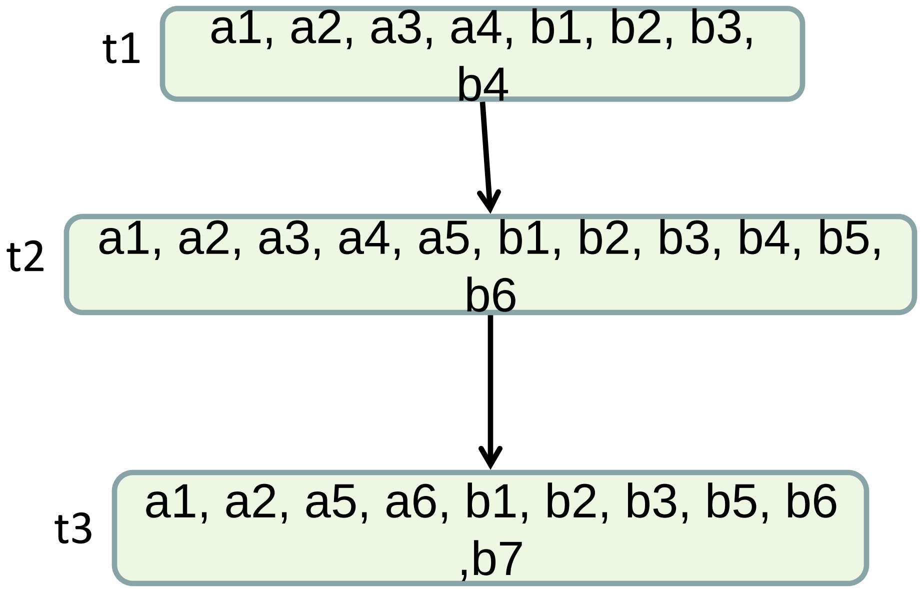
Example

G*



U={u1,u2,u3,u4}

T



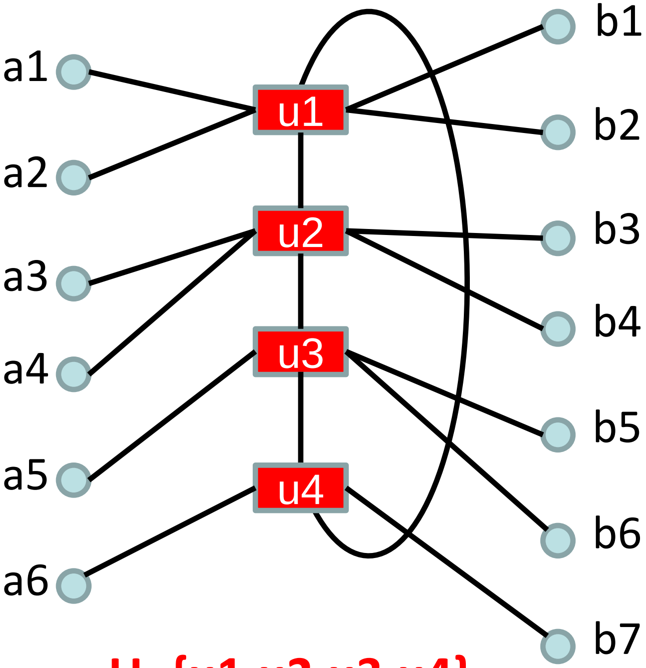
t1 a1, a2, a3, a4, b1, b2, b3, b4

t2 a1, a2, a3, a4, a5, b1, b2, b3, b4, b5, b6

t3 a1, a2, a5, a6, b1, b2, b3, b5, b6 ,b7

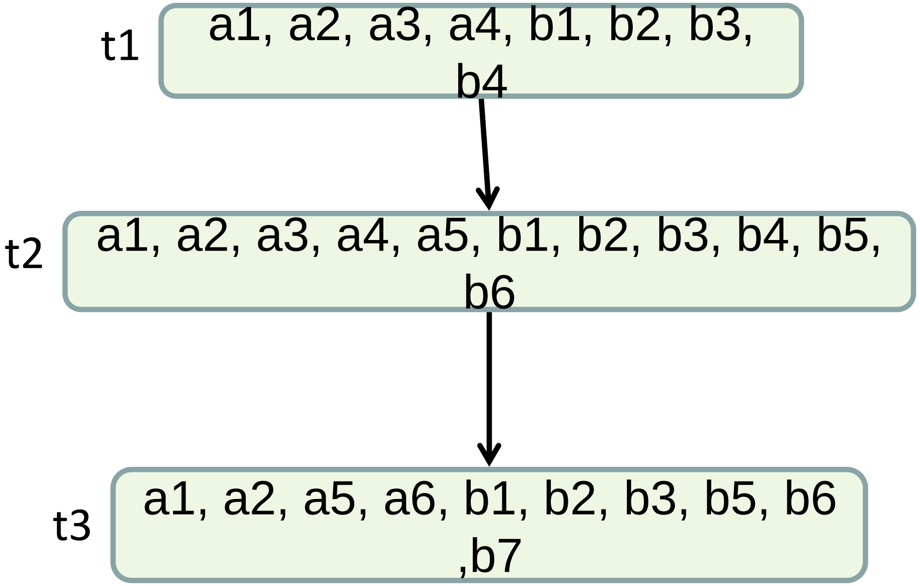
Example

G*



U={u1,u2,u3,u4}

T



t1 a1, a2, a3, a4, b1, b2, b3, b4

t2 a1, a2, a3, a4, a5, b1, b2, b3, b4, b5, b6

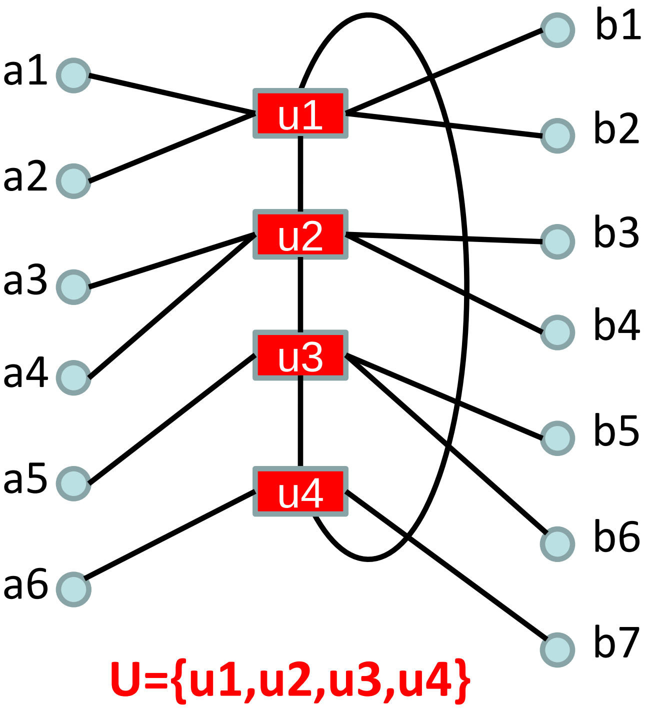
t3 a1, a2, a5, a6, b1, b2, b3, b5, b6 ,b7

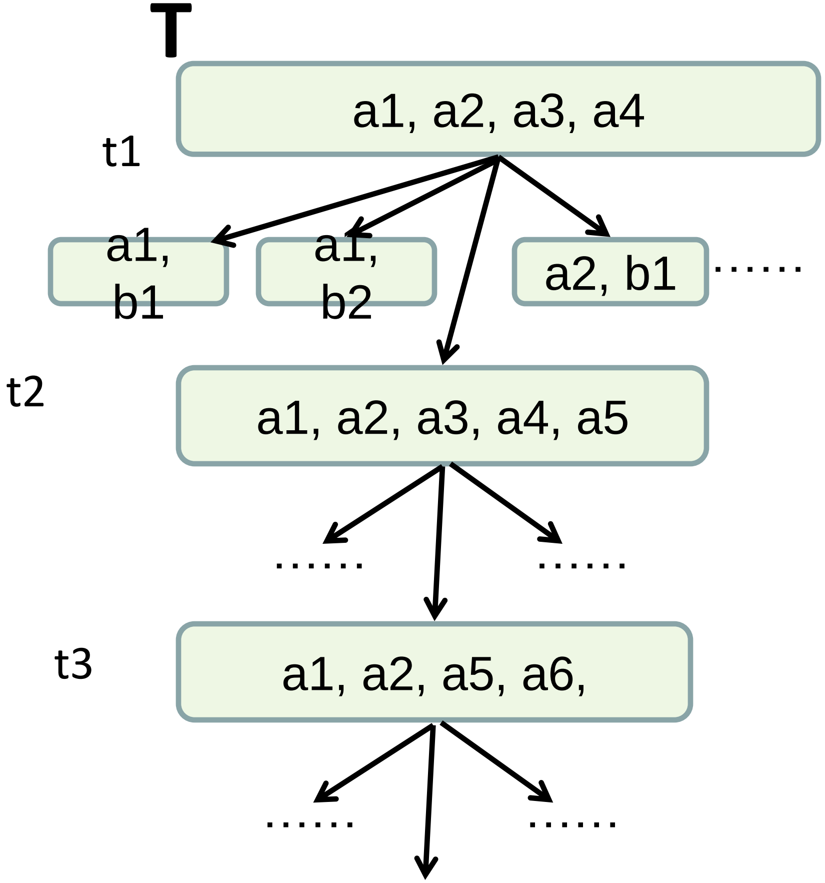
Note: We cannot do better, thus the bound 11 is actually tight!

Example

We now show (2)

Strip off the Vb-elements and put them into separate bags.

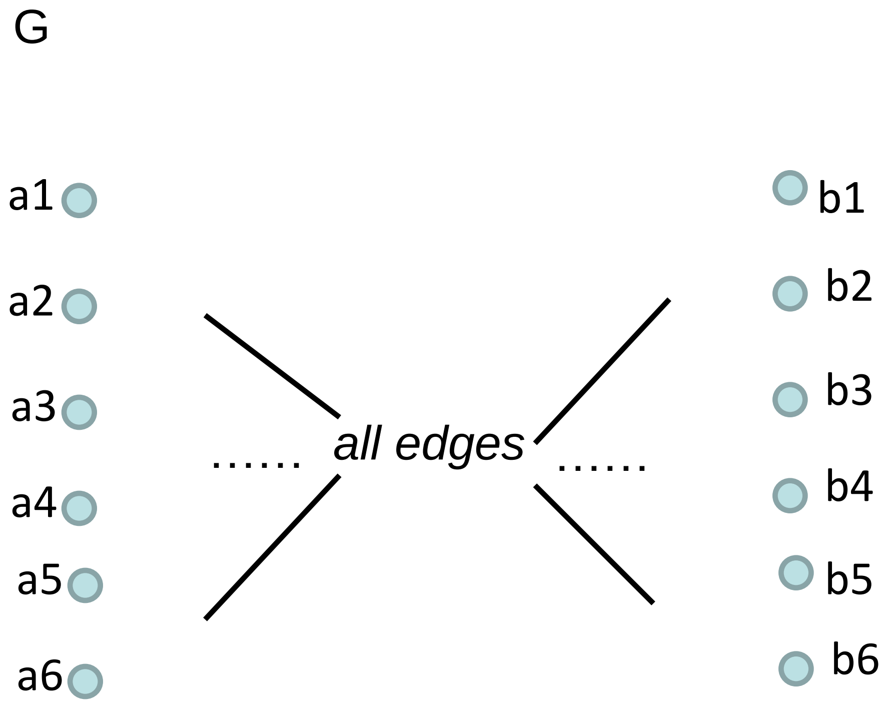




Note: Other examples show, we cannot do better, thus the bound 5 is actually tight

Example

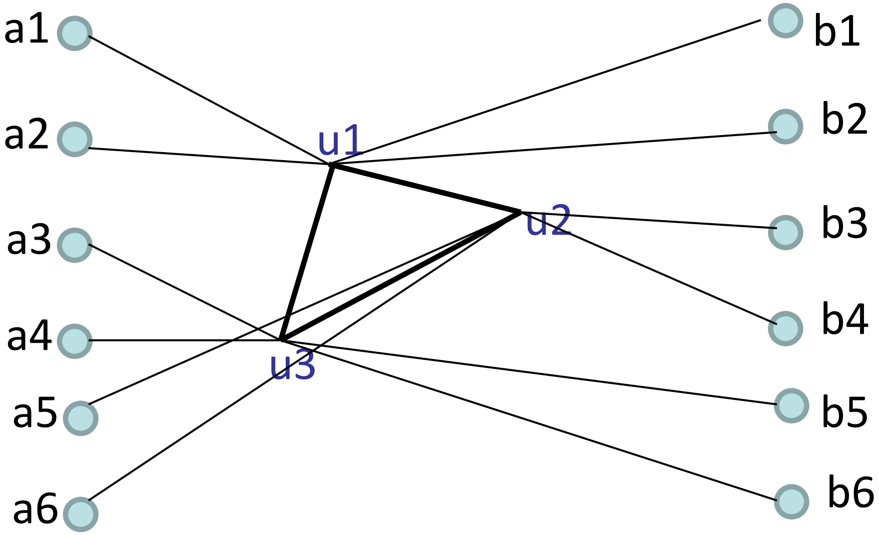
Example for lower bound 5



tw=5

Example for lower bound 5

… and this G is actually solvable:



Result

Theorem : PUDP is in polynomial time and is solvable by dynamic programming techniques.

QED

Partner Units Results

Name	Sensors	Zones	Edges	Cost	CSP	DECPUP
dbl-20	28	20	56	14	*	0.01
dbl-40	58	40	116	29	*	0.05
dbl-60	88	60	176	44	*	0.08
dblv-30	28	30	92	15	*	65.49
dblv-60	58	60	192	30	*	*
triple-30	40	30	78	20	*	0.50
triple-34	40	34	93	/	*	*
grid-90	50	68	97	34	*	0.03

Case N>2

For constant N totally open. Could well be NP-hard.
In fact, Unit Graph does not need to have bounded treewidth!

If N is not-constant, then NP-complete:

For Siemens, it seems that very small values of N are relevant.

Outline of PART II

Beyond Tree Decompositions

Applications to Databases and CSPs

Structural and Consistency Properties

Outline of PART II

Beyond Tree Decompositions

Applications to Databases and CSPs

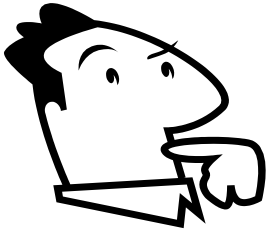
Structural and Consistency Properties

Beyond Treewidth

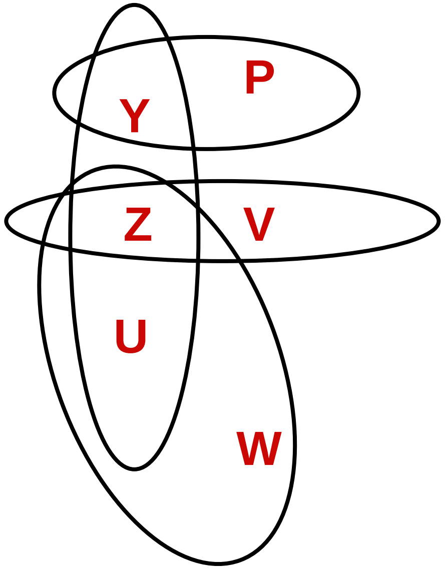
- Treewidth is currently the most successful measure of graph cyclicity. It subsumes most other methods.
- However, there are “simple” graphs that are heavily cyclic. For example, a clique.

Beyond Treewidth

- Treewidth is currently the most successful measure of graph cyclicity. It subsumes most other methods.
- However, there are “simple” graphs that are heavily cyclic. For example, a clique.



There are also problems whose structure is better described by **hypergraphs** rather than by graphs…



Database queries

- Database schema (scopes):
  - *Enrolled (Pers#, Course, Reg-Date)*
  - *Teaches (Pers#, Course, Assigned)*
  - *Parent (Pers1, Pers2)*

- Is there any teacher having a child enrolled in her course?

ans ← *Enrolled(S,C,R)* ∧ *Teaches(P,C,A)* ∧ *Parent(P,S)*

Database queries

Enrolled

John	Algebra	2003
Anita	Logic	2003
Sara	DB	2002
Luisa	DB	2003
………	…..	…….

Teaches

Nicola	Algebra	March
Georg	Logic	May
Frank	DB	June
Mimmo	DB	May
………	…..	…….

Parent

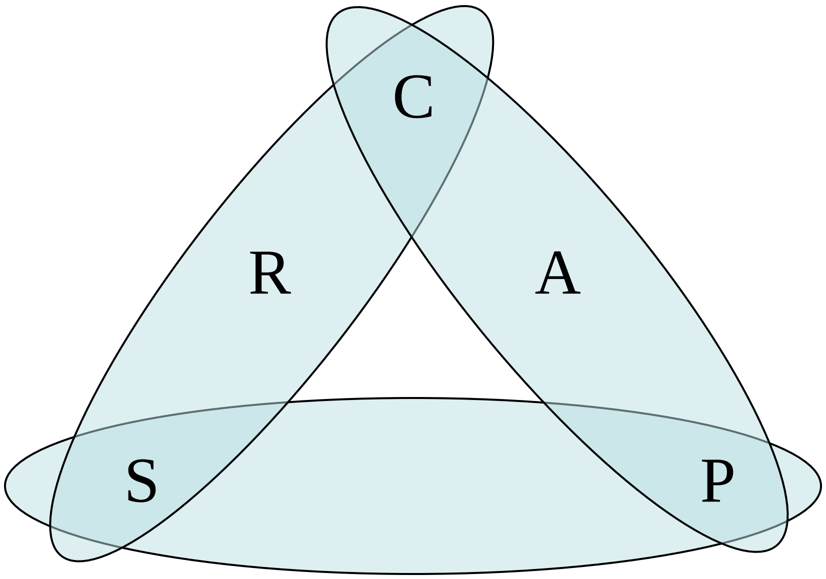
Mimmo	Luisa
Georg	Anita
Frank	Sara
………	…..

QUERY: Is there any teacher having a child enrolled in her course?

$$ans \leftarrow Enrolled(S,C,R) \wedge Teaches(P,C,A) \wedge Parent(P,S)$$

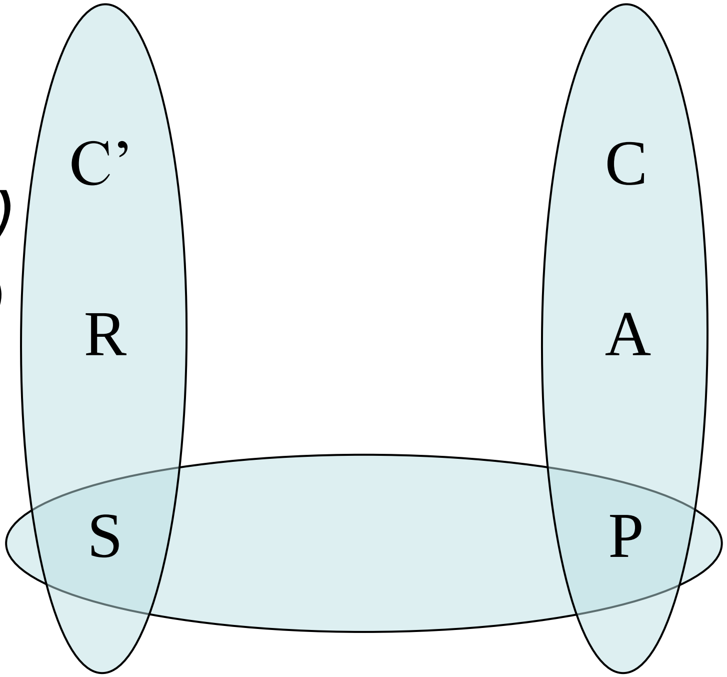
Queries and Hypergraphs

$Ans \Leftarrow Enrolled(S,C,R) \wedge Teaches(P,C,A) \wedge Parent(P,S)$



Queries and Hypergraphs (2)

- Database schema (scopes):
  - *Enrolled (Pers#, Course, Reg-Date)*
  - *Teaches (Pers#, Course, Assigned)*
  - *Parent (Pers1, Pers2)*

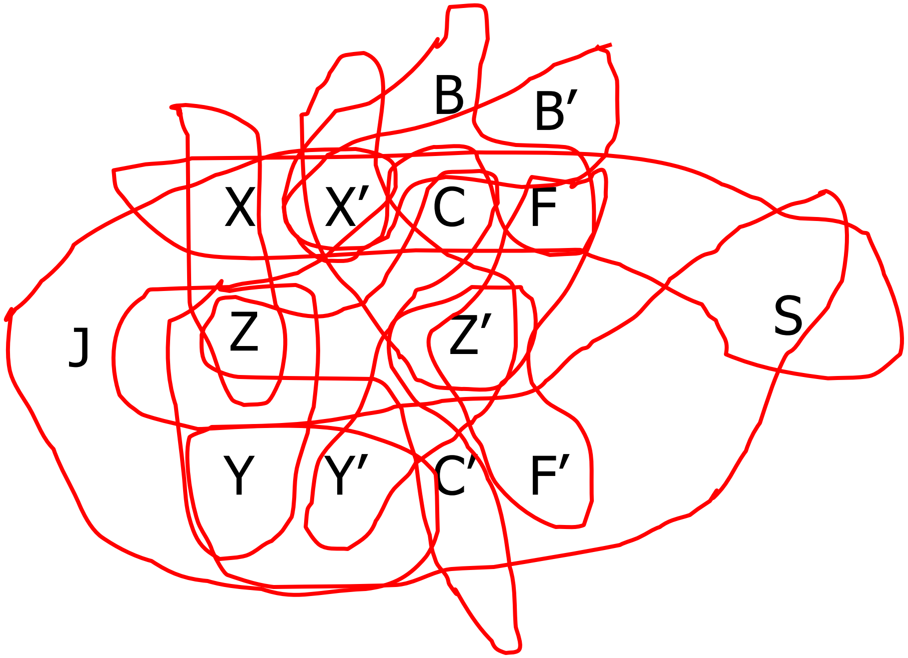


- Is there any teacher whose child attend some course?

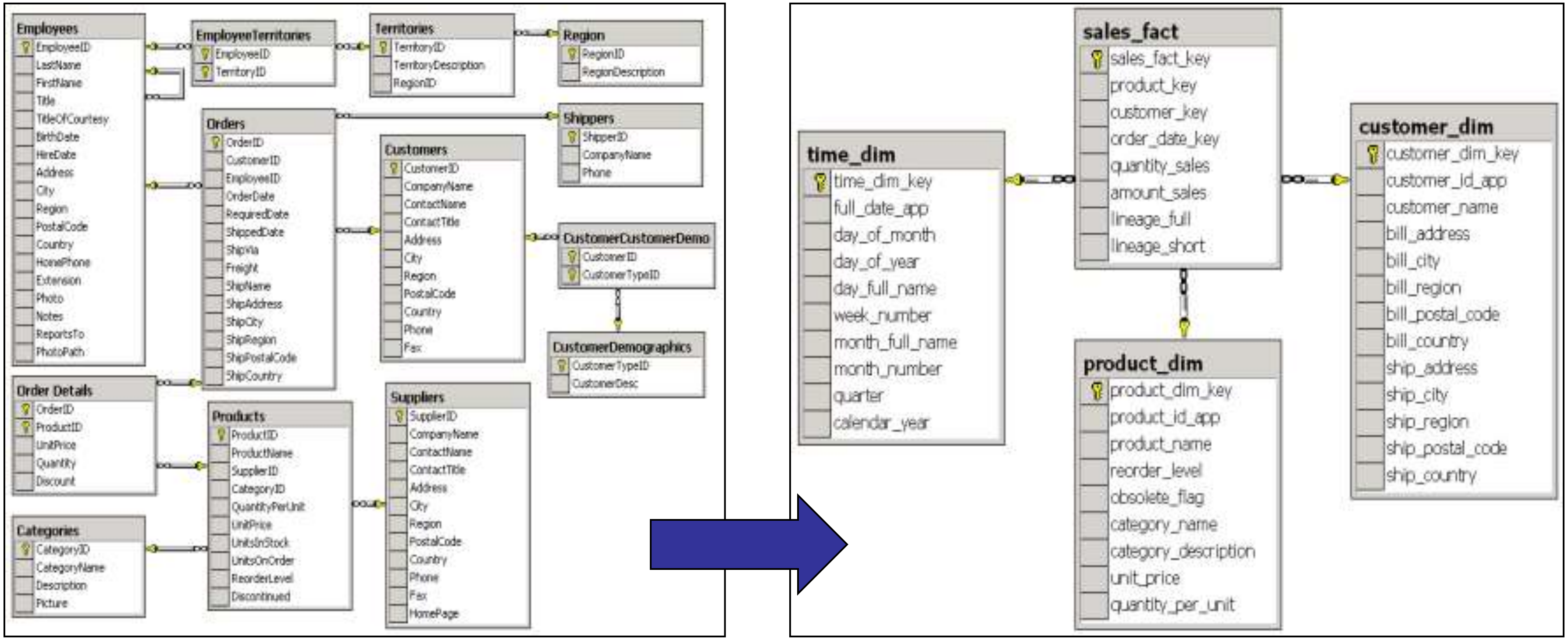
  Ans 🡐 *Enrolled(S,C',R)* ∧ *Teaches(P,C,A)* ∧ *Parent(P,S)*

A more intricate query

$$ans \leftarrow a(S,X,X',C,F) \wedge b(S,Y,Y',C',F') \wedge c(C,C',Z) \wedge d(X,Z) \wedge$$
$$e(Y,Z) \wedge f(F,F',Z') \wedge g(X',Z') \wedge h(Y',Z') \wedge$$
$$j(J,X,Y,X',Y') \wedge p(B,X',F) \wedge q(B',X',F)$$



Populating datawarehouses



Employees
EmployeeID
LastName
FirstName
Title
TitleOfCourtesy
BirthDate
HireDate
Address
City
Region
PostalCode
Country
HomePhone
Extension
Photo
Notes
ReportsTo
PhotoPath

EmployeeTerritories
EmployeeID
TerritoryID

Territories
TerritoryID
TerritoryDescription
RegionID

Region
RegionID
RegionDescription

Orders
OrderID
CustomerID
EmployeeID
OrderDate
RequiredDate
ShippedDate
ShipVia
Freight
ShipName
ShipAddress
ShipCity
ShipRegion
ShipPostalCode
ShipCountry

Shippers
ShipperID
CompanyName
Phone

Customers
CustomerID
CompanyName
ContactName
ContactTitle
Address
City
Region
PostalCode
Country
Phone
Fax

CustomerCustomerDemo
CustomerID
CustomerTypeID

CustomerDemographics
CustomerTypeID
CustomerDesc

Order Details
OrderID
ProductID
UnitPrice
Quantity
Discount

Products
ProductID
ProductName
SupplierID
CategoryID
QuantityPerUnit
UnitPrice
UnitsInStock
UnitsOnOrder
ReorderLevel
Discontinued

Suppliers
SupplierID
CompanyName
ContactName
ContactTitle
Address
City
Region
PostalCode
Country
Phone
Fax
HomePage

Categories
CategoryID
CategoryName
Description
Picture

sales_fact
sales_fact_key
product_key
customer_key
order_date_key
quantity_sales
amount_sales
lineage_full
lineage_short

time_dim
time_dim_key
full_date_app
day_of_month
day_of_year
day_full_name
week_number
month_full_name
month_number
quarter
calendar_year

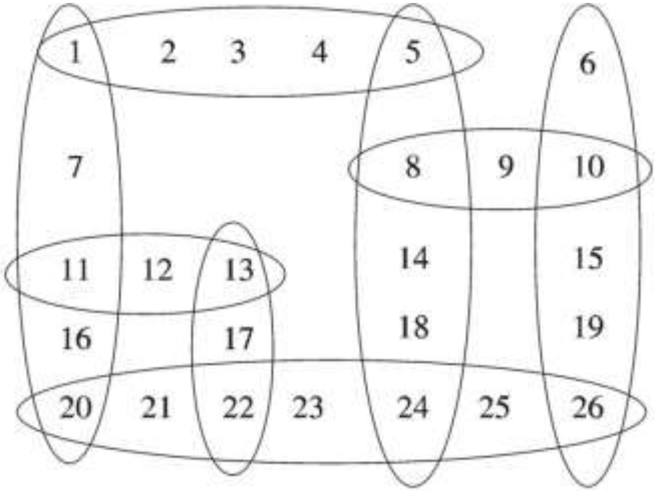
customer_dim
customer_dim_key
customer_id_app
customer_name
bill_address
bill_city
bill_region
bill_postal_code
bill_country
ship_address
ship_city
ship_region
ship_postal_code
ship_country

product_dim
product_dim_key
product_id_app
product_name
reorder_level
obsolete_flag
category_name
category_description
unit_price
quantity_per_unit

Constraint Satisfaction Problems

Crossword puzzle

1	2	3	4	5	■	6
7	■	■	■	8	9	10
11	12	13	■	14	■	15
16	■	17	■	18	■	19
20	21	22	23	24	25	26



1h:

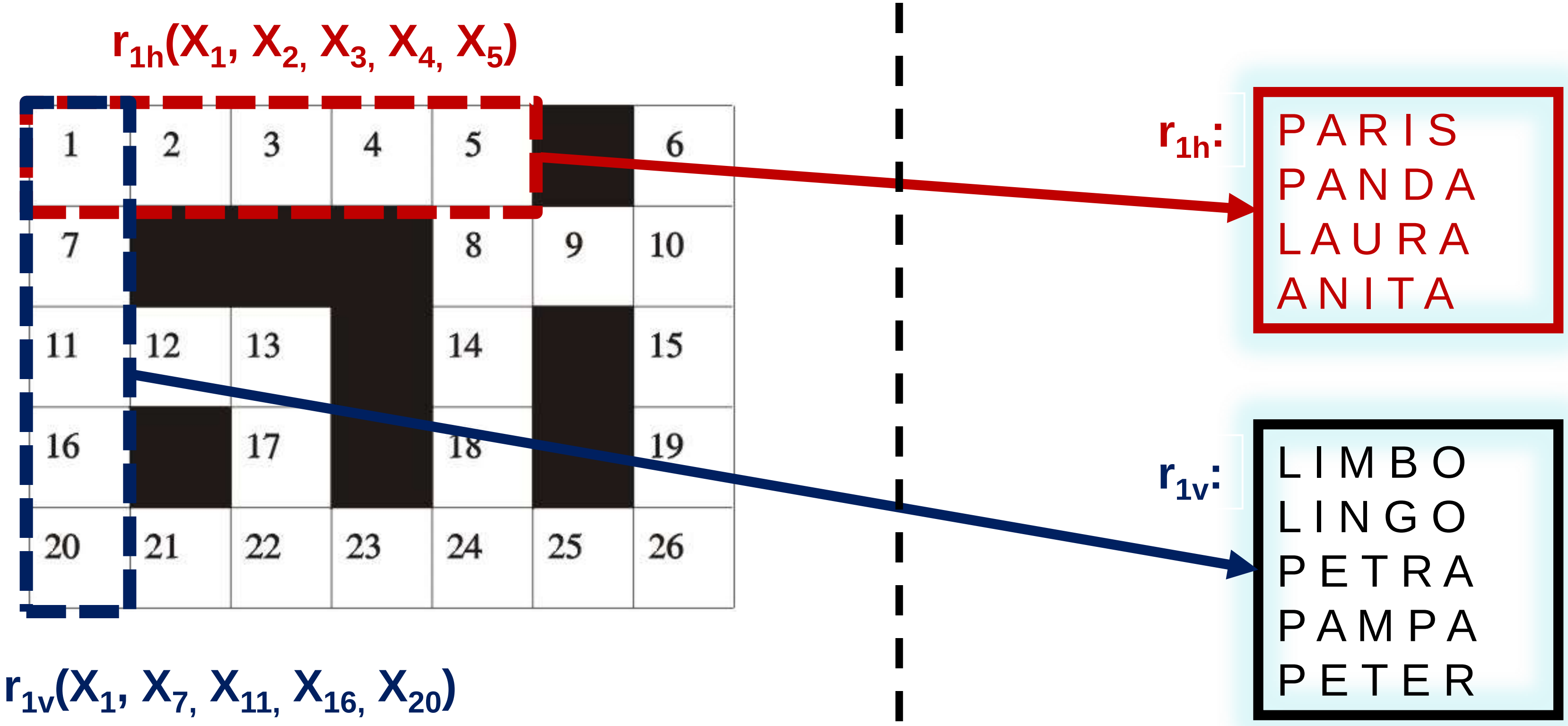
PARIS
PANDA
LAURA
ANITA

1v:

LIMBO
LINGO
PETRA
PAMPA
PETER

and so on

Constraint Satisfaction Problems



$r_{1h}(X_1, X_2, X_3, X_4, X_5)$

$r_{1v}(X_1, X_7, X_{11}, X_{16}, X_{20})$

r_{1h}:

PARIS
PANDA
LAURA
ANITA

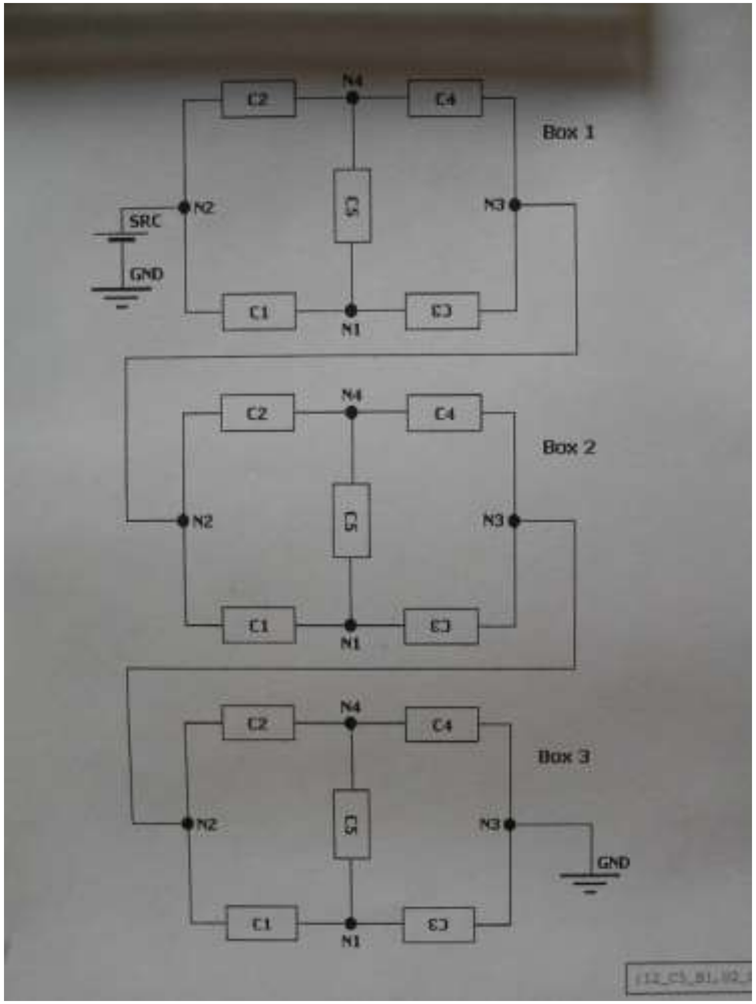
r_{1v}:

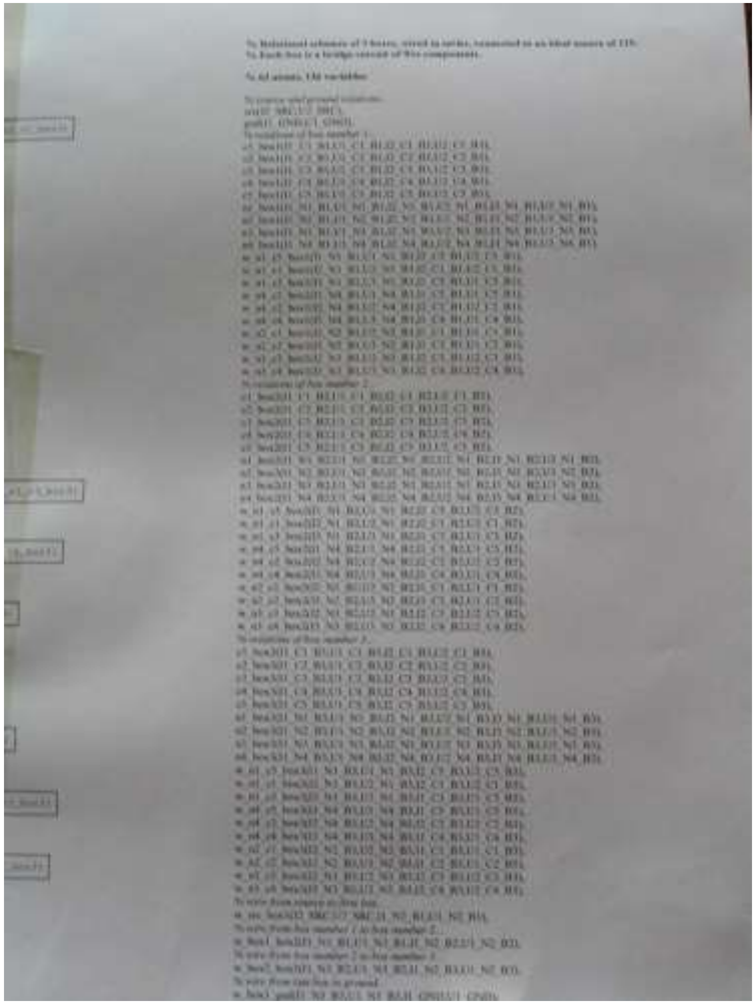
LIMBO
LINGO
PETRA
PAMPA
PETER

- Set of variables $\{X_1,\ldots,X_{26}\}$
- Set of constraint scopes

- Set of constraint relations

Problems on Electric Circuits





% [illegible] of 3 boxes, wired in series, [illegible]

% Each box is a bridge circuit of five components.

% [illegible]

[illegible]

A problem from Nasa

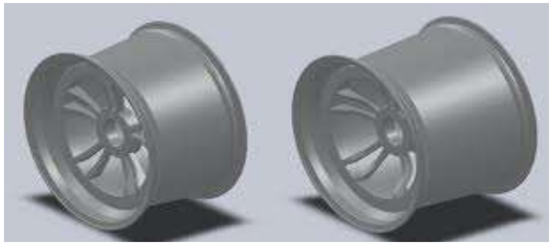
Part of relations for the Nasa problem

```
...

cid_260(Vid_49, Vid_366, Vid_224),
cid_261(Vid_100, Vid_391, Vid_392),
cid_262(Vid_273, Vid_393, Vid_246),
cid_263(Vid_329, Vid_394, Vid_249),
cid_264(Vid_133, Vid_360, Vid_356),
cid_265(Vid_314, Vid_348, Vid_395),
cid_266(Vid_67, Vid_352, Vid_396),
cid_267(Vid_182, Vid_364, Vid_397),
cid_268(Vid_313, Vid_349, Vid_398),
cid_269(Vid_339, Vid_348, Vid_399),
cid_270(Vid_98, Vid_366, Vid_400),
cid_271(Vid_161, Vid_364, Vid_401),
cid_272(Vid_131, Vid_353, Vid_234),
cid_273(Vid_126, Vid_402, Vid_245),
cid_274(Vid_146, Vid_252, Vid_228),
cid_275(Vid_330, Vid_360, Vid_361),

...
```

- 680 constraints
- 579 variables

Configuration problems (Renault example)

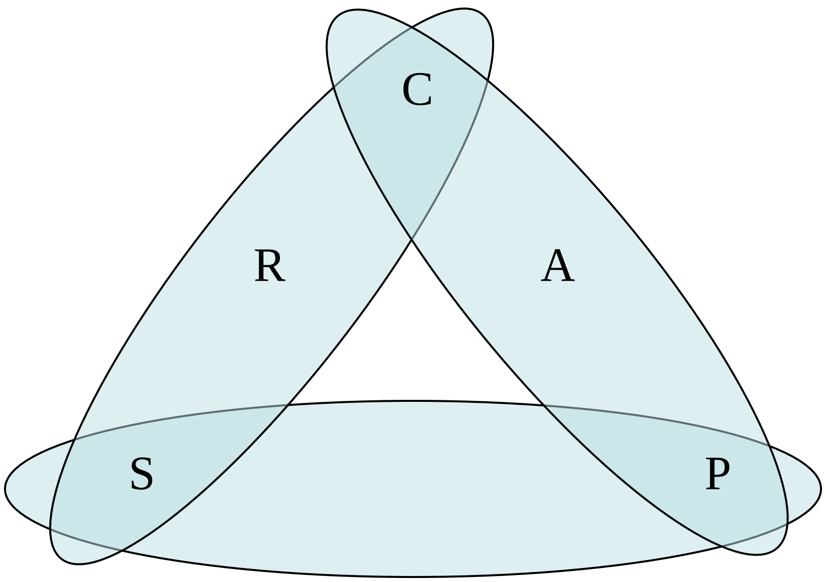
- Renault Megane configuration
  [Amilhastre, Fargier, Marquis AIJ, 2002]
  Used in CSP competitions and as a benchmark problem
- Variables encode type of engine, country, options like air cooling, etc.
- 99 variables with domains ranging from 2 to 43.
- 858 constraints, which can be compressed to 113 constraints.
- The maximum arity is 10 (hyperedge cardinality/size of constraint scopes)
- Represented as extensive relations, the 113 constraints comprise about 200 000 tuples
- 2.84 × 10^{12} solutions.



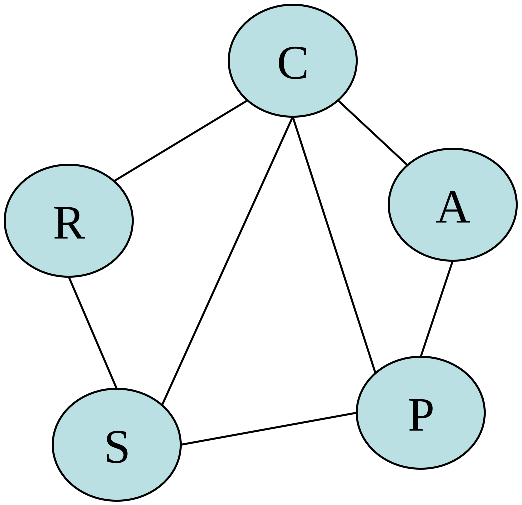
Further examples…

In the third part

Representing Hypergraphs via Graphs

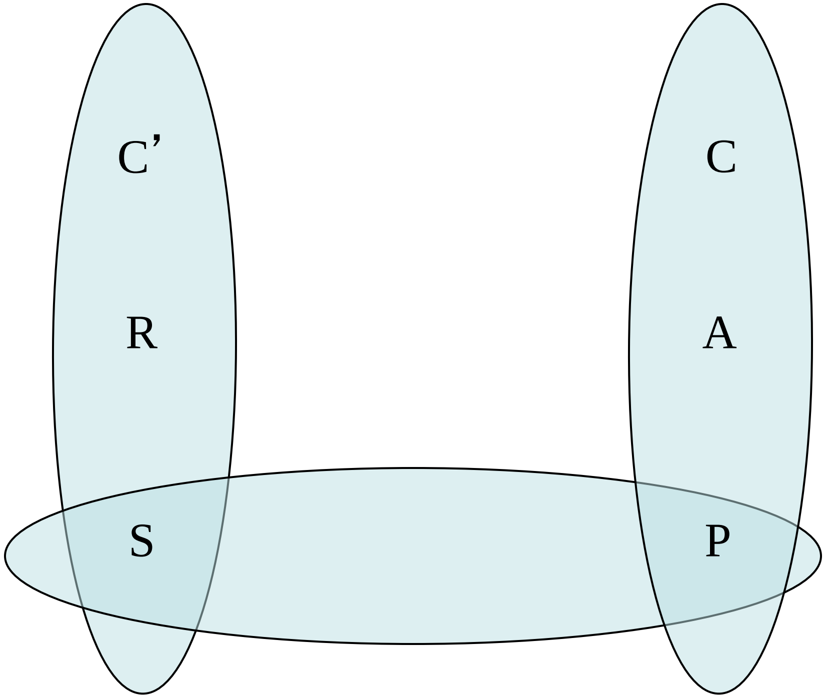


Hypergraph *H(Q)*

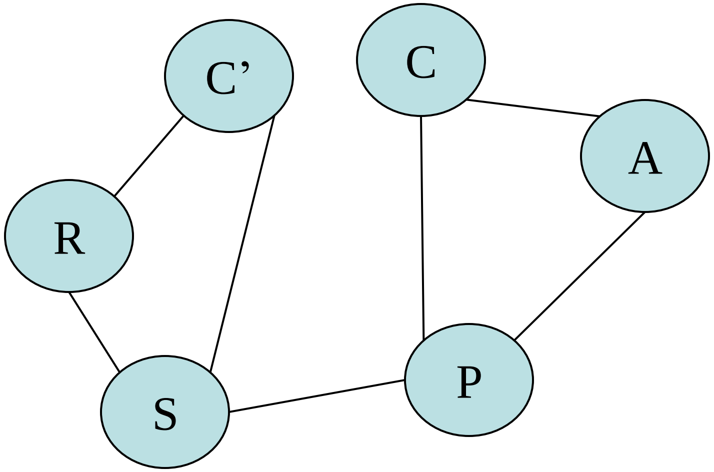


Primal graph *G(Q)*

Hypergraphs vs Graphs

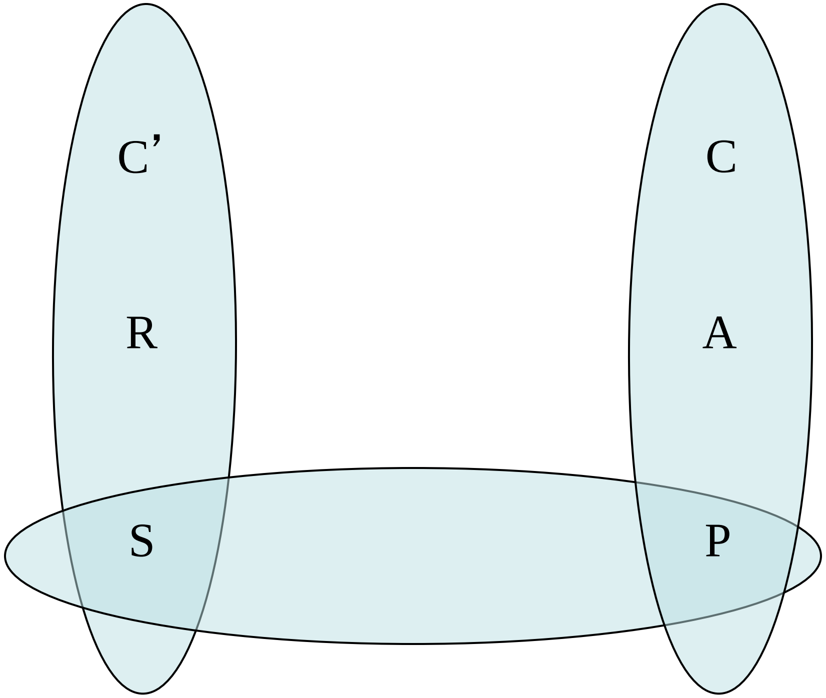


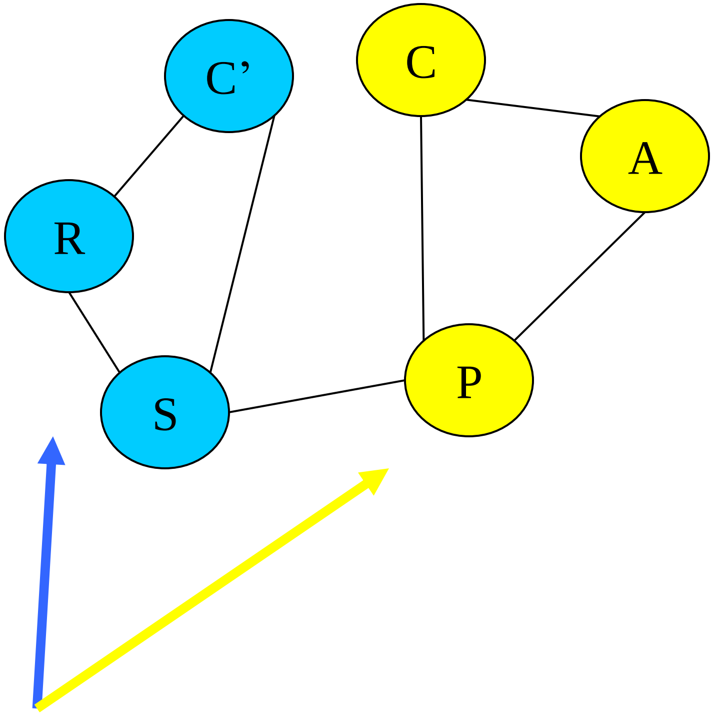
An acyclic hypergraph



Its cyclic primal graph

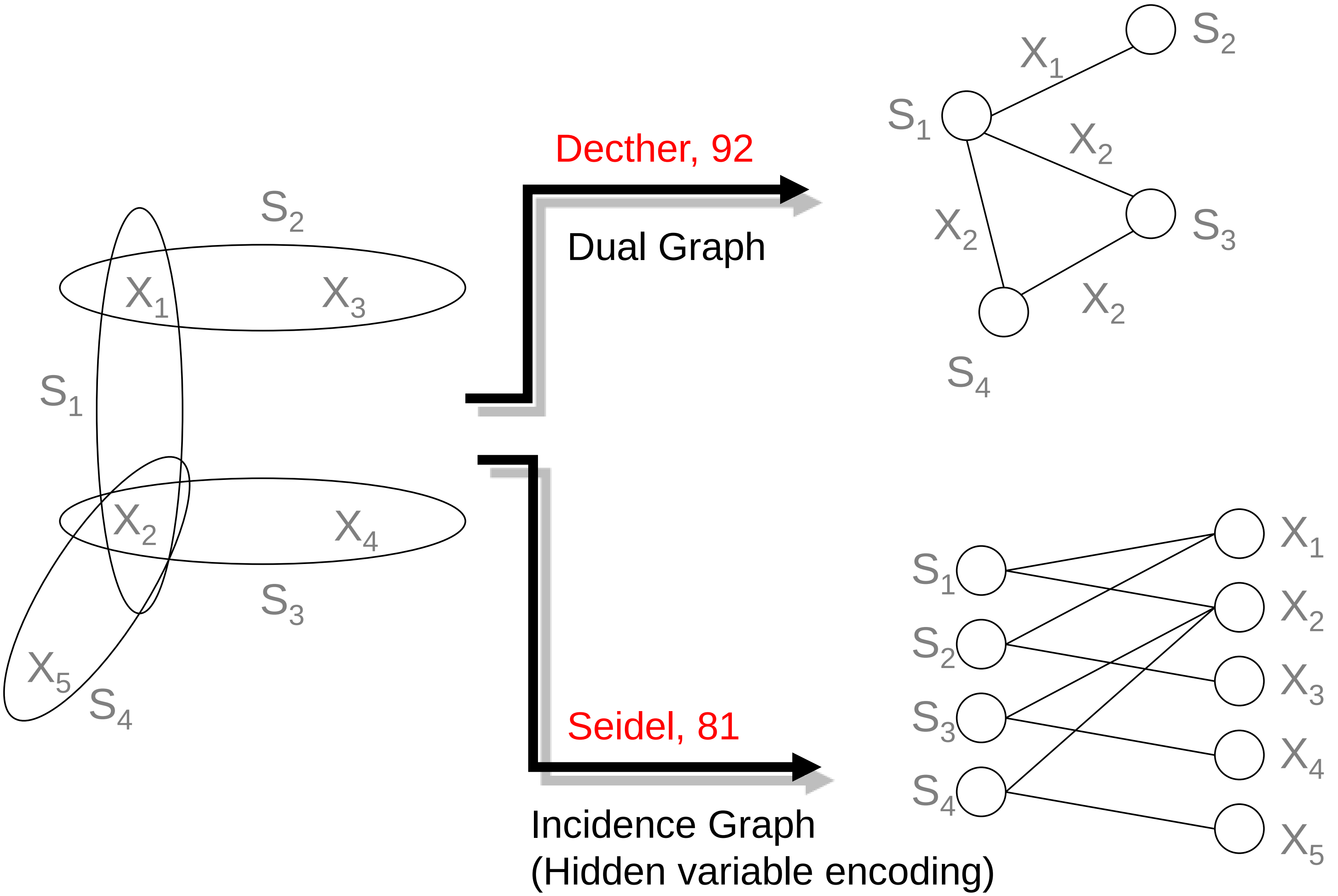
Hypergraphs vs Graphs





There are two cliques.
We cannot know where they come from

Further Graph Representations



Decther, 92

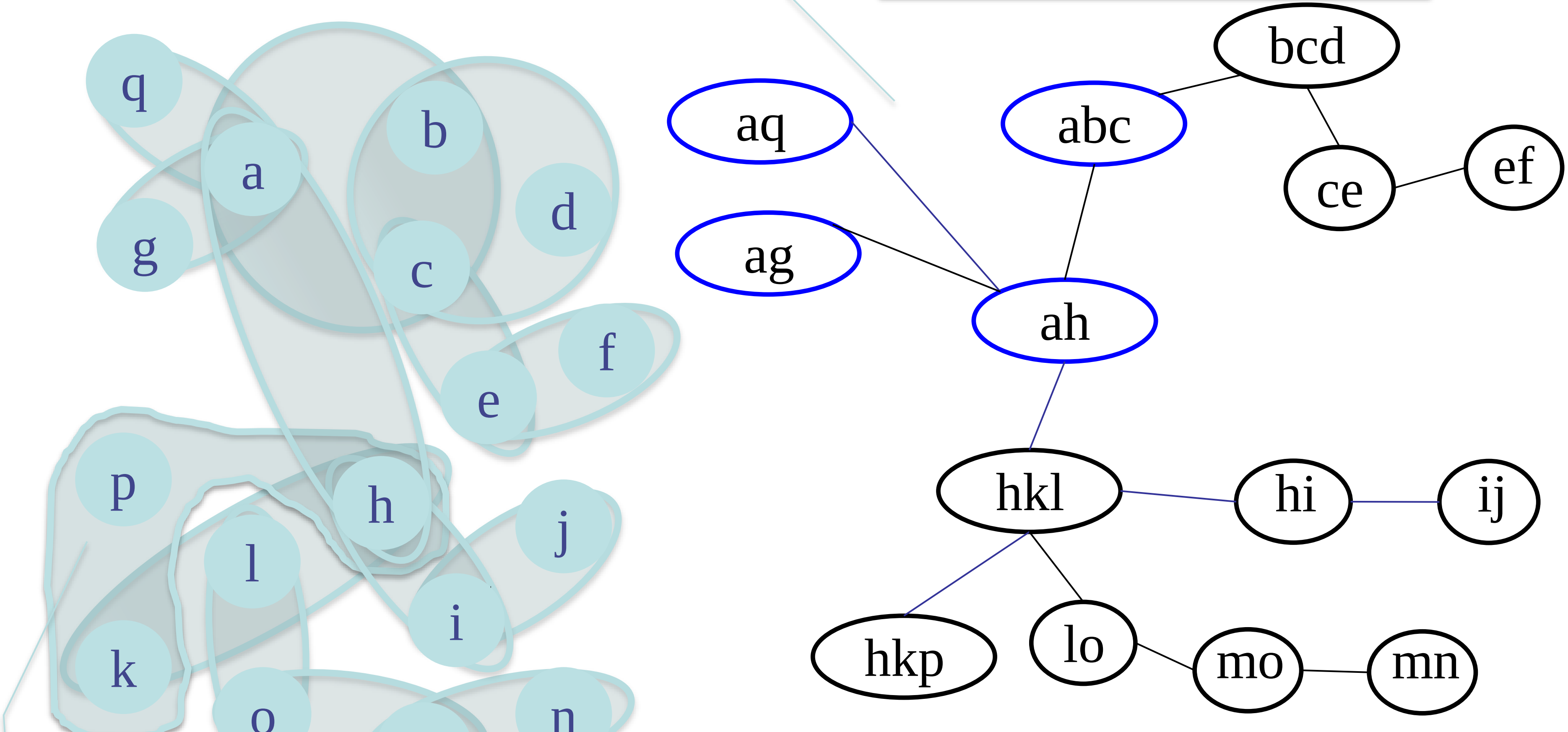
Dual Graph

Seidel, 81

Incidence Graph
(Hidden variable encoding)

α-acyclic Hypergraphs

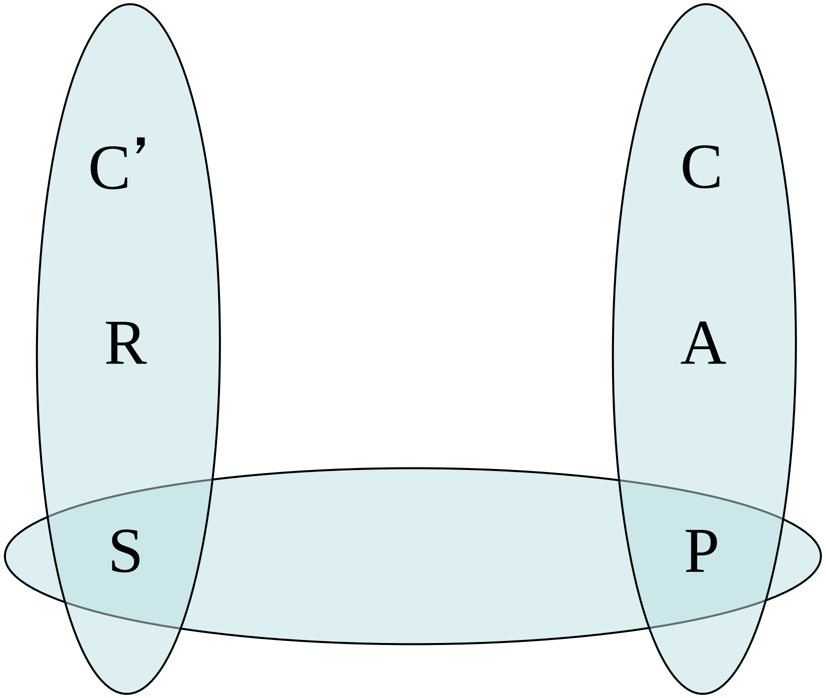
Note the connectedness condition for *a*

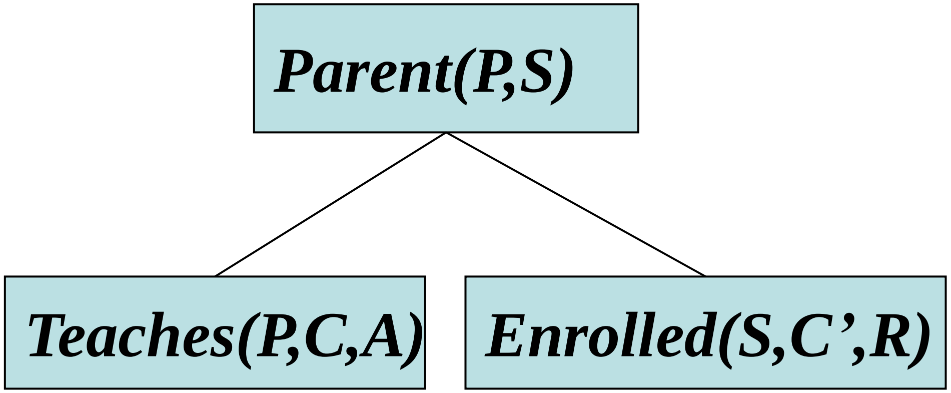


Acyclic hypergraphs may contain cycles

Again on the simplest query

Ans ← *Enrolled(S,C',R)* ∧ *Teaches(P,C,A)* ∧ *Parent(P,S)*





α-acyclic hypergraph

Join Tree

Deciding Hypergraph Acyclicity

- Can be done in linear time
  by **GYO-Reduction**

[Yu and Özsoyoğlu, IEEE Compsac'79; see also Graham, Tech Rep'79]

Input: Hypergraph H

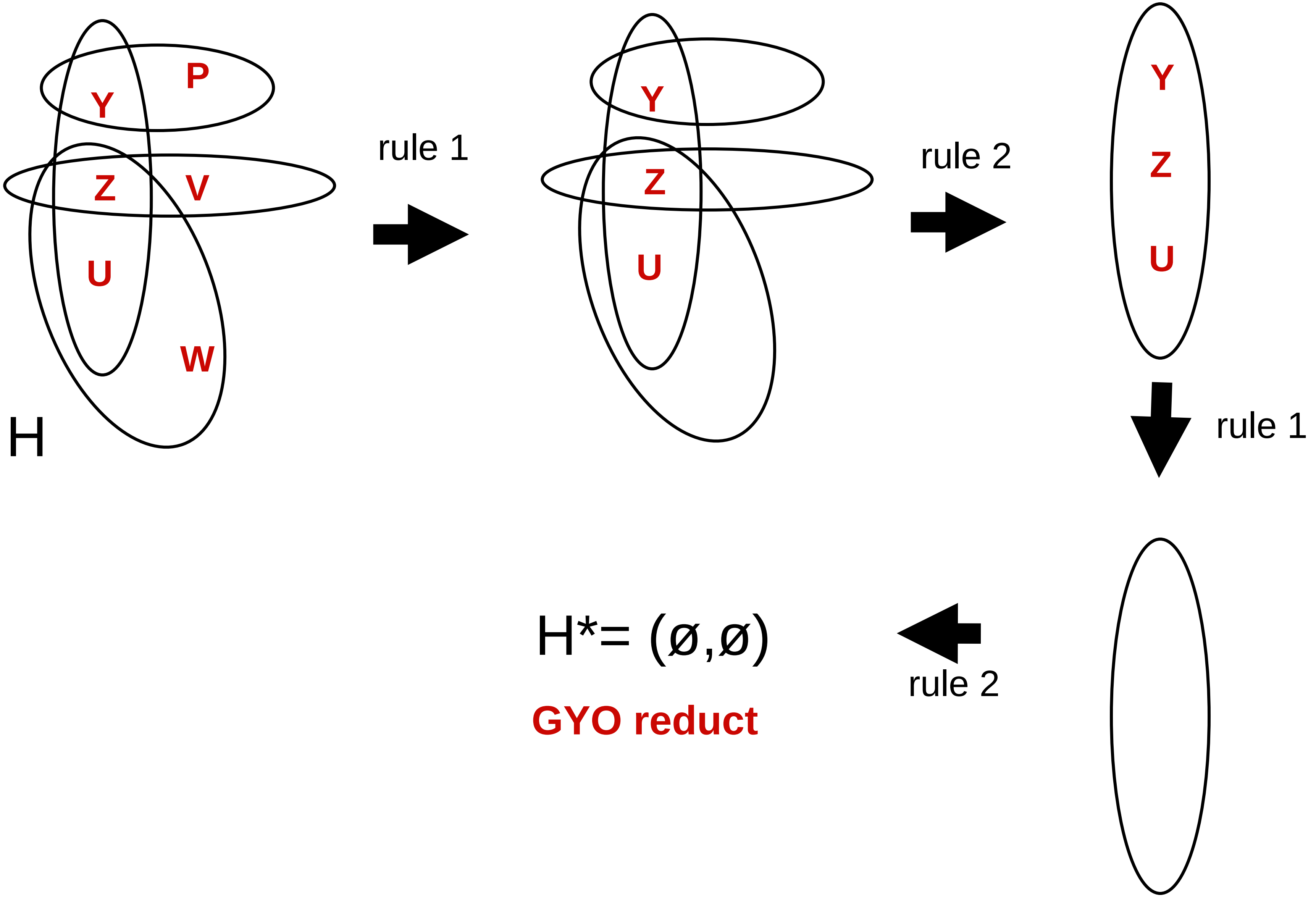
Method: Apply the following two rules as long as possible:

(1) Eliminate vertices that are contained in at most one hyperedge
(2) Eliminate hyperedges that are empty or contained in other hyperedges

H is (α-)acyclic iff the resulting hypergraph empty

Proof: Easy by considering leaves of join tree

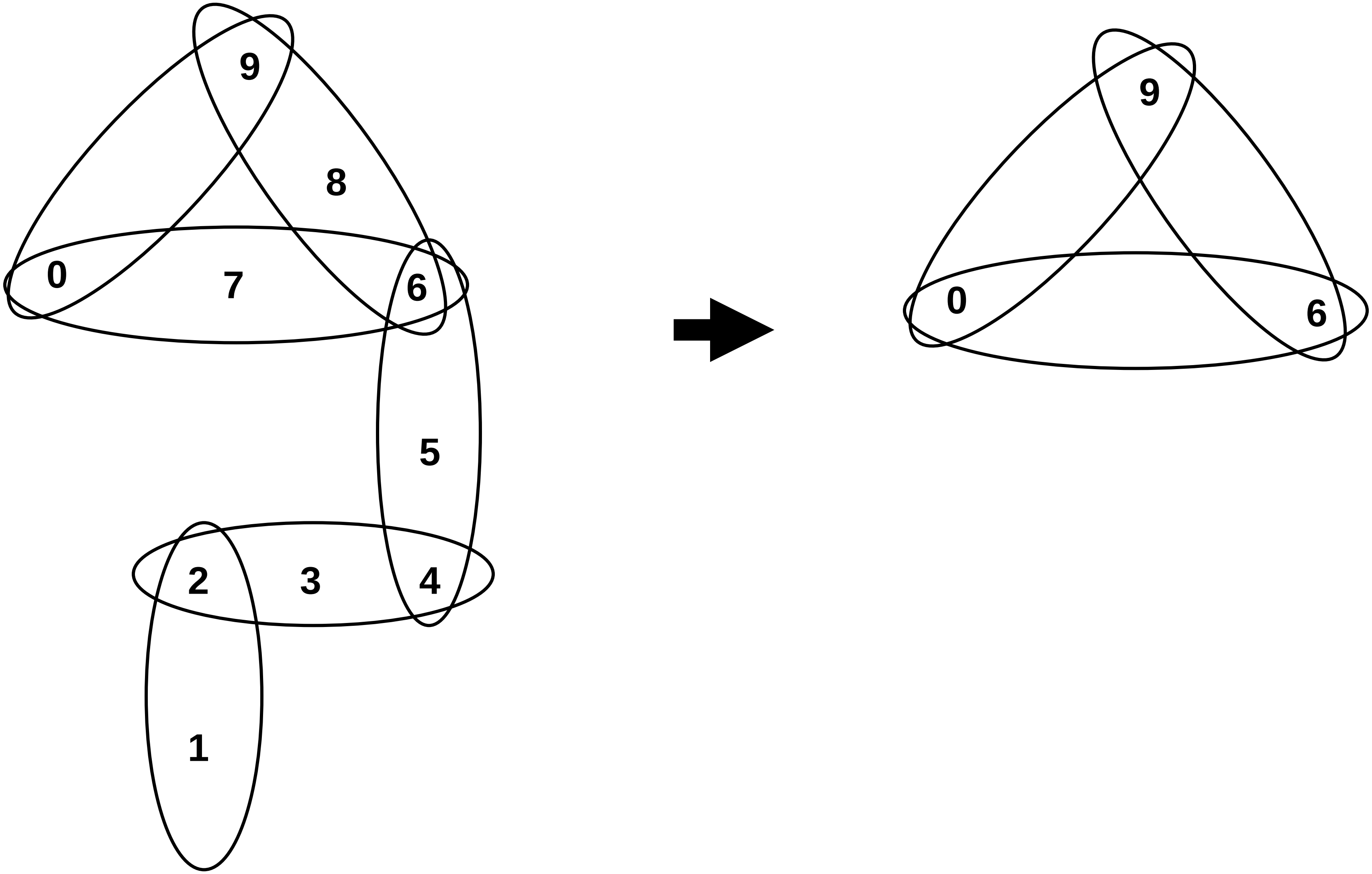
Example of GYO-Reduction



$H^* = (\emptyset,\emptyset)$

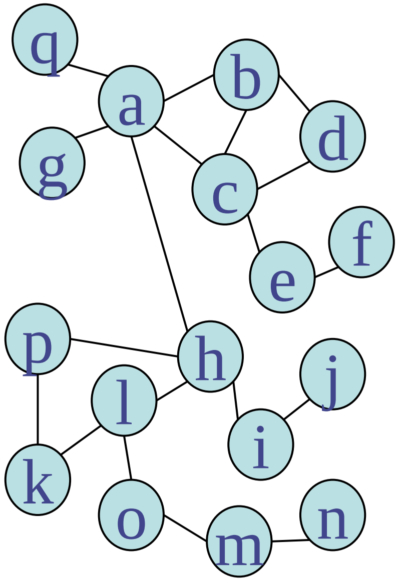
GYO reduct

Example of GYO-irreducible Hypergraph

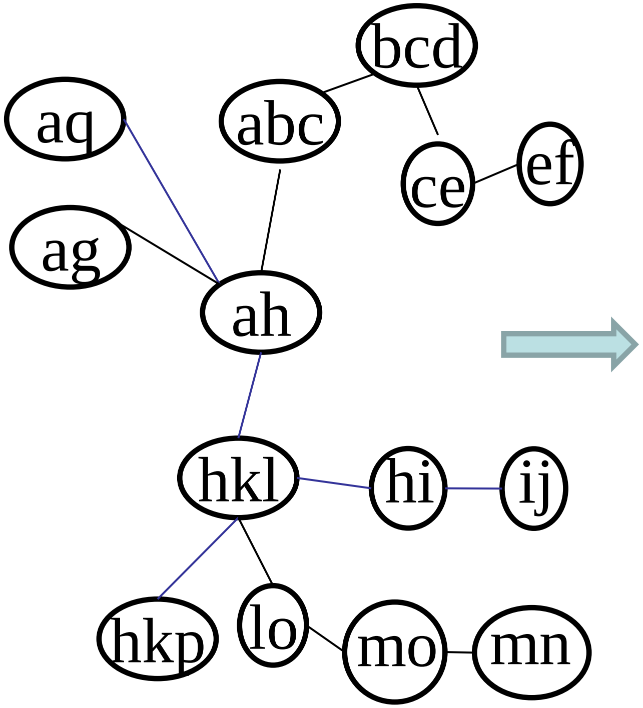


Tree decompositions as Join trees

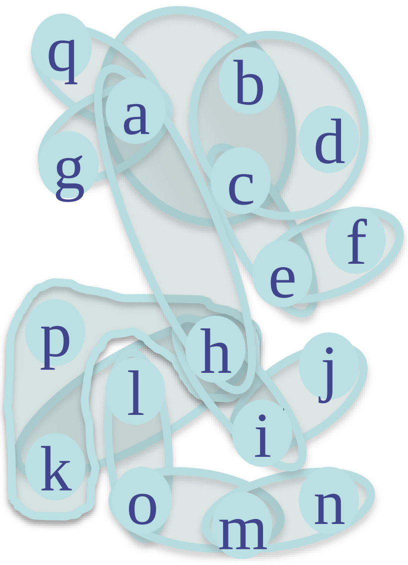
- Tree decomposition as a way of clustering vertices to obtain a join tree (acyclic hypergraph)
- Implicitly defines an equivalent acyclic instance



Graph



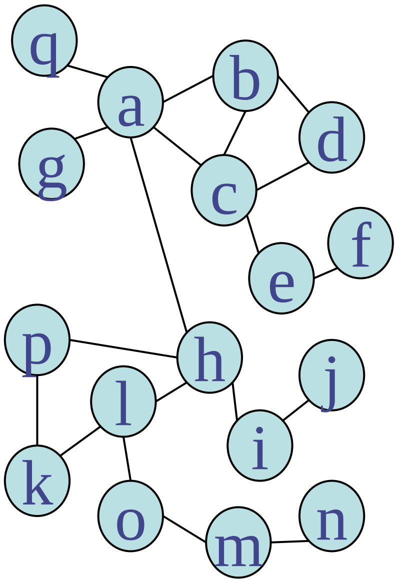
width 2 tree decomposition



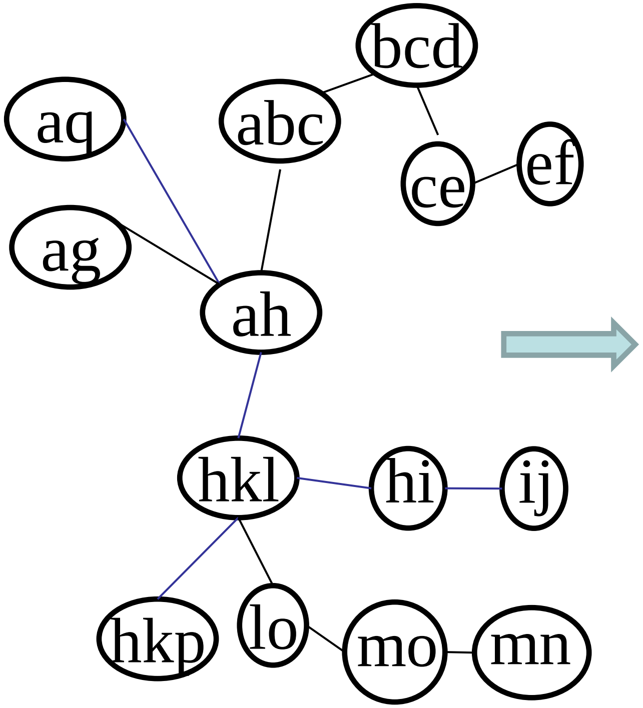
Acyclic instance

From graphs to acyclic hypergraphs

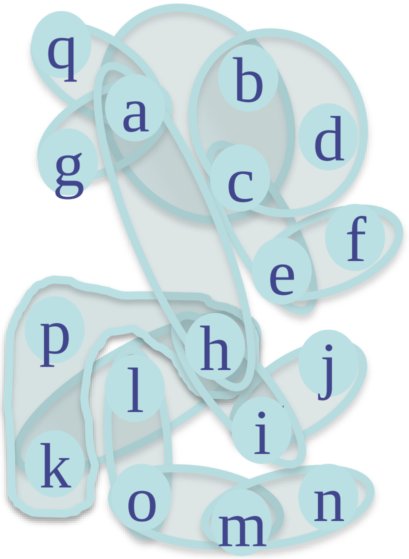
- The “degree of cyclicity” is the treewidth
  (maximum number of vertices in a cluster -1)
- In this example, the treewidth is 2
- That’s ok! We started with a cyclic graph…



Input Graph



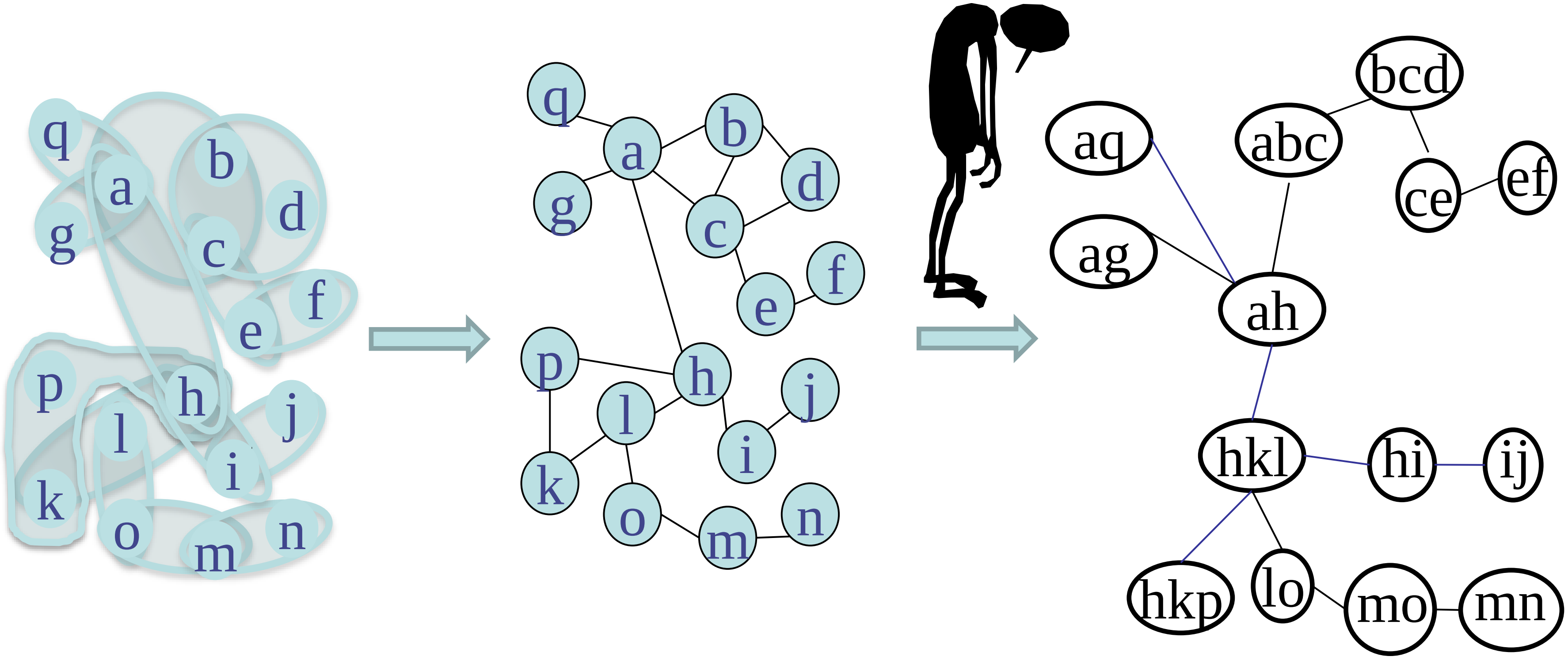
width 2 tree decomposition



Equivalent acyclic instance

Not good for hypergraph-based problems

- Here the input instance is acyclic (hence, easy)
- However, its treewidth is 2!
  (similar troubles for all graph representations)



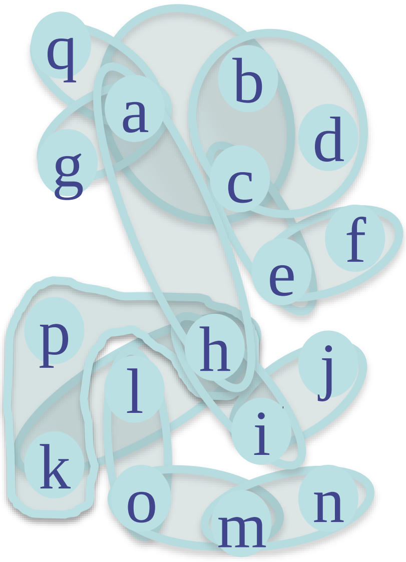
Input: acyclic hypergraph

Primal graph

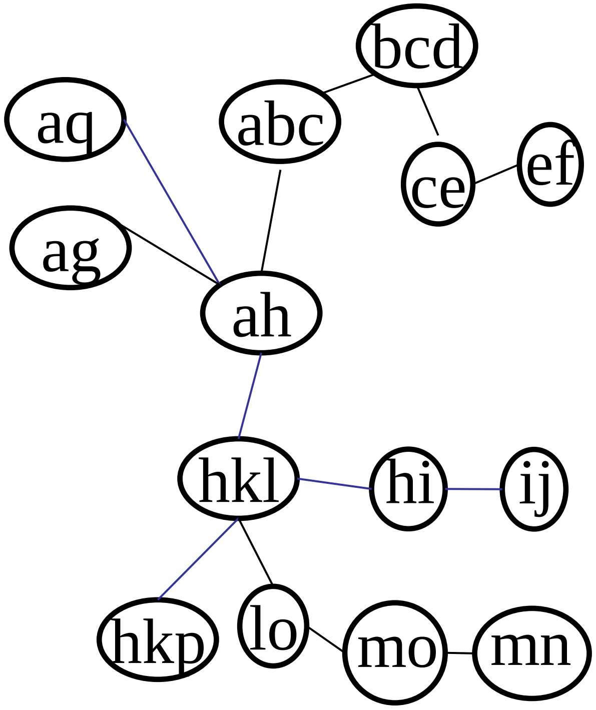
width-2 tree decomposition

A different notion of “width”

- Exploit the fact that a single hyperedge *covers* many vertices
- Degree of cyclicity: maximum number of hyperedges needed to cover every cluster



Input: acyclic instance



One hyperedge covers each cluster: width 1

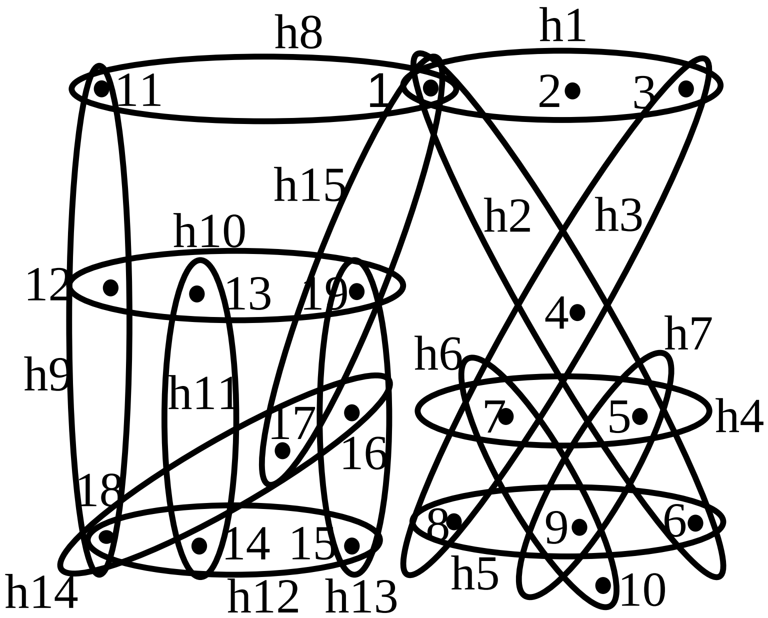
Generalizing acyclicity and treewidth

- Tree decomposition as a way of clustering vertices to obtain a join tree (acyclic hypergraph)
- Implicitly defines an equivalent acyclic instance
- Width of a decomposition: maximum number of hyperedges needed to cover each bag of the tree decomposition
- **Generalized Hypertree Width** (ghw): minimum width over all possible decompositions [Gottlob, Leone, Scarcello, JCSS'03]
  - also known as (acyclic) cover width

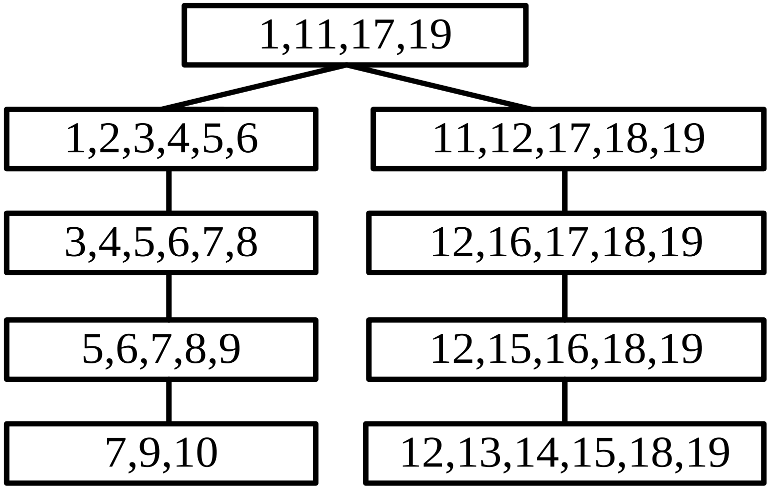
- Generalizes both acyclicity and treewidth:
  - Acyclic hypergraphs are precisely those having ghw = 1
  - The "covering power" of a hyperedge is always greater than the covering power of a vertex (used in the treewidth)

Tree Decomposition of a Hypergraph

H



Tree decomp of G(H)



1,11,17,19

1,2,3,4,5,6

3,4,5,6,7,8

5,6,7,8,9

7,9,10

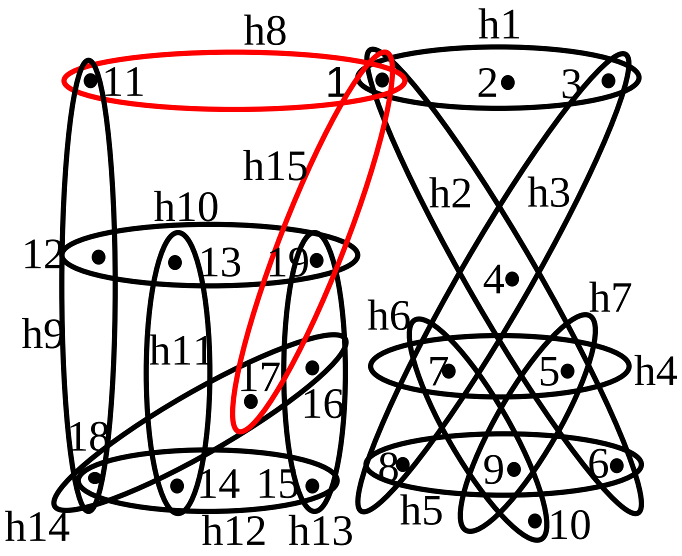
11,12,17,18,19

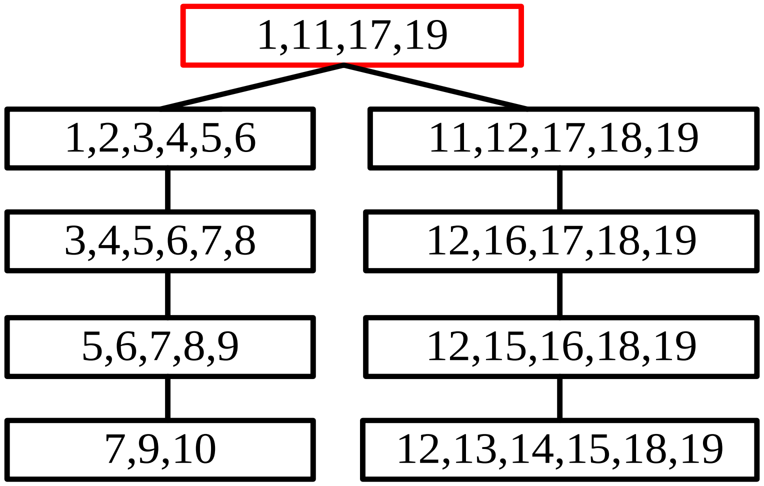
12,16,17,18,19

12,15,16,18,19

12,13,14,15,18,19

2 hyperedges suffice for each bag





1,11,17,19

1,2,3,4,5,6

3,4,5,6,7,8

5,6,7,8,9

7,9,10

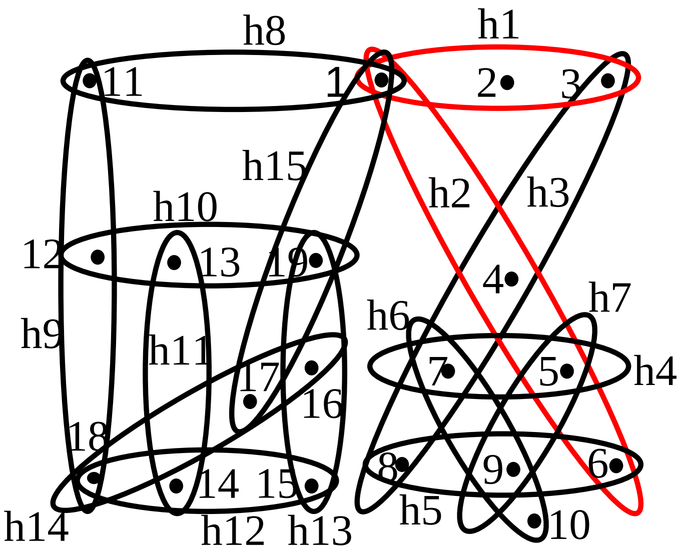
11,12,17,18,19

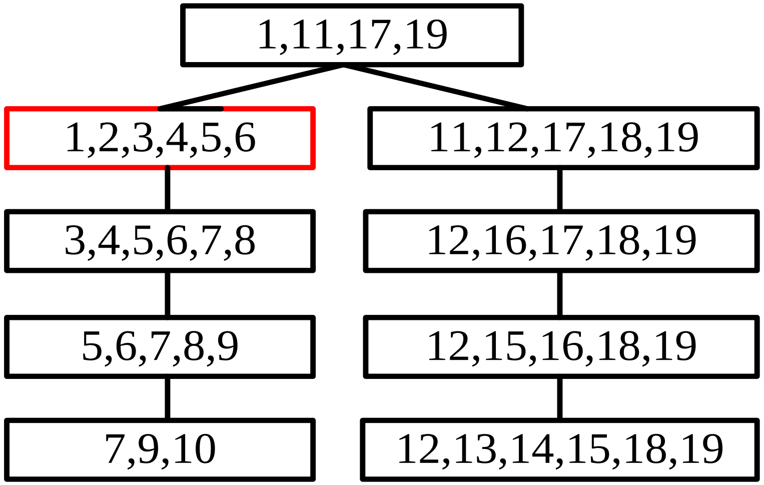
12,16,17,18,19

12,15,16,18,19

12,13,14,15,18,19

2 hyperedges suffice for each bag





1,11,17,19

1,2,3,4,5,6

11,12,17,18,19

3,4,5,6,7,8

12,16,17,18,19

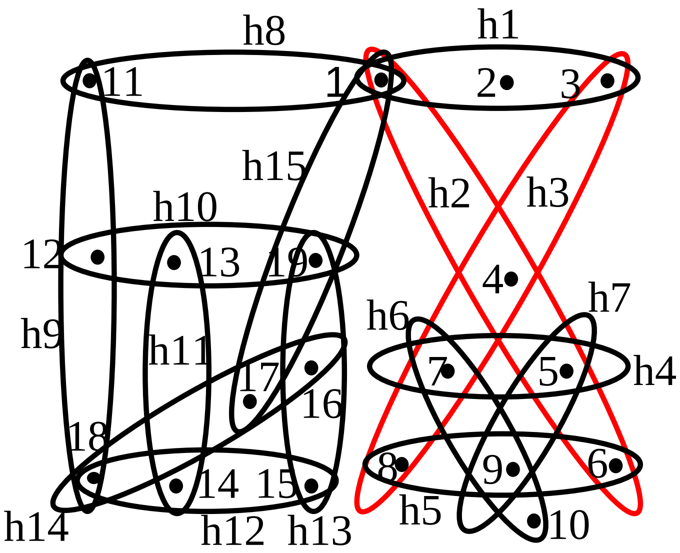
5,6,7,8,9

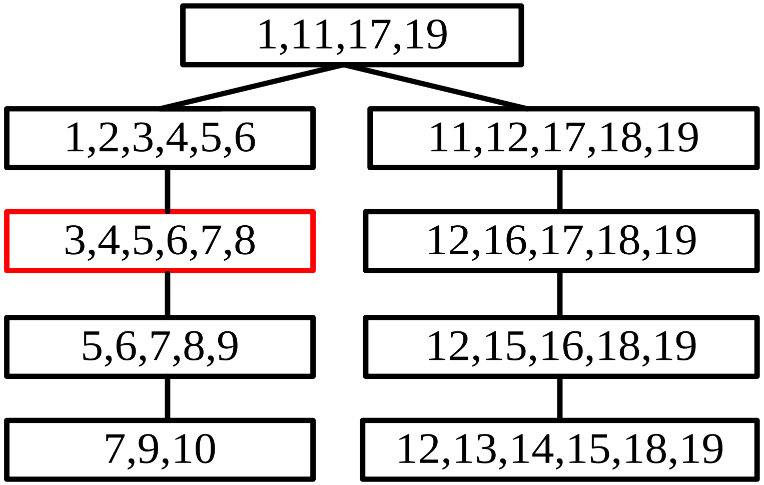
12,15,16,18,19

7,9,10

12,13,14,15,18,19

2 hyperedges suffice for each bag





1,11,17,19

1,2,3,4,5,6

11,12,17,18,19

3,4,5,6,7,8

12,16,17,18,19

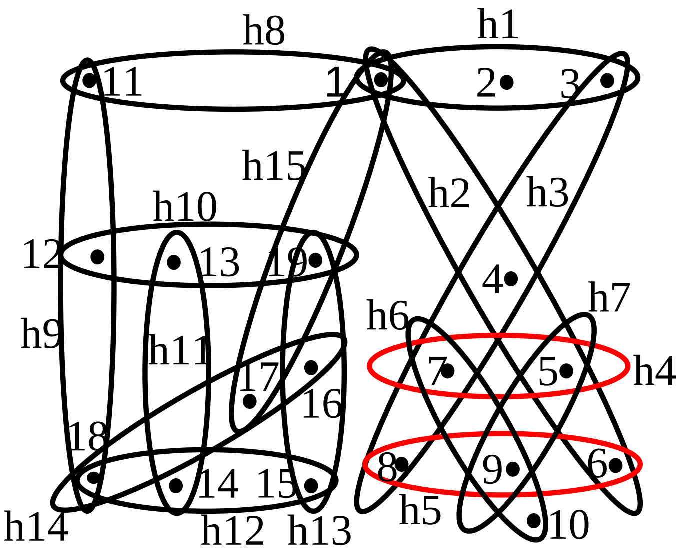
5,6,7,8,9

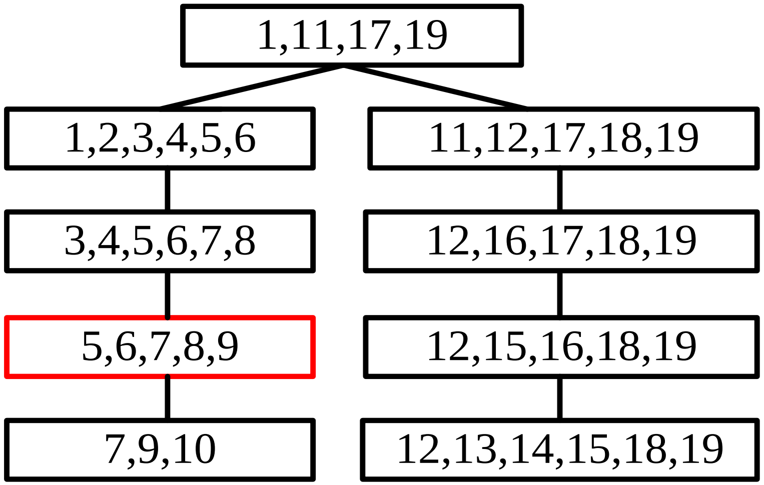
12,15,16,18,19

7,9,10

12,13,14,15,18,19

2 hyperedges suffice for each bag





1,11,17,19

1,2,3,4,5,6

11,12,17,18,19

3,4,5,6,7,8

12,16,17,18,19

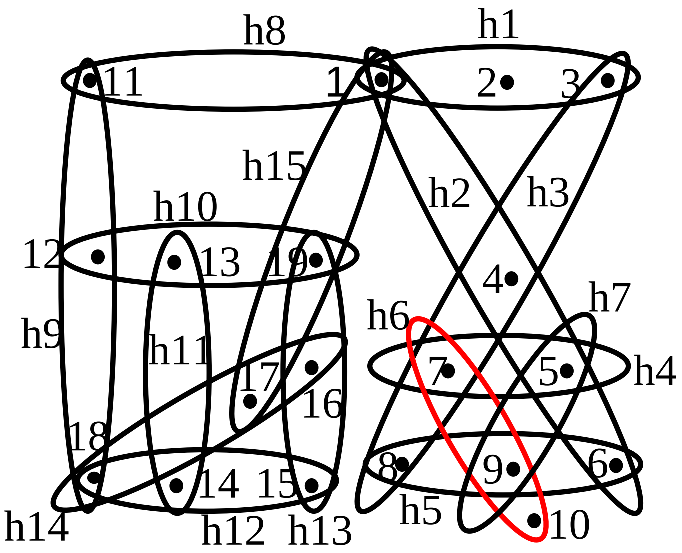
5,6,7,8,9

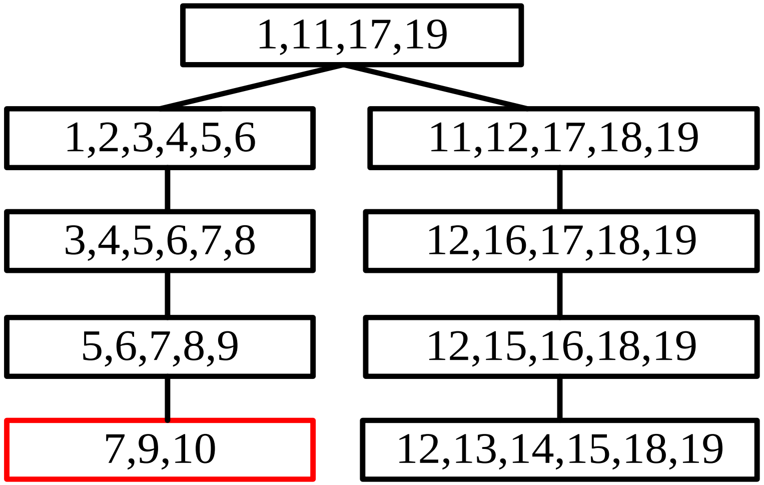
12,15,16,18,19

7,9,10

12,13,14,15,18,19

2 hyperedges suffice for each bag




1,11,17,19

1,2,3,4,5,6

11,12,17,18,19

3,4,5,6,7,8

12,16,17,18,19

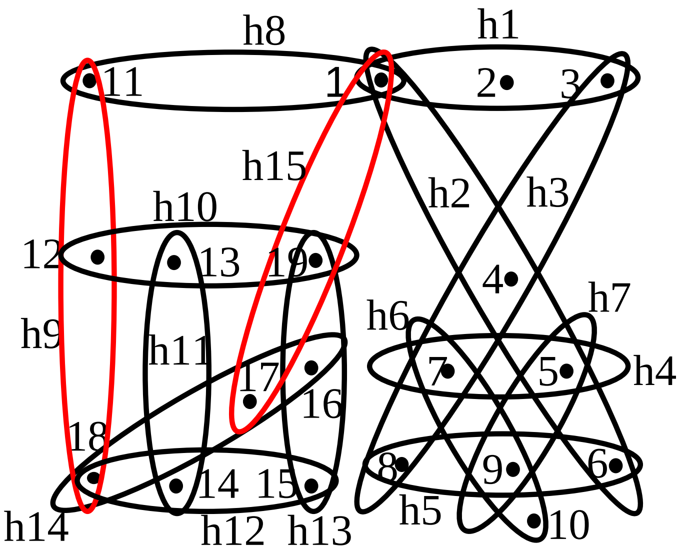
5,6,7,8,9

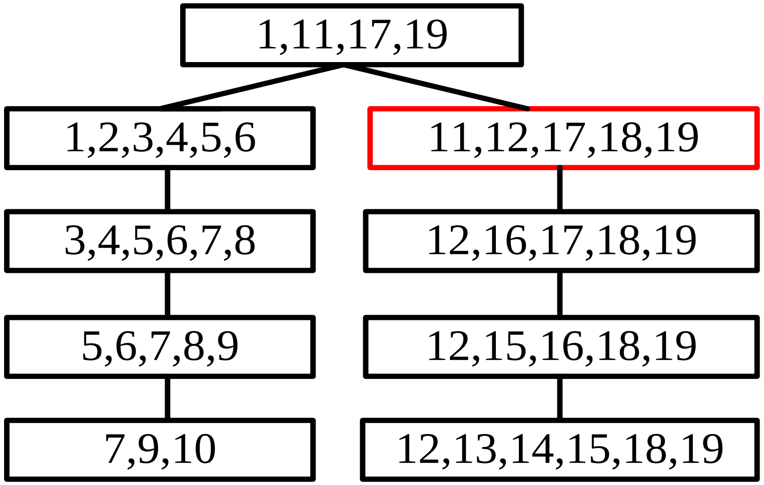
12,15,16,18,19

7,9,10

12,13,14,15,18,19

2 hyperedges suffice for each bag




1,11,17,19

1,2,3,4,5,6

11,12,17,18,19

3,4,5,6,7,8

12,16,17,18,19

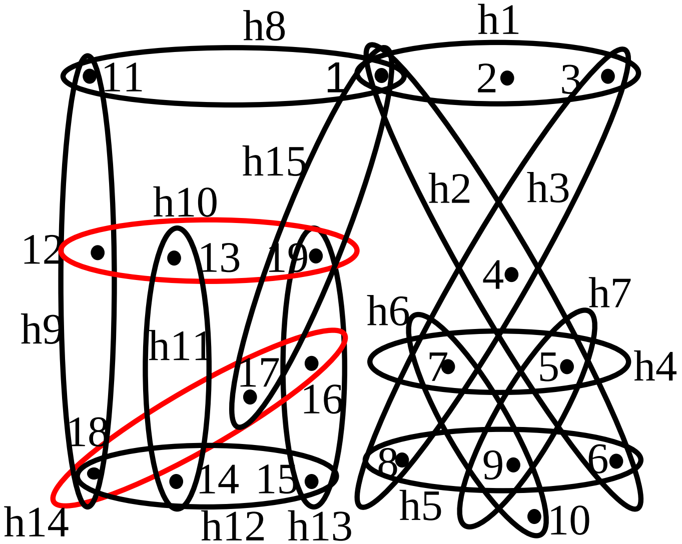
5,6,7,8,9

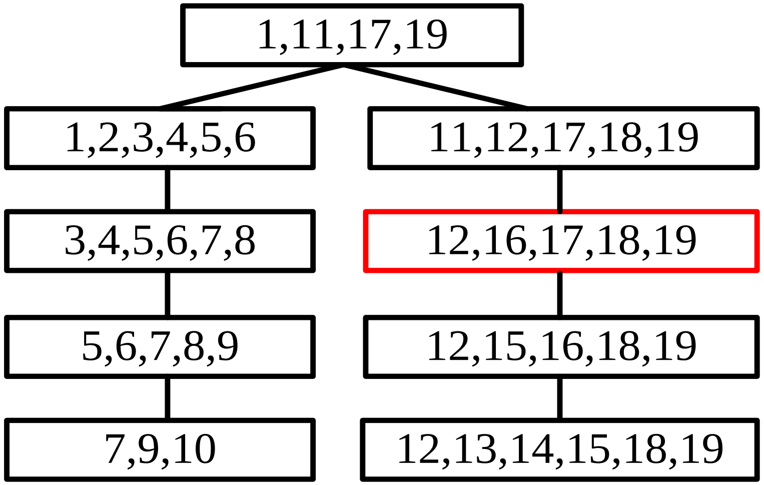
12,15,16,18,19

7,9,10

12,13,14,15,18,19

2 hyperedges suffice for each bag





1,11,17,19

1,2,3,4,5,6

11,12,17,18,19

3,4,5,6,7,8

12,16,17,18,19

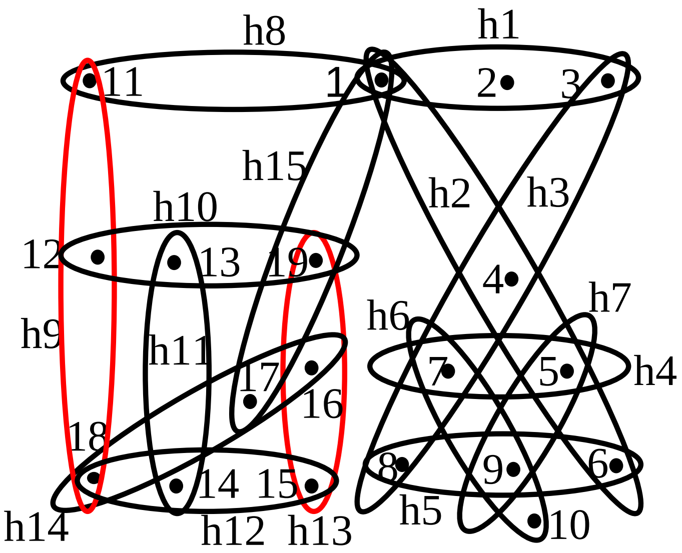
5,6,7,8,9

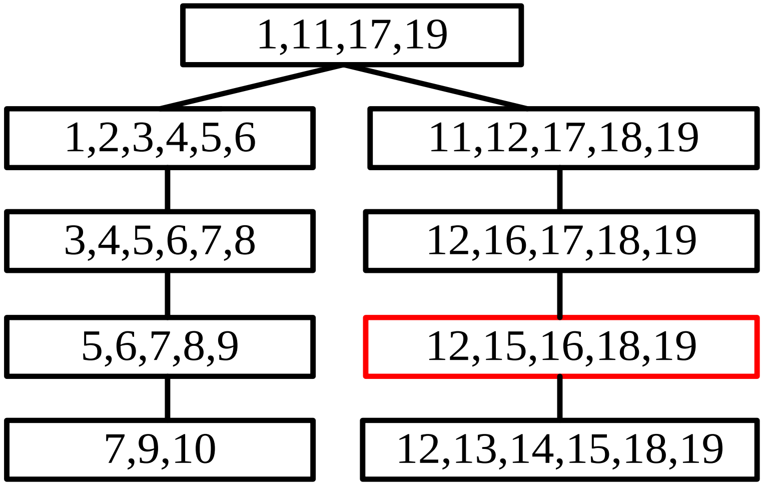
12,15,16,18,19

7,9,10

12,13,14,15,18,19

2 hyperedges suffice for each bag





1,11,17,19

1,2,3,4,5,6

11,12,17,18,19

3,4,5,6,7,8

12,16,17,18,19

5,6,7,8,9

12,15,16,18,19

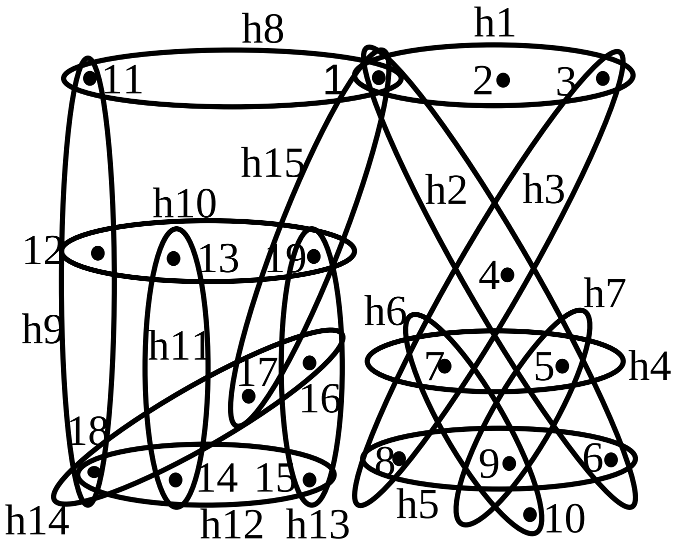
7,9,10

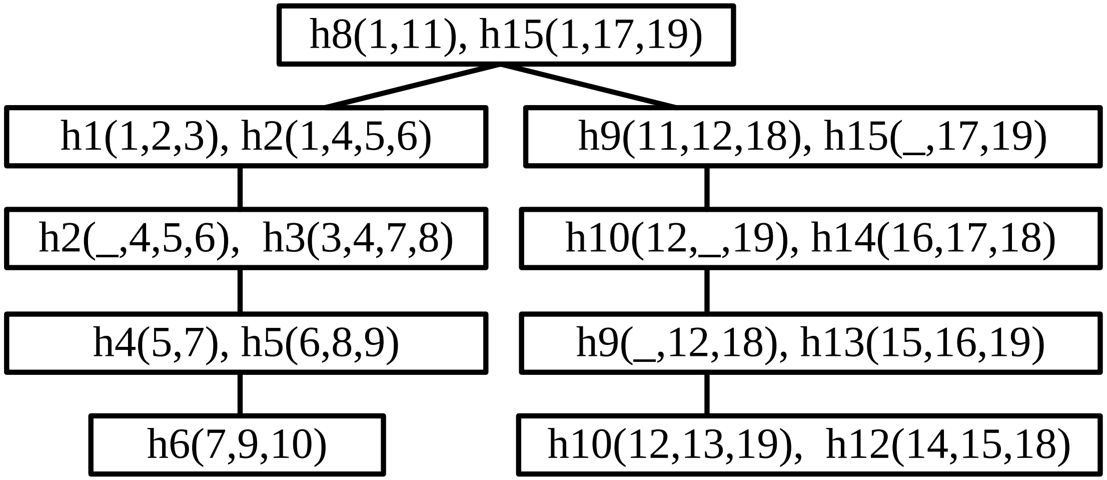
12,13,14,15,18,19

Generalized Hypertree Decomposition

Notation:

- label decomposition vertices by hyperedges
- omit hyperedge elements not used for bag covering (hidden elements are replaced by "_")





h8(1,11), h15(1,17,19)

h1(1,2,3), h2(1,4,5,6)

h2(_,4,5,6), h3(3,4,7,8)

h4(5,7), h5(6,8,9)

h6(7,9,10)

h9(11,12,18), h15(_,17,19)

h10(12,_,19), h14(16,17,18)

h9(_,12,18), h13(15,16,19)

h10(12,13,19), h12(14,15,18)

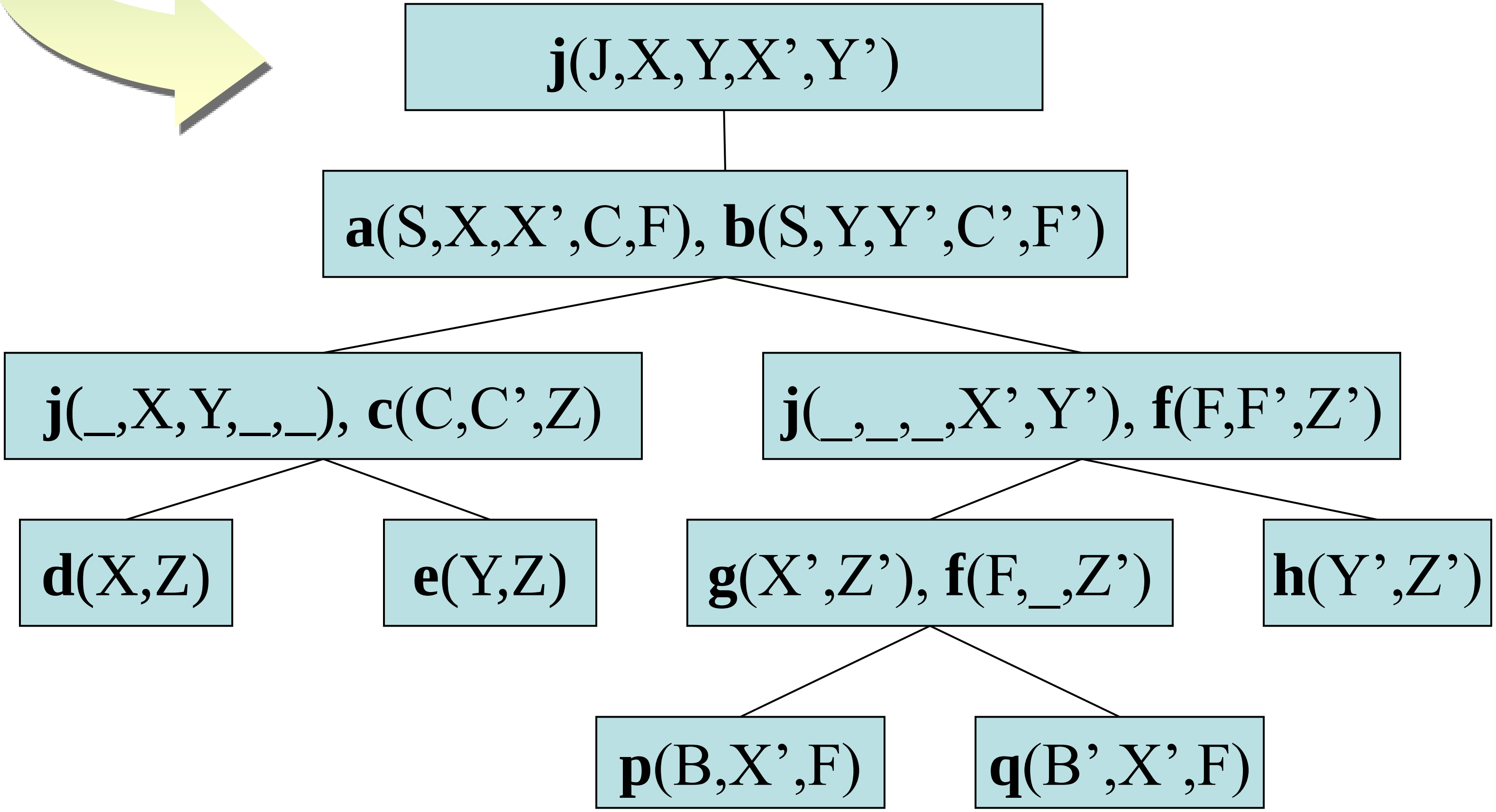
Generalized hypetree decomposition of width 2

Generalized Hypertree Decompositions

$a(S,X,X',C,F)$ $b(S,Y,Y',C',F')$ $c(C,C',Z)$ $d(X,Z)$

$e(Y,Z)$ $f(F,F',Z')$ $g(X',Z')$ $h(Y',Z')$

$j(J,X,Y,X',Y')$ $p(B,X',F)$ $q(B',X',F)$



j(J,X,Y,X’,Y’)

a(S,X,X’,C,F), **b**(S,Y,Y’,C’,F’)

j(_,X,Y,_,_), **c**(C,C’,Z)

j(_,_,_,X’,Y’), **f**(F,F’,Z’)

d(X,Z)

e(Y,Z)

g(X’,Z’), **f**(F,_,Z’)

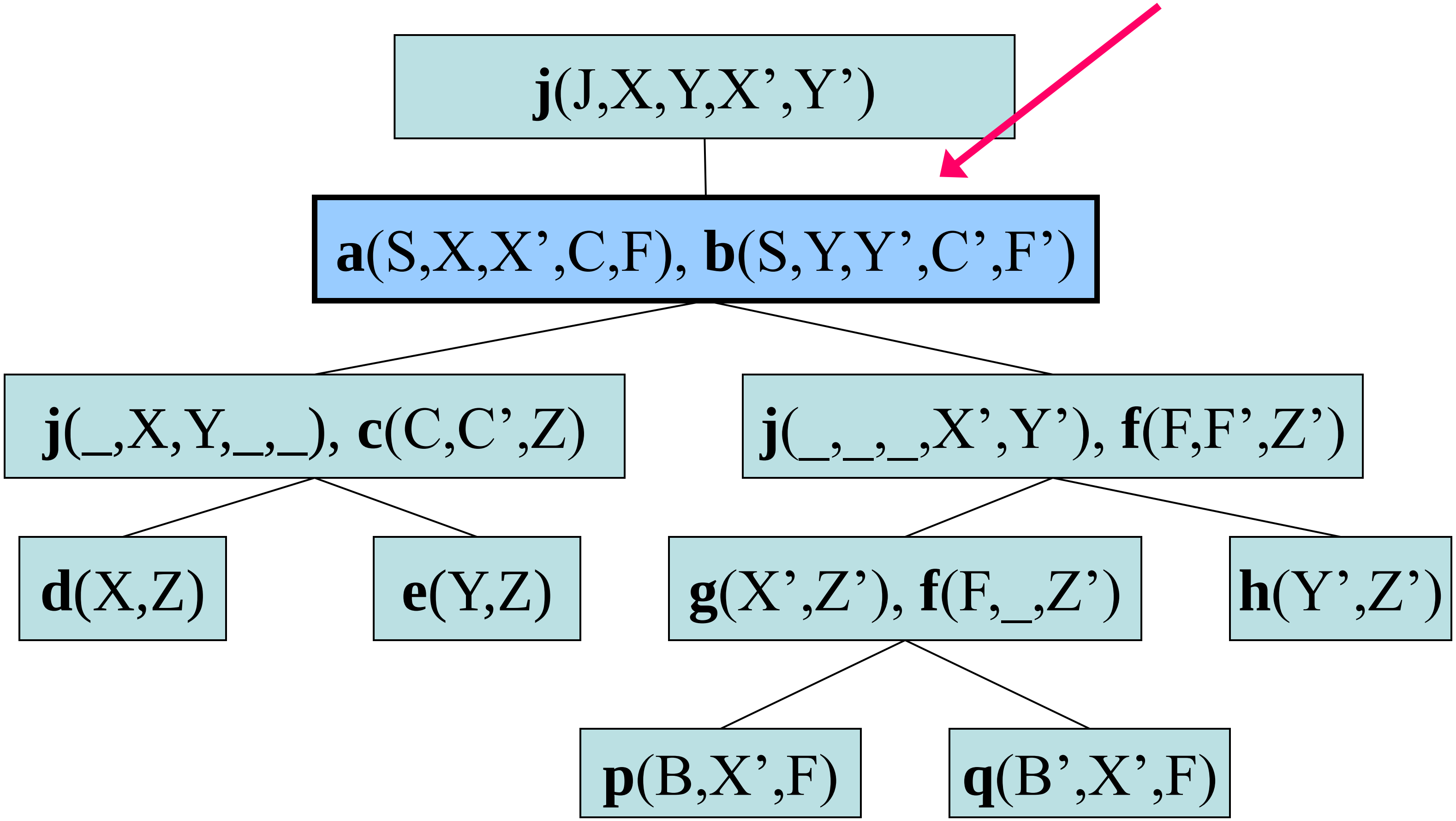
h(Y’,Z’)

p(B,X’,F)

q(B’,X’,F)

Basic Conditions$_{(1/3)}$

Original (direct) definition



We group edges

j(J,X,Y,X’,Y’)

a(S,X,X’,C,F), **b**(S,Y,Y’,C’,F’)

j(_,X,Y,_,_), **c**(C,C’,Z)

j(_,_,_,X’,Y’), **f**(F,F’,Z’)

d(X,Z)

e(Y,Z)

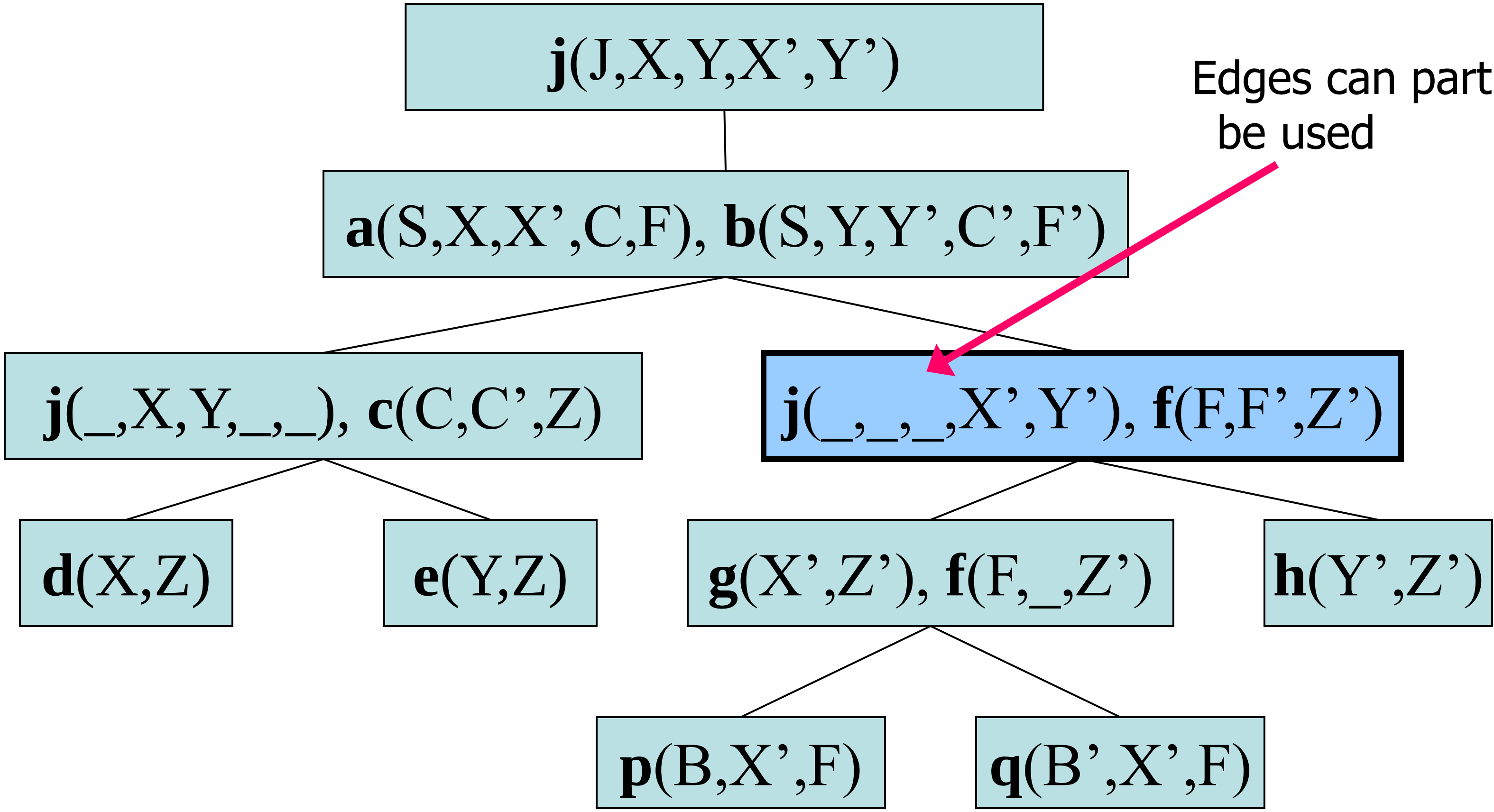
g(X’,Z’), **f**(F,_,Z’)

h(Y’,Z’)

p(B,X’,F)

q(B’,X’,F)

Basic Conditions$_{(2/3)}$



j(J,X,Y,X’,Y’)

a(S,X,X’,C,F), **b**(S,Y,Y’,C’,F’)

j(_,X,Y,_,_), **c**(C,C’,Z)

j(_,_,_,X’,Y’), **f**(F,F’,Z’)

d(X,Z)

e(Y,Z)

g(X’,Z’), **f**(F,_,Z’)

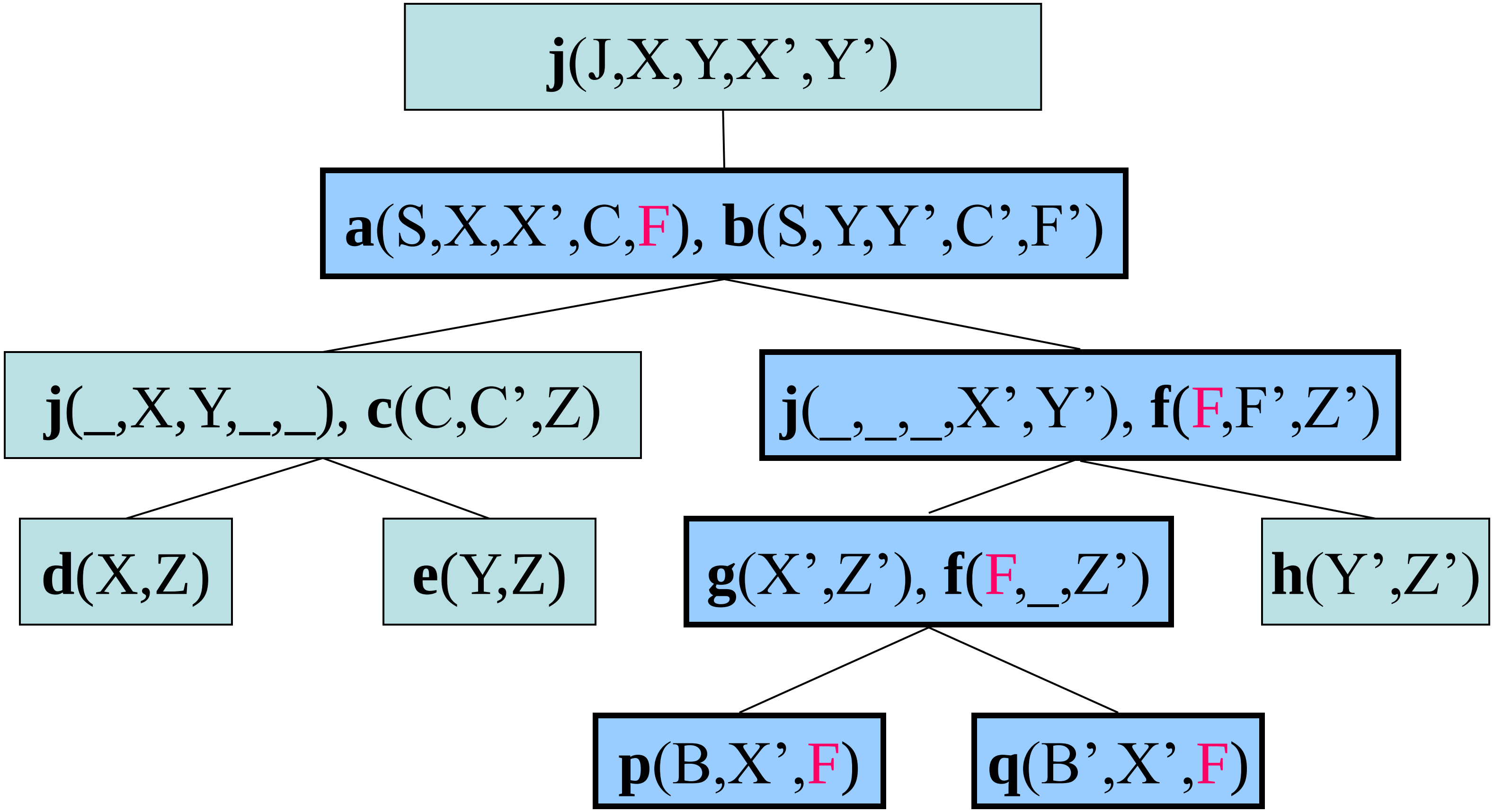
h(Y’,Z’)

p(B,X’,F)

q(B’,X’,F)

Edges can partially be used

Connectedness Condition$_{(3/3)}$

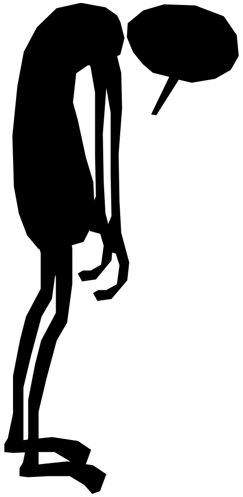

j(J,X,Y,X’,Y’)
a(S,X,X’,C,F), **b**(S,Y,Y’,C’,F’)
j(_,X,Y,_,_), **c**(C,C’,Z)
j(_,_,_,X’,Y’), **f**(F,F’,Z’)
d(X,Z)
e(Y,Z)
g(X’,Z’), **f**(F,_,Z’)
h(Y’,Z’)
p(B,X’,F)
q(B’,X’,F)

Computational Question

- Can we determine in polynomial time whether ghw(H) < k for constant k ?

Computational Question

- Can we determine in polynomial time whether ghw(H) < k for constant k ?



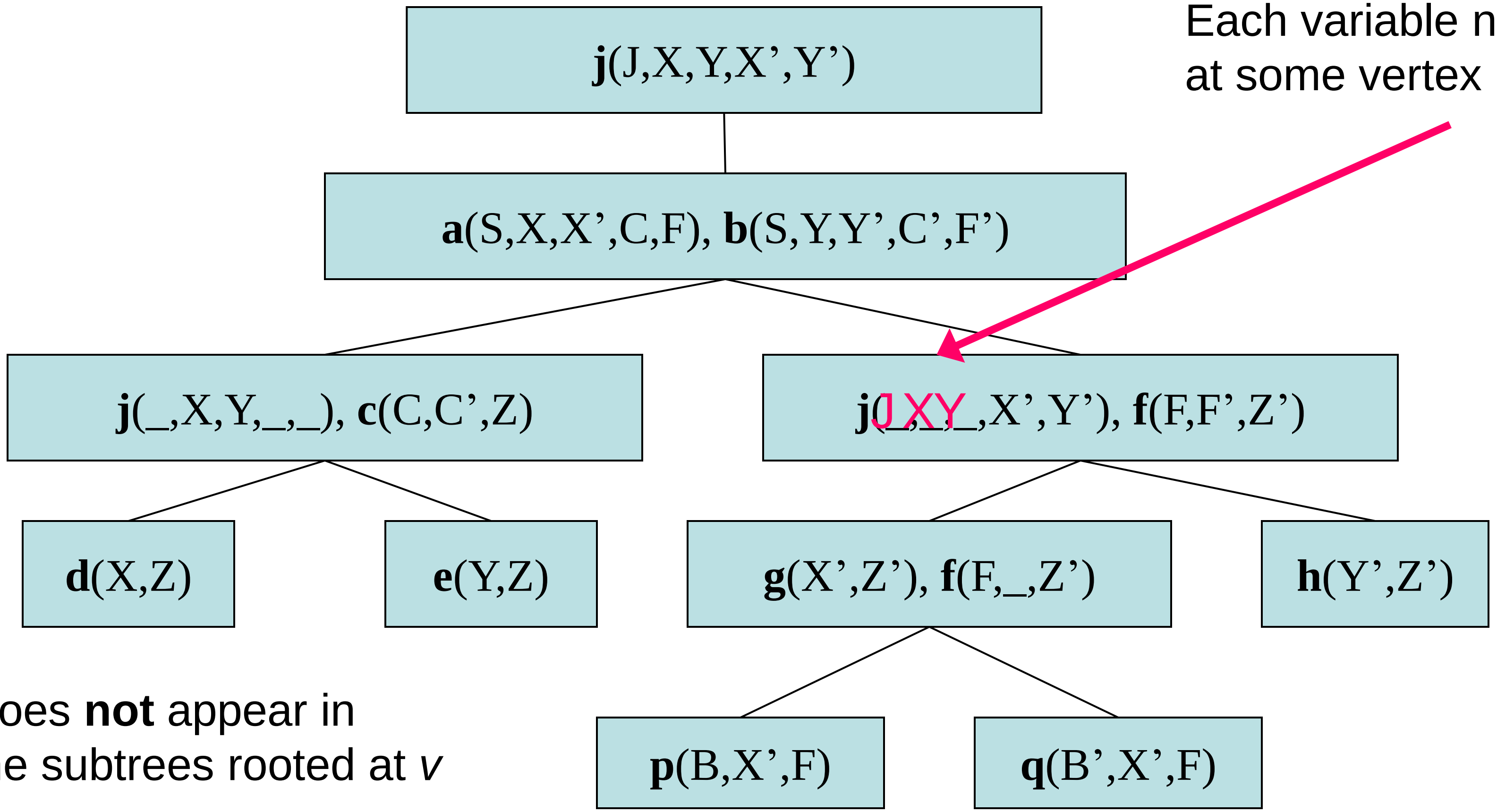
Bad news: ghw(H) < 4? NP-complete

[Gottlob, Miklós, and Schwentick, J.ACM'09]

Hypertree Decomposition (HTD)

HTD = Generalized HTD +Special Condition

[Gottlob, Leone, Scarcello, PODS'99; JCSS'02]



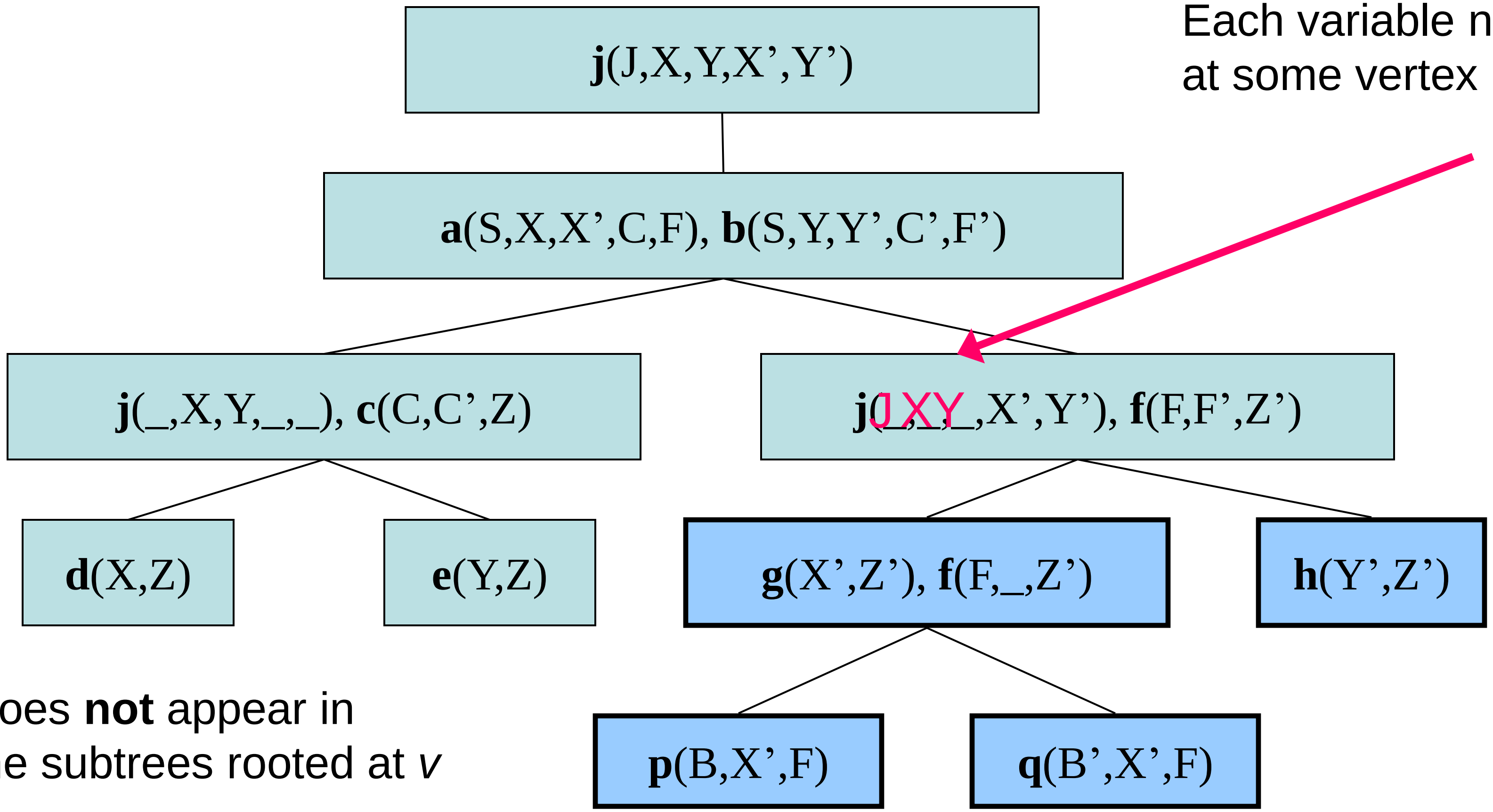
j(J,X,Y,X',Y')
a(S,X,X',C,F), **b**(S,Y,Y',C',F')
j(_,X,Y,_,_), **c**(C,C',Z)
j(_,_,_,X',Y'), **f**(F,F',Z')
J,X,Y
d(X,Z)
e(Y,Z)
g(X',Z'), **f**(F,_,Z')
h(Y',Z')
p(B,X',F)
q(B',X',F)

Each variable not used at some vertex *v*

Does **not** appear in the subtrees rooted at *v*

Special Condition

Each variable not used at some vertex *v*



j(J,X,Y,X’,Y’)

a(S,X,X’,C,F), **b**(S,Y,Y’,C’,F’)

j(_,X,Y,_,_), **c**(C,C’,Z)

j(_,_,_,X’,Y’), **f**(F,F’,Z’)

J XY

d(X,Z)

e(Y,Z)

g(X’,Z’), **f**(F,_,Z’)

h(Y’,Z’)

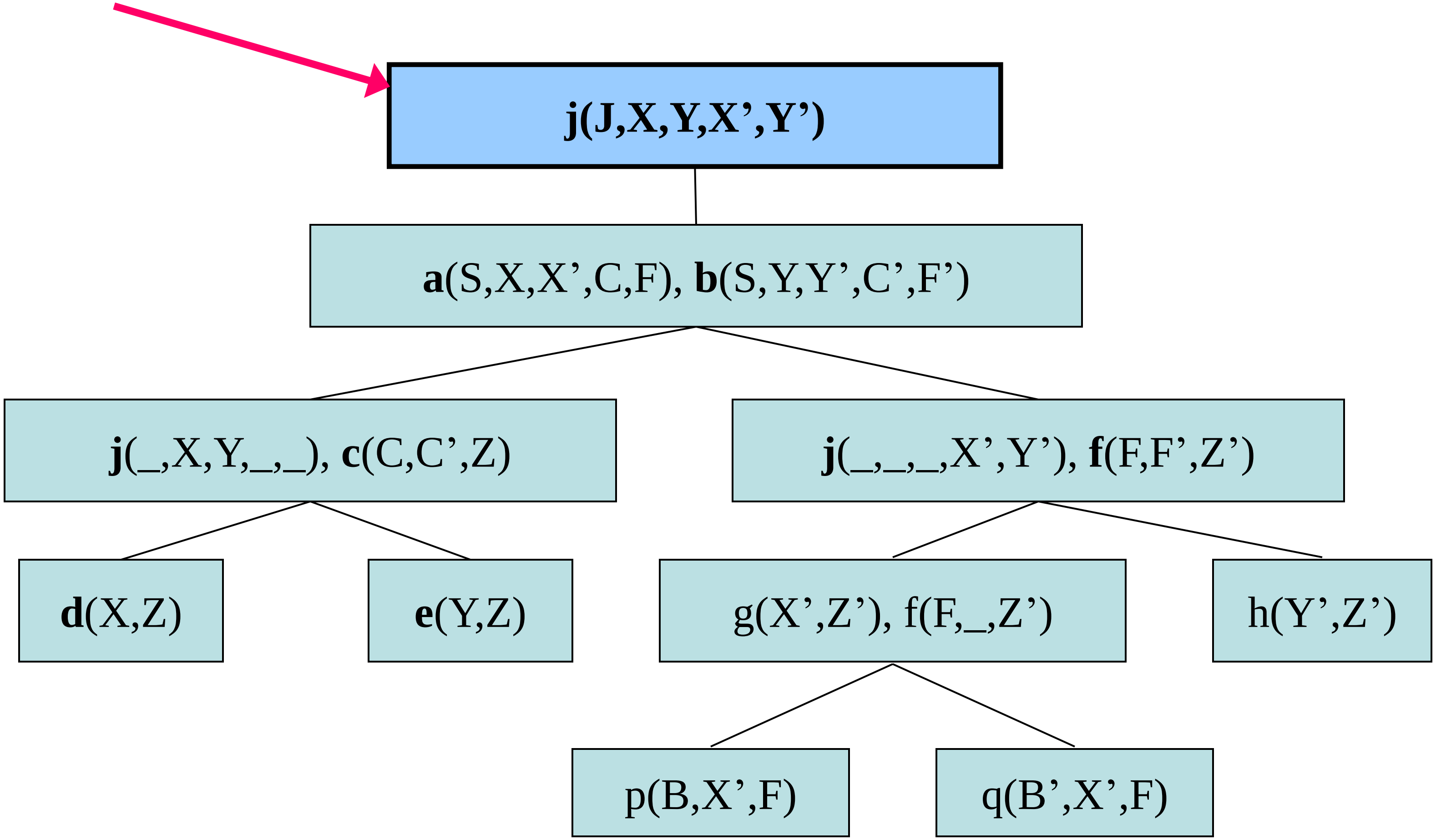
p(B,X’,F)

q(B’,X’,F)

Does **not** appear in
the subtrees rooted at *v*

Special Condition

Thus, e.g., all available variables in the root must be used



j(J,X,Y,X’,Y’)

a(S,X,X’,C,F), **b**(S,Y,Y’,C’,F’)

j(_,X,Y,_,_), **c**(C,C’,Z)

j(_,_,_,X’,Y’), **f**(F,F’,Z’)

d(X,Z)

e(Y,Z)

g(X’,Z’), f(F,_,Z’)

h(Y’,Z’)

p(B,X’,F)

q(B’,X’,F)

Positive Results on Hypertree Decompositions

- For fixed *k*, deciding whether *hw(Q)* ≤ *k* is in polynomial time (LOGCFL)
- Computing hypertree decompositions is feasible in polynomial time (for fixed *k*).

But: FP-intractable wrt k: W[2]-hard.

Relationship GHW vs HW

Observation:

$$ghw(H) = hw(H^*)$$

where $H^* = H \cup \{E' \mid \exists E \text{ in } edges(H): E' \subseteq E\}$

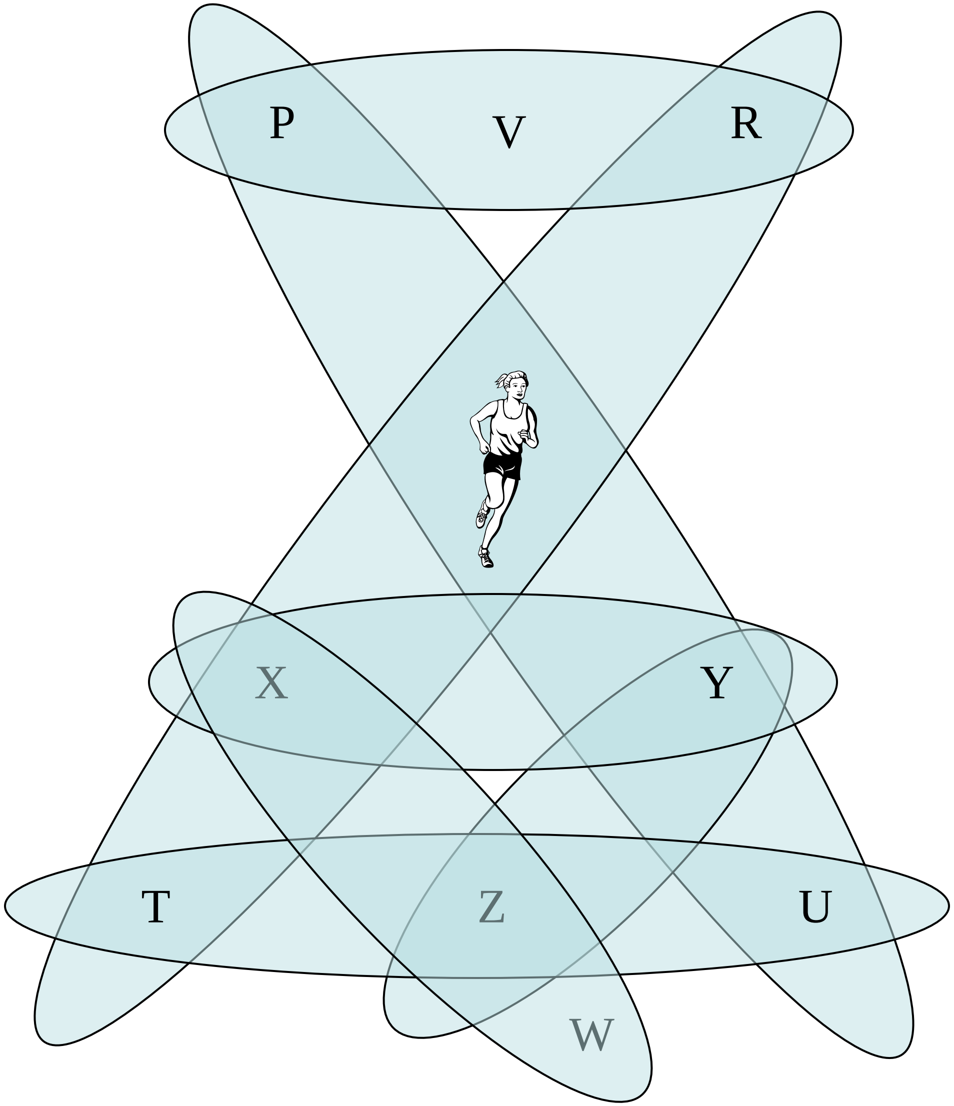
Exponential!

Approximation Theorem [Adler,Gottlob,Grohe ,05] :

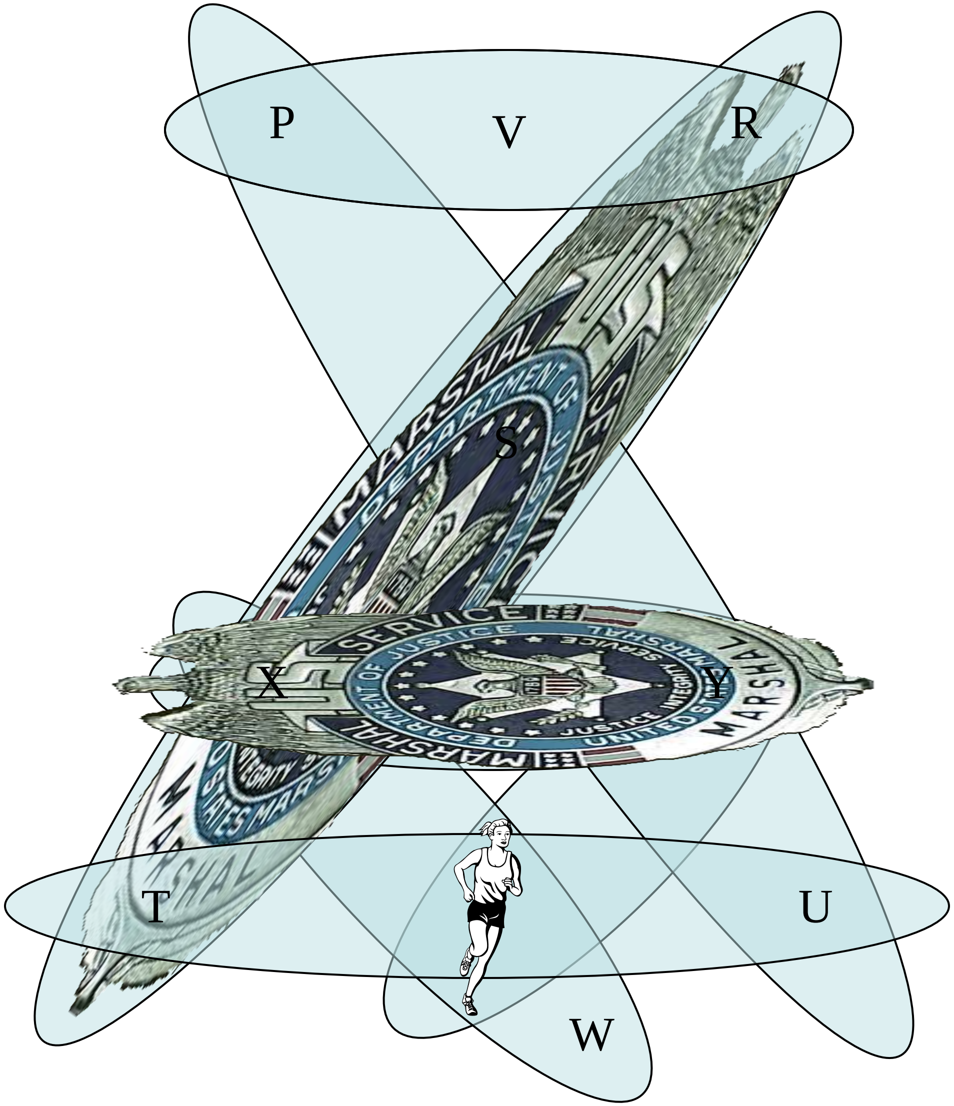
$$\boxed{ghw(H) <= 3hw(H)+1}$$

GHW and HW identify the same
set of classes having bounded width

Game Characterization: Robber and Marshals



Marshals block hyperedges



Game Characterization: Robber and Marshals

- A robber and *k* marshals play the game on a hypergraph

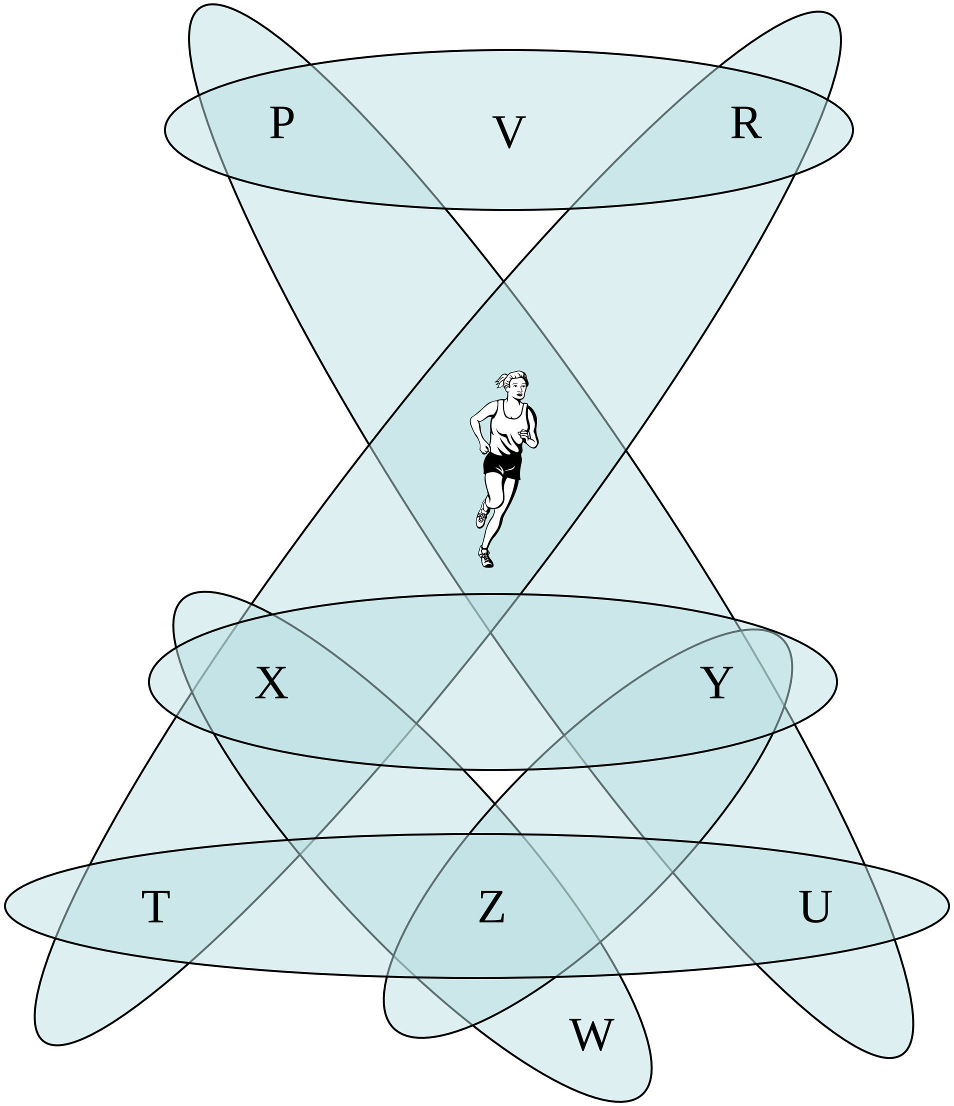
- The marshals have to capture the robber

- The robber tries to elude her capture, by running arbitrarily fast on the vertices of the hypergraph

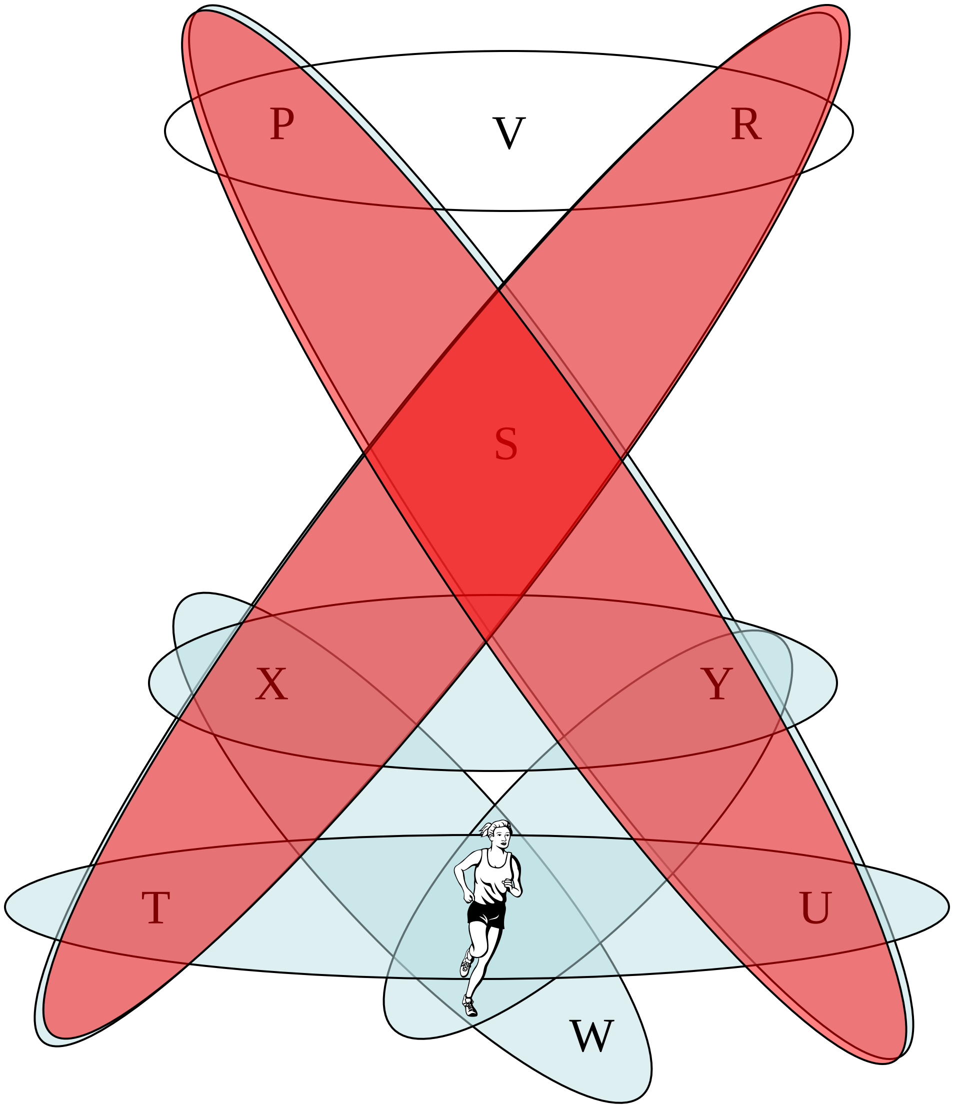
Robbers and Marshals: The Rules

- Each marshal stays on an edge of the hypergraph and controls all of its vertices at once
- The robber can go from a vertex to another vertex running along the edges, but she cannot pass through vertices controlled by some marshal
- The marshals win the game if they are able to monotonically shrink the moving space of the robber, and thus eventually capture her
- Consequently, the robber wins if she can go back to some vertex previously controlled by marshals

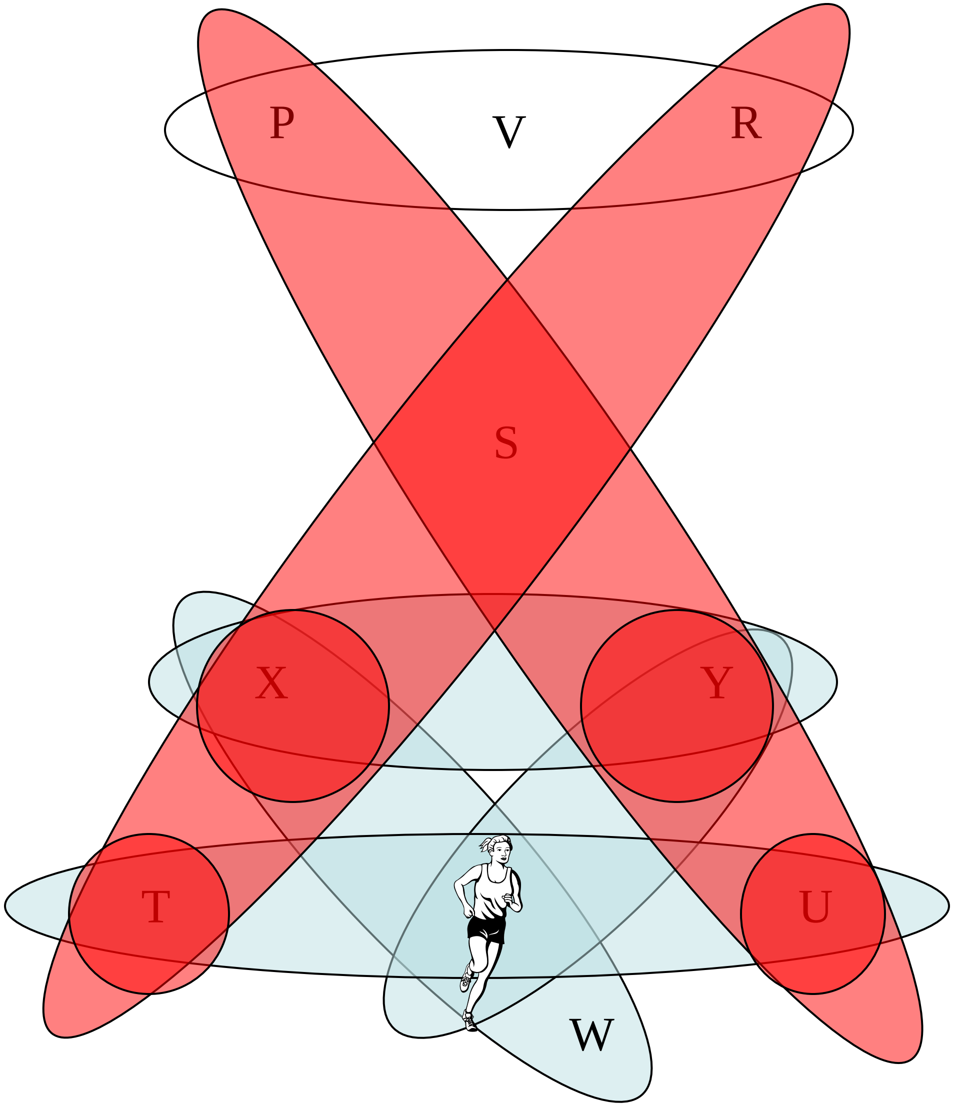
Step 0: the empty hypergraph



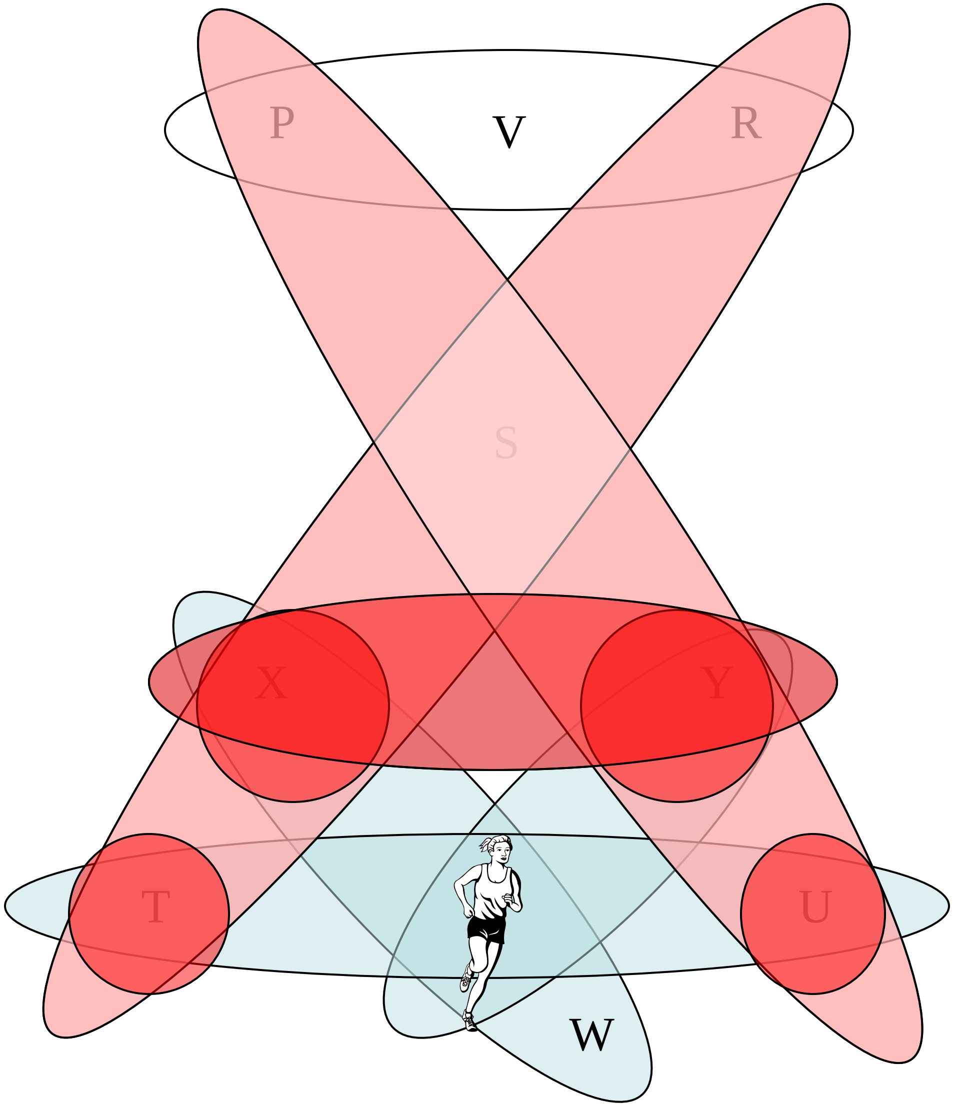
Step 1: first move of the marshals



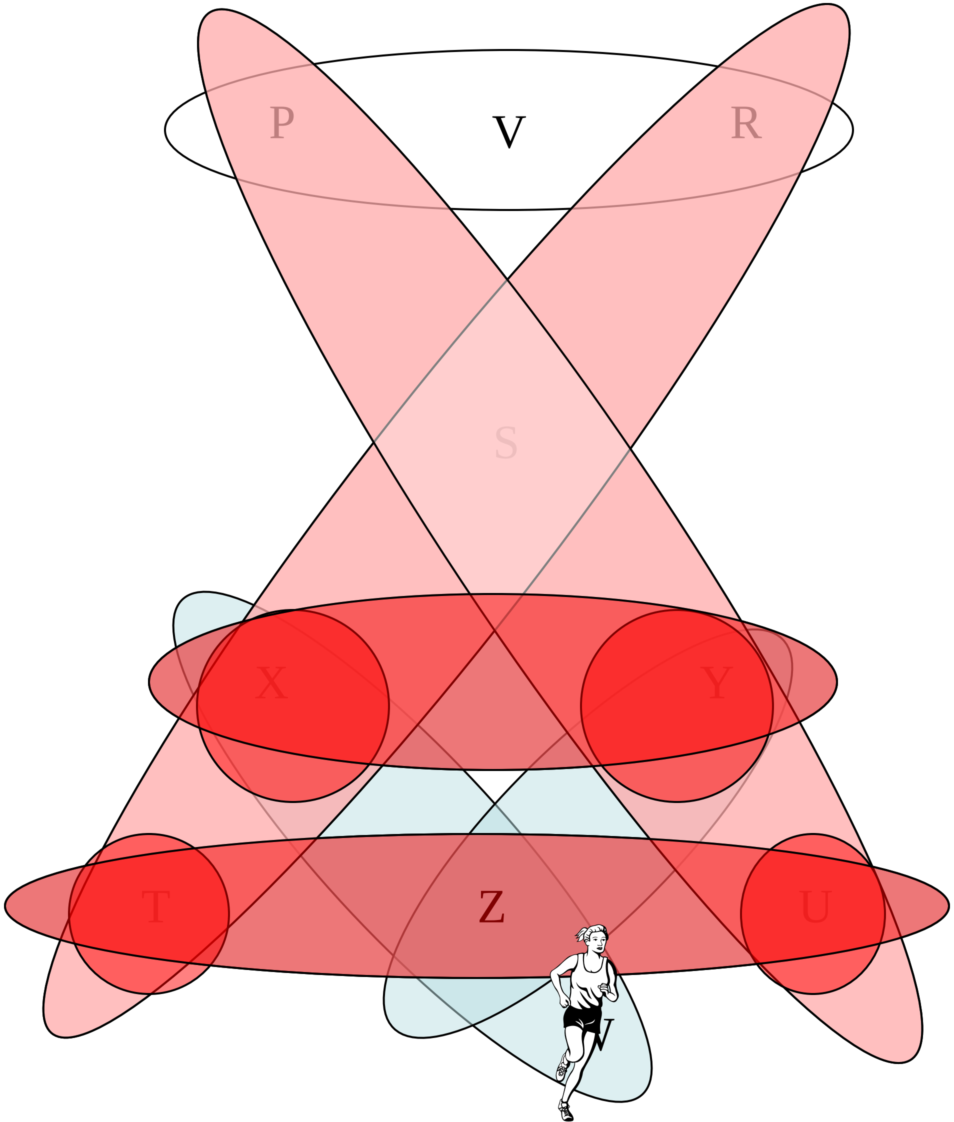
Step 2a: shrinking the space



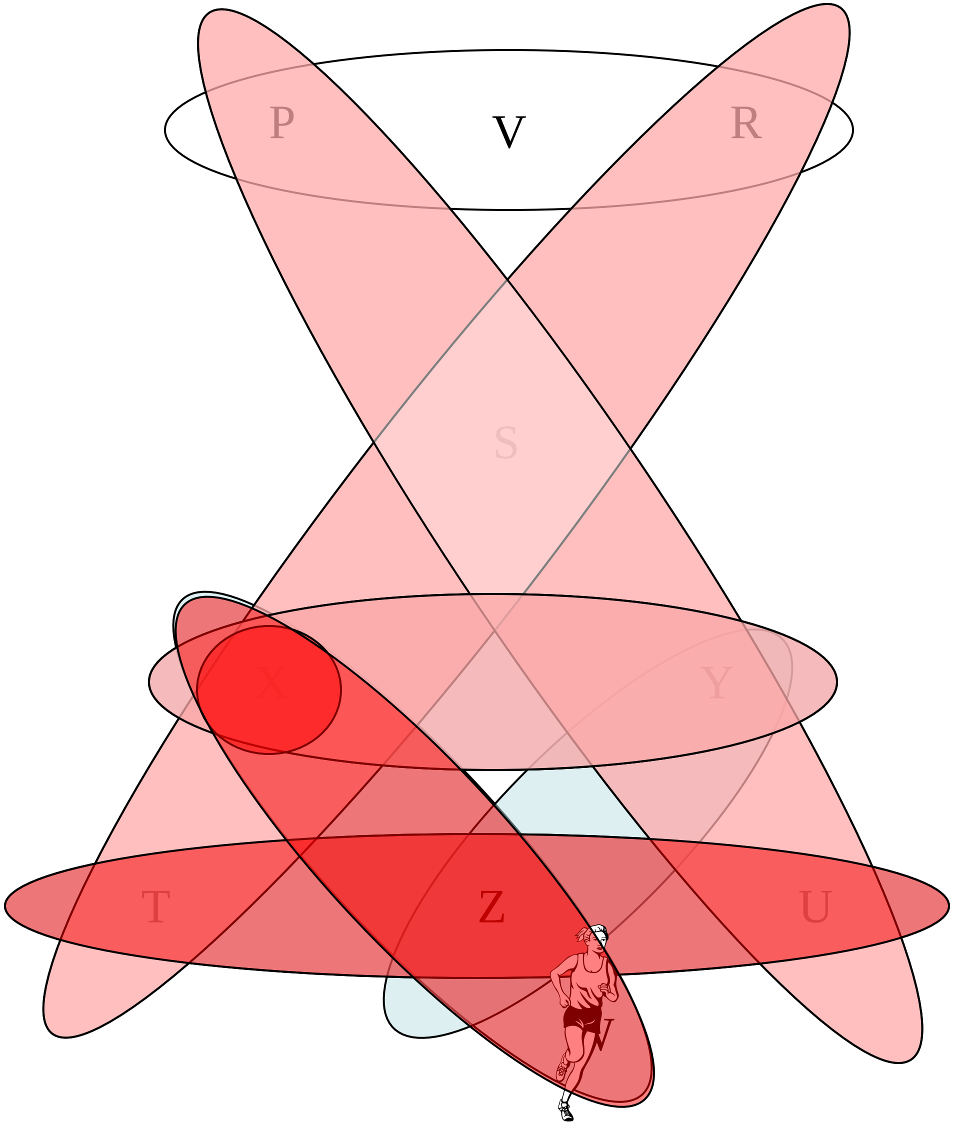
Step 2a: shrinking the space



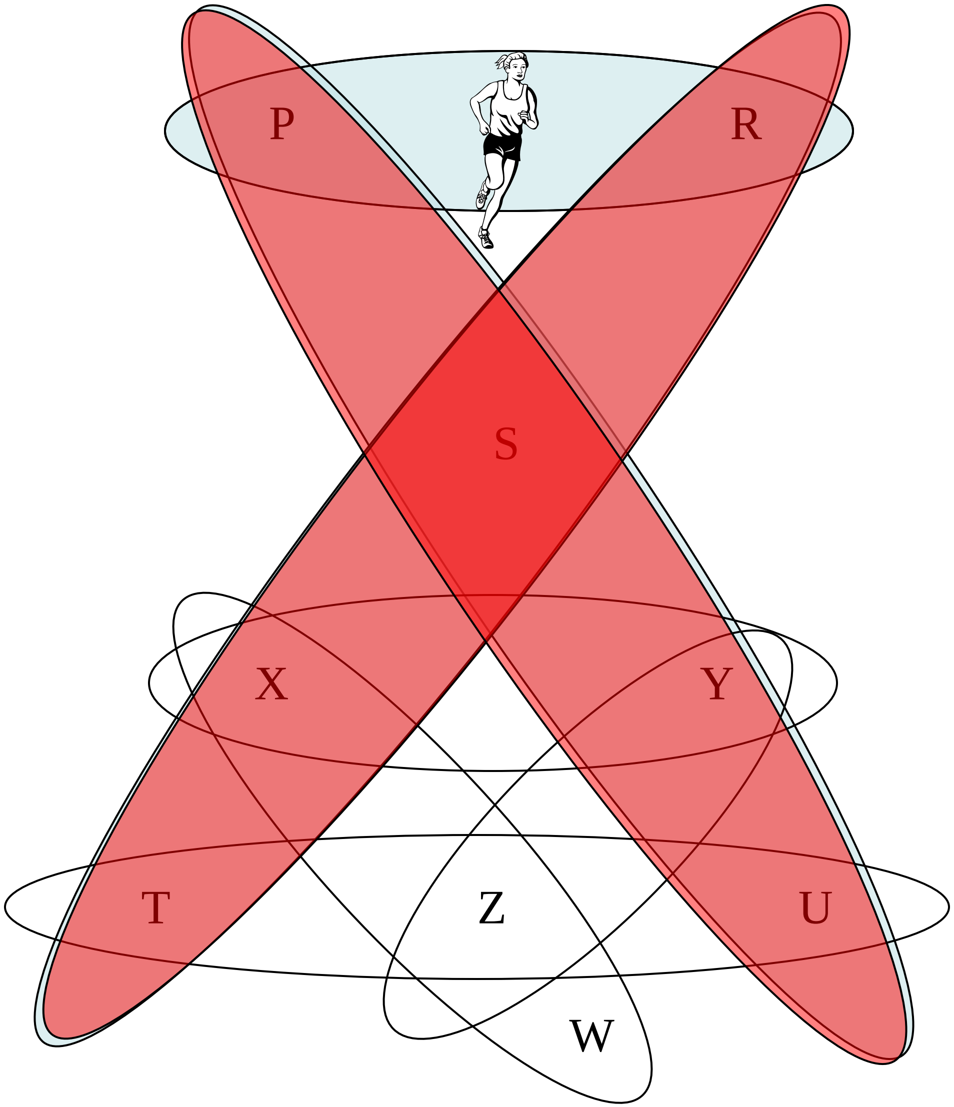
Step 2a: shrinking the space



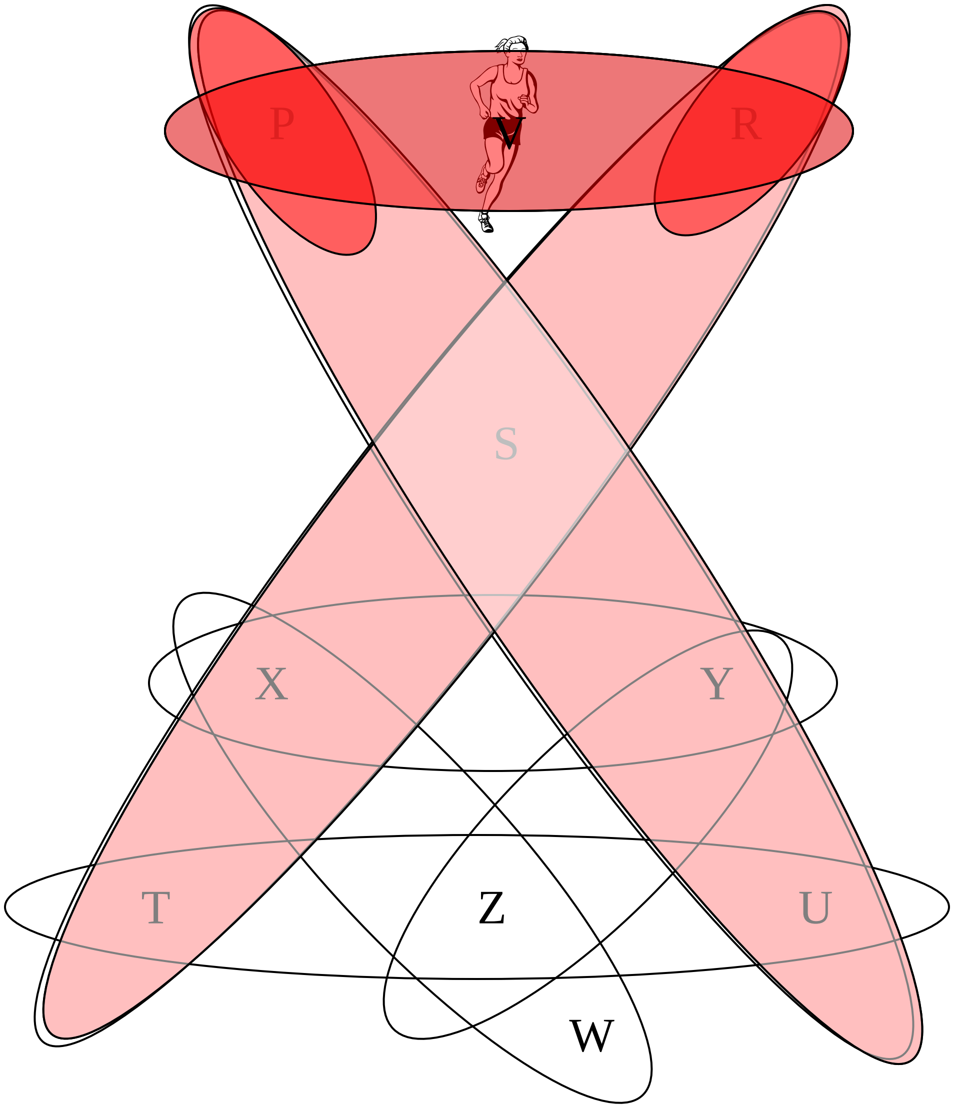
The capture



A different robber's choice

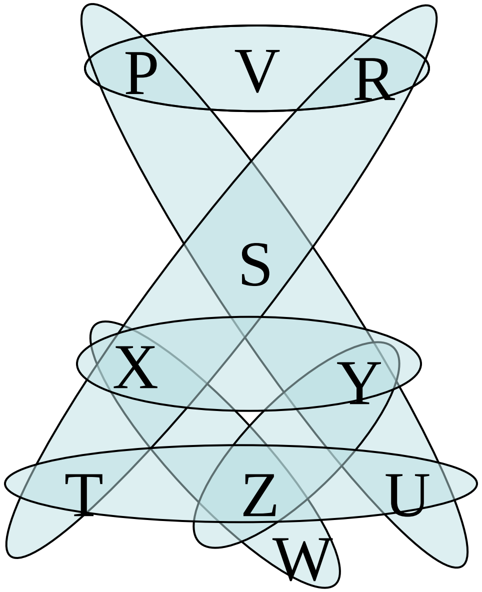


Step 2b: the capture



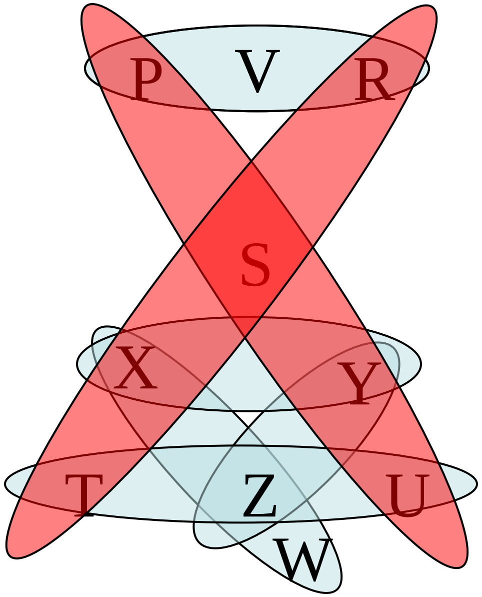
Strategies and Decompositions

$$ans \leftarrow a(S,X,T,R) \wedge b(S,Y,U,P) \wedge c(T,U,Z) \wedge e(Y,Z) \wedge$$
$$g(X,Y) \wedge f(R,P,V) \wedge \wedge d(W,X,Z)$$



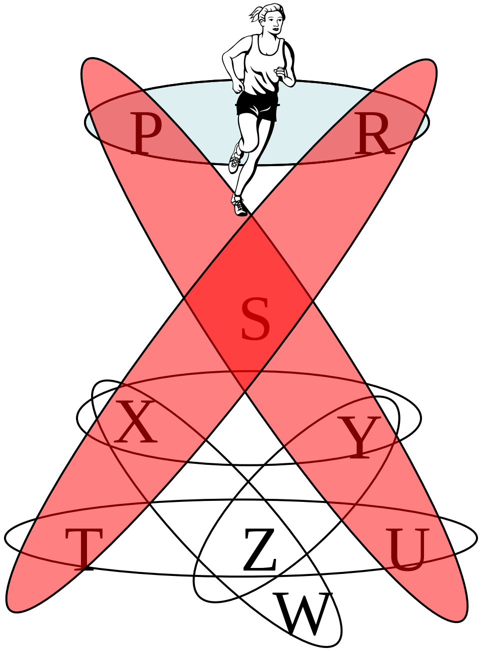
First choice of the two marshals

a(S,X,T,R), **b**(S,Y,U,P)

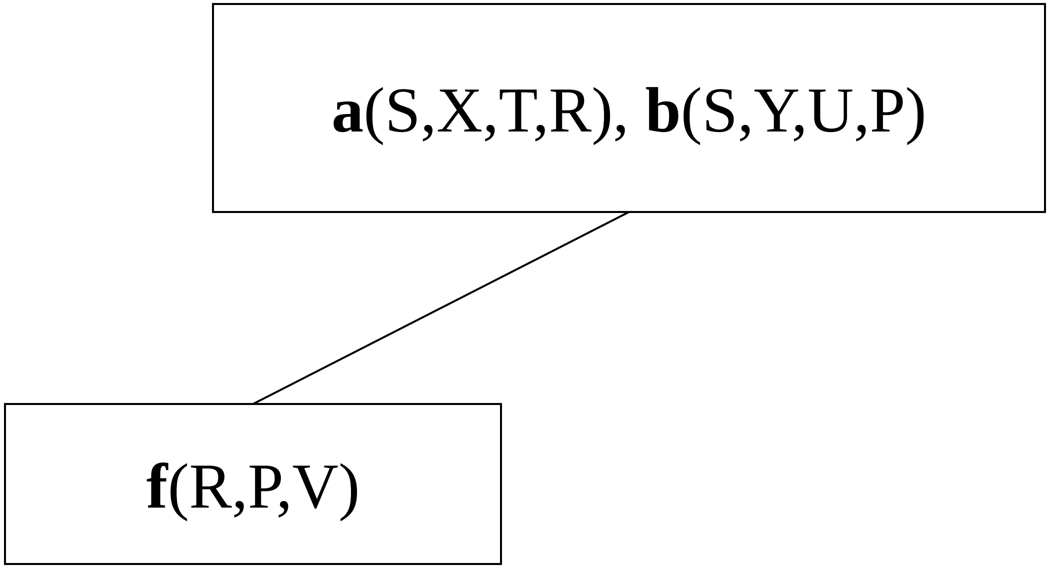


A possible choice for the robber

a(S,X,T,R), **b**(S,Y,U,P)

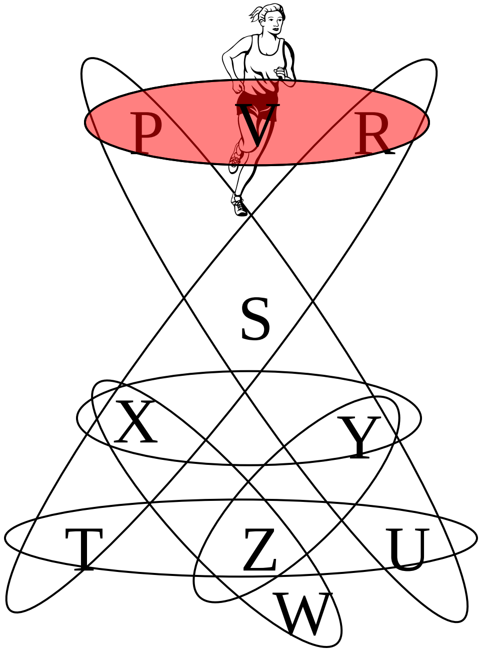


The capture

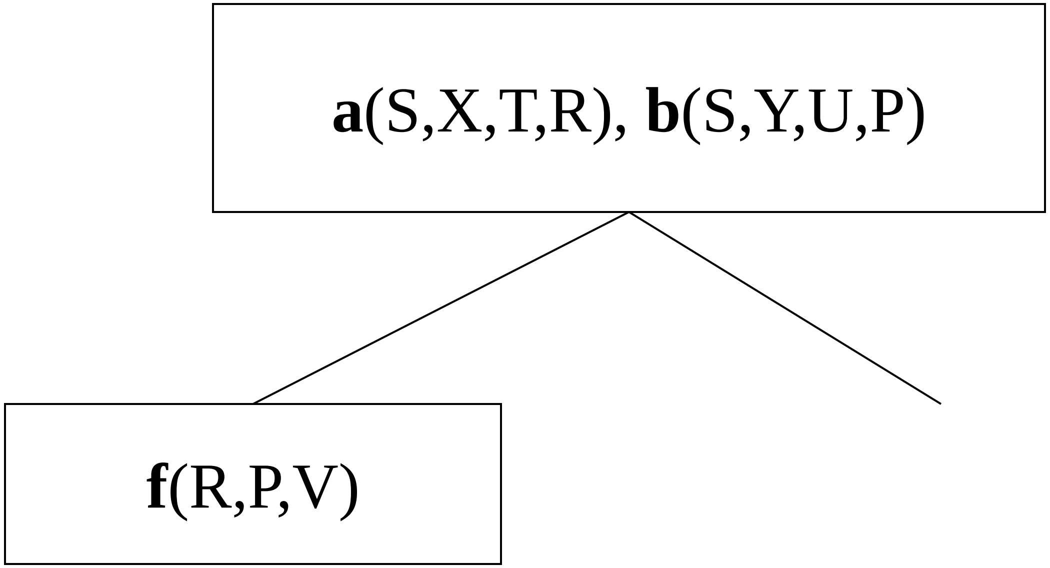


a(S,X,T,R), **b**(S,Y,U,P)

f(R,P,V)

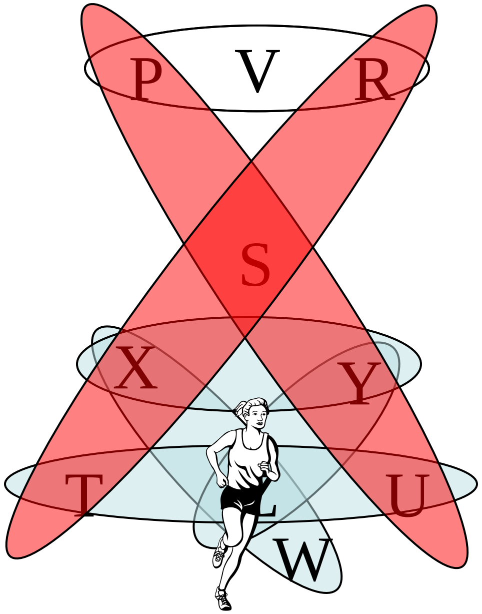


The second choice for the robber

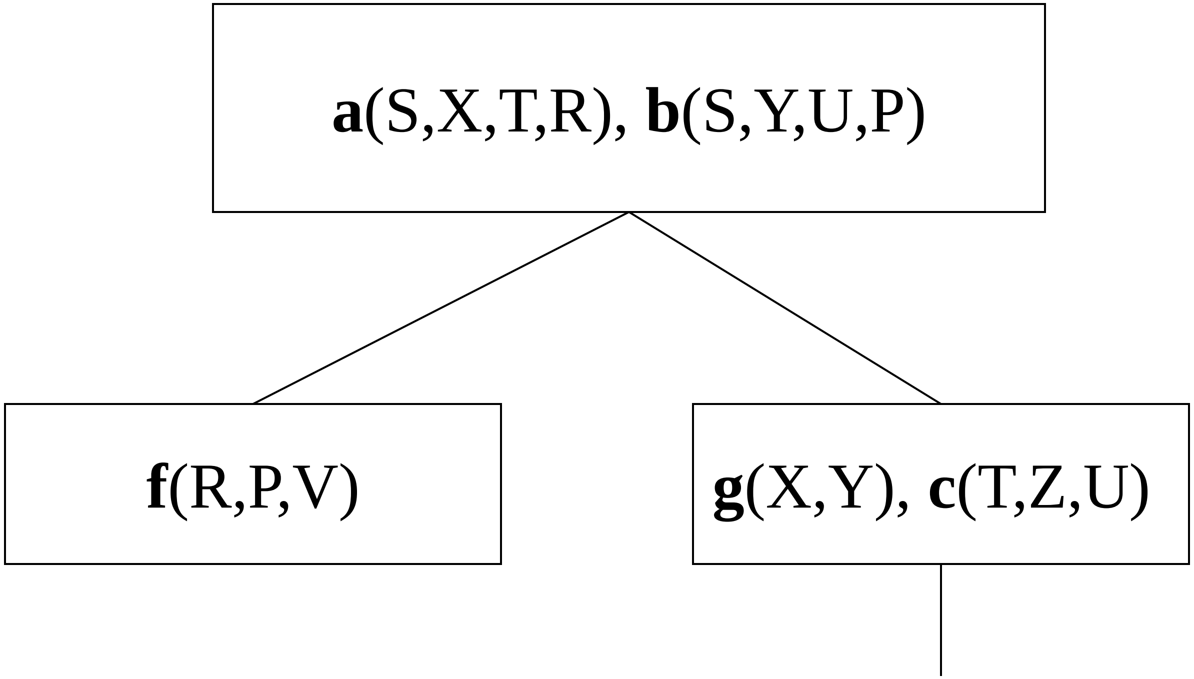


a(S,X,T,R), **b**(S,Y,U,P)

f(R,P,V)



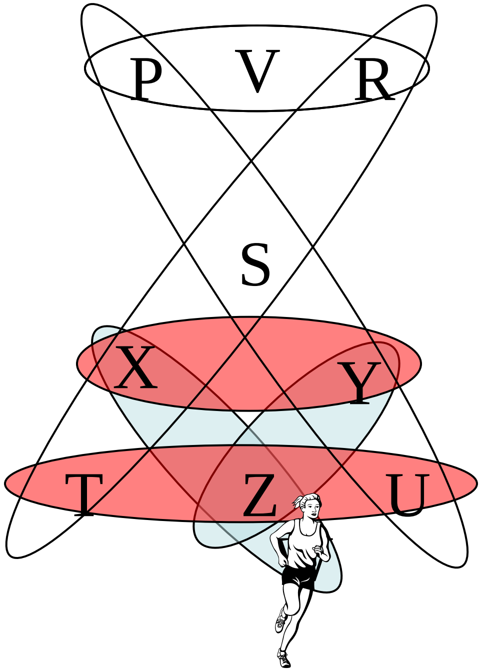
The marshals corner the robber



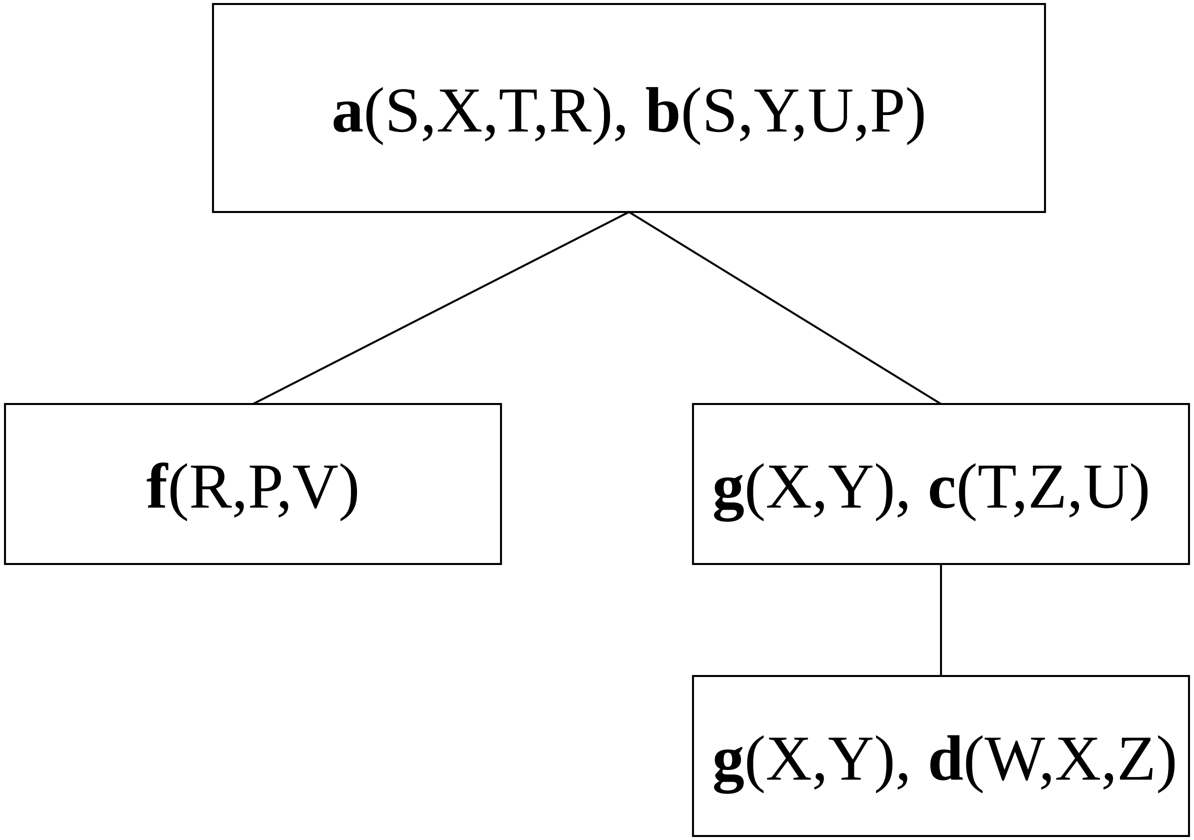
a(S,X,T,R), **b**(S,Y,U,P)

f(R,P,V)

g(X,Y), **c**(T,Z,U)



The capture

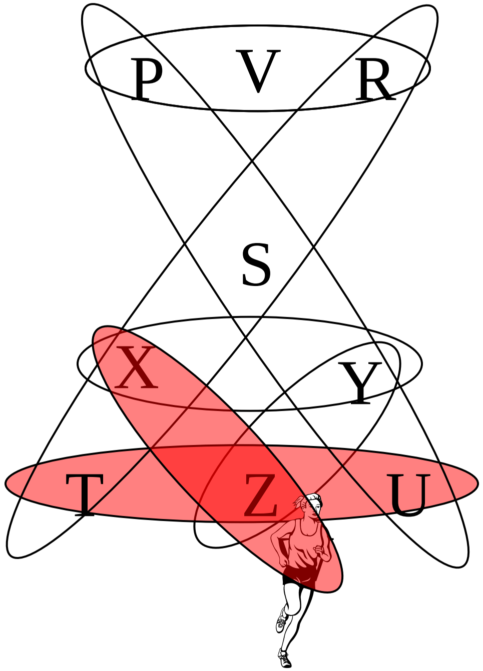


a(S,X,T,R), **b**(S,Y,U,P)

f(R,P,V)

g(X,Y), **c**(T,Z,U)

g(X,Y), **d**(W,X,Z)



R&M Game and Hypertree Width

Let *H* be a hypergraph.

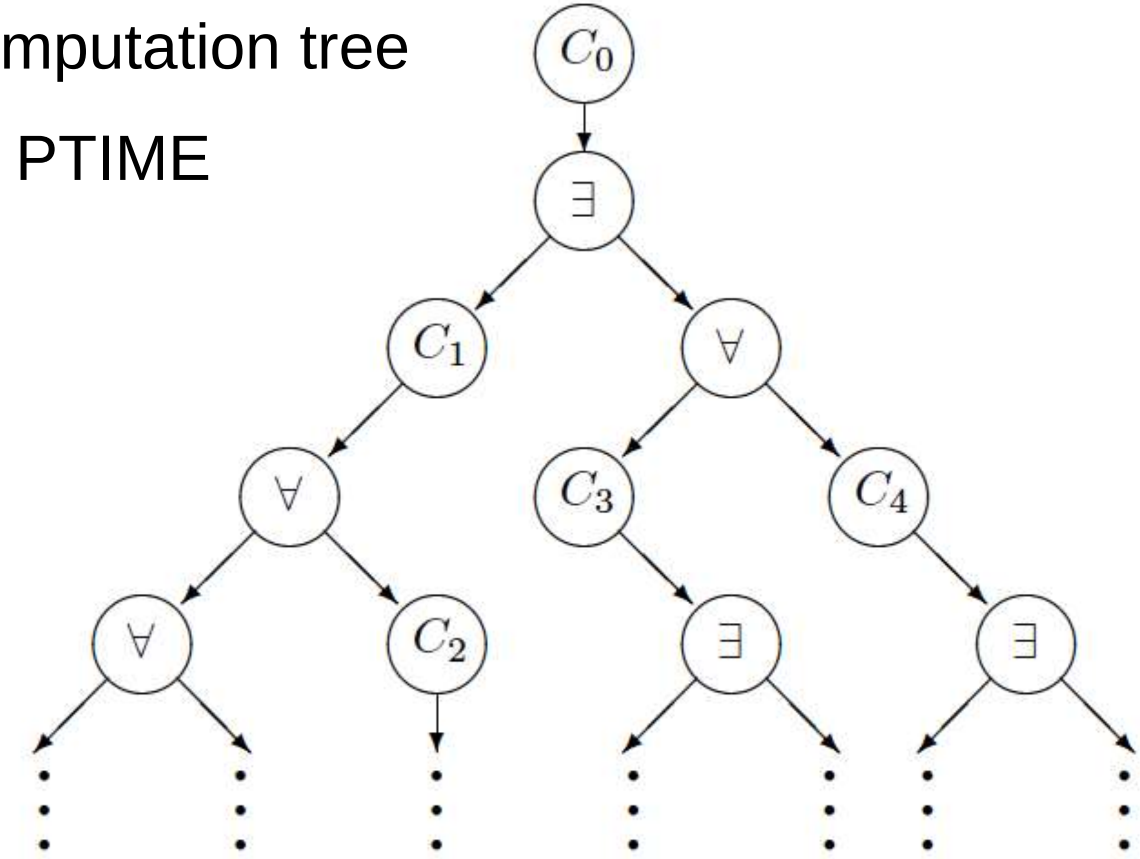
- **Theorem**: *H* has hypertree width ≤ *k* if and only if *k* marshals have a winning strategy on *H*.
- **Corollary**: *H* is acyclic if and only if one marshal has a winning strategy on *H*.

- Winning strategies on H correspond to hypertree decompositions of H and vice versa.

[Gottlob, Leone, Scarcello, PODS'01, JCSS'03]

A Useful Tool: Alternating Turing Machines

- Generalization of non-deterministic Turing machines
- There are two special states: $ and "
- Acceptation: Computation tree
- ALOGSPACE = PTIME



ATMs and LOGCFL

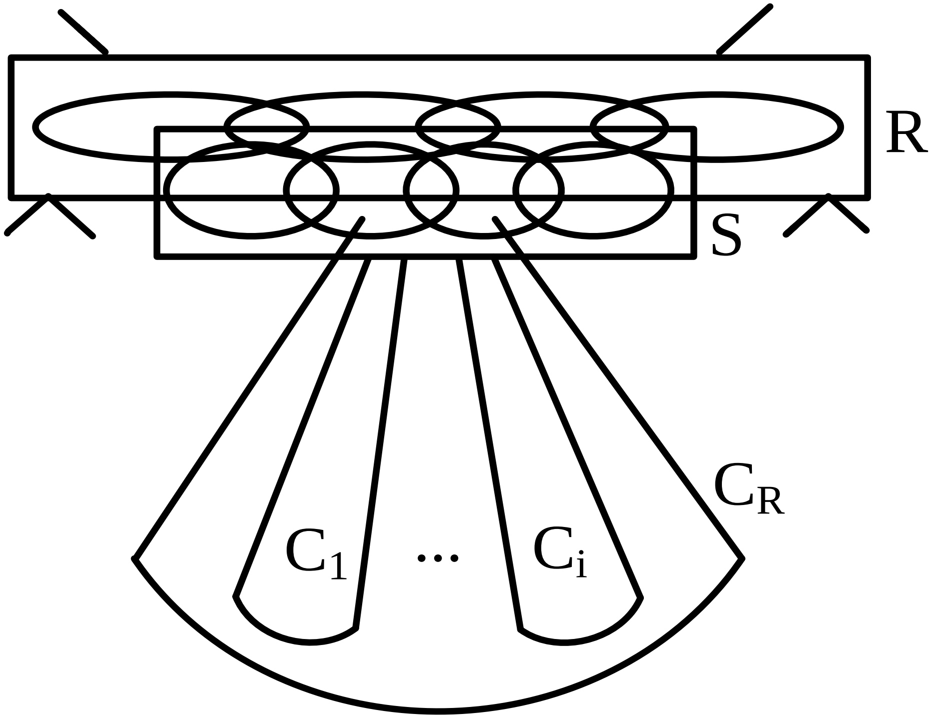
- LOGCFL: class of problems/languages that are logspace-reducible to a CFL
- Admit efficient parallel algorithms

$$\mathrm{AC}_0 \subseteq \mathrm{NL} \subseteq \text{LOGCFL} = \mathrm{SAC}_1 \subseteq \mathrm{AC}_1 \subseteq \mathrm{NC}_2 \subseteq \cdots \subseteq \mathrm{NC} = \mathrm{AC} \subseteq \mathrm{P} \subseteq \mathrm{NP}$$

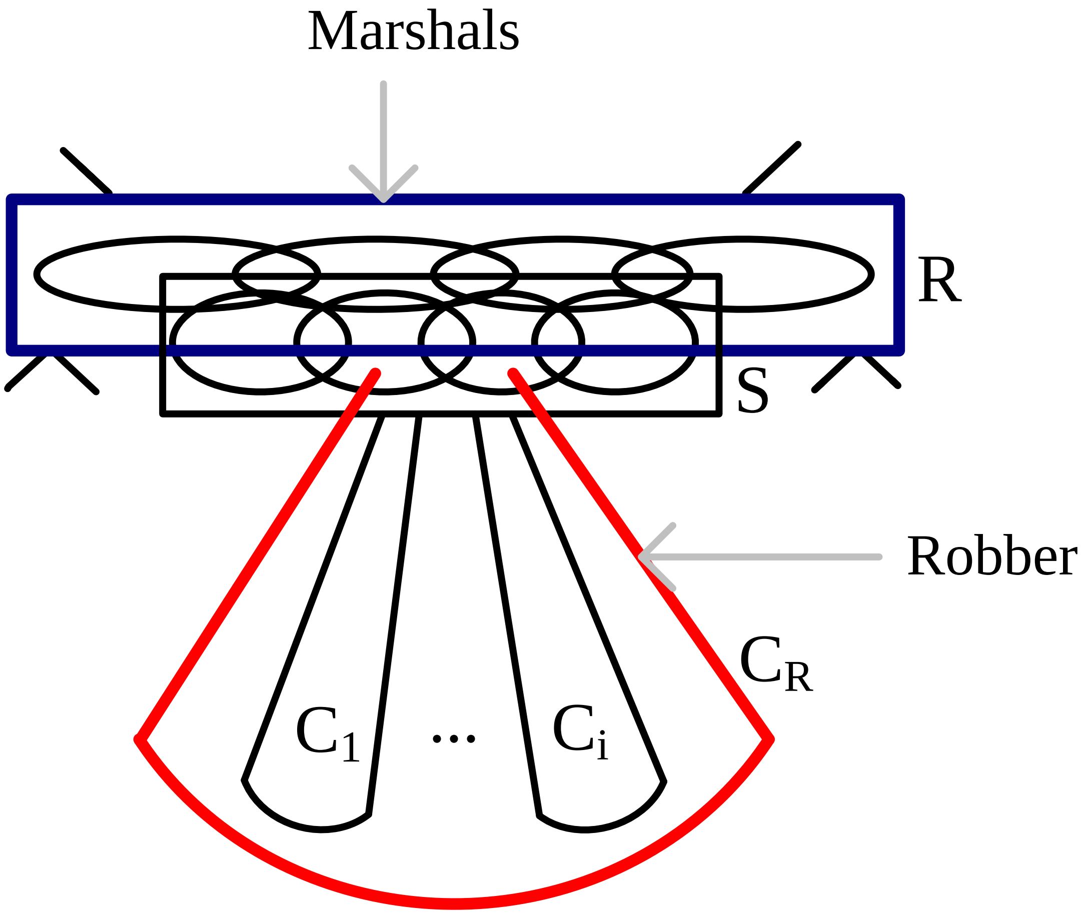
Characterization of LOGCFL [Ruzzo '80]:

LOGCFL = Class of all problems solvable with a logspace ATM
with polynomial tree-size

Coming back to Marshals...

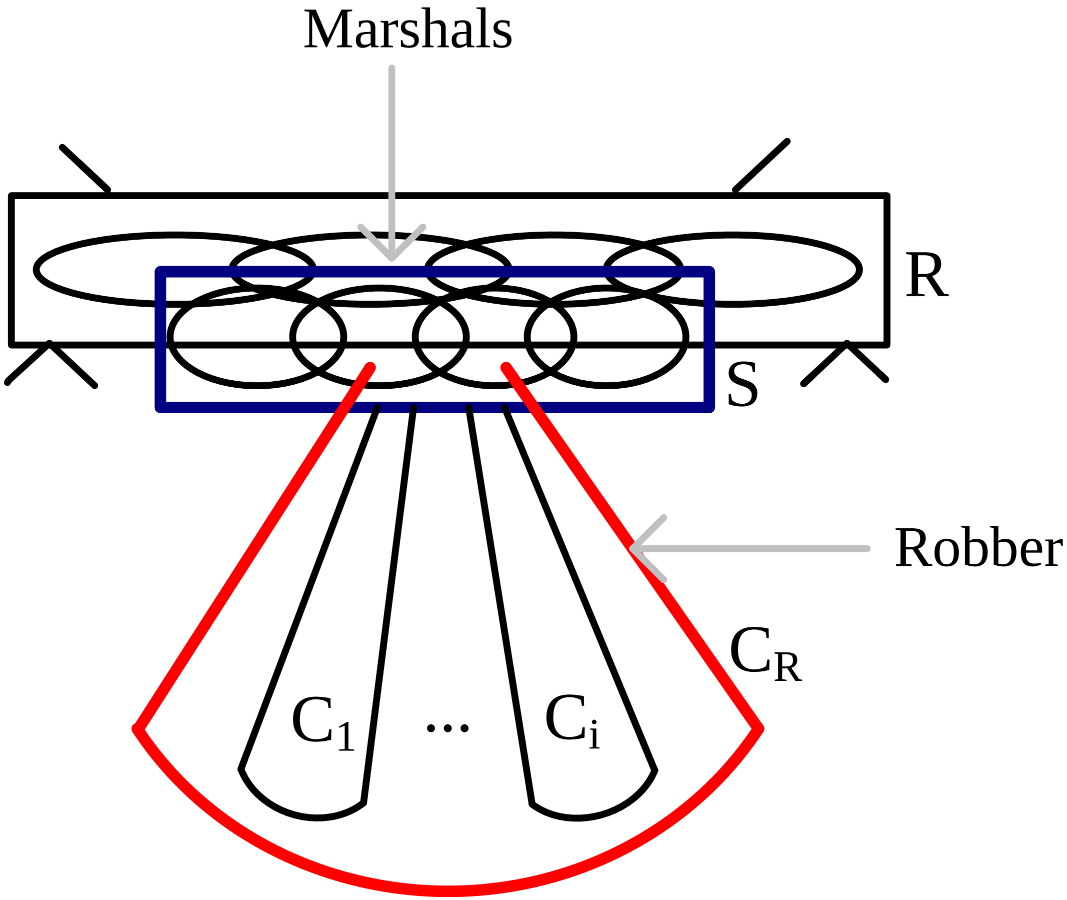


A polynomial algorithm: ALOGSPACE



Actually, LOGCFL

Once I have guessed R, how to guess the next marshal position S ?



Monotonicity: $\forall$ E$\in$ edges(C_R): (E$\cap$UR) $\subseteq$ US

Strict shrinking: (US) $\cap$ $C_R \neq \emptyset$

- LOGSPACE checkable
- Polynomial proof-tree

Outline of PART II

Beyond Tree Decompositions

Applications to Databases and CSPs

Structural and Consistency Properties

Some hypergraph based problems

HOM: The homomorphism problem

BCQ: Boolean conjunctive query evaluation

CSP: Constraint satisfaction problem

Important problems in different areas.
All these problems are hypergraph based.

[e.g., Kolaitis & Vardi, JCSS’98]

The Homomorphism Problem

- Given two relational structures

$$\mathbb{A} = (U, R_1, R_2, \ldots, R_k)$$

$$\mathbb{B} = (V, S_1, S_2, \ldots, S_k)$$

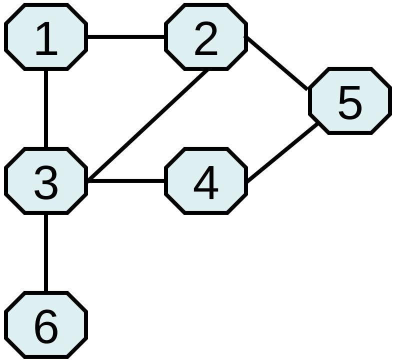
- Decide whether there exists a *homomorphism* ***h*** from $\mathbb{A}$ to $\mathbb{B}$

$$h : U \longrightarrow V$$

$$\text{such that} \quad \forall \mathbf{x}, \forall i$$

$$\mathbf{x} \in R_i \quad \Longrightarrow \quad h(\mathbf{x}) \in S_i$$

Example: graph colorability



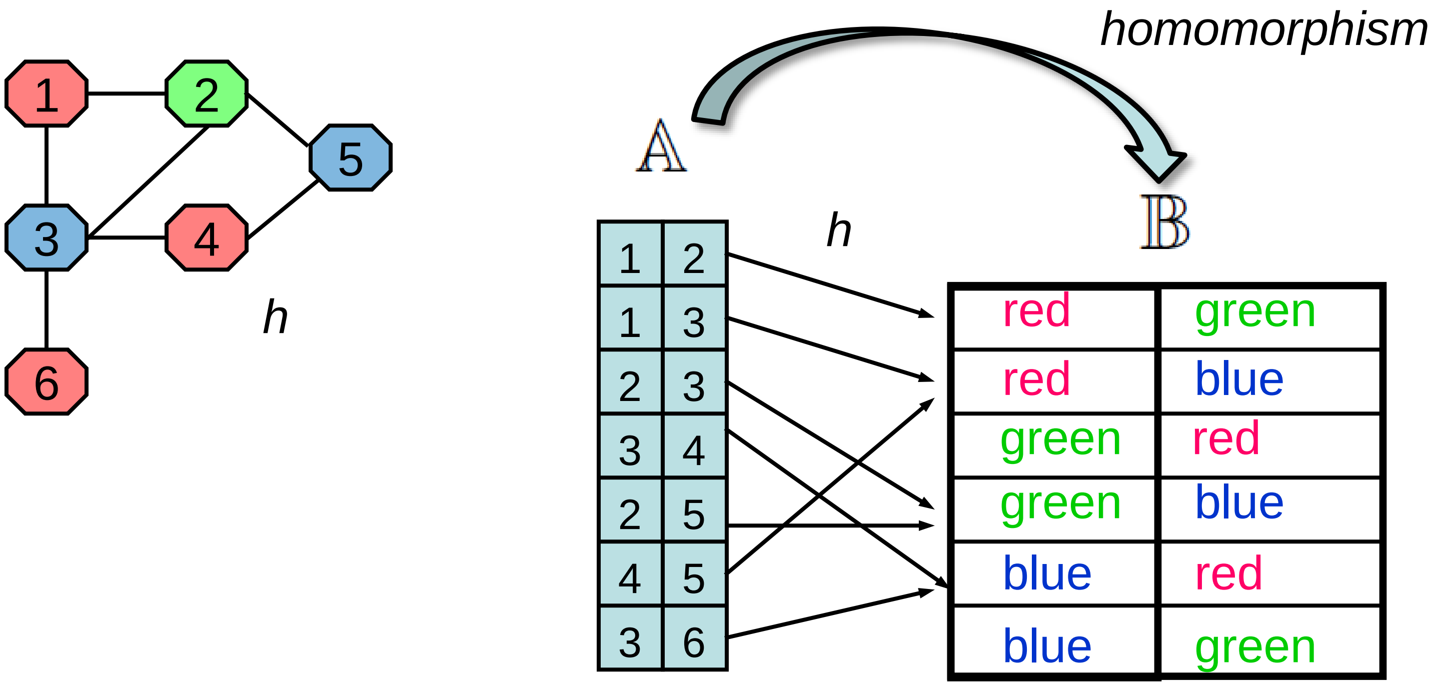
$\mathbb{A}$

1	2
1	3
2	3
3	4
2	5
4	5
3	6

$\mathbb{B}$

red	green
red	blue
green	red
green	blue
blue	red
blue	green

Example: graph colorability



1	2
1	3
2	3
3	4
2	5
4	5
3	6

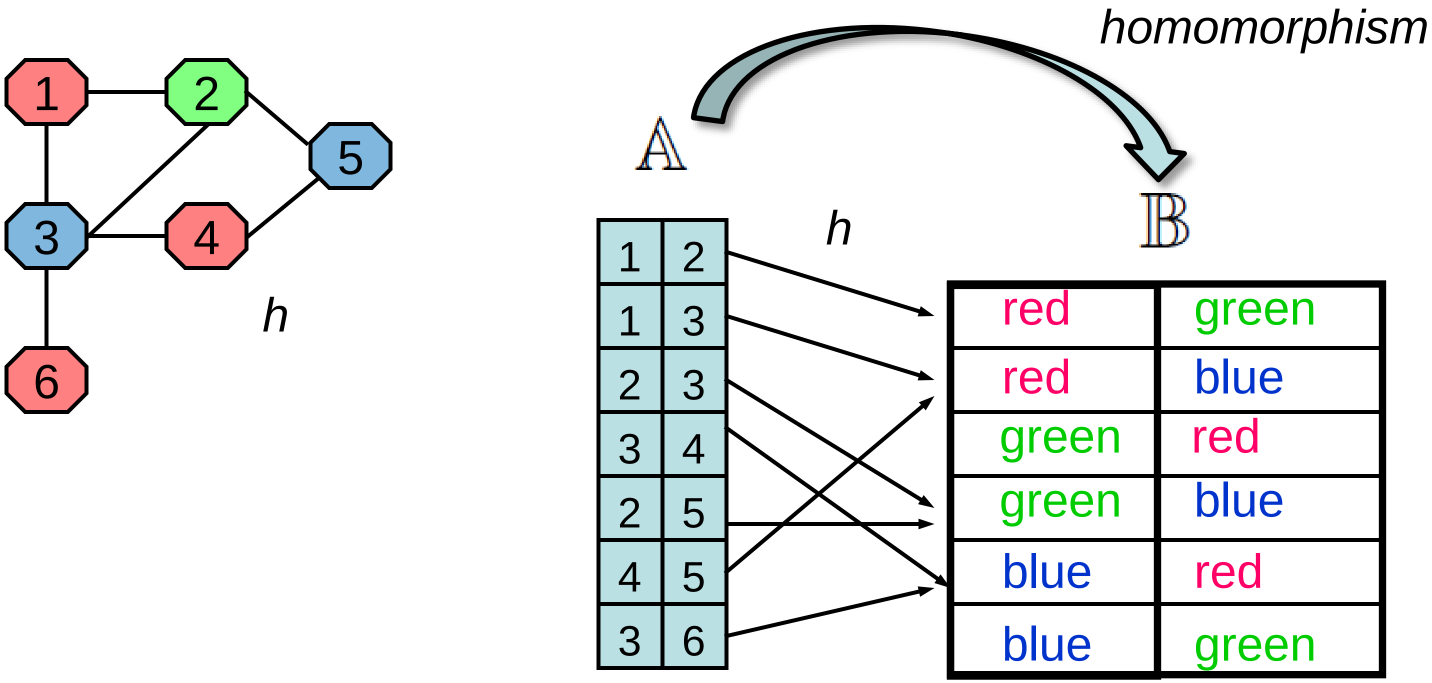
red	green
red	blue
green	red
green	blue
blue	red
blue	green

Complexity: HOM is NP-complete

(well-known, independently proved in various contexts)

Membership: Obvious, guess *h.*

Hardness: Transformation from 3COL.



1	2
1	3
2	3
3	4
2	5
4	5
3	6

red	green
red	blue
green	red
green	blue
blue	red
blue	green

Graph 3-colourable *iff* HOM(*A*,*B*) yes-instance.

Conjunctive Database Queries

DATABASE:

Enrolled

John	Algebra	2003
Robert	Logic	2003
Mary	DB	2002
Lisa	DB	2003
......	.....	....

Teaches

McLane	Algebra	March
Verdi	Logic	May
Lausen	DB	June
Rahm	DB	May
....	.....	....

Parent

McLane	Lisa
Verdi	Robert
Rahm	Mary
....	.....

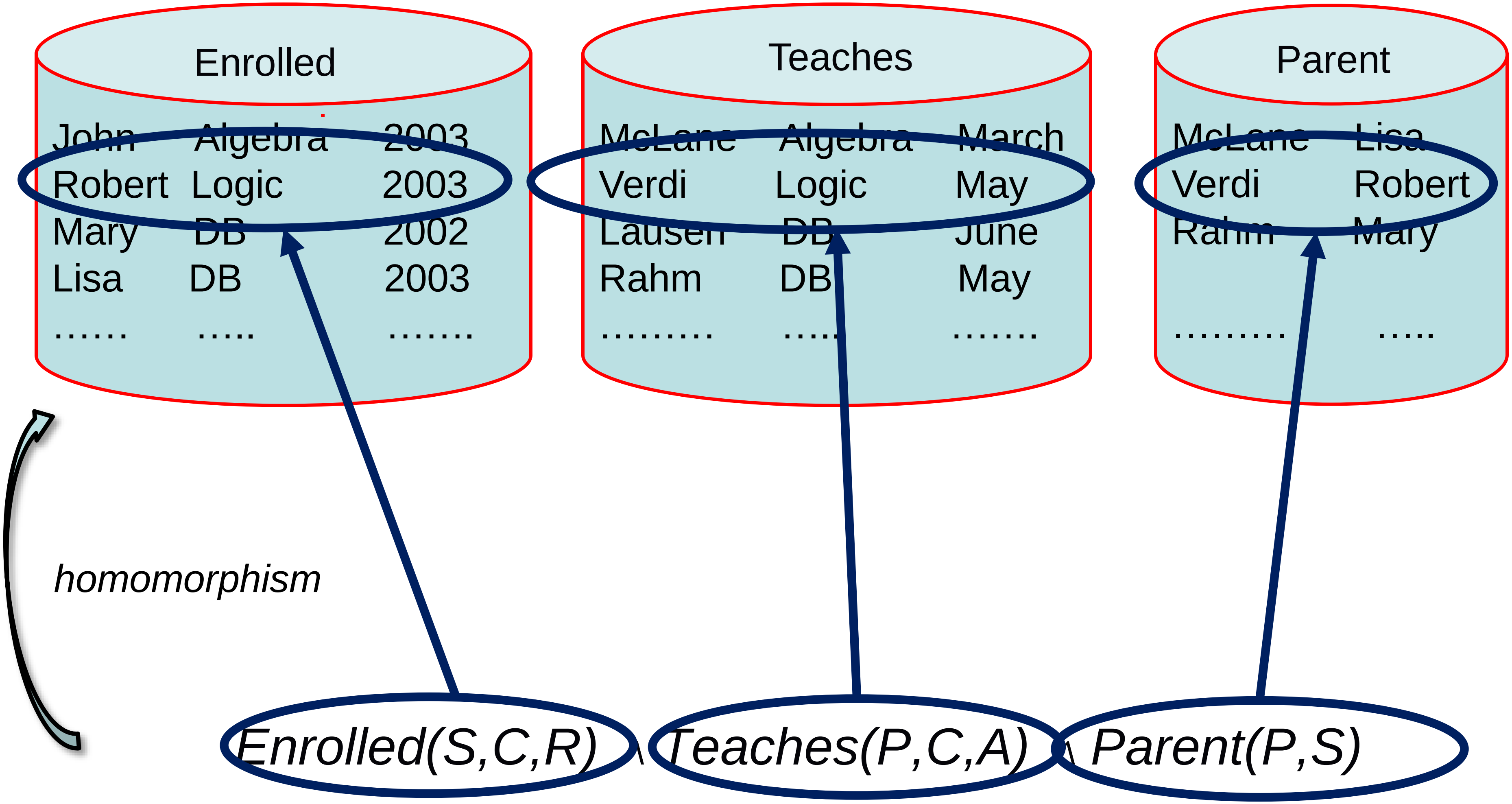
QUERY:

Is there any teacher having a child enrolled in her course?

$$ans \leftarrow Enrolled(S,C,R) \wedge Teaches(P,C,A) \wedge Parent(P,S)$$

Conjunctive Database Queries

DATABASE:



Enrolled

John	Algebra	2003
Robert	Logic	2003
Mary	DB	2002
Lisa	DB	2003
……	…..	…….

Teaches

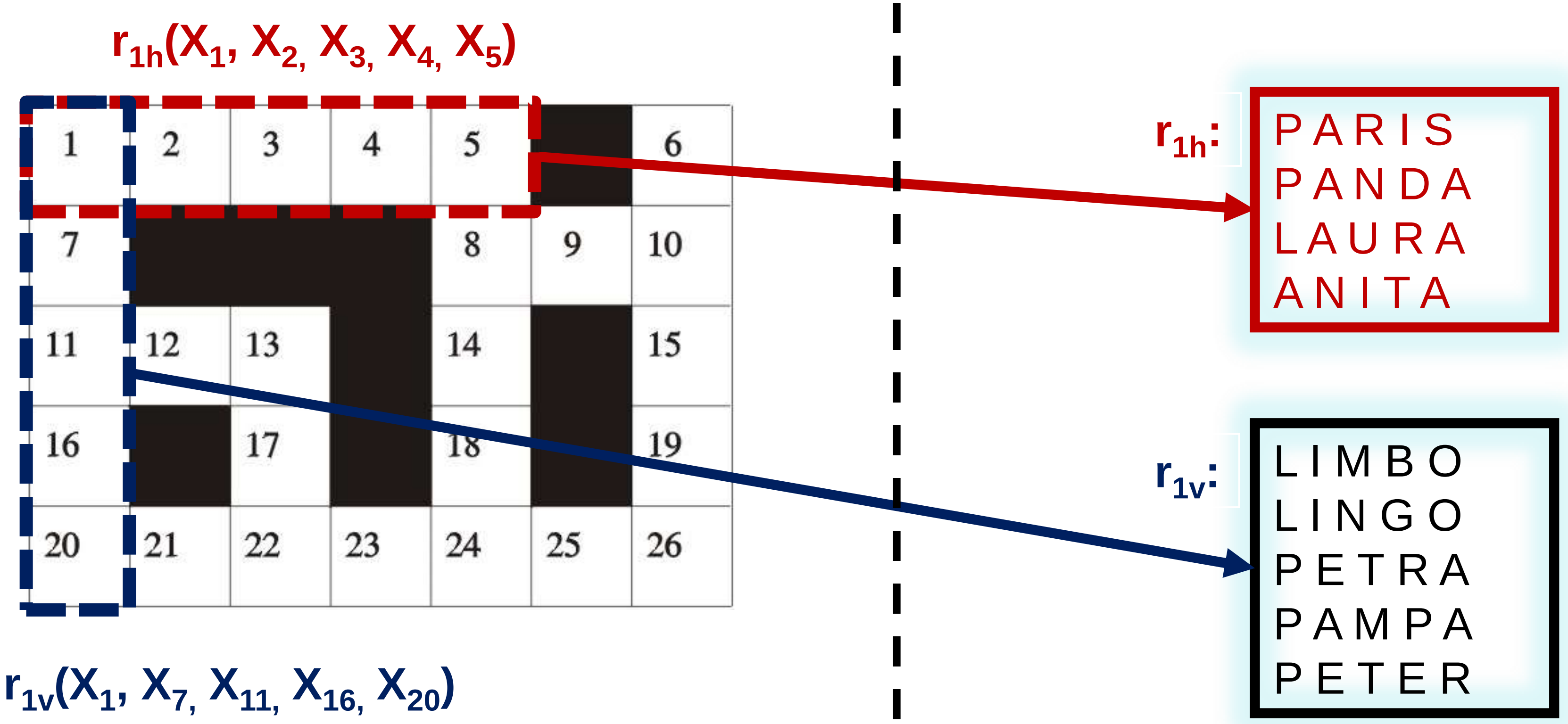
McLane	Algebra	March
Verdi	Logic	May
Lausen	DB	June
Rahm	DB	May
………	…..	…….

Parent

McLane	Lisa
Verdi	Robert
Rahm	Mary
………	…..

Enrolled(S,C,R) ∧ *Teaches(P,C,A)* ∧ *Parent(P,S)*

CSPs as Homomorphism Problems



$r_{1h}(X_1, X_2, X_3, X_4, X_5)$

$r_{1v}(X_1, X_7, X_{11}, X_{16}, X_{20})$

r_{1h}:

PARIS
PANDA
LAURA
ANITA

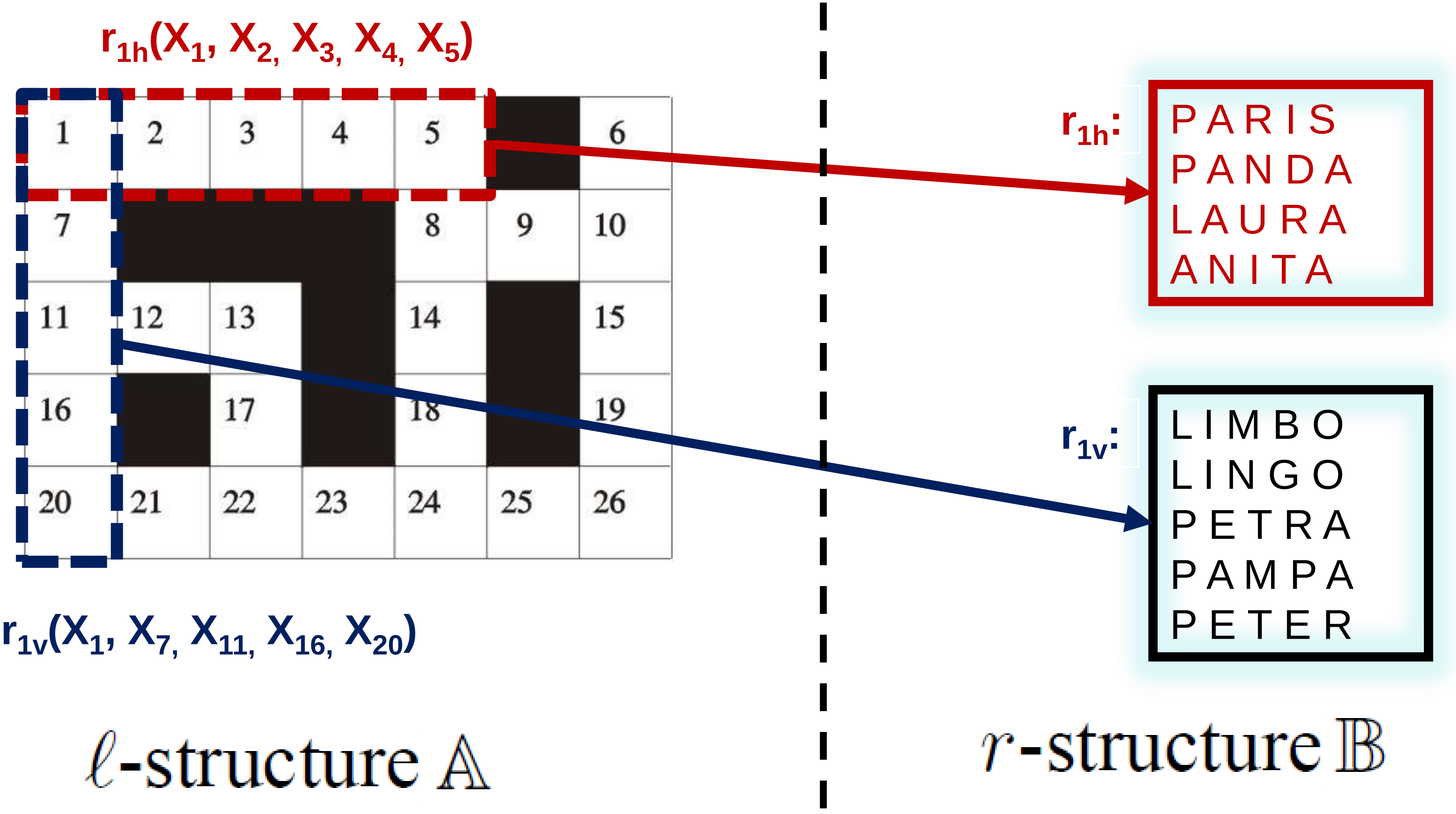
r_{1v}:

LIMBO
LINGO
PETRA
PAMPA
PETER

- Set of variables $\{X_1,\dots,X_{26}\}$
- Set of constraint scopes

- Set of (finite) constraint relations

CSPs as Homomorphism Problems



$r_{1h}(X_1, X_2, X_3, X_4, X_5)$

$r_{1v}(X_1, X_7, X_{11}, X_{16}, X_{20})$

r_{1h}:

PARIS
PANDA
LAURA
ANITA

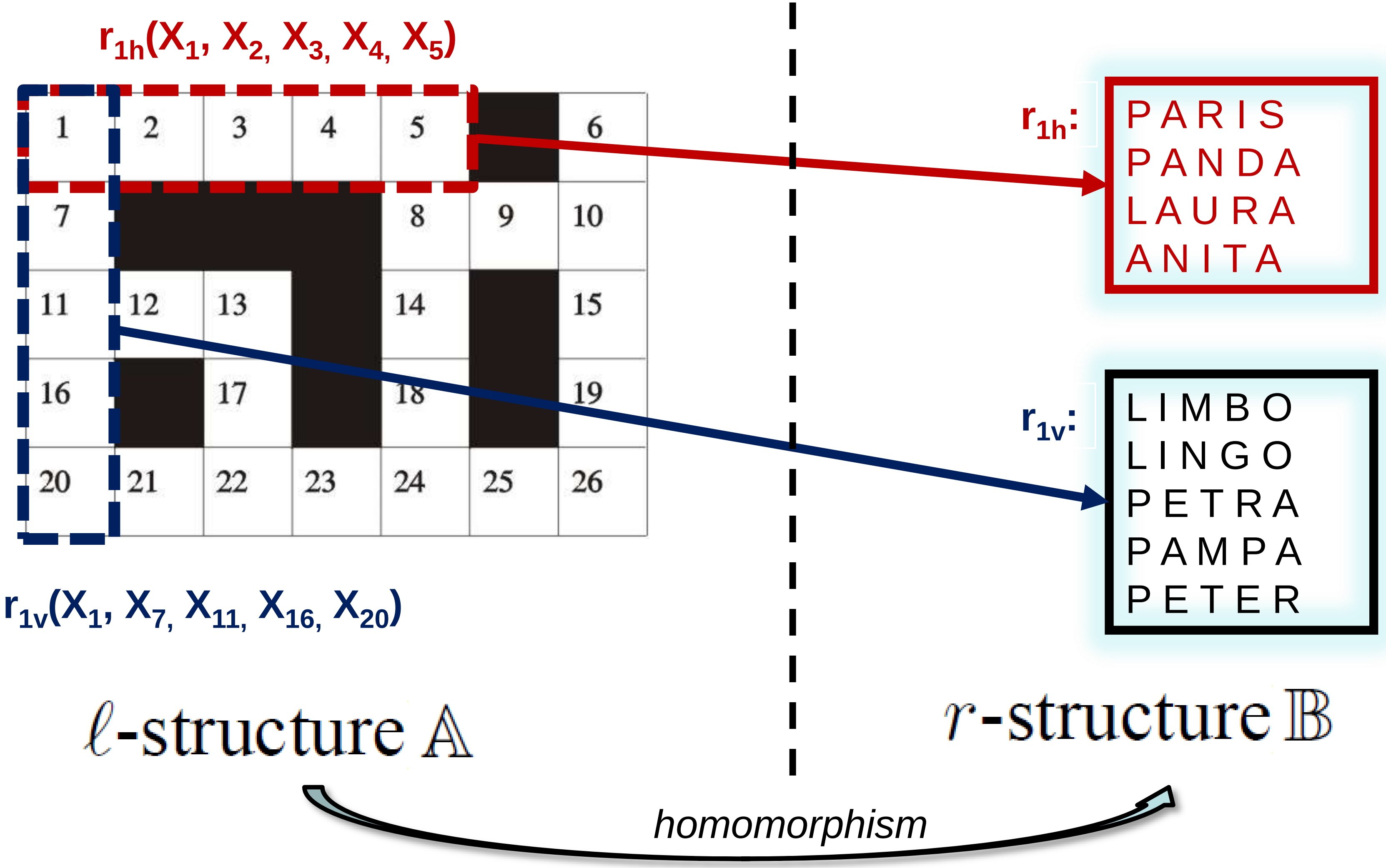
r_{1v}:

LIMBO
LINGO
PETRA
PAMPA
PETER

ℓ-structure $\mathbb{A}$

r-structure $\mathbb{B}$

CSPs as Homomorphism Problems



$r_{1h}(X_1, X_2, X_3, X_4, X_5)$

1	2	3	4	5	■	6
7	■	■	■	8	9	10
11	12	13	■	14	■	15
16	■	17	■	18	■	19
20	21	22	23	24	25	26

$r_{1v}(X_1, X_7, X_{11}, X_{16}, X_{20})$

r_{1h}:

PARIS
PANDA
LAURA
ANITA

r_{1v}:

LIMBO
LINGO
PETRA
PAMPA
PETER

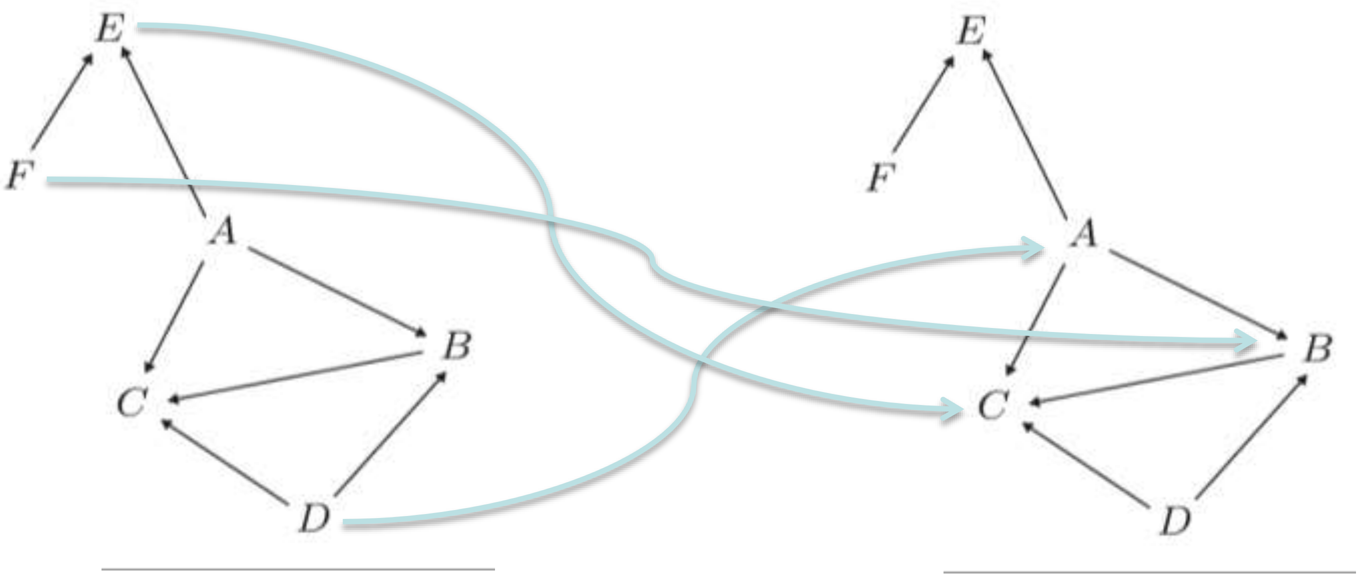
ℓ-structure $\mathbb{A}$

r-structure $\mathbb{B}$

homomorphism

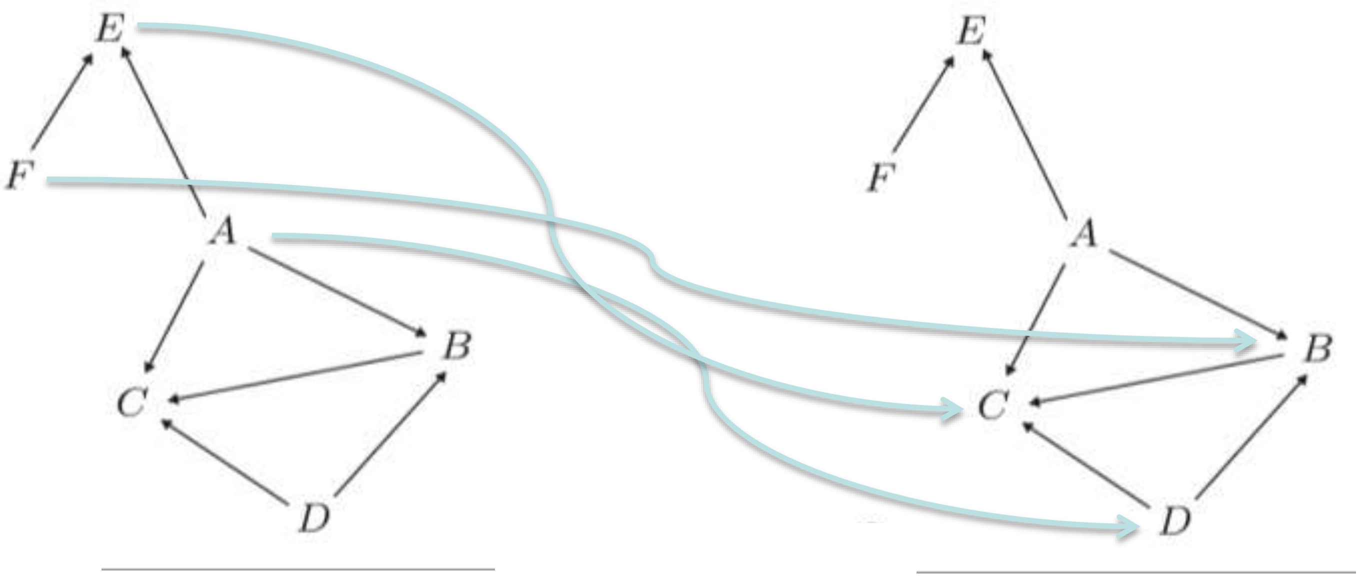
Endomorphisms and cores

- Sometimes the two structures coincide
- Core: minimal substructure to which there is an endomorphism
- Cores are isomorphic to each other



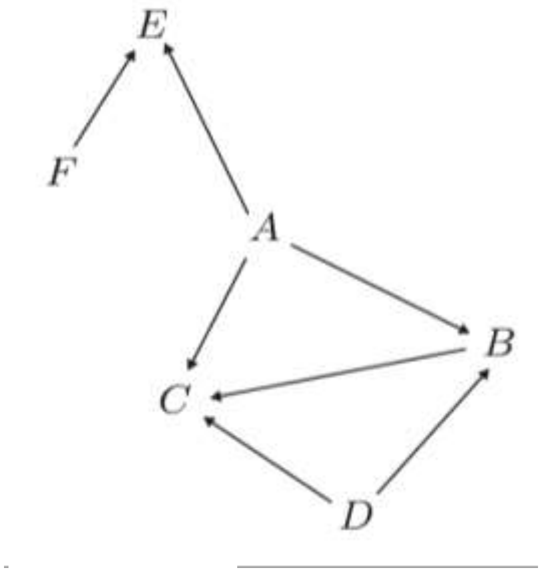
Endomorphisms and cores

- Sometimes the two structures coincide
- Core: minimal substructure to which there is an endomorphism
- Cores are isomorphic to each other

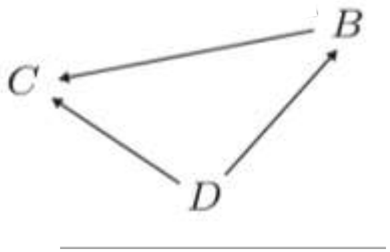


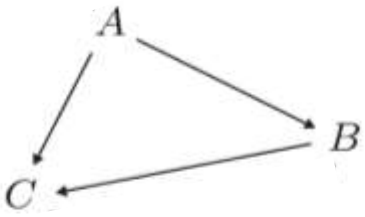
Endomorphisms and cores

- Sometimes the two structures coincide
- Core: minimal substructure to which there is an endomorphism
- Cores are isomorphic to each other



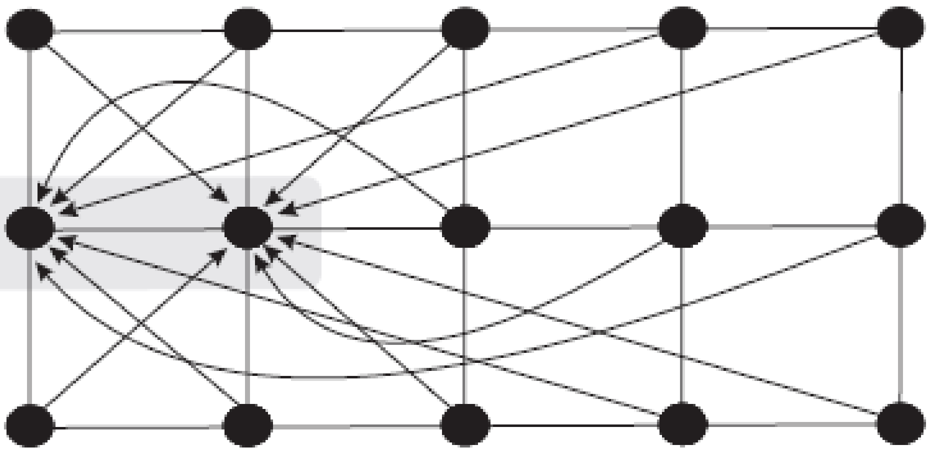
Two isomorphic cores





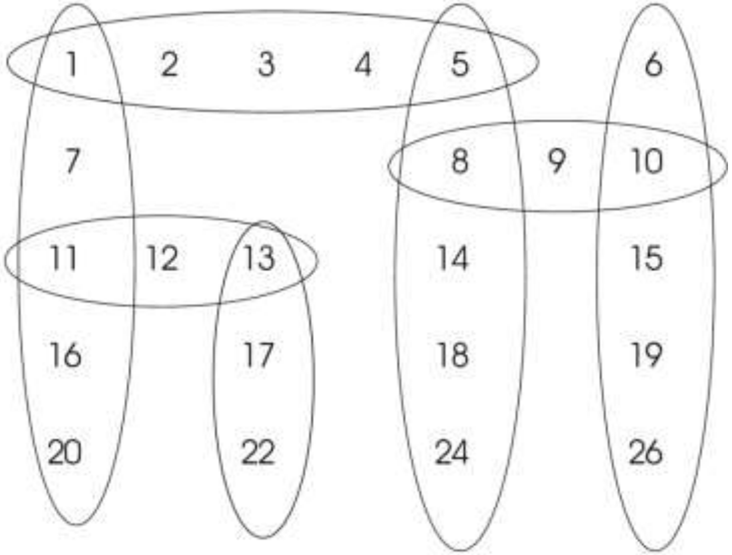
Cores and equivalent instances

- Can be used to simplify problems
- There is a homomorphism from **A** to **B** if and only if there is a homomorphism from a/any core of **A** to **B**
- Sometimes terrific simplifications:



- This undirected grid is equivalent to a single edge. That is, it is equivalent to an acyclic instance!

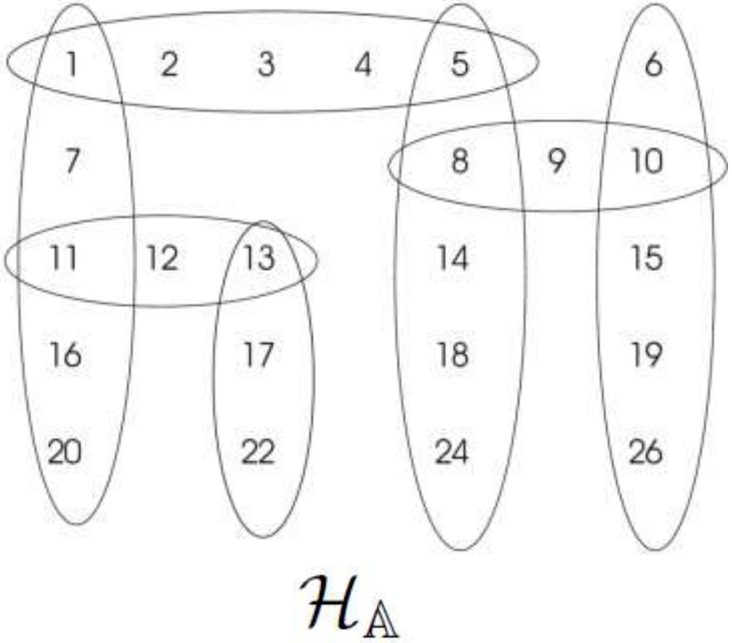
Structurally Restricted CSPs



$\mathcal{H}_{\mathbb{A}}$

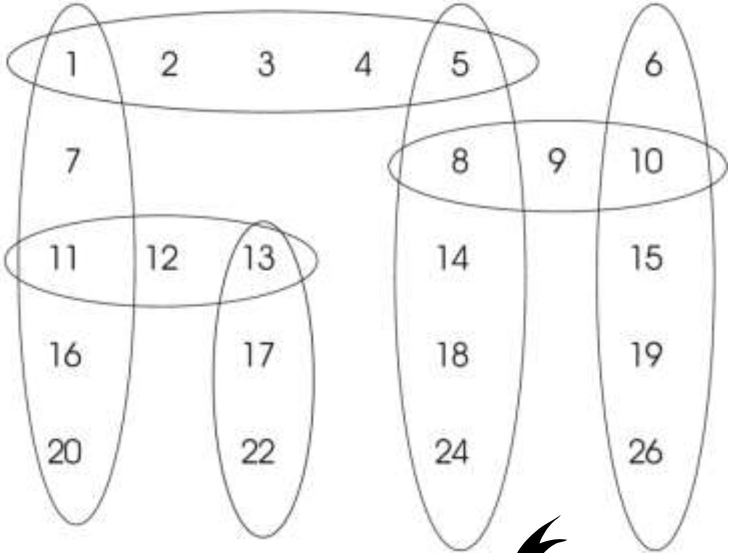
Structurally Restricted CSPs

The hypergraph is acyclic

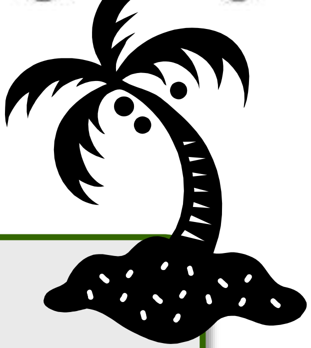


Structurally Restricted CSPs

The hypergraph is acyclic



$\mathcal{H}_{\mathbb{A}}$



- We have seen that *Acyclicity is efficiently recognizable*
- We shall see that *Acyclic CSPs can be efficiently solved*

Basic Question

INPUT: CSP instance $(\mathbb{A}, \mathbb{B})$

- Is there a homomorphism from $\mathbb{A}$ to $\mathbb{B}$?

Basic Question (on Acyclic Instances)

INPUT: CSP instance $(\mathbb{A}, \mathbb{B})$

- Is there a homomorphism from $\mathbb{A}$ to $\mathbb{B}$?

- Feasible in polynomial time $O(||\mathbb{A}||\ ||\mathbb{B}||\ \log||\mathbb{B}||)$
- LOGCFL-complete

Basic Question (on Acyclic Instances)

INPUT: CSP instance $(\mathbb{A}, \mathbb{B})$

- Is there a homomorphism from $\mathbb{A}$ to $\mathbb{B}$?

- Feasible in polynomial time O(||$\mathbb{A}$|| ||$\mathbb{B}$|| log||$\mathbb{B}$||)
- LOGCFL-complete

[Yannakakis, VLDB'81]

Basic Question (on Acyclic Instances)

INPUT: CSP instance $(\mathbb{A}, \mathbb{B})$

- Is there a homomorphism from $\mathbb{A}$ to $\mathbb{B}$?

- Feasible in polynomial time O(||$\mathbb{A}$|| ||$\mathbb{B}$|| log||$\mathbb{B}$||)
- LOGCFL-complete

[Gottlob, Leone, Scarcello, J.ACM’00]

A Polynomial-time Algorithm

HOM: The homomorphism problem

BCQ: Boolean conjunctive query evaluation

CSP: Constraint satisfaction problem

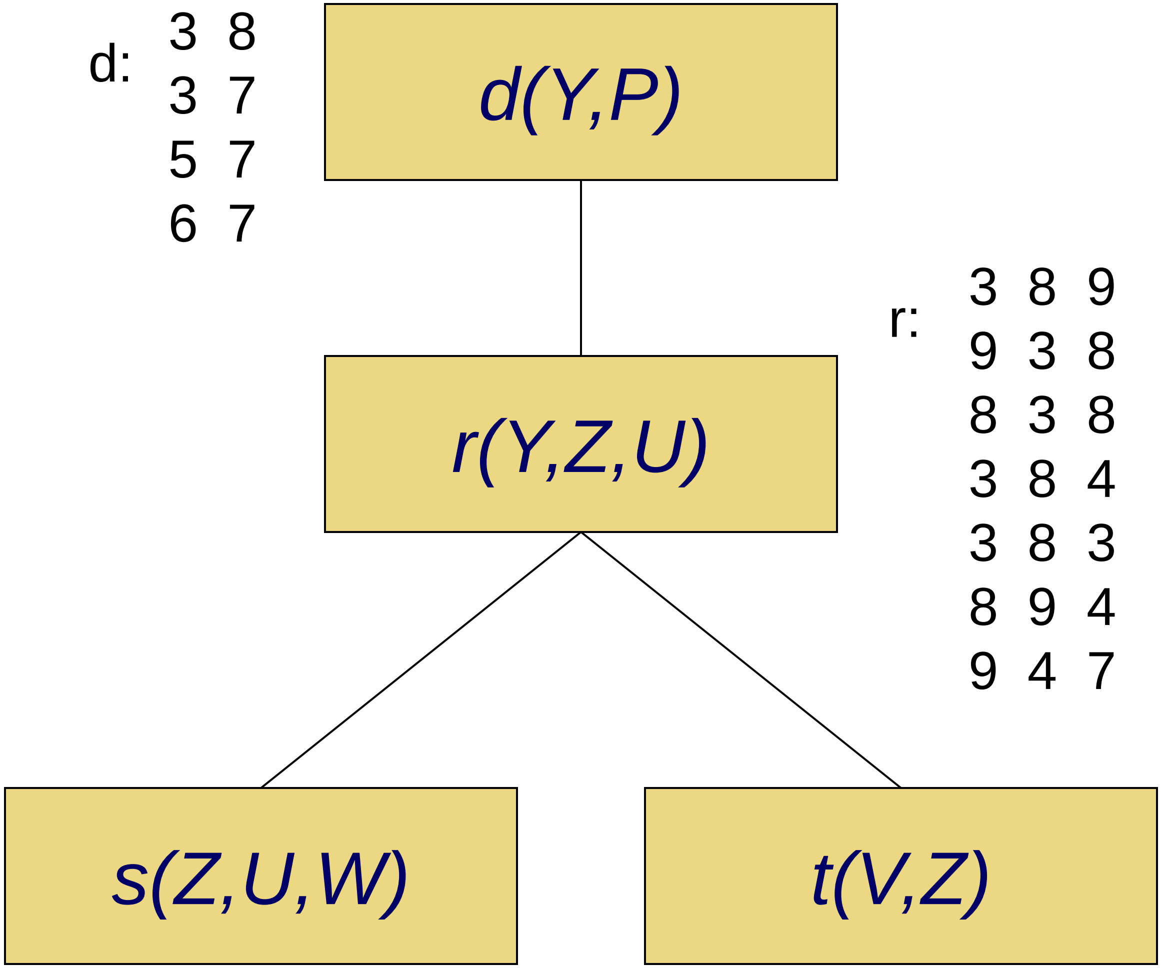
Yannakakis's Algorithm (Acyclic structures):

- Dynamic Programming over a Join Tree,
  where each vertex contains the relation associated
  with the corresponding hyperedge

- Therefore, if there are more constraints over the
  same relation, it may occur (as a copy) at different
  vertices

d:

3	8
3	7
5	7
6	7



d(Y,P)

r(Y,Z,U)

s(Z,U,W)

t(V,Z)

r:

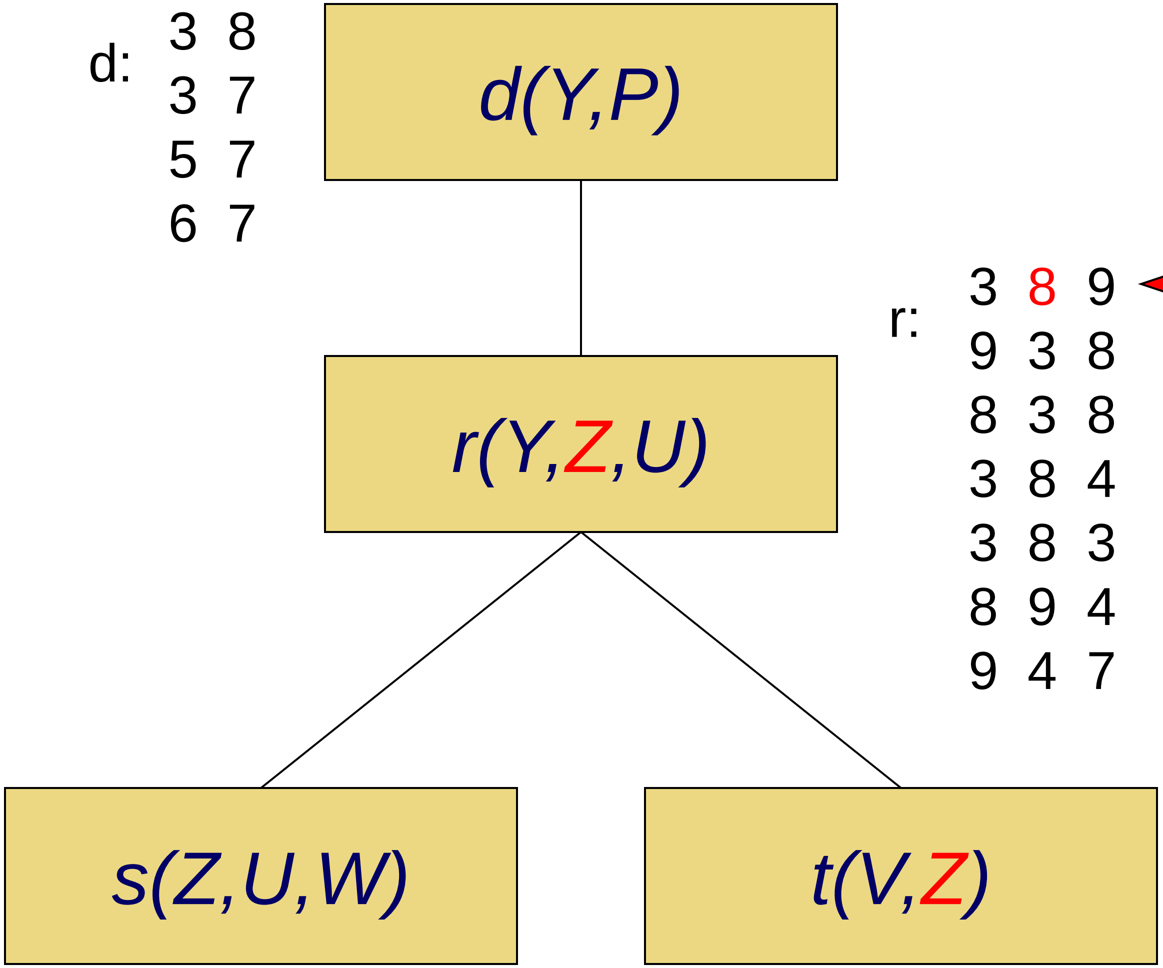
3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

s:

3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

t:

9	8
9	3
9	5



d:

3	8
3	7
5	7
6	7

r:

3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

s:

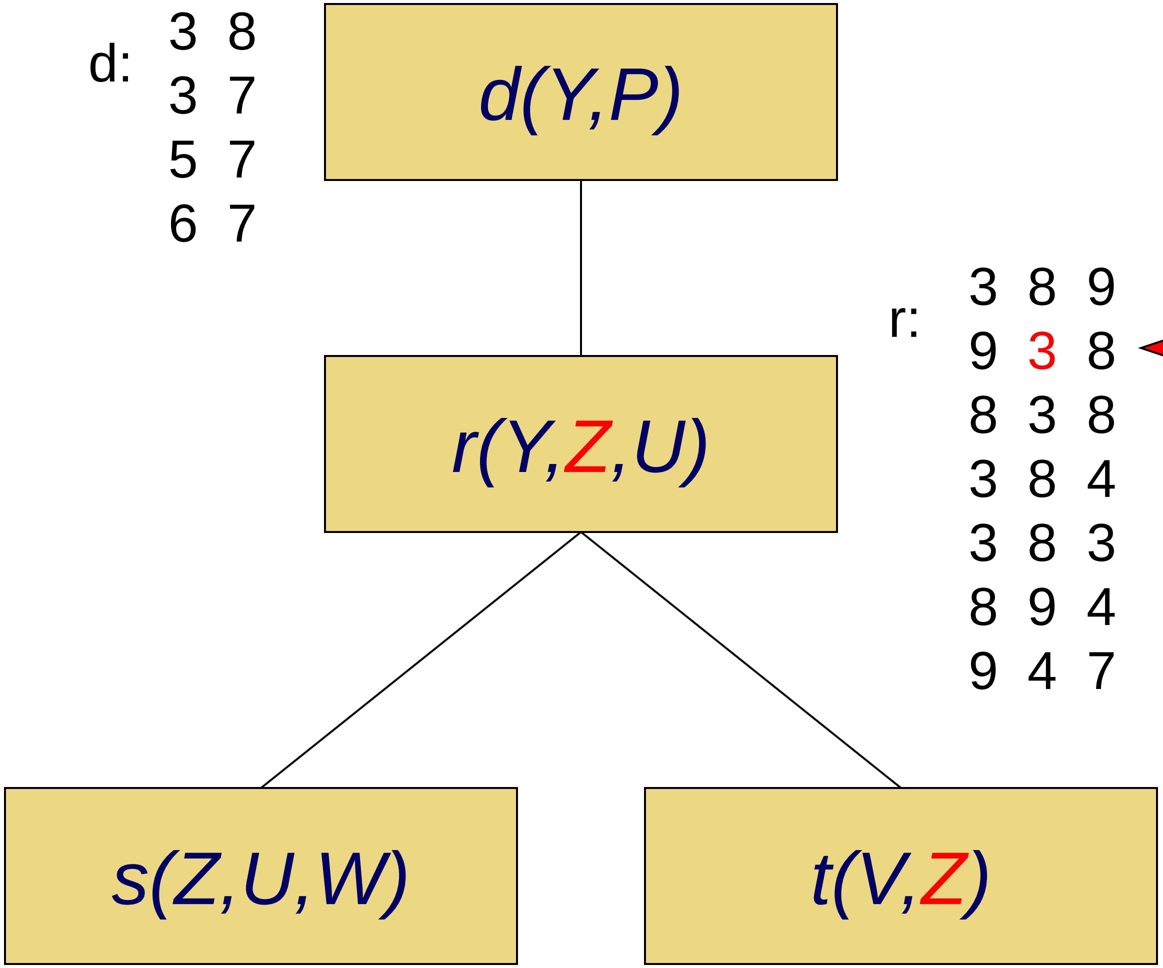
3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

t:

9	8
9	3
9	5

d:

3	8
3	7
5	7
6	7



$d(Y,P)$

$r(Y,Z,U)$

$s(Z,U,W)$

$t(V,Z)$

r:

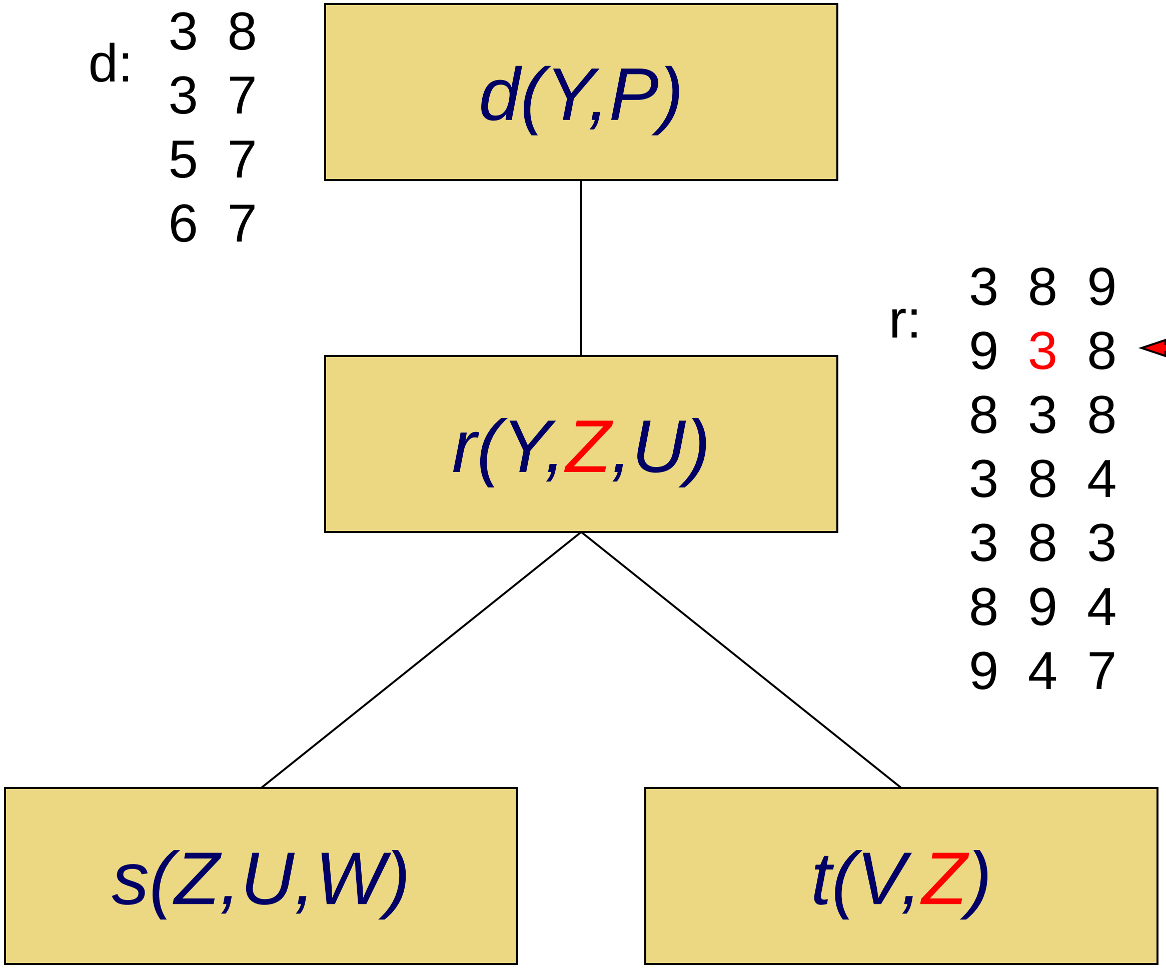
3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

s:

3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

t:

9	8
9	3
9	5



d:

3	8
3	7
5	7
6	7

r:

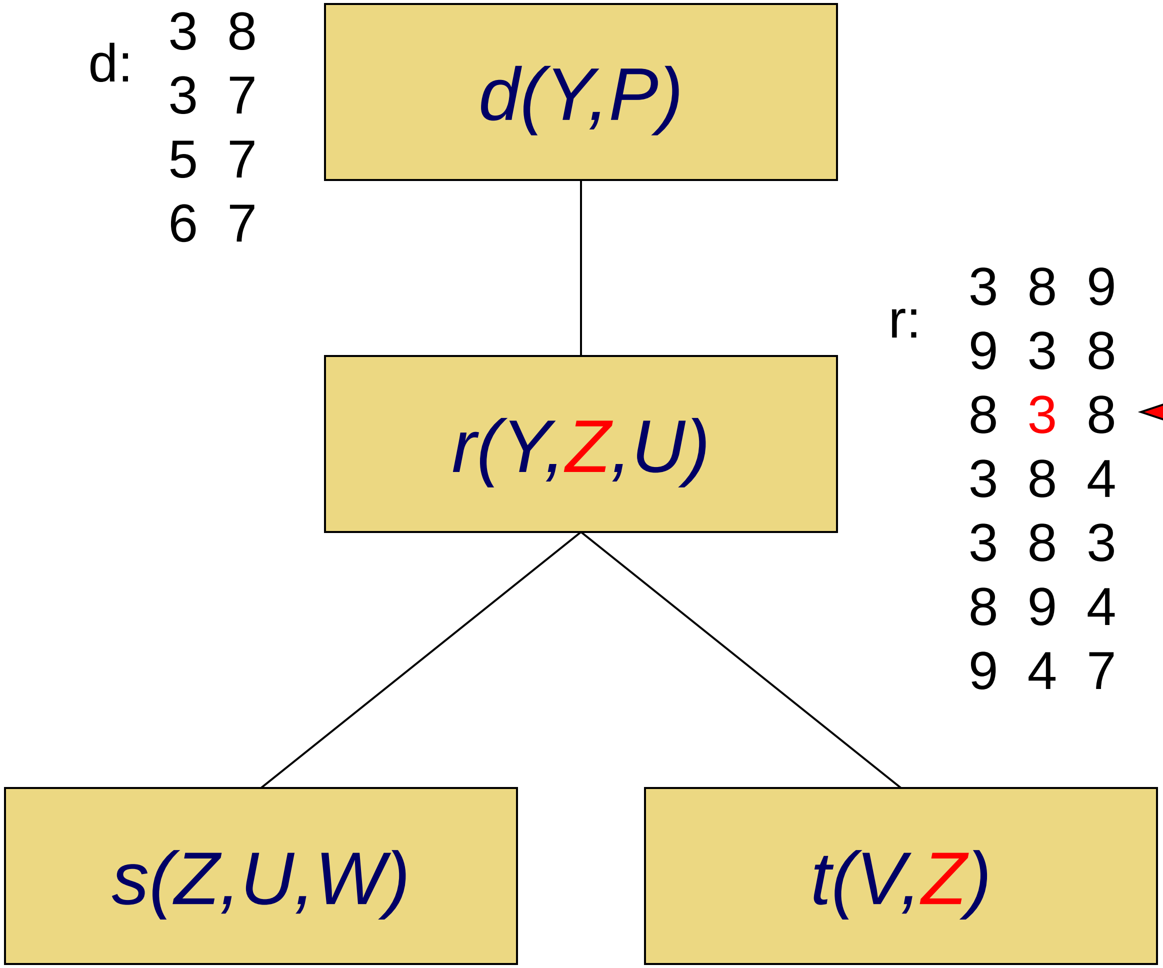
3	8	9	
9	3	8	←
8	3	8	
3	8	4	
3	8	3	
8	9	4	
9	4	7	

s:

3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

t:

9	8	
9	3	←
9	5	



d:

3	8
3	7
5	7
6	7

r:

3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

…

s:

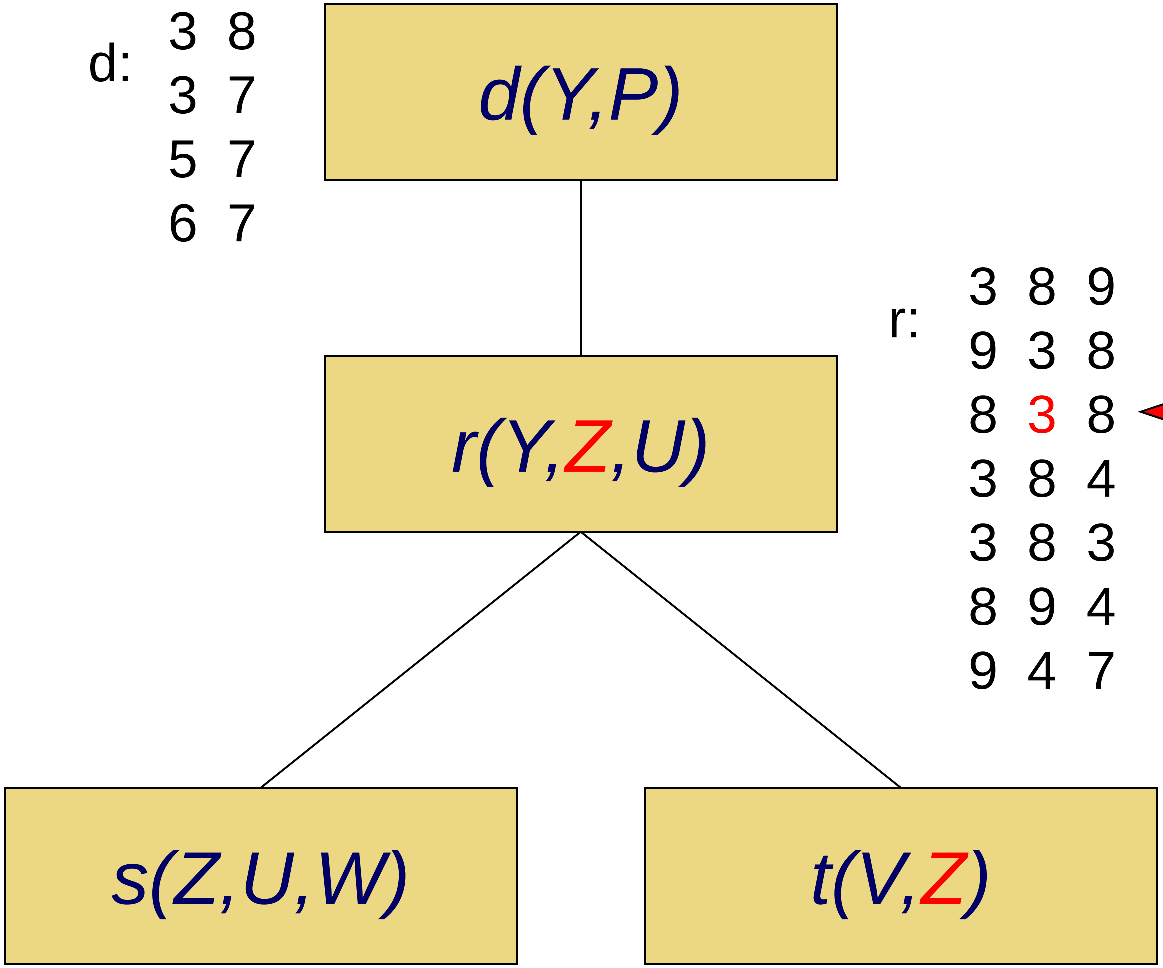
3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

t:

9	8
9	3
9	5

d:

3	8
3	7
5	7
6	7



r:

3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

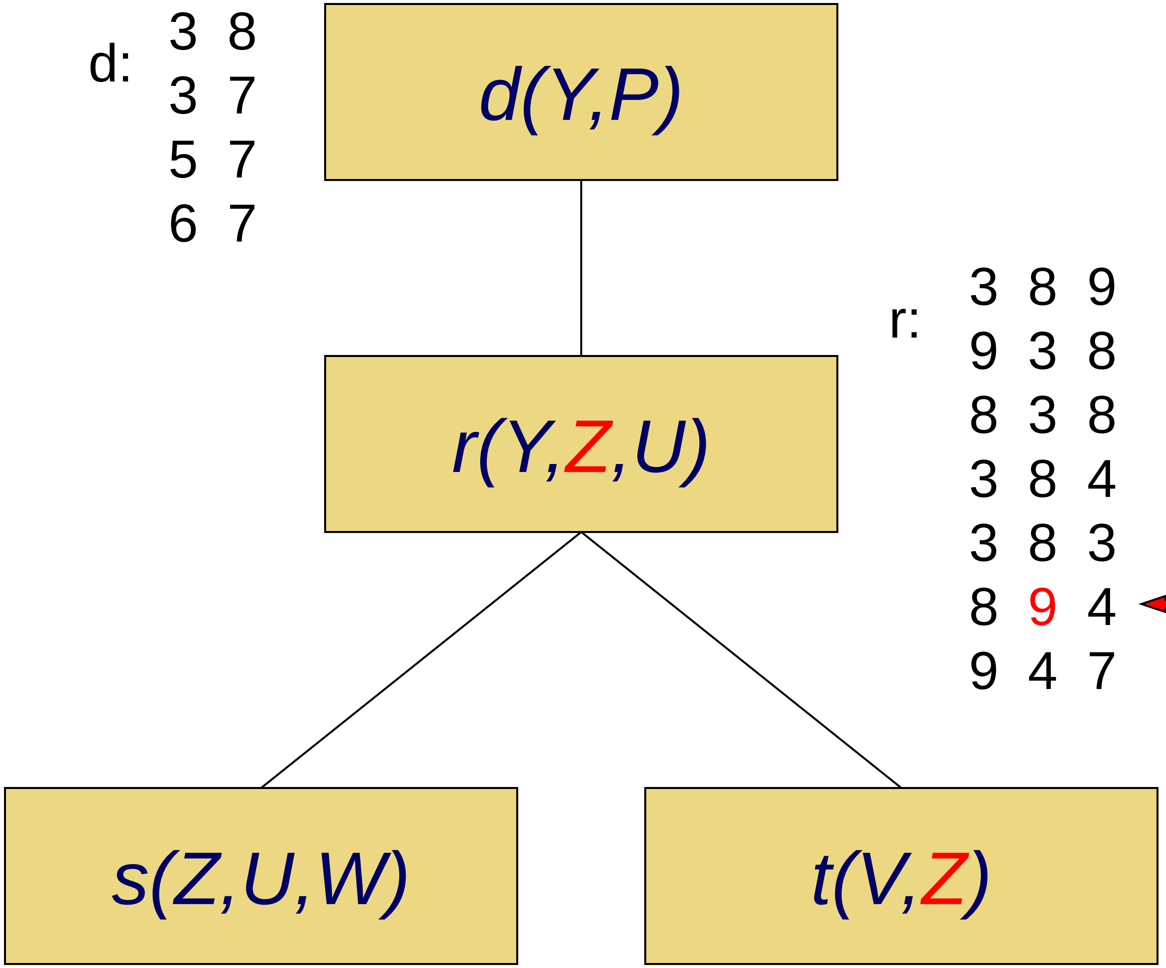
…

s:

3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

t:

9	8
9	3
9	5



d:

3	8
3	7
5	7
6	7

r:

3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

s:

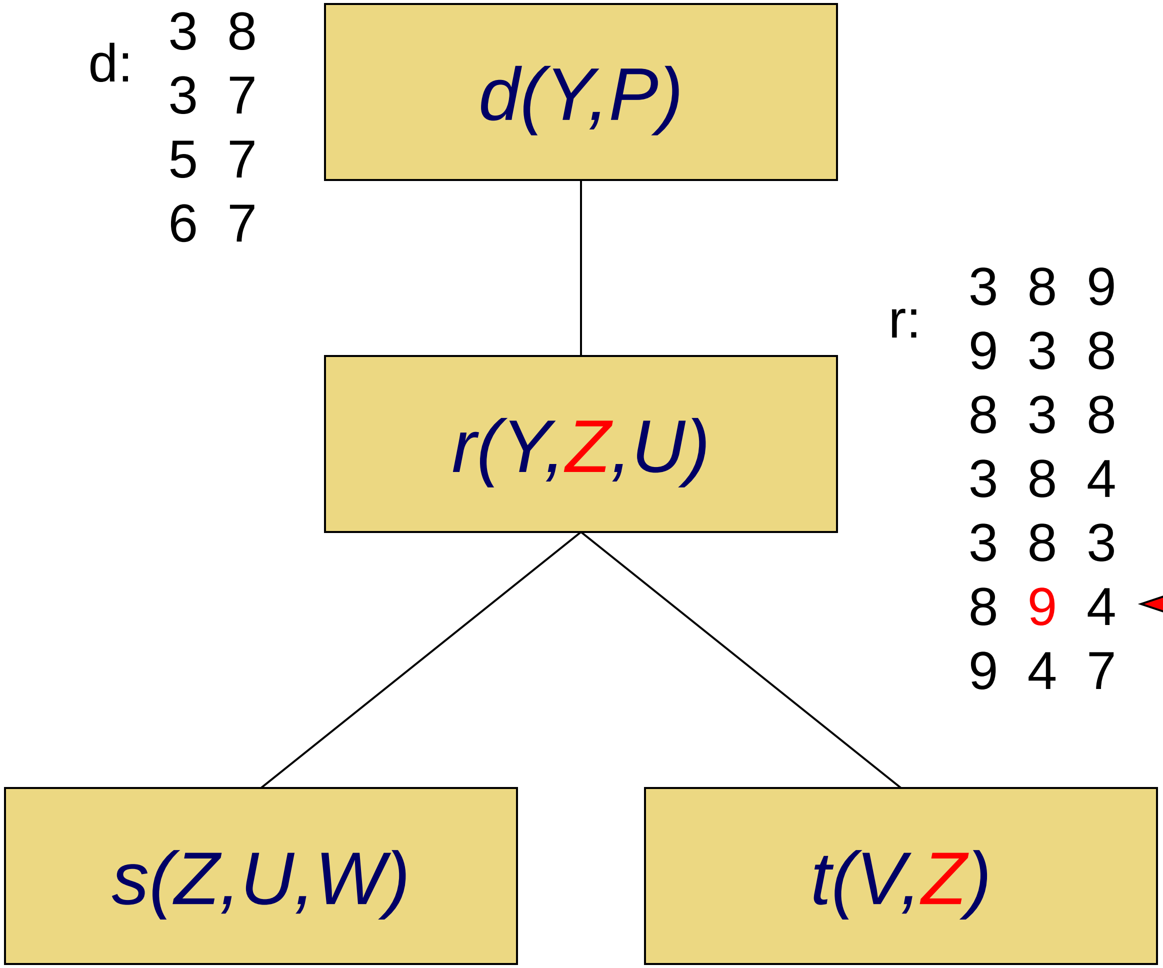
3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

t:

9	8
9	3
9	5

d:

3	8
3	7
5	7
6	7



r:

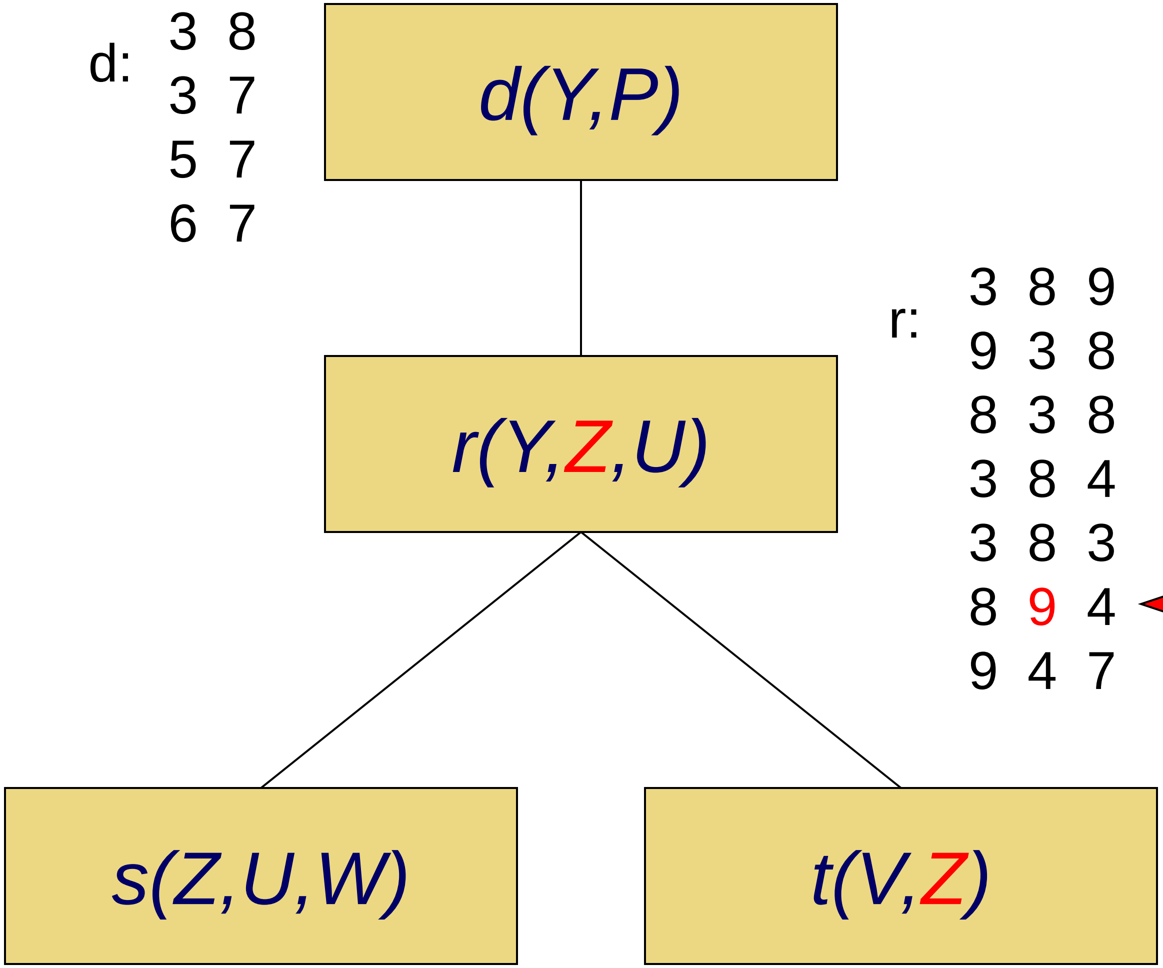
3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

s:

3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

t:

9	8
9	3
9	5



d(Y,P)

r(Y,Z,U)

s(Z,U,W)

t(V,Z)

d:

3	8
3	7
5	7
6	7

r:

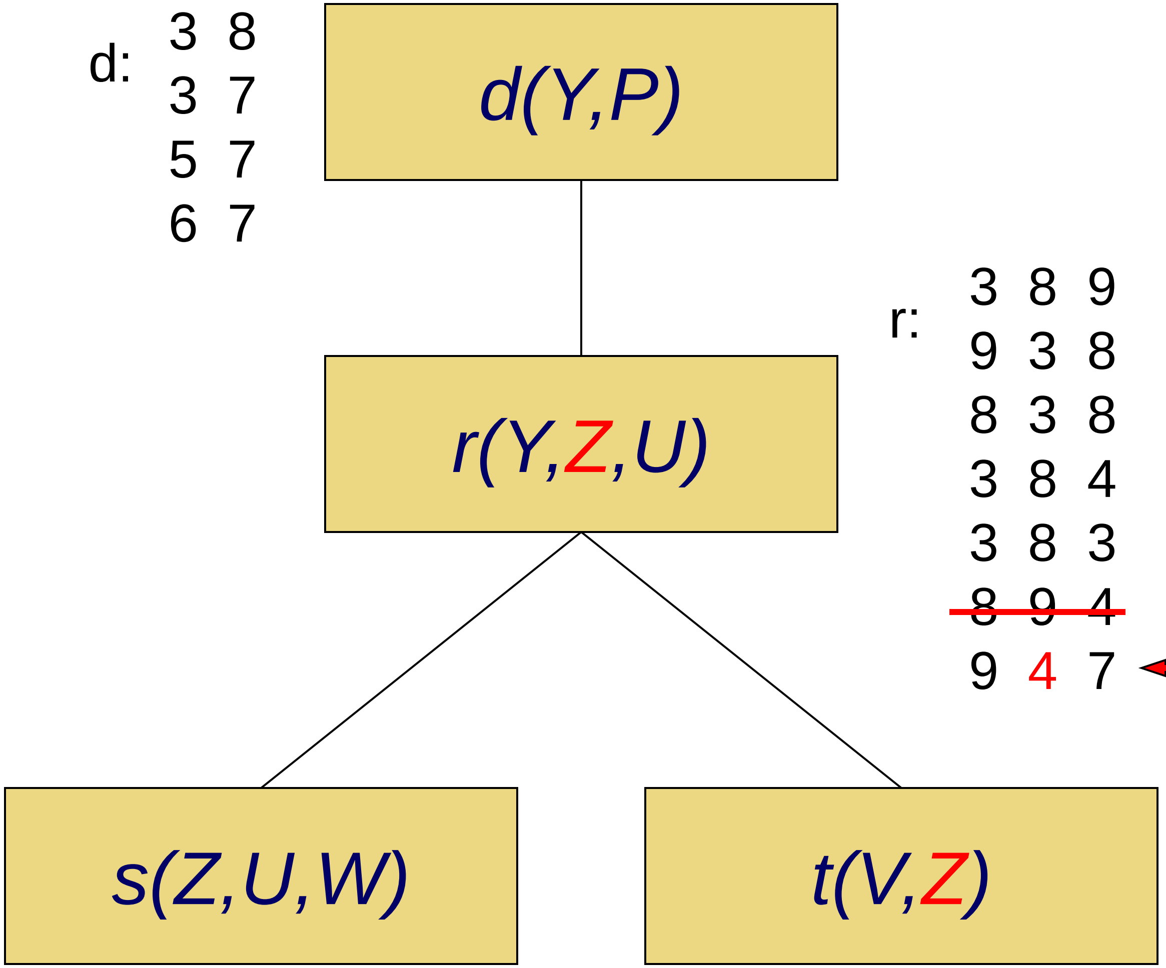
3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

s:

3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

t:

9	8
9	3
9	5



d:

3	8
3	7
5	7
6	7

r:

3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
~~8~~	~~9~~	~~4~~
9	4	7

…

s:

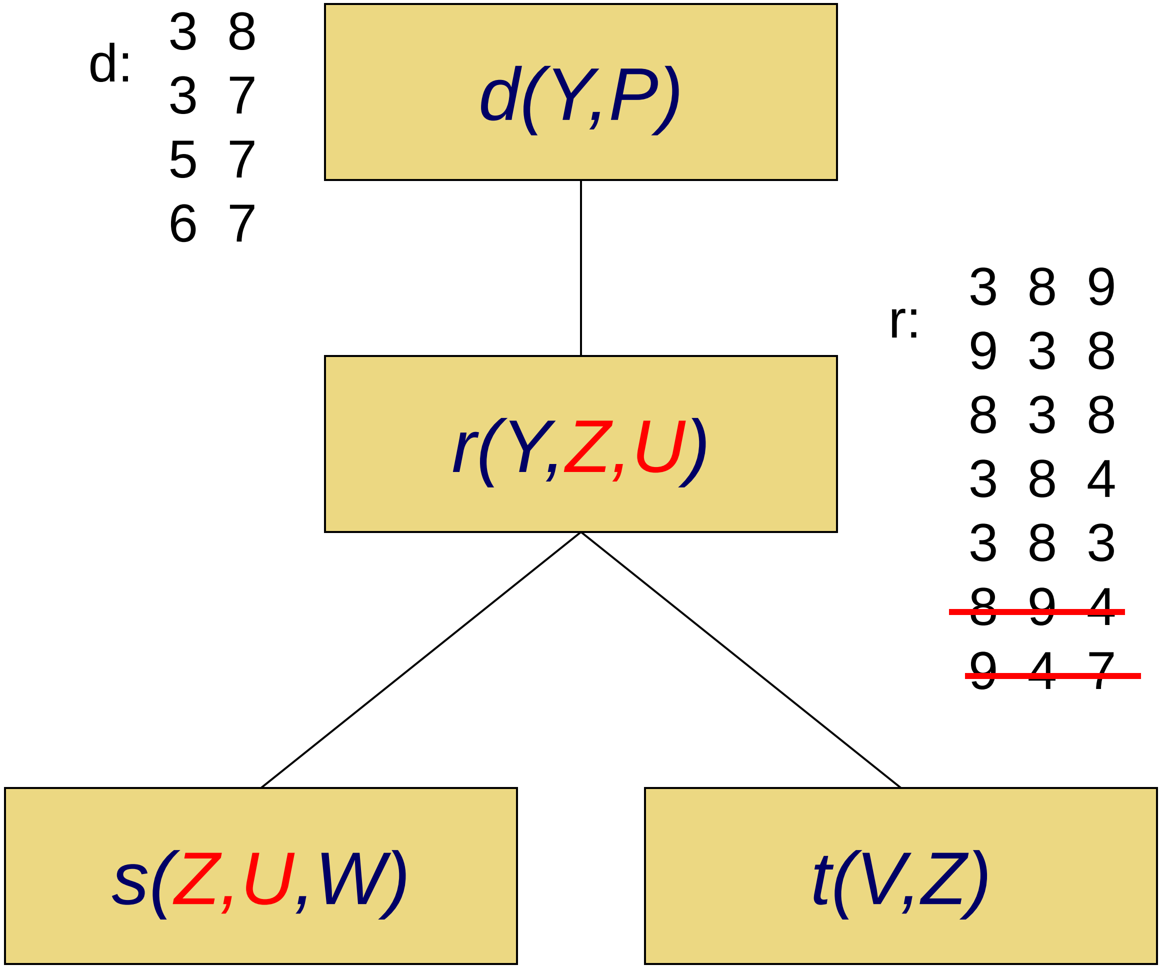
3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

t:

9	8
9	3
9	5

d:

3	8
3	7
5	7
6	7



r:

3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
~~8~~	~~9~~	~~4~~
~~9~~	~~4~~	~~7~~

s:

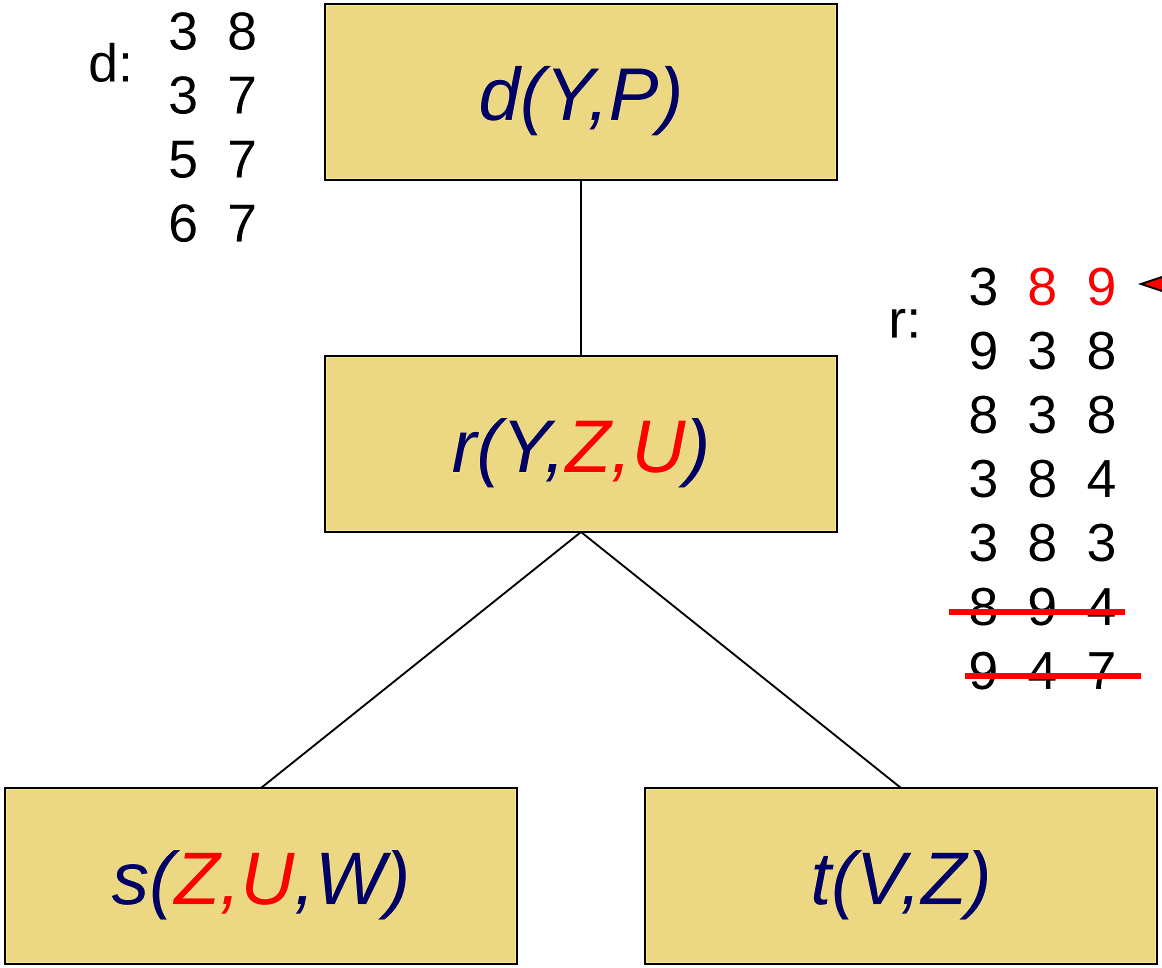
3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

t:

9	8
9	3
9	5

d:

3	8
3	7
5	7
6	7



d(Y,P)

r(Y,Z,U)

s(Z,U,W)

t(V,Z)

r:

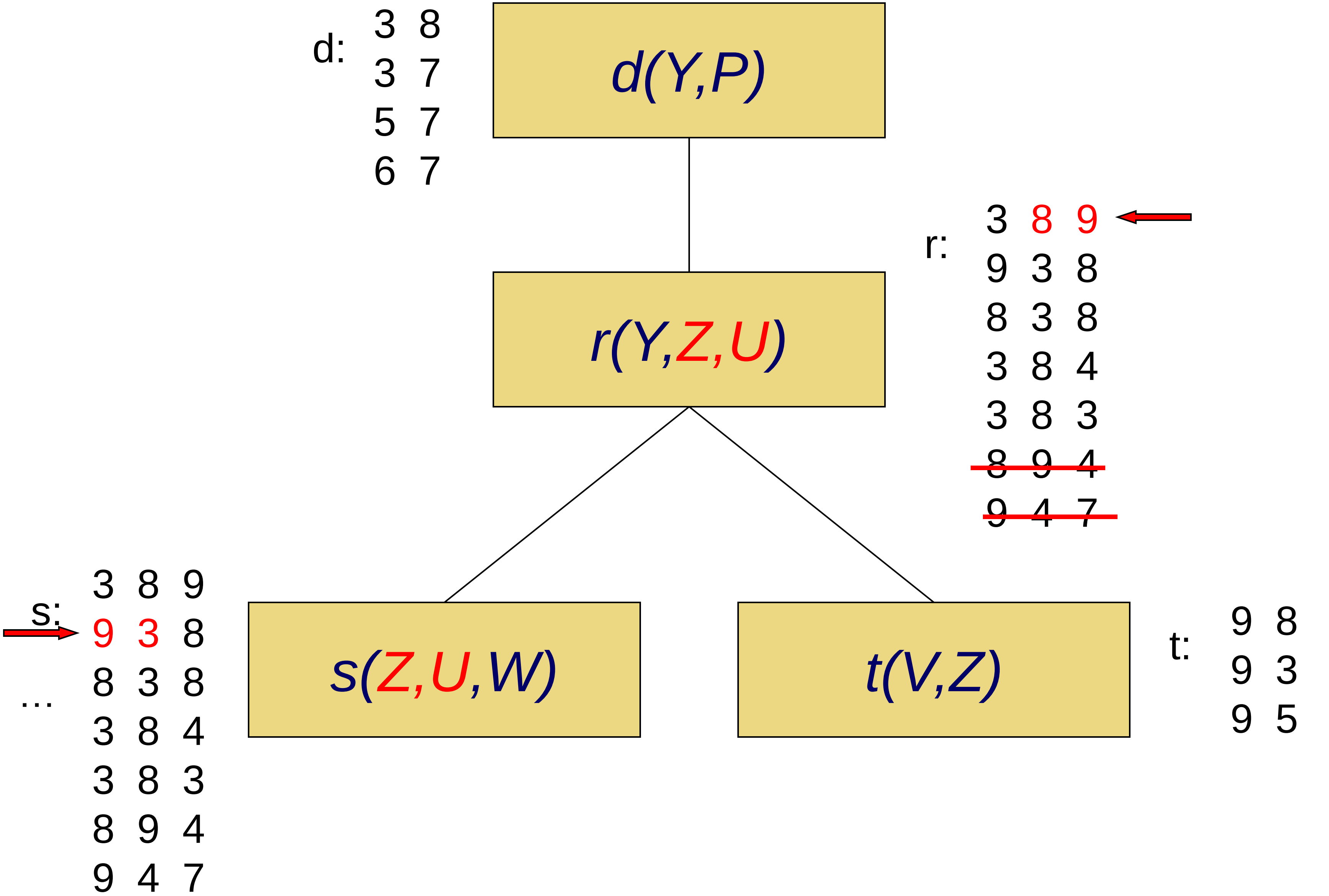
3	8	9	←
9	3	8	
8	3	8	
3	8	4	
3	8	3	
~~8~~	~~9~~	~~4~~	
~~9~~	~~4~~	~~7~~	

s:

→	3	8	9
	9	3	8
	8	3	8
	3	8	4
	3	8	3
	8	9	4
	9	4	7

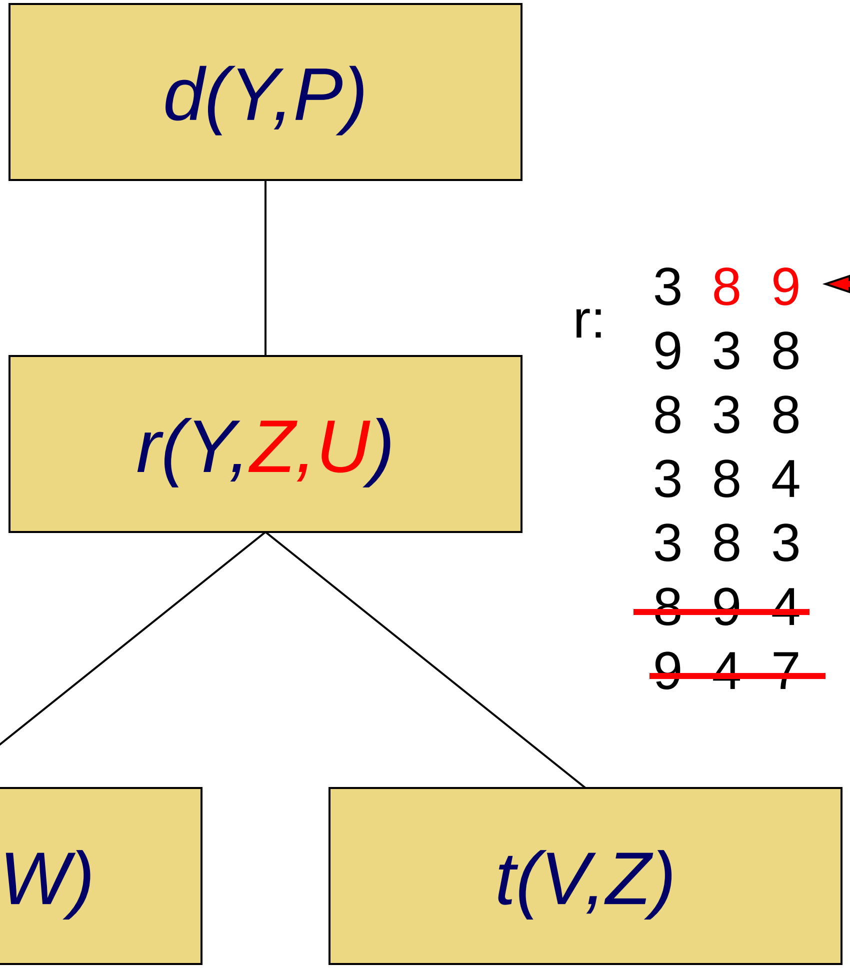
t:

9	8
9	3
9	5



d:

3	8
3	7
5	7
6	7



r:

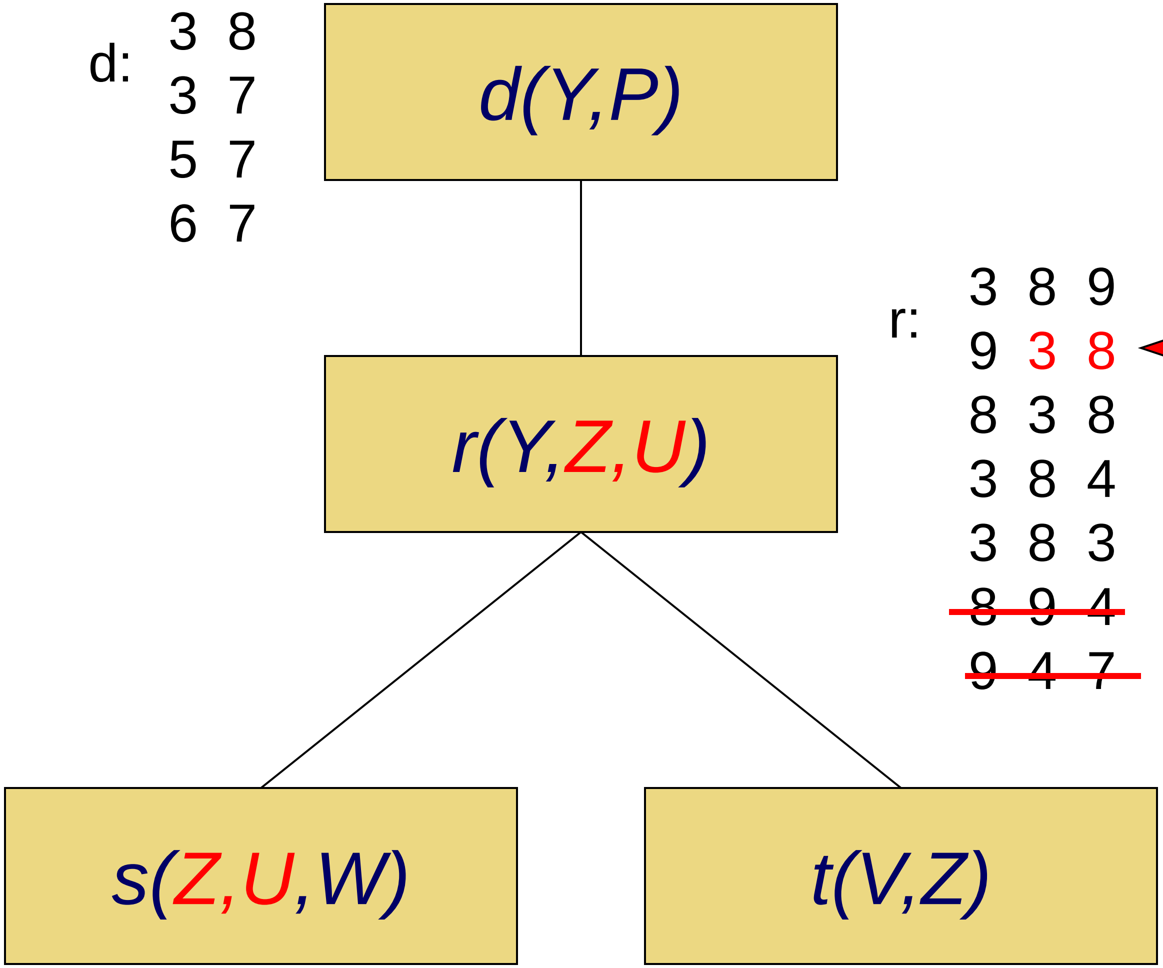
3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
~~8~~	~~9~~	~~4~~
~~9~~	~~4~~	~~7~~

s:

3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

t:

9	8
9	3
9	5



d(Y,P)

r(Y,Z,U)

s(Z,U,W)

t(V,Z)

d:

3	8
3	7
5	7
6	7

r:

3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
~~8~~	~~9~~	~~4~~
~~9~~	~~4~~	~~7~~

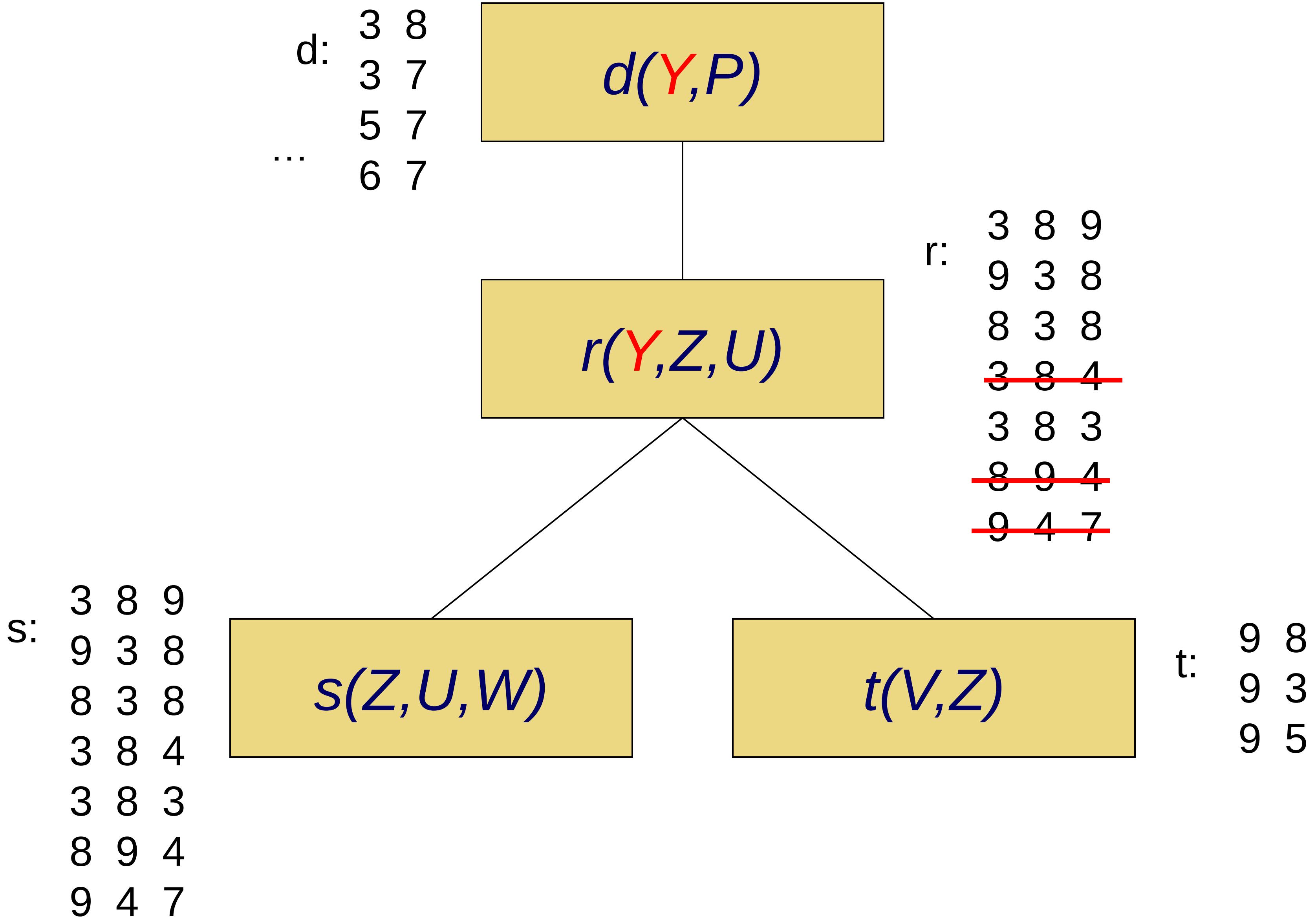
…

s:

3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

t:

9	8
9	3
9	5



d(Y,P)

r(Y,Z,U)

s(Z,U,W)

t(V,Z)

d:

3	8
3	7
5	7
6	7

…

r:

3	8	9
9	3	8
8	3	8
~~3~~	~~8~~	~~4~~
3	8	3
~~8~~	~~9~~	~~4~~
~~9~~	~~4~~	~~7~~

s:

3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

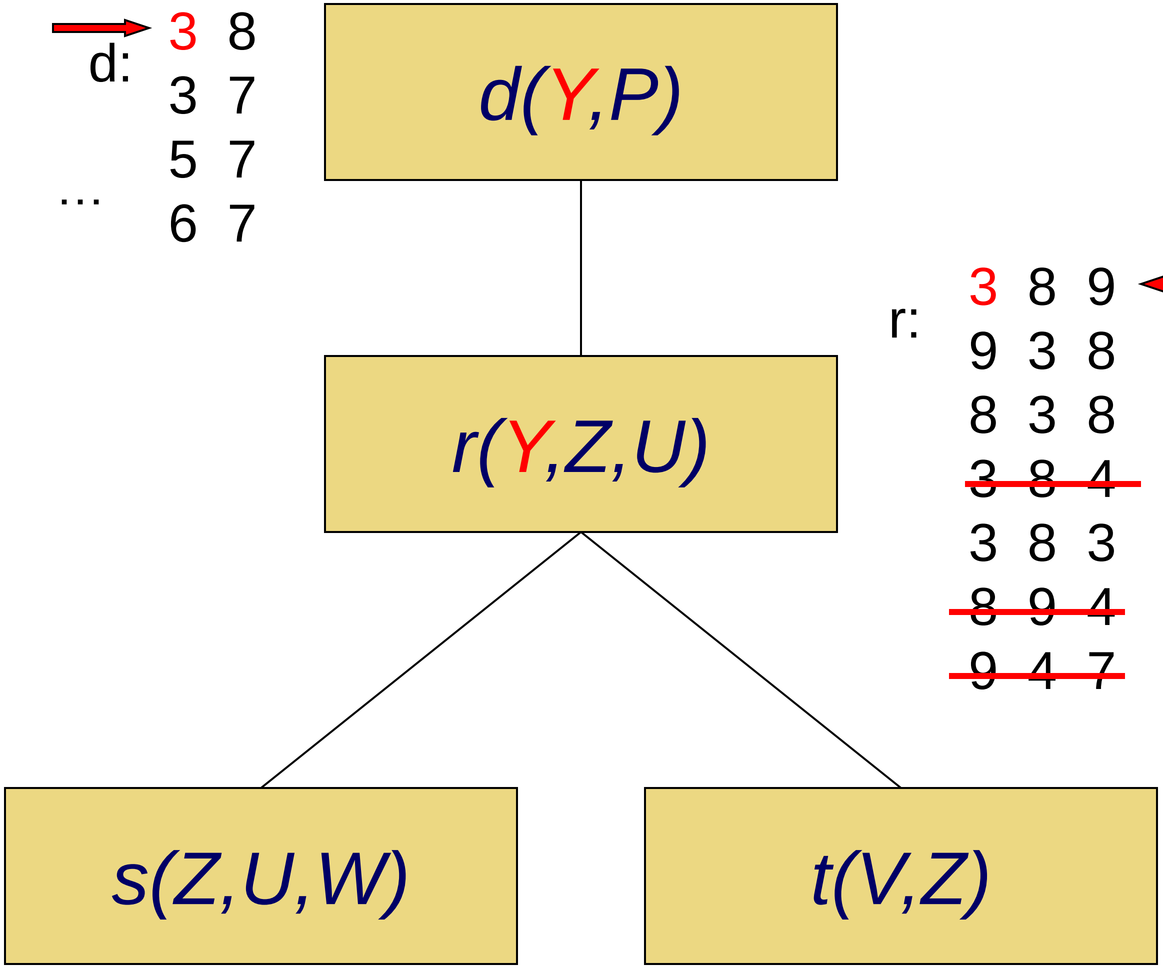
t:

9	8
9	3
9	5

d:

3	8
3	7
5	7
6	7

…



d(Y,P)

r(Y,Z,U)

s(Z,U,W)

t(V,Z)

r:

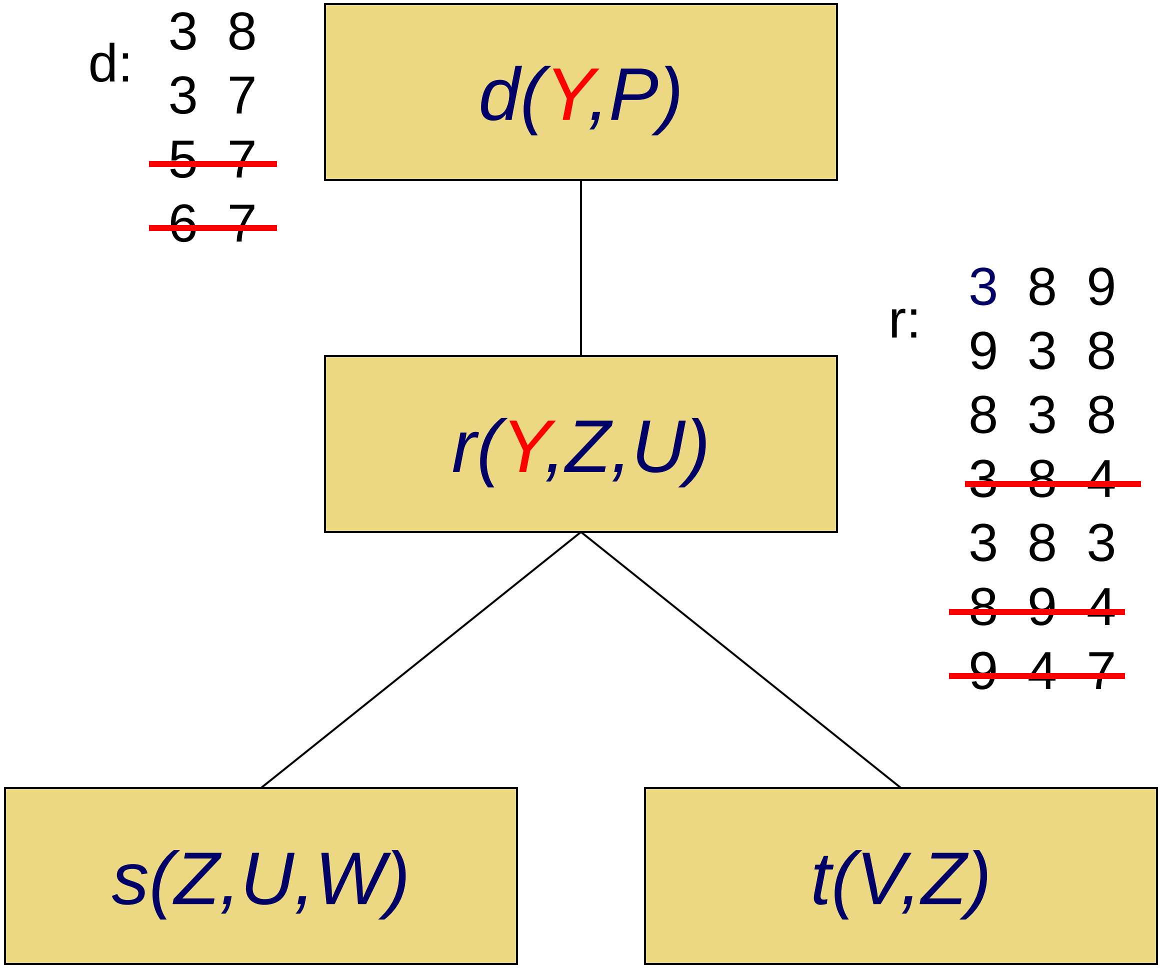
3	8	9
9	3	8
8	3	8
~~3~~	~~8~~	~~4~~
3	8	3
~~8~~	~~9~~	~~4~~
~~9~~	~~4~~	~~7~~

s:

3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

t:

9	8
9	3
9	5


d(Y,P)

r(Y,Z,U)

s(Z,U,W)

t(V,Z)

d:

3	8
3	7
~~5~~	~~7~~
~~6~~	~~7~~

r:

3	8	9
9	3	8
8	3	8
~~3~~	~~8~~	~~4~~
3	8	3
~~8~~	~~9~~	~~4~~
~~9~~	~~4~~	~~7~~

s:

3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

t:

9	8
9	3
9	5

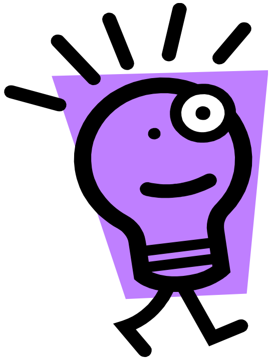
«Answering» Acyclic Instances

HOM: The homomorphism problem

BCQ: Boolean conjunctive query evaluation

CSP: Constraint satisfaction problem

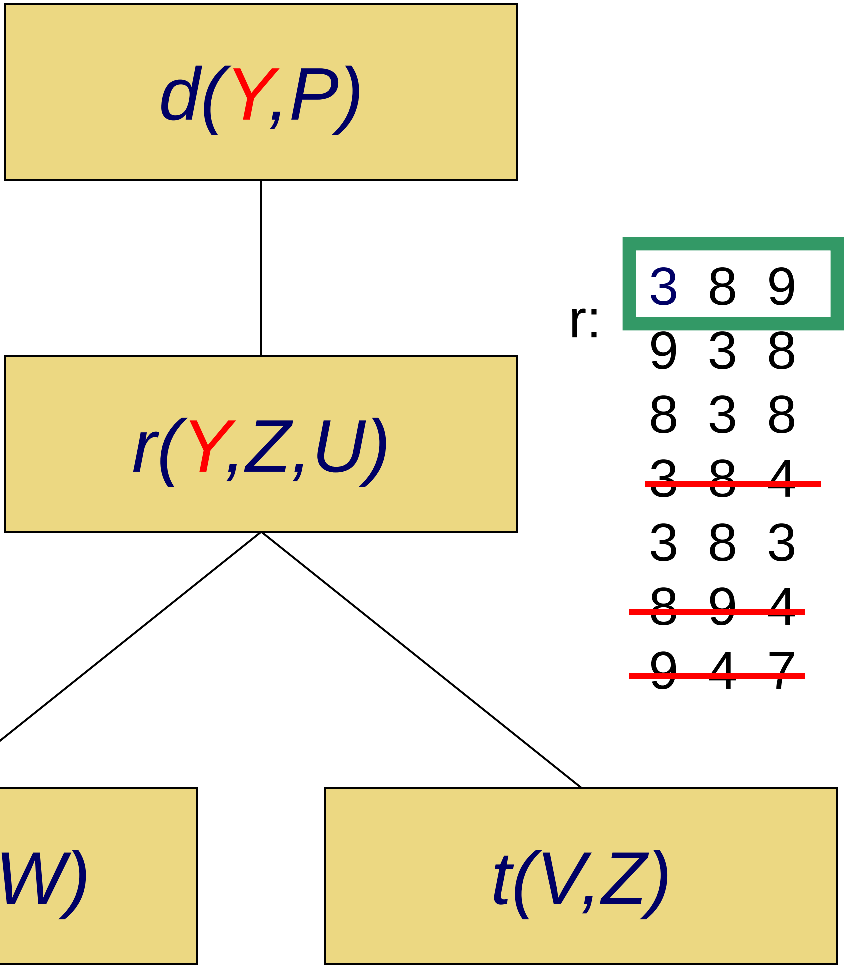
Yannakakis’s Algorithm (Acyclic structures):
Dynamic Programming over a Join Tree



Solutions can be computed by adding a top-down phase to Yannakakis’ algorithm for acyclic instances

d:

3	8
3	7
~~5~~	~~7~~
~~6~~	~~7~~



r:

3	8	9
9	3	8
8	3	8
~~3~~	~~8~~	~~4~~
3	8	3
~~8~~	~~9~~	~~4~~
~~9~~	~~4~~	~~7~~

s:

3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

t:

9	8
9	3
9	5

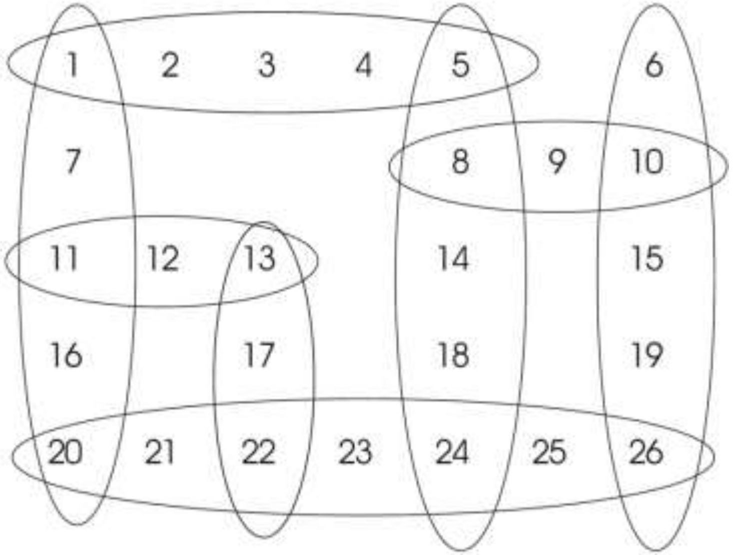
A solution: Y=3, P=7, Z=8, U=9, W=4, V=9

Computing the result (Acyclic)

- The result size can be exponential (even in the acyclic case).
- Even when the result is of polynomial size, it is in general hard to compute.
- In case of acyclic instances, the result can be computed in time polynomial in the result size
  (and **with polynomial delay**: first solution, if any, in polynomial time, and each subsequent solution within polynomial time from the previous one).
- This will remain true for the subsequent generalizations of acyclicity.
- Add a top-down phase to Yannakakis' algorithm for acyclic instances, thus obtaining a full reducer, and join the partial results (or perform a backtrack free visit)

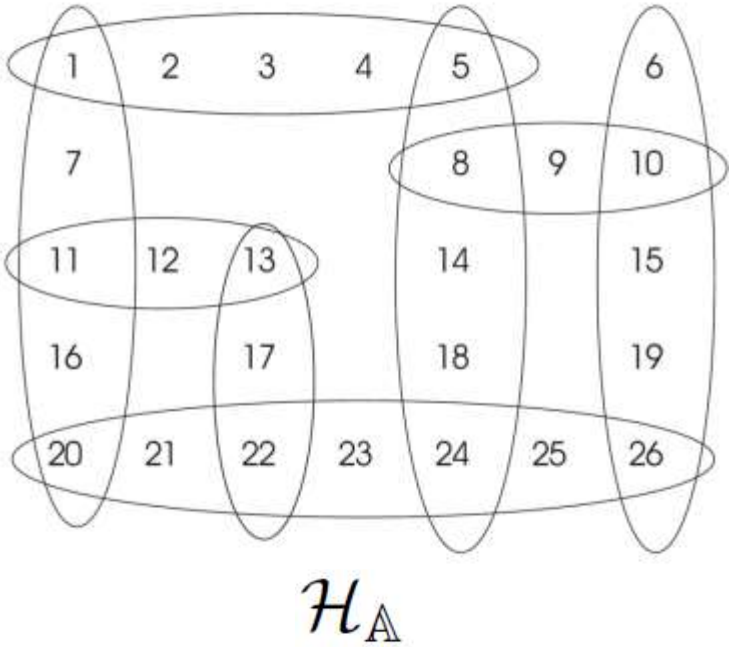
Decomposition Methods

1	2	3	4	5	■	6
7	■	■	■	8	9	10
11	12	13	■	14	■	15
16	■	17	■	18	■	19
20	21	22	23	24	25	26



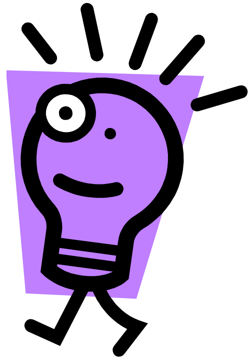
$\mathcal{H}_{\mathbb{A}}$

Decomposition Methods

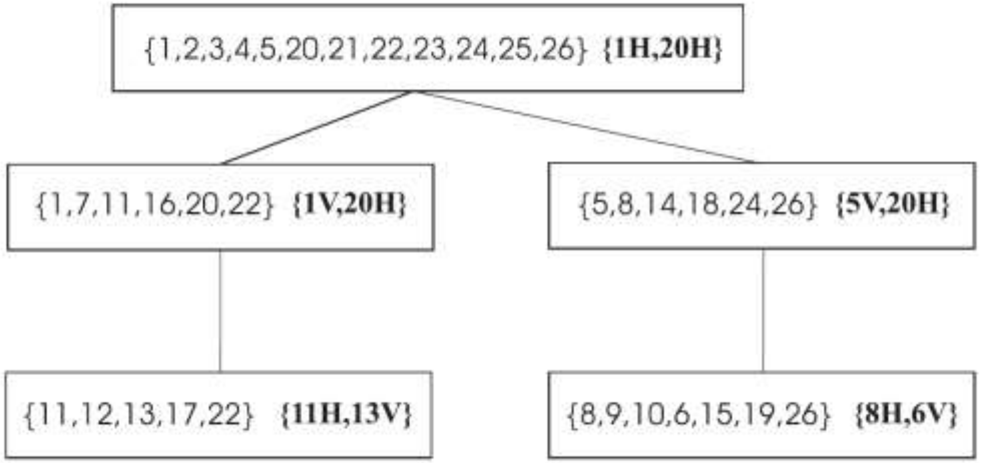


Transform the hypergraph into an acyclic one:

- Organize its edges (or nodes) in clusters
- Arrange the clusters as a tree,
  by satisfying the connectedness condition



(Generalized) Hypertree Decompositions



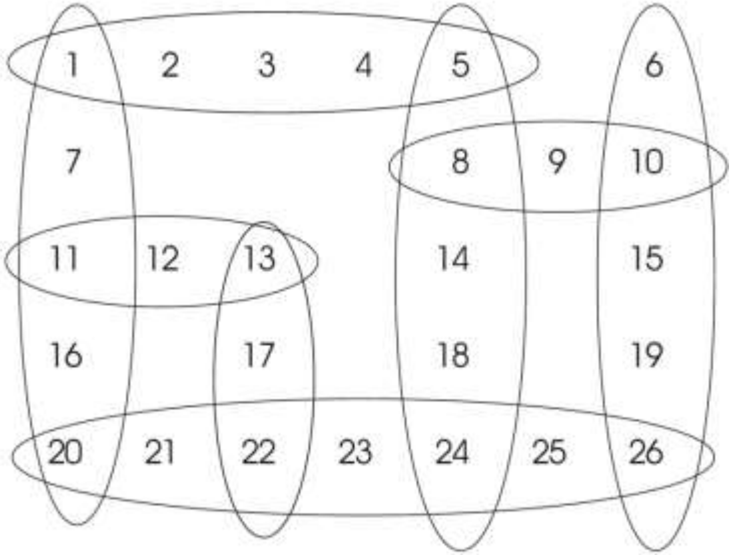
{1,2,3,4,5,20,21,22,23,24,25,26} **{1H,20H}**

{1,7,11,16,20,22} **{1V,20H}**

{5,8,14,18,24,26} **{5V,20H}**

{11,12,13,17,22} **{11H,13V}**

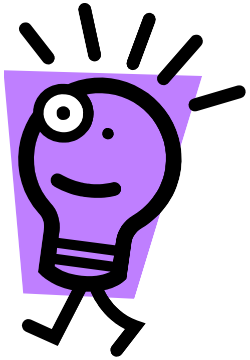
{8,9,10,6,15,19,26} **{8H,6V}**



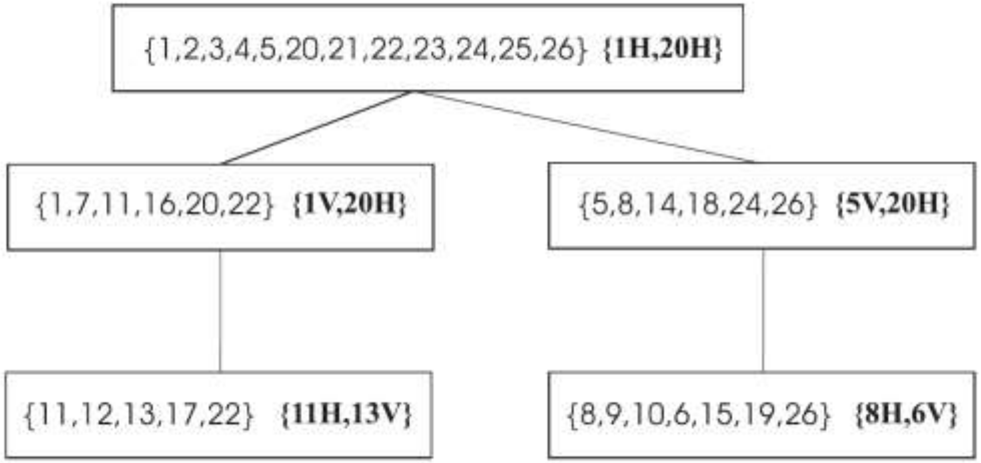
$\mathcal{H}_{\mathbb{A}}$

Transform the hypergraph into an acyclic one:

- Organize its edges (or nodes) in clusters
- Arrange the clusters as a tree,
  by satisfying the connectedness condition



(Generalized) Hypertree Decompositions



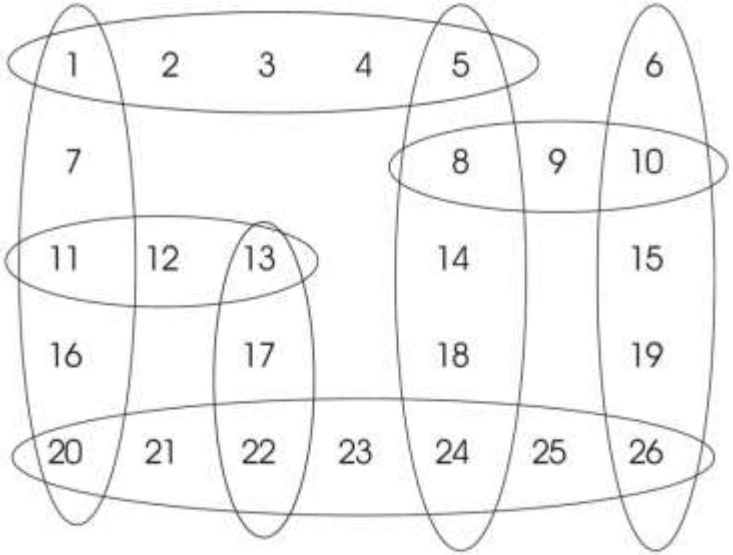
{1,2,3,4,5,20,21,22,23,24,25,26} **{1H,20H}**

{1,7,11,16,20,22} **{1V,20H}**

{5,8,14,18,24,26} **{5V,20H}**

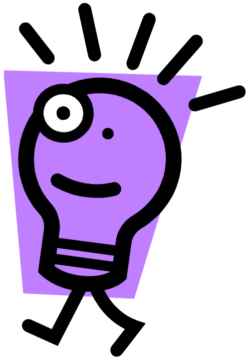
{11,12,13,17,22} **{11H,13V}**

{8,9,10,6,15,19,26} **{8H,6V}**



$\mathcal{H}_\mathbb{A}$

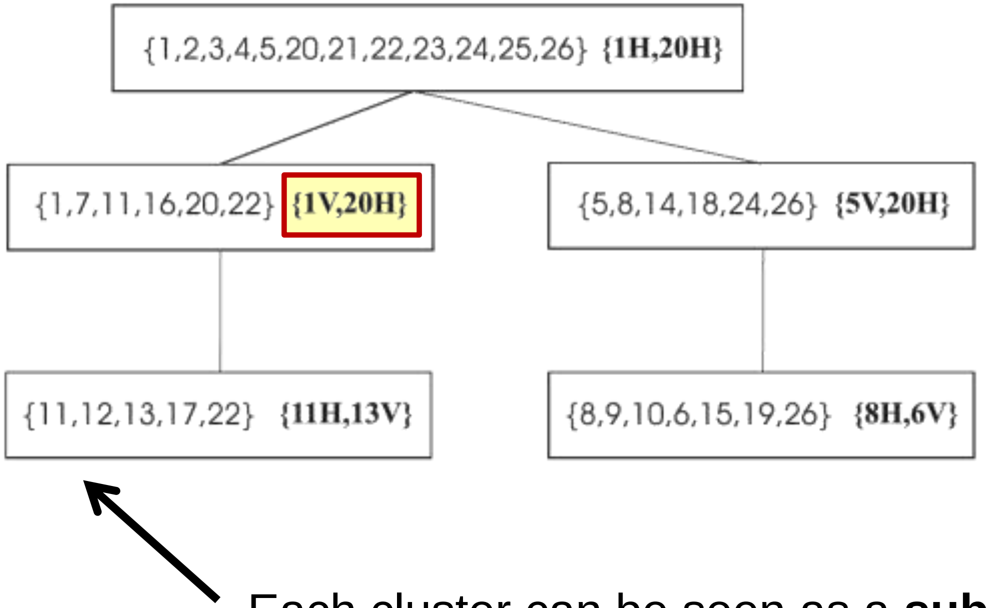
Each cluster can be seen as a **subproblem**



Transform the hypergraph into an acyclic one:

- Organize its edges (or nodes) in clusters
- Arrange the clusters as a tree,
  by satisfying the connectedness condition

(Generalized) Hypertree Decompositions



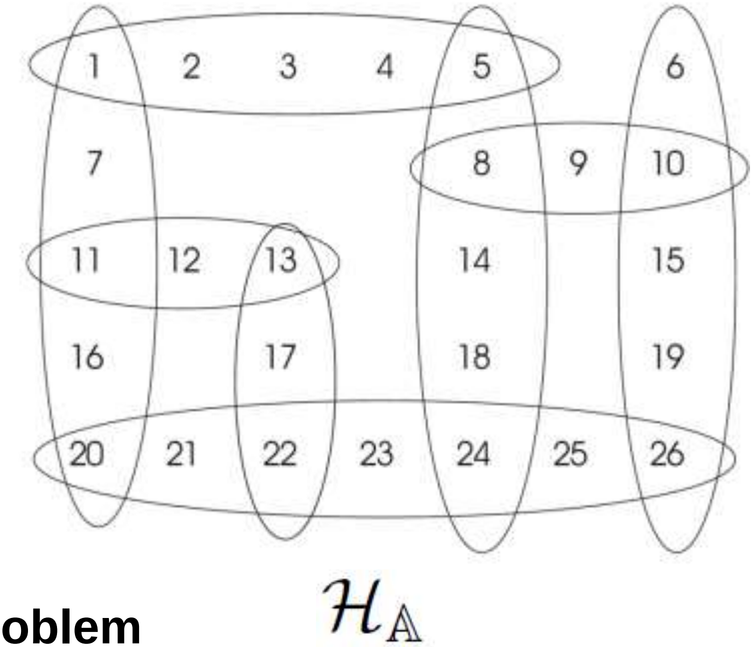
{1,2,3,4,5,20,21,22,23,24,25,26} {1H,20H}

{1,7,11,16,20,22} {1V,20H}

{5,8,14,18,24,26} {5V,20H}

{11,12,13,17,22} {11H,13V}

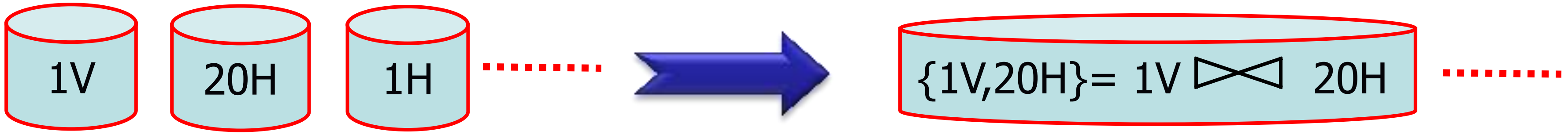
{8,9,10,6,15,19,26} {8H,6V}



Each cluster can be seen as a **subproblem**

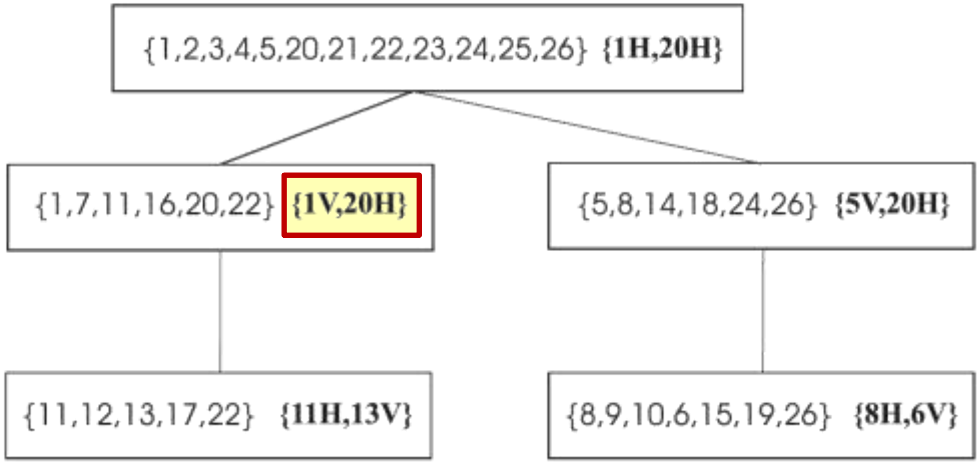
Relations:

Relations:



{1V,20H}= 1V $\bowtie$ 20H

Toward an equivalent acyclic instance



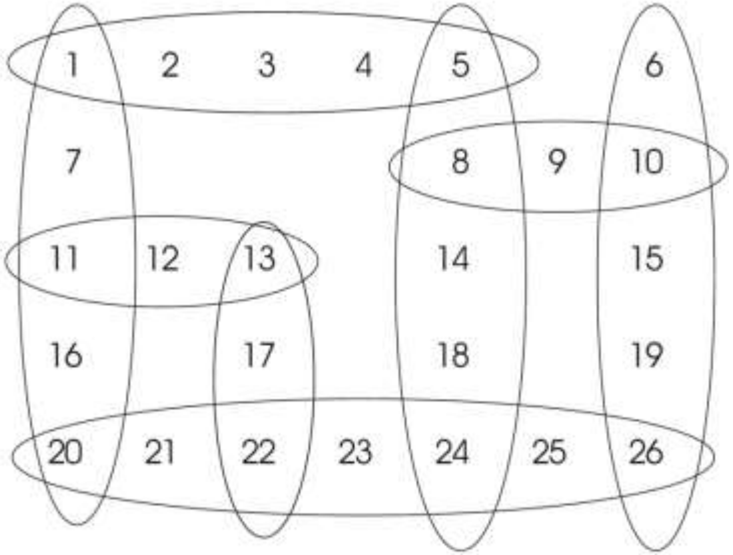
{1,2,3,4,5,20,21,22,23,24,25,26} **{1H,20H}**

{1,7,11,16,20,22} **{1V,20H}**

{5,8,14,18,24,26} **{5V,20H}**

{11,12,13,17,22} **{11H,13V}**

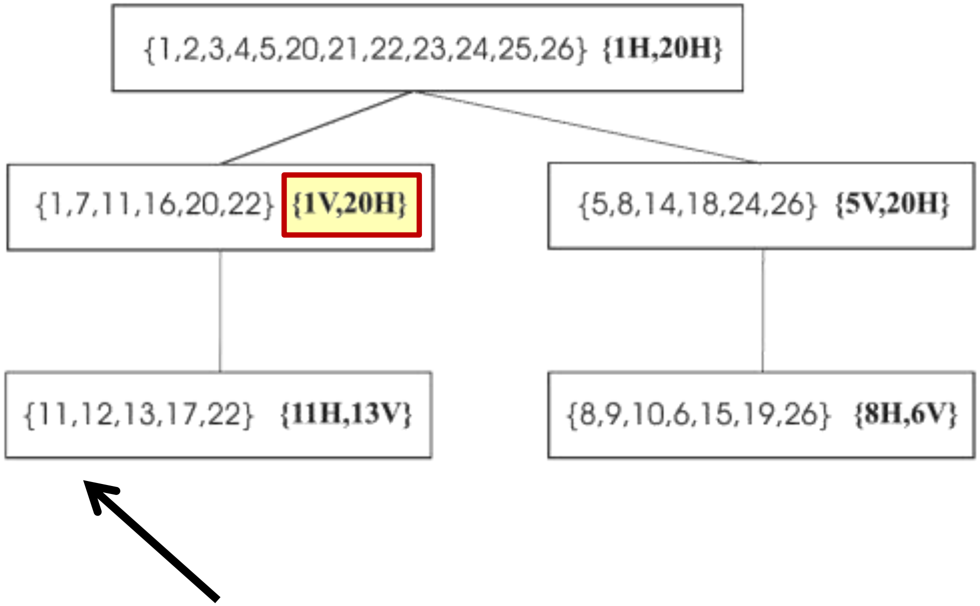
{8,9,10,6,15,19,26} **{8H,6V}**



$\mathcal{H}_{\mathbb{A}}$

- Each cluster can be seen as a **subproblem**
- Associate each subproblem with a fresh constraint

Toward an equivalent acyclic instance



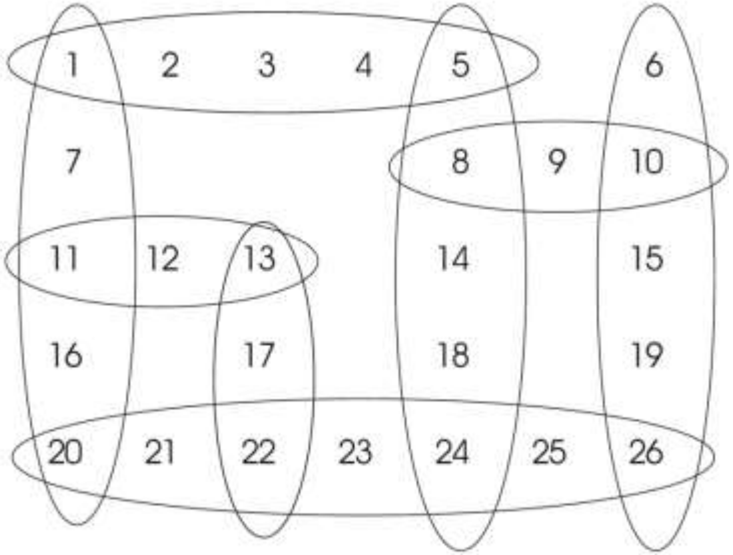
{1,2,3,4,5,20,21,22,23,24,25,26} **{1H,20H}**

{1,7,11,16,20,22} **{1V,20H}**

{5,8,14,18,24,26} **{5V,20H}**

{11,12,13,17,22} **{11H,13V}**

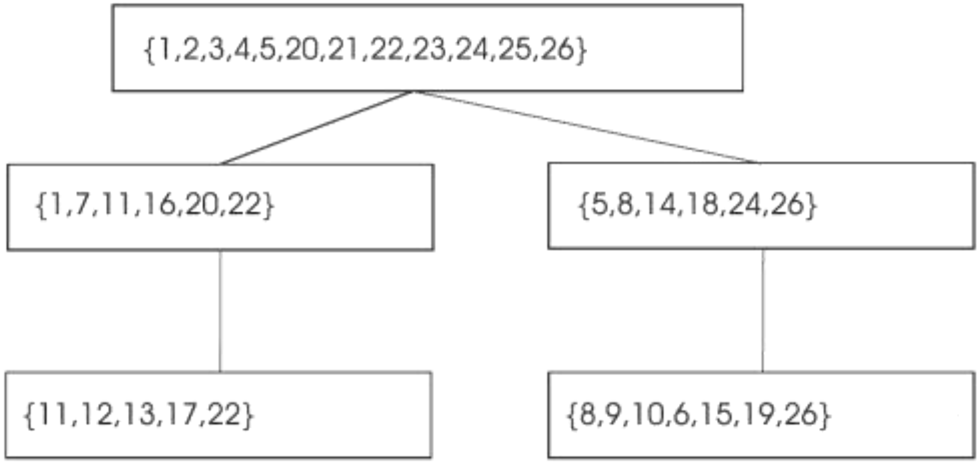
{8,9,10,6,15,19,26} **{8H,6V}**



$\mathcal{H}_{\mathbb{A}}$

- Each cluster can be seen as a **subproblem**
- Compute solutions for subproblems (exponential dependency on the width)
- Associate each subproblem with a fresh constraint
- Get an equivalent problem (all original constraints are there…)

The structure of the equivalent instance



{1,2,3,4,5,20,21,22,23,24,25,26}

{1,7,11,16,20,22}

{5,8,14,18,24,26}

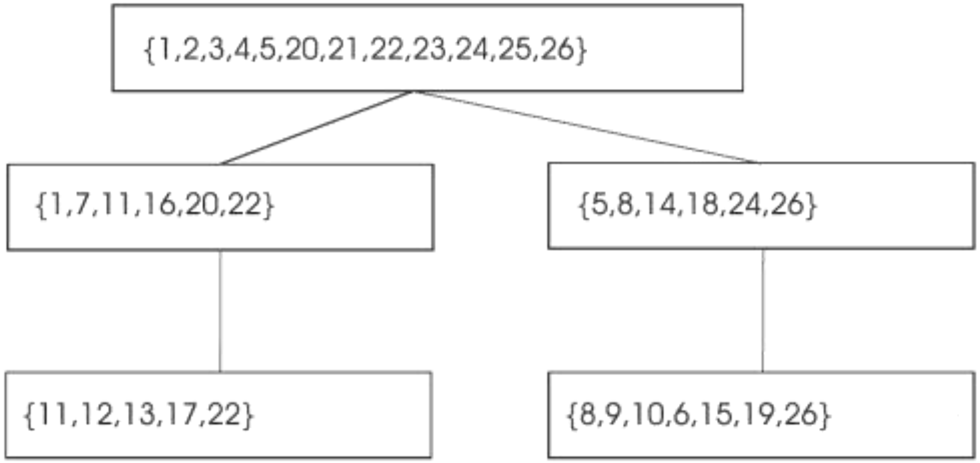
{11,12,13,17,22}

{8,9,10,6,15,19,26}

A join tree of the new instance

- Each cluster can be seen as a **subproblem**
- Compute solutions for subproblems (exponential dependency on the width)
- Associate each subproblem with a fresh constraint
- Get an equivalent problem (all original constraints are there…)

An acyclic equivalent instance



{1,2,3,4,5,20,21,22,23,24,25,26}

{1,7,11,16,20,22}

{5,8,14,18,24,26}

{11,12,13,17,22}

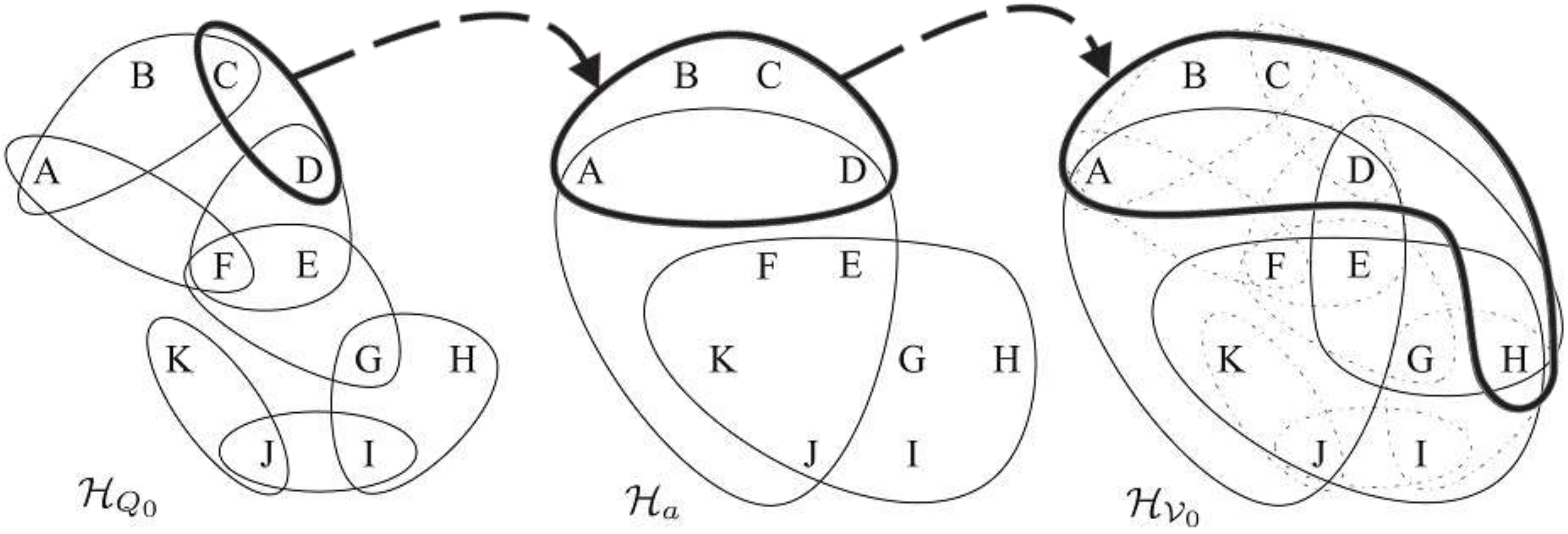
{8,9,10,6,15,19,26}

- Each cluster can be seen as a **subproblem**
- Compute solutions for subproblems (exponential dependency on the width)
- Associate each subproblem with a fresh constraint
- Get an equivalent problem (all original constraints are there…)

Solve the acyclic instance with any known technique

Tree Projection (idea)

- Generalization where suproblems are arbitrary (not necessarily clusters of k edges or vertices)



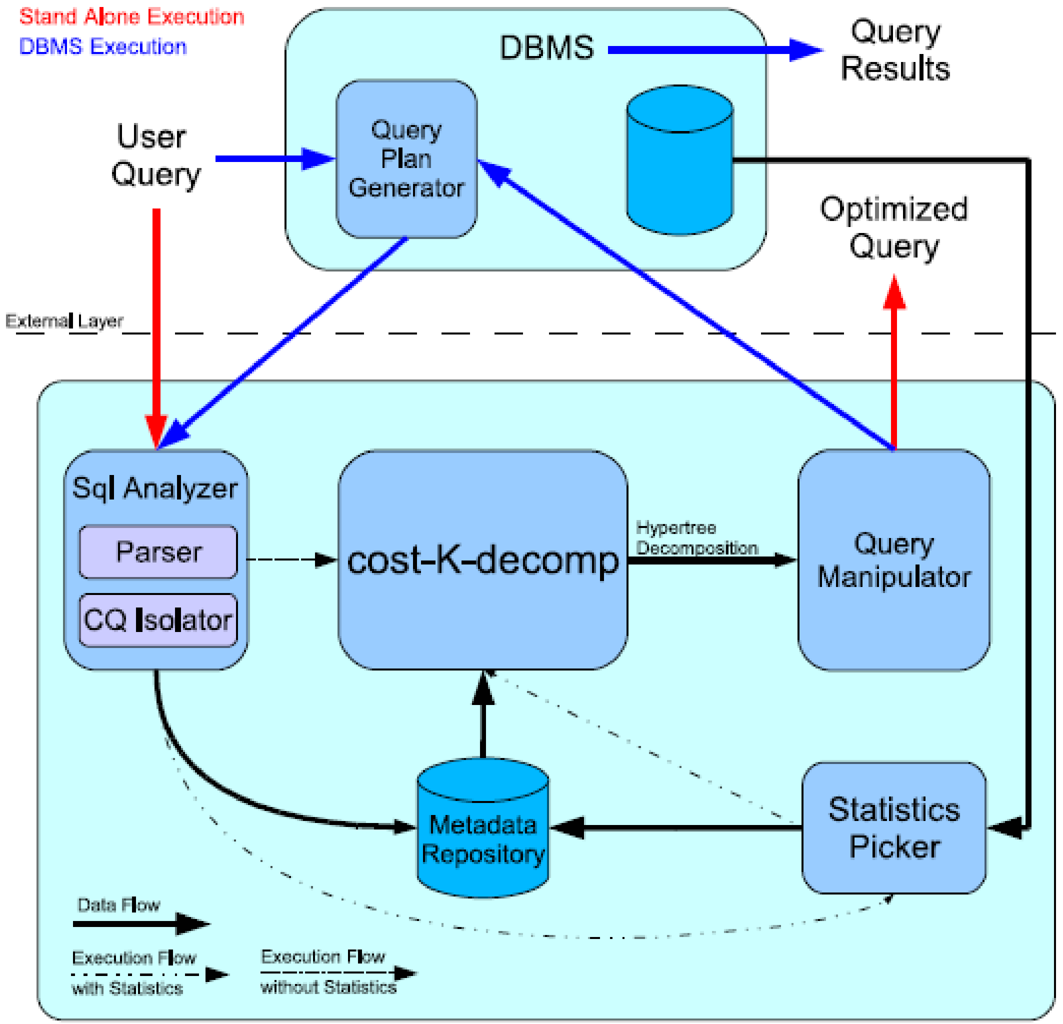
Structure

Sandwich acyclic hypergraph (Tree Projection)

Available Subproblems

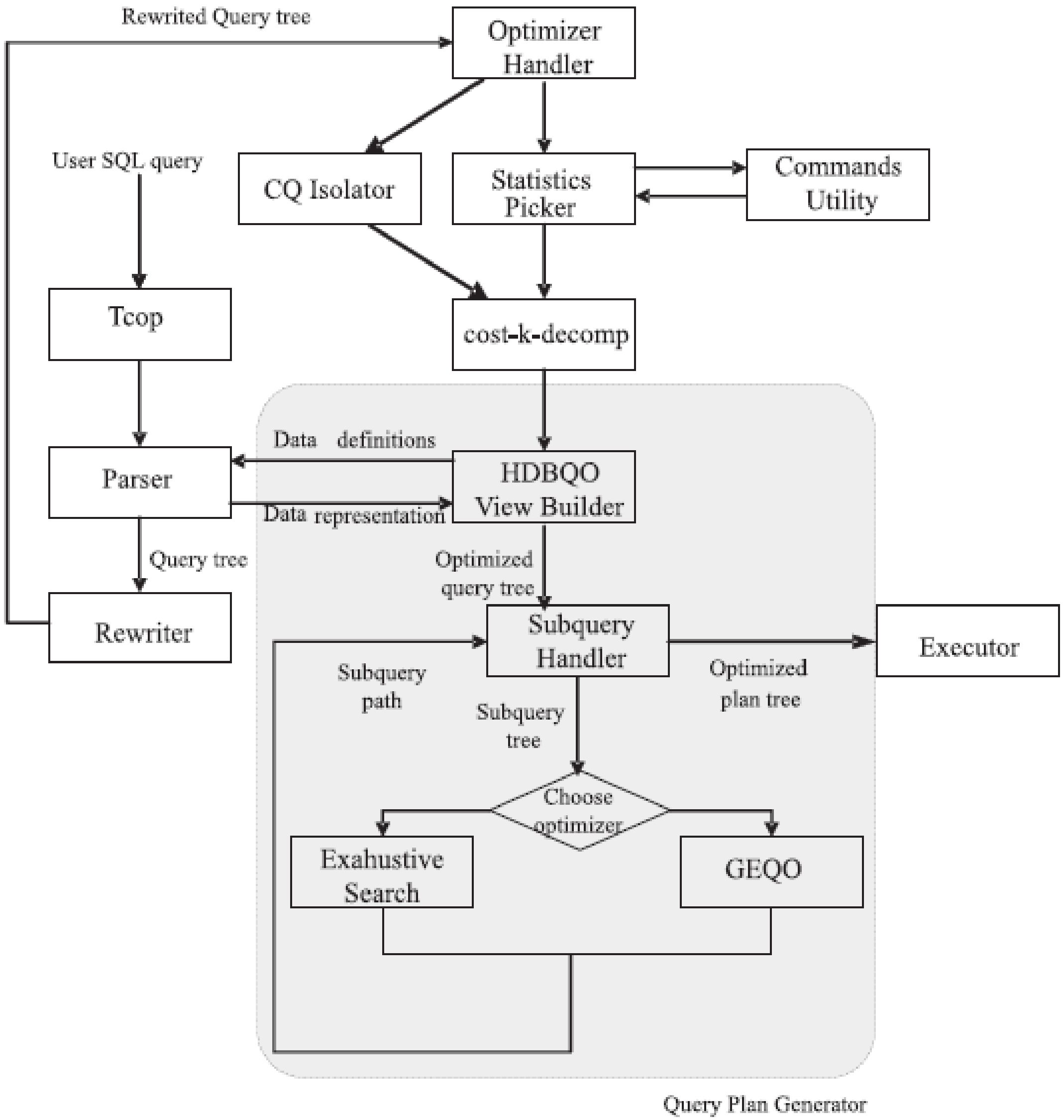
- More information in the appendix

Hypertrees for Databases

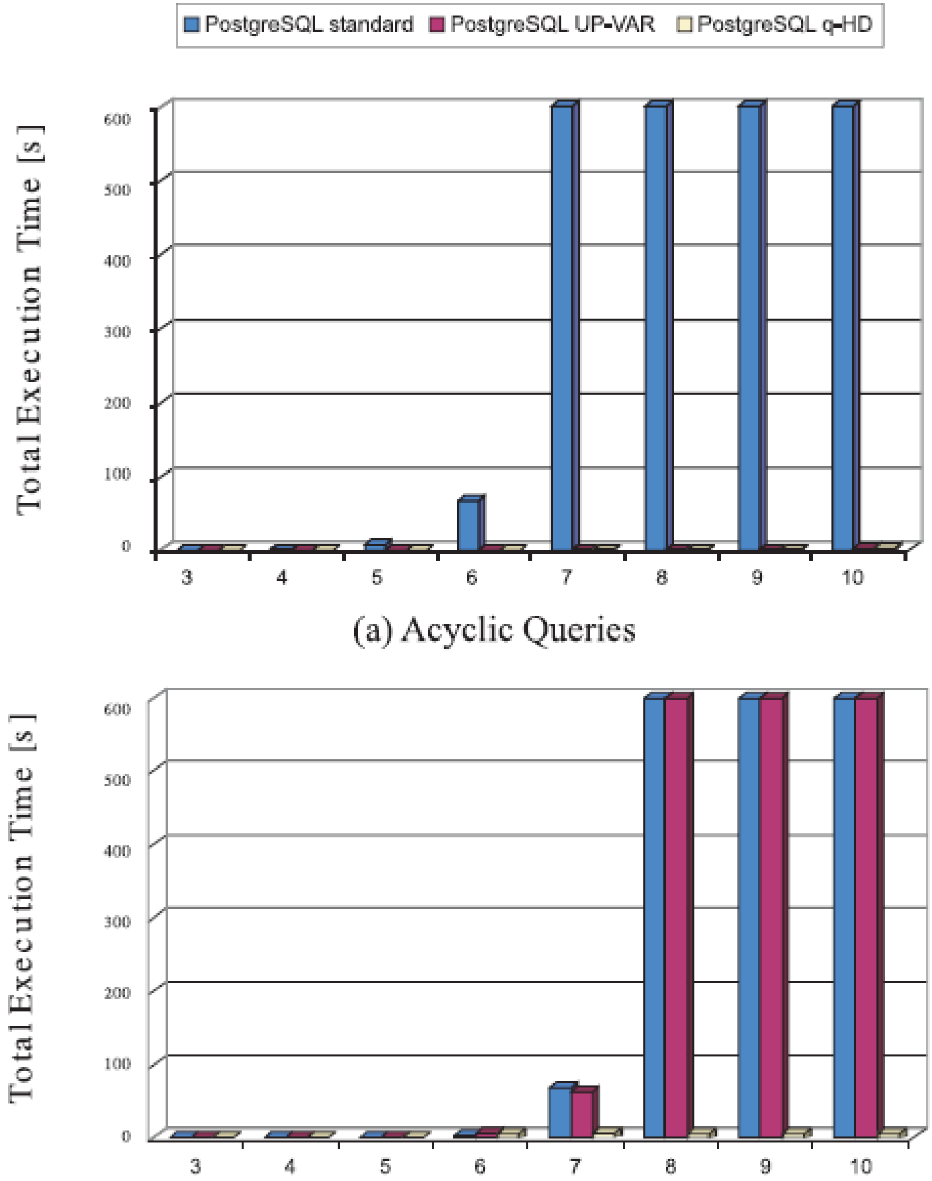


Weighted HDs, which exploit quantitative data, too.

Inside PostgreSQL



Some experiments



(a) Acyclic Queries

(b) Chain Queries

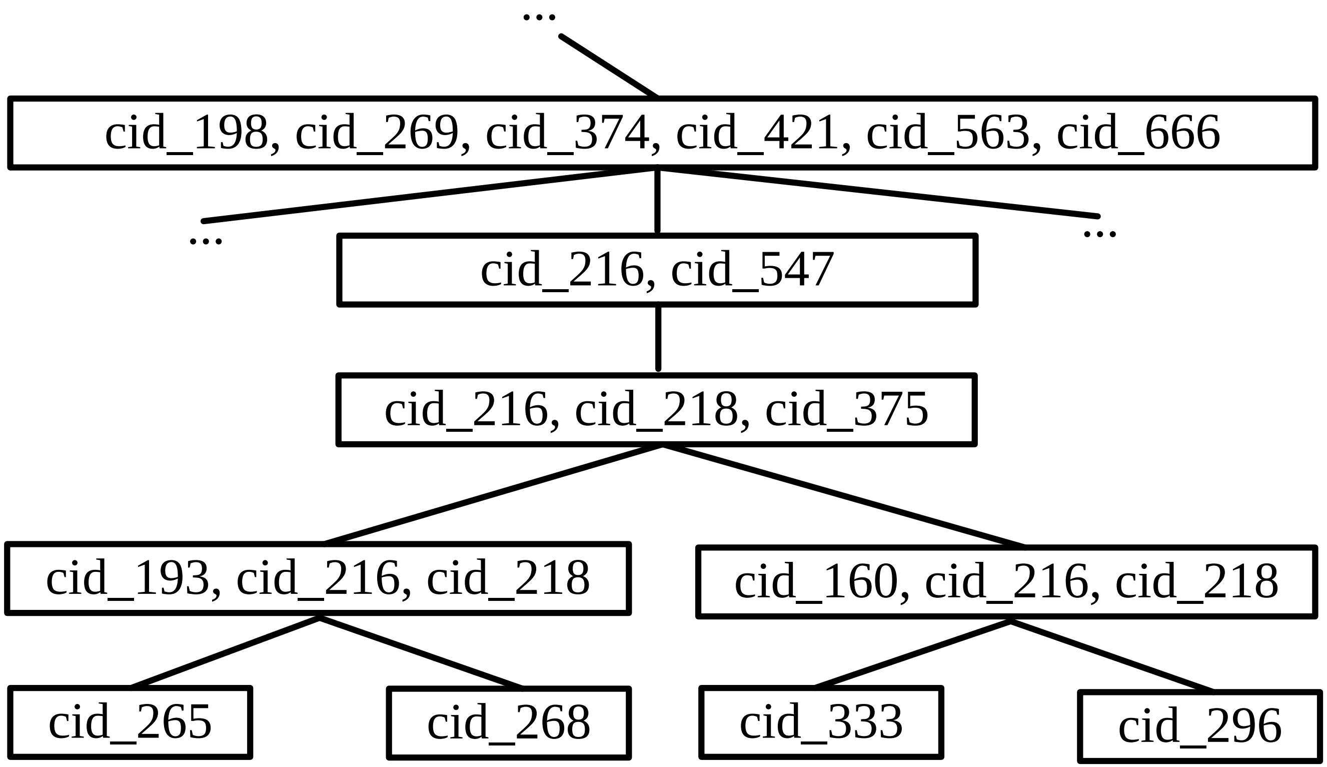
Large width example: Nasa problem

Part of relations for the Nasa problem

```
…
cid_260(Vid_49, Vid_366, Vid_224),
cid_261(Vid_100, Vid_391, Vid_392),
cid_262(Vid_273, Vid_393, Vid_246),
cid_263(Vid_329, Vid_394, Vid_249),
cid_264(Vid_133, Vid_360, Vid_356),
cid_265(Vid_314, Vid_348, Vid_395),
cid_266(Vid_67, Vid_352, Vid_396),
cid_267(Vid_182, Vid_364, Vid_397),
cid_268(Vid_313, Vid_349, Vid_398),
cid_269(Vid_339, Vid_348, Vid_399),
cid_270(Vid_98, Vid_366, Vid_400),
cid_271(Vid_161, Vid_364, Vid_401),
cid_272(Vid_131, Vid_353, Vid_234),
cid_273(Vid_126, Vid_402, Vid_245),
cid_274(Vid_146, Vid_252, Vid_228),
cid_275(Vid_330, Vid_360, Vid_361),
…
```

- 680 relations
- 579 variables

Nasa problem: Hypertree


…
cid_198, cid_269, cid_374, cid_421, cid_563, cid_666
…
cid_216, cid_547
…
cid_216, cid_218, cid_375
cid_193, cid_216, cid_218
cid_160, cid_216, cid_218
cid_265
cid_268
cid_333
cid_296

Part of hypertree for the Nasa problem

Best known hypertree-width for the Nasa problem is 22

Further Structural Methods

- Many proposals in the literature, besides (generalized) hypertree width (see [Gottlob, Leone, Scarcello. Art. Int.'00])
- For the binary case, the method based on tree decompositions (first proposed as heuristics in [Dechter and Pearl. Art.Int.'88 and Art.Int.'89]) is the most powerful [Grohe. J.ACM'07]
- Let us recall some recent proposals for the general (non-binary) case:
  - Fractional hypertree width [Grohe and Marx. SODA'06]
  - Spread-cut decompositions [Cohen, Jeavons, and Gyssens. J.CSS'08]
  - Component Decompositions [Gottlob,Miklòs,and Schwentick. J.ACM'09]
  - Greedy tree projections [Greco and Scarcello, PODS'10, ArXiv'12]
- Computing a width-k decomposition is in PTIME for all of them (for any fixed k>0).
- If we relax the above requirement, we can consider fixed-parameter tractable methods. If the size of the hypergraph structure is the fixed parameter, the most powerful is the Submodular width [Marx. STOC'10]

Heuristics for large width instances (CSPs)

1. Computing decompositions
    - Heuristics to get variants of (hyper)tree decompositions
2. Evaluating instances
    - Computing all solutions of the subproblems involved at each node may be prohibitive
    - Memory explosion

- Solution: combine with other techniques. E.g., in CSPs,
    - use (hyper)tree decompositions for bounding the search space [Otten and Dechter. UAI'08]
    - use (hyper)tree decompositions for improving the performance of consistency algorithms (hence, speeding-up propagations) [Karakashian, Woodward, and Choueiry. AAAI'13]
    - …

Alternative constraint encodings

- Some tractability results hold only on constraint encodings where allowed tuples are listed as finite relations
- Alternative encodings make sense
- For instance,
  - constraint satisfaction with succinctly specified relations [Chen and Grohe. J.CSS'10]
  - see also [Cohen, Green, and Houghton. CP'09]

Outline of PART II

Beyond Tree Decompositions

Applications to Databases and CSPs

Structural and Consistency Properties

Local (pairwise) consistency

- For every relation/constraint:
  each tuple matches some tuple in every other relation
- Can be enforced in polynomial time:
  take the join of all pairs of relations/constraints until a fixpoint is reached, or some relation becomes empty

See [Beeri, Fagin, Maier, and Yannakakis. J.ACM’83] or [Janssen, Jégou, Nougier, and Vilarem. IEEE WS Tools for AI’89],

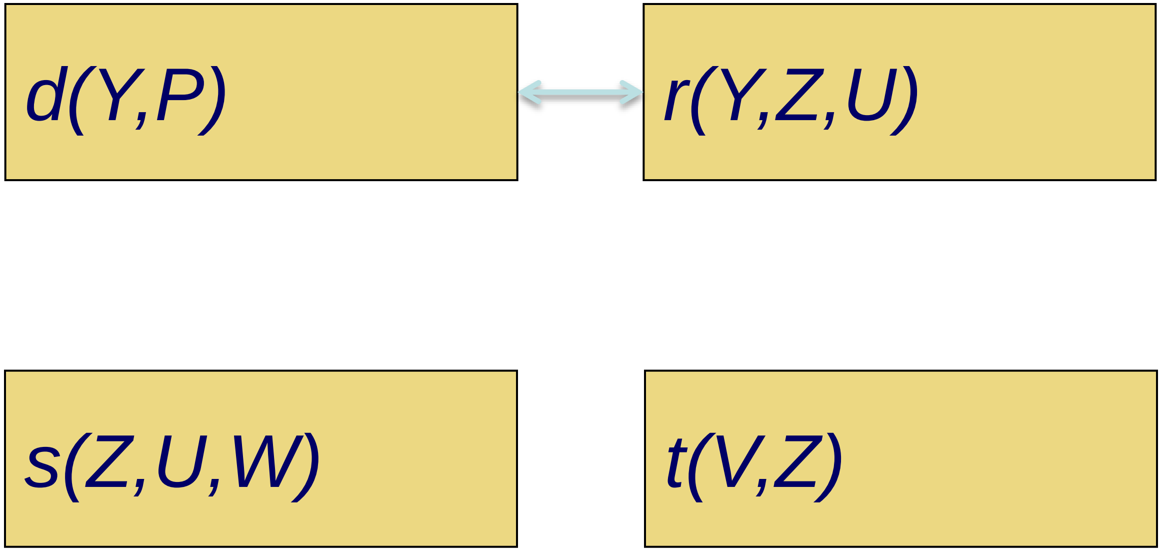
Enforcing pairwise consistency

d:

3	8
3	7
~~5~~	~~7~~
~~6~~	~~7~~

r:

3	8	9
~~9~~	~~3~~	~~8~~
~~8~~	~~3~~	~~8~~
3	8	4
3	8	3
~~8~~	~~9~~	~~4~~
~~9~~	~~4~~	~~7~~



d(Y,P)

r(Y,Z,U)

s(Z,U,W)

t(V,Z)

s:

3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

t:

9	8
9	3
9	5

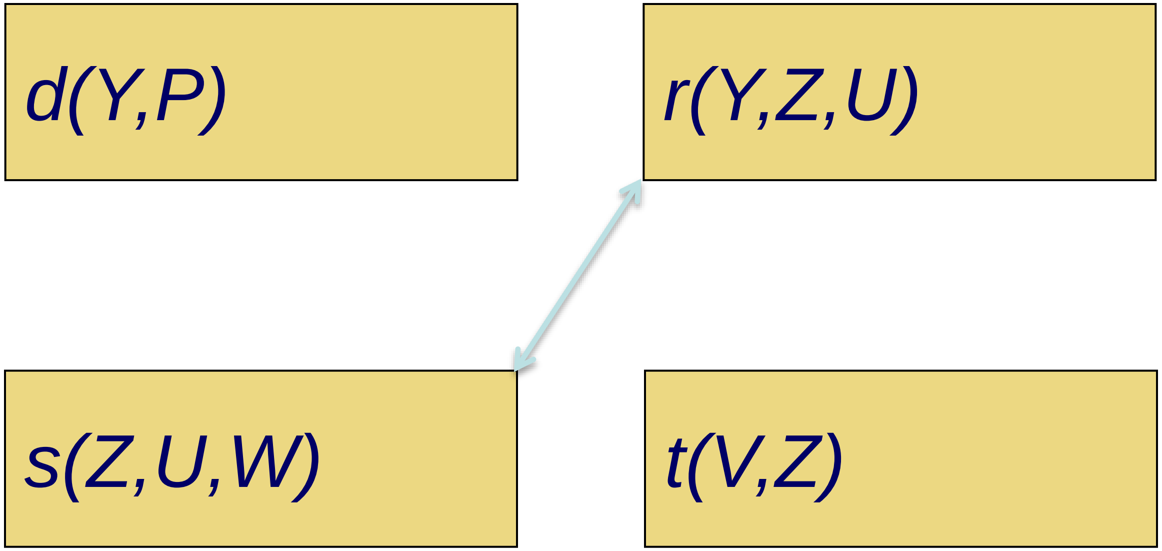
Enforcing pairwise consistency

d:

3	8
3	7
~~5~~	~~7~~
~~6~~	~~7~~

r:

3	8	9
~~9~~	~~3~~	~~8~~
~~8~~	~~3~~	~~8~~
~~3~~	~~8~~	~~4~~
3	8	3
~~8~~	~~9~~	~~4~~
~~9~~	~~4~~	~~7~~



d(Y,P)

r(Y,Z,U)

s(Z,U,W)

t(V,Z)

s:

~~3~~	~~8~~	~~9~~
~~9~~	~~3~~	~~8~~
8	3	8
~~3~~	~~8~~	~~4~~
~~3~~	~~8~~	~~3~~
8	9	4
~~9~~	~~4~~	~~7~~

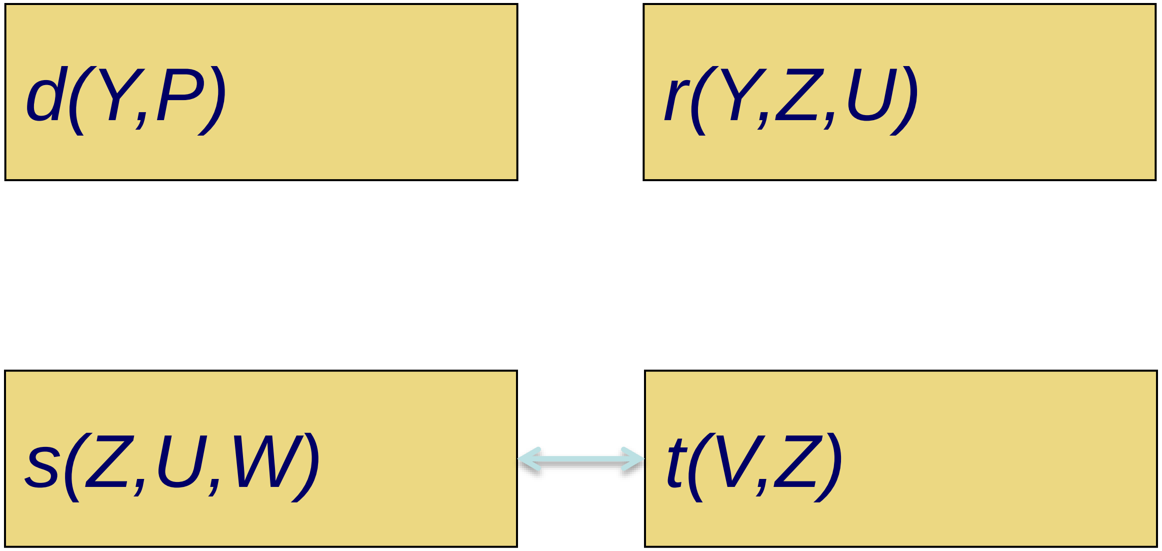
t:

9	8
9	3
9	5

Enforcing pairwise consistency

d:

3	8
3	7
~~5~~	~~7~~
~~6~~	~~7~~



r:

3	8	9
~~9~~	~~3~~	~~8~~
~~8~~	~~3~~	~~8~~
~~3~~	~~8~~	~~4~~
3	8	3
~~8~~	~~9~~	~~4~~
~~9~~	~~4~~	~~7~~

s:

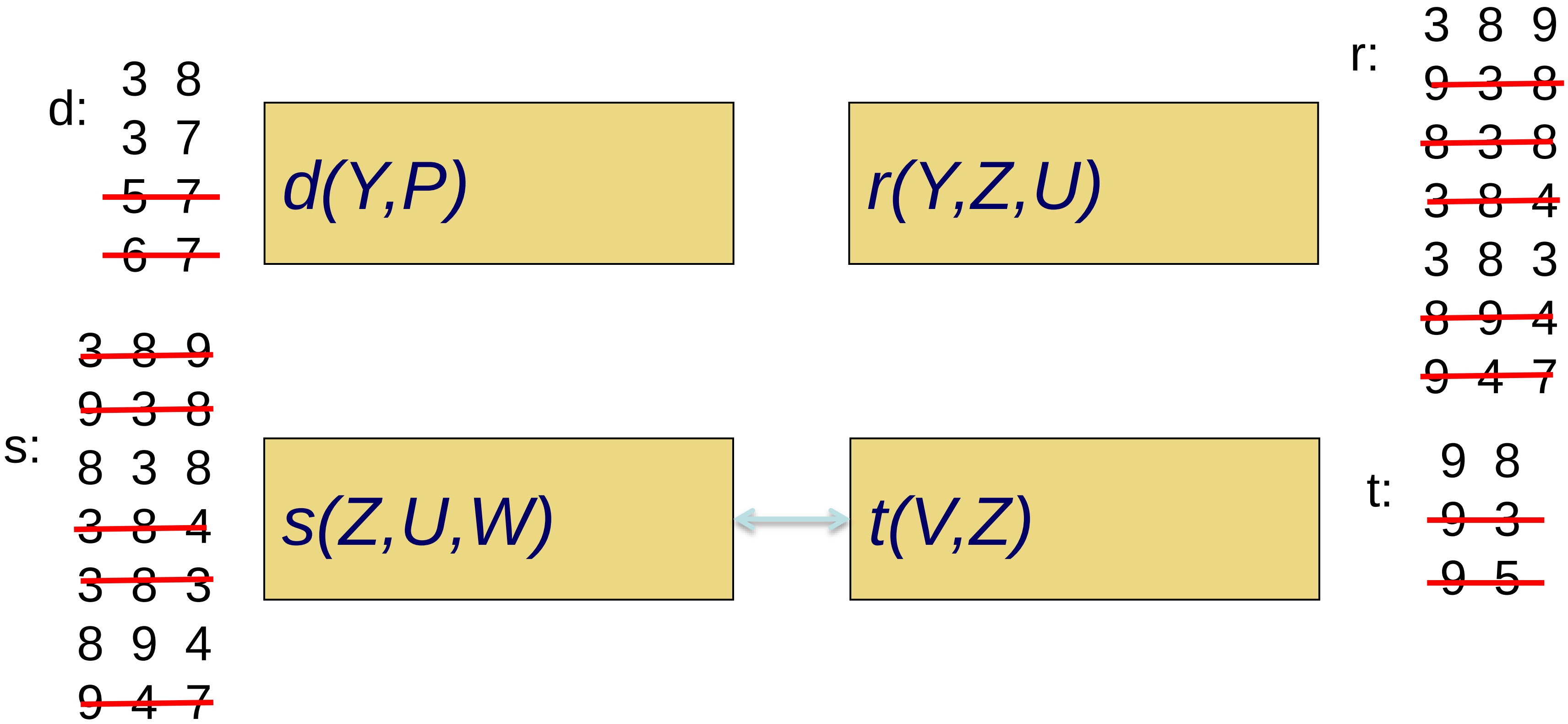
~~3~~	~~8~~	~~9~~
~~9~~	~~3~~	~~8~~
8	3	8
~~3~~	~~8~~	~~4~~
~~3~~	~~8~~	~~3~~
8	9	4
~~9~~	~~4~~	~~7~~

t:

9	8
~~9~~	~~3~~
~~9~~	~~5~~

Enforcing pairwise consistency

- Further steps are useless, because the instance is now locally consistent
- On acyclic instances,
  same result as Yannakakis' algorithm on the join tree!



d:
3 8
3 7
~~5 7~~
~~6 7~~

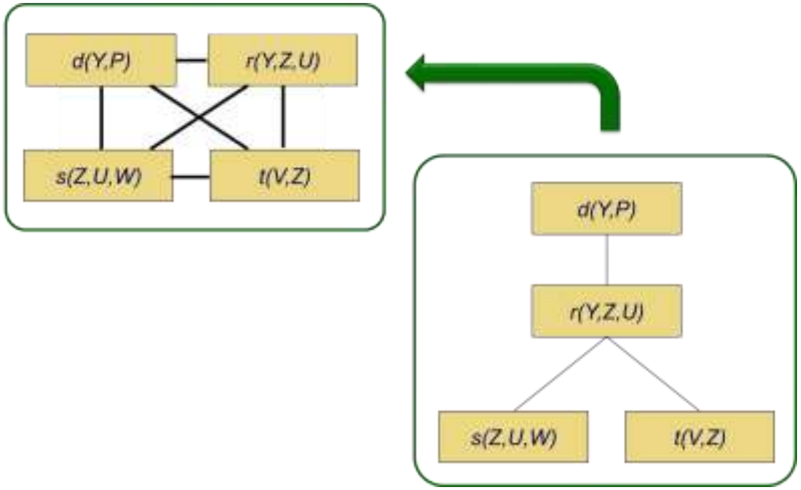
r:
3 8 9
~~9 3 8~~
~~8 3 8~~
~~3 8 4~~
3 8 3
~~8 9 4~~
~~9 4 7~~

s:
~~3 8 9~~
~~9 3 8~~
8 3 8
~~3 8 4~~
~~3 8 3~~
8 9 4
~~9 4 7~~

t:
9 8
~~9 3~~
~~9 5~~

Easy on Acyclic Instances

- Computing a join tree
  (in linear time, and logspace-complete [GLS'98+ SL=L])
  may be viewed as a clever way to enforce pairwise consistency



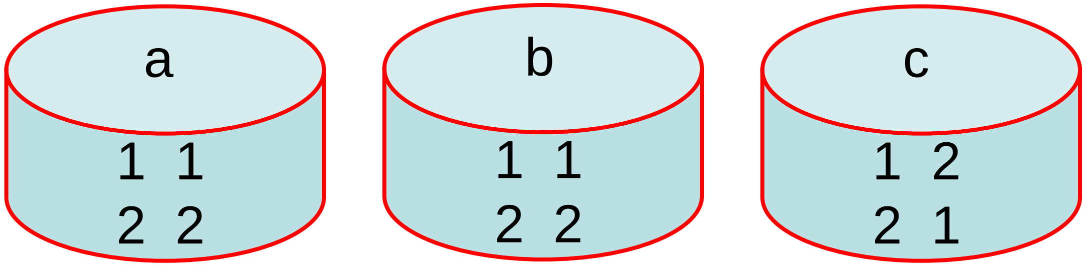
- Cost for the computation of the locally consistent instance: $O(m\ n^2 \log n)$ vs $O(m\ n \log n)$

- N.B. n is the (maximum) number of tuples in a relation and may be very large (esp. in database applications)

Global and pairwise Consistency

- Yannakakis' algorithm actually solves acyclic instances because of their following crucial property:
  - Local (pairwise) consistency ➔ Global consistency [Beeri, Fagin, Maier, and Yannakakis. J.ACM'83]
  - Global consistency: Every tuple in each relation can be extended to a full (global) solution
  - In particular, if all relations/constraints are pairwise consistent, then the result is not empty
- Not true in the general case:

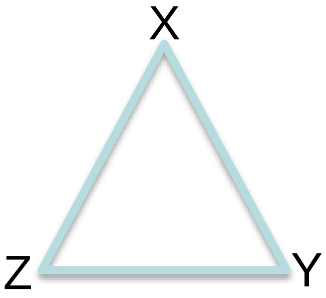
$$ans :\text{-}\ a(X,Y) \wedge b(Y,Z) \wedge c(Z,X)$$



a	
1	1
2	2

b	
1	1
2	2

c	
1	2
2	1



Consistency in Databases and CSPs

- Huge number of works in the database and constraint satisfaction literature about different kinds (and levels) of consistencies
  (e.g., recall the seminal paper [Mackworth. Art. Int., 1977]
  or [Beeri, Fagin, Maier, and Yannakakis. J.ACM'83],
  [Dechter. Art. Int., 1992], and [Dechter and van Beek. TCS'97])
- Most theoretical papers in the database community
- Also practical papers in the constraint satisfaction community:
  - Local consistencies are crucial for filtering domains and constraints
  - Allow tremendous speed-up in constraint solvers
  - Sometimes allow backtrack-free computations

Global consistency in Databases and CSPs

- Global consistency (GC): Every tuple in each relation can be extended to a full (global) solution
  [Beeri, Fagin, Maier, and Yannakakis. J.ACM'83]
- For instances with *m* constraints, it is also known as
  - m-wise consistency [Gyssens. TODS'86]
  - relational (i;m)-consistency [Dechter and van Beek. TCS'97]
  - ***R(*,m)C*** [Karakashian, Woodward, Reeson, Choueiry and Bessiere. AAAI'10]
  - …
- Remark:

  In the CSP literature, "global consistent network" sometimes means "strongly n-consistent network", which is a different notion (see, e.g., [Constraint Processing, Dechter, 2003]).

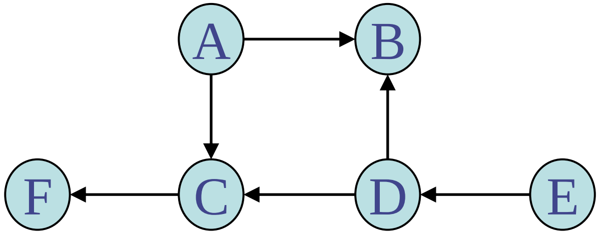
On the desirability of Global Consistency

- If an instance is globally consistent, we can immediately read partial solutions from the constraint/database relations
- full solutions are often computed efficiently
- can be exploited in heuristics by constraint solvers. For a very recent example, see
  - [Karakashian, Woodward, and Choueiry. AAAI’13]: enforce global consistency on groups of subproblems (tree-like arranged) for bolstering propagations

When pairwise consistency entails GC

- We have seen that it happens in acyclic instances…
- Is it the case that this condition is also necessary?
- What is the real power of local (pairwise) consistency? i.e., relational **arc-consistency** (more precisely, arc-consistency on the dual graph)

  Also known as
  - 2-wise consistency [Gyssens. TODS'86],
  - R(*,2)C [Karakashian, Woodward, Reeson, Choueiry and Bessiere. AAAI'10]
  - …



When pairwise consistency entails GC

- We have seen that it happens in acyclic instances…
- The classical result that this is also necessary
  [Beeri, Fagin, Maier, and Yannakakis. J.ACM'83]
  actually holds only if relations cannot be used in more than one constraint/query atoms
- In fact, it works even on some cyclic instances
- We now have a precise structural characterization of the instances where local consistency entails global consistency
  - it applies to the binary case, too
  - it applies to the more general case where pairwise consistency is enforced between each pair of arbitrary defined subproblems (see appendix)!

[Greco and Scarcello. PODS'10]

The Power of Pairwise Consistency

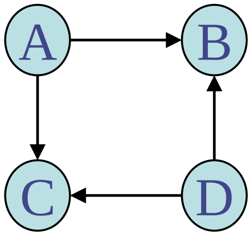
- Let us describe when local (pairwise) consistency (LC) entails global consistency (GC), on the basis of the constraint structure

- That is, we describe the condition such that:
  - whenever it holds, LC entails GC for every possible CSP instance (i.e., no matter on the constraint relations)
  - if it does not hold, there exists an instance where LC fails

- For binary (or fixed arity) instances: if we are interested only in the decision problem (is the CSP satisfiable?) than this condition is the existence of an acyclic core [Atserias, Bulatov, and Dalmau. ICALP’07]

The Power of Pairwise Consistency

- Does pairwise consistency entail global consistency in this case?

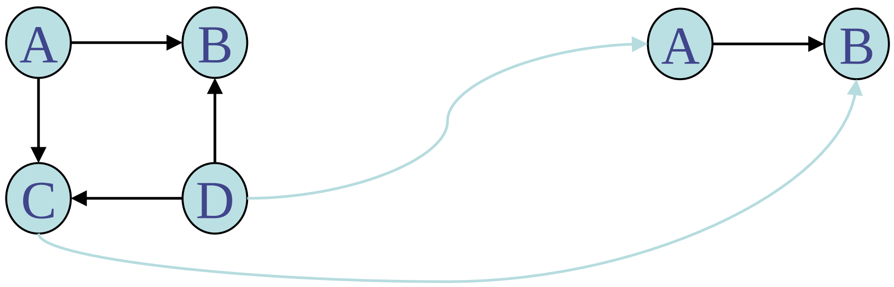


Constraints

e(A,B)
e(A,C)
e(D,C)
e(D,B)

The Power of Pairwise Consistency

- Does pairwise consistency entail global consistency in this case?
- Yes! No matter of the tuples in the constraint relation ***e***
- Every constraint is a core of the instance

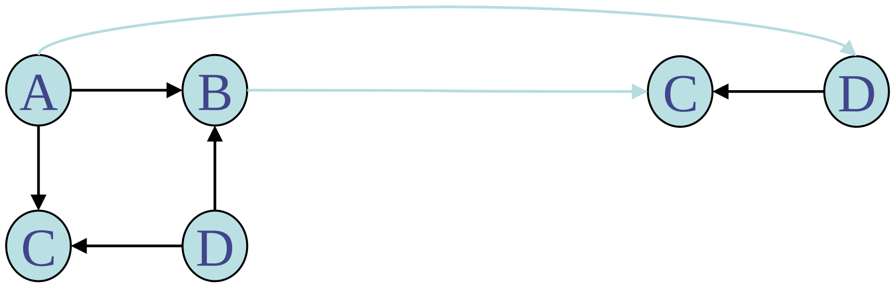


Constraints

e(A,B)
e(A,C)
e(D,C)
e(D,B)

The Power of Pairwise Consistency

- Does pairwise consistency entail global consistency in this case?
- Yes! No matter of the tuples in the constraint relation ***e***
- Every constraint is a core of the instance



Constraints

e(A,B)
e(A,C)
e(D,C)
e(D,B)

tp-covering (acyclic version)

- The constraint e(X,Y) is **tp-covered** in an acyclic hypergraph if,
  - add a fresh constraint e’(X,Y) (where e’ is a fresh relational symbol),
  - a core of the new instance has an acyclic hypergraph

- Intuitively the “coloring” of *e(X,Y)* forces the core of the new structure to deal with the ordered pair (X,Y)
  - Indeed, every core must contain *e’(X,Y)*
- Instead, the usual notion of the core does not preserve the meaning of variables
  - this is crucial for computing solutions, but not for the decision problem

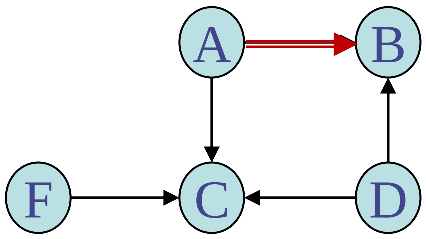
The Power of Pairwise Consistency

- The constraint e(X,Y) is **tp-covered** in an acyclic hypergraph if,
    - add a fresh constraint e’(X,Y) (where e’ is a fresh relational symbol),
    - a core of the new instance has an acyclic hypergraph

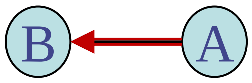
Local (pairwise) consistency entails Global consistency if and only if every constraint is tp-covered in an acyclic hypergraph

tp-covering by Example

- The constraint e(X,Y) is tp-covered in an acyclic hypergraph if,
  - add a fresh constraint e’(X,Y) (where e’ is a fresh relational symbol),
  - a core of the new instance has an acyclic hypergraph



e(A,B) is tp-covered

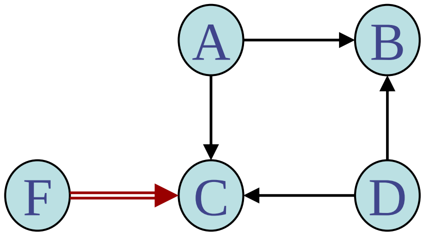


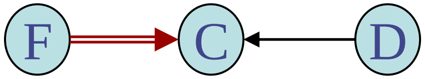
Note that e(F,C) does not occur in any core

tp-covering by Example

- The constraint e(X,Y) is tp-covered in an acyclic hypergraph if,
  - add a fresh constraint e'(X,Y) (where e' is a fresh relational symbol),
  - a core of the new instance has an acyclic hypergraph

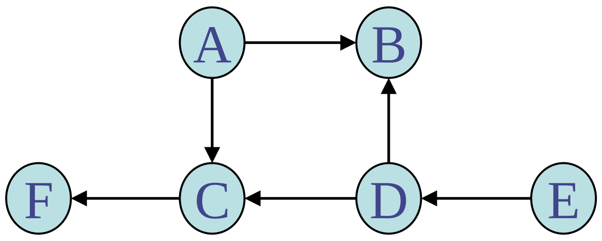
e(F,C) is tp-covered

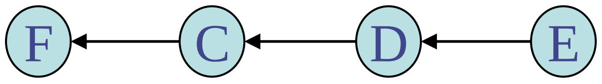




tp-covering by Example

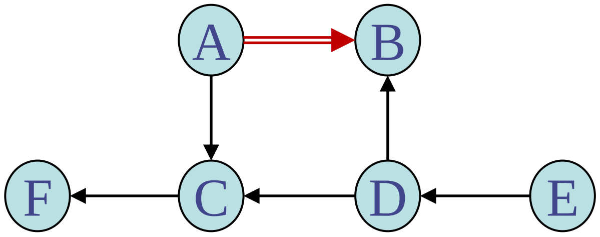
- Here pairwise consistency solves the satisfaction problem
- The structure of any core is an undirected acyclic graph





The power of Pairwise Consistency

- Here pairwise consistency solves the satisfaction problem
- The structure of any core is an undirected acyclic graph
- However, it does not entail global consistency
- There is an instance that is pairwise consistent but *e(A,B)* contains wrong tuples



e(A,B) is not tp-covered:
the core of the new structure
is cyclic

A generalization: Local k-consistency

- Consider subproblems of k constraints
- *Local k-consistency*: pairwise consistency over such (k-constraints) subproblems
  Equivalent to *relational k-consistency* [Dechter and van Beek. TCS'97]

Local k-consistency entails Global consistency if and only if every constraint is tp-covered in a hypergraph having Generalized Hypertree width k

[Greco and Scarcello. PODS'10]

- See the appendix for a further generalization to arbitrary subproblems in the general framework of
  tree projections

Outline of Part III

Applications to Optimization Problems

Application: Nash Equilibria

Application: Coalitional Games

Application: Combinatorial Auctions

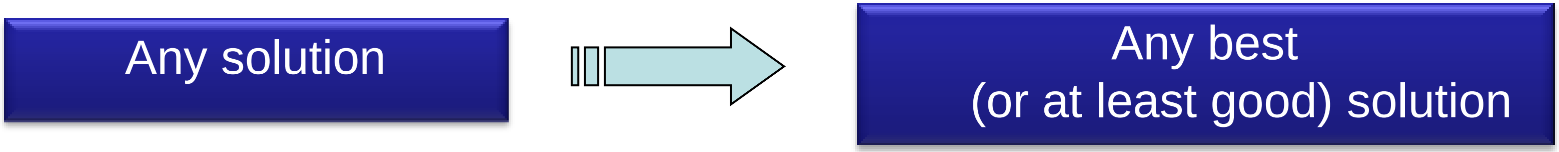
Appendix: Beyond Hypertree Width

Outline of Part III

- Applications to Optimization Problems
- Application: Nash Equilibria
- Application: Coalitional Games
- Application: Combinatorial Auctions
- Appendix: Beyond Hypertree Width

Constraint Optimization Problems

- Classically, CSP: Constraint Satisfaction Problem
- However, sometimes a solution is
  enough to “satisfy” (constraints),
  but not enough to make (users) “happy”

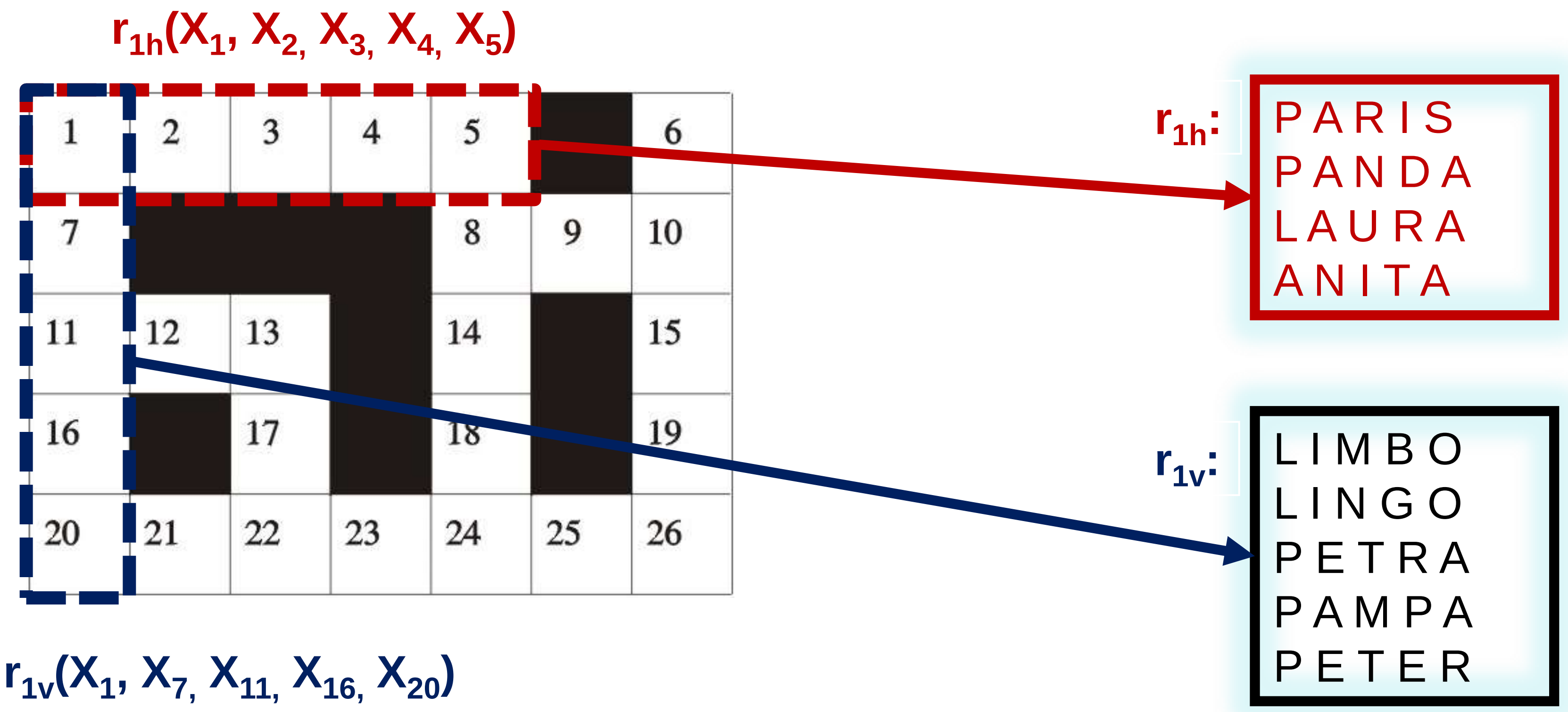


Any solution

Any best
(or at least good) solution

- Hence, several variants of the basic CSP framework:
  - E.g., fuzzy, probabilistic, weighted, lexicographic, penalty, valued, semiring-based, …

Classical CSPs



$r_{1h}(X_1, X_2, X_3, X_4, X_5)$

$r_{1v}(X_1, X_7, X_{11}, X_{16}, X_{20})$

r_{1h}:

PARIS
PANDA
LAURA
ANITA

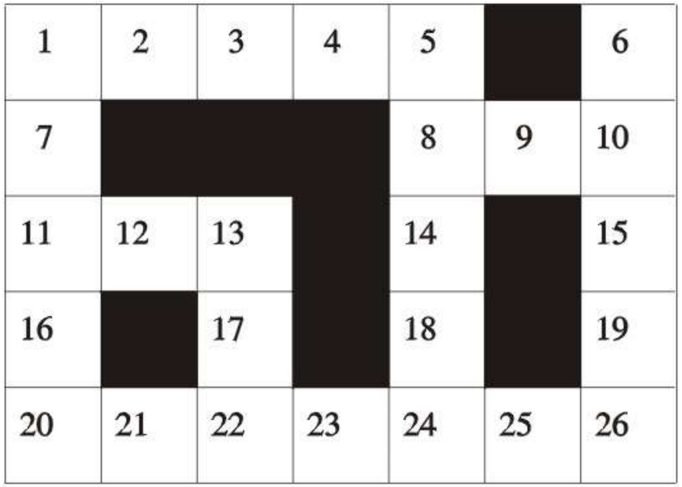
r_{1v}:

LIMBO
LINGO
PETRA
PAMPA
PETER

- Set of variables $\{X_1,\ldots,X_{26}\}$
- Set of constraint scopes

- Set of constraint relations

Puzzles for Experts…



1	2	3	4	5	■	6
7	■	■	■	8	9	10
11	12	13	■	14	■	15
16	■	17	■	18	■	19
20	21	22	23	24	25	26

The puzzle in general admits more than one solution...

- E.g., find the solution that **minimizes** the total number of vowels occurring in the words

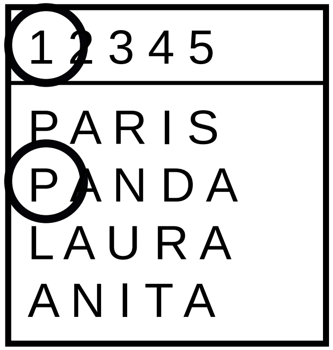
A Classification for Optimization Problems

CSOP

Each mapping variable-value has a cost.

Then, find an assignment:

- Satisfying all the constraints, and
- Having the minimum total cost.



1 2 3 4 5

PARIS

PANDA

LAURA

ANITA

A Classification for Optimization Problems

IJCAI-13

CSOP

Each mapping variable-value has a cost.

Then, find an assignment:

- Satisfying all the constraints, and
- Having the minimum total cost.

WCSP

Each tuple has a cost.

Then, find an assignment:

- Satisfying all the constraints, and
- Having the minimum total cost.

12345
PARIS
[✓ PANDA]
LAURA
ANITA

A Classification for Optimization Problems

CSOP

Each mapping variable-value has a cost.

Then, find an assignment:

- Satisfying all the constraints, and
- Having the minimum total cost.

WCSP

Each tuple has a cost.

Then, find an assignment:

- Satisfying all the constraints, and
- Having the minimum total cost.

MAX-CSP

Each constraint relation has a cost.

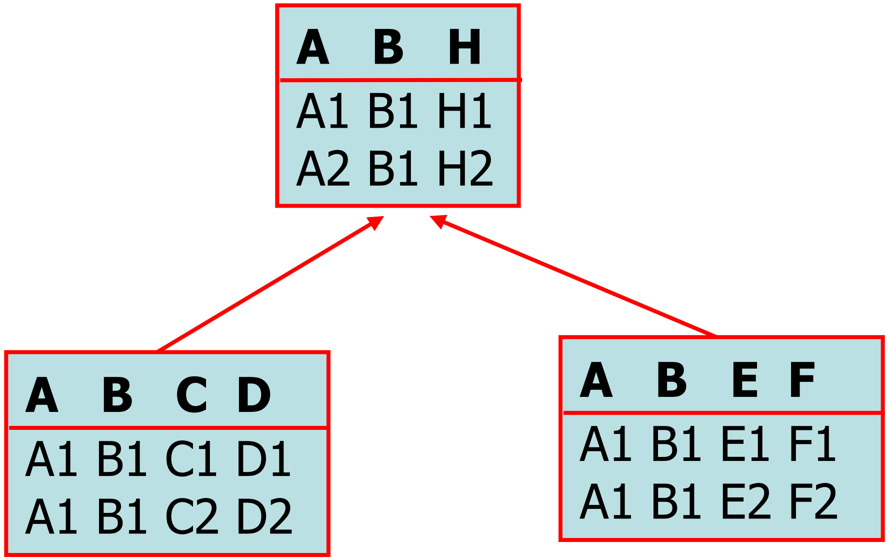
Then, find an assignment:

- Minimizing the cost of violated relations.

1	2	3	4	5
P	A	R	I	S
P	A	N	D	A
L	A	U	R	A
A	N	I	T	A

CSOP: Tractability of Acyclic Instances

- Adapt the dynamic programming approach in **(Yannakakis'81)**



A	B	H
A1	B1	H1
A2	B1	H2

A	B	C	D
A1	B1	C1	D1
A1	B1	C2	D2

A	B	E	F
A1	B1	E1	F1
A1	B1	E2	F2

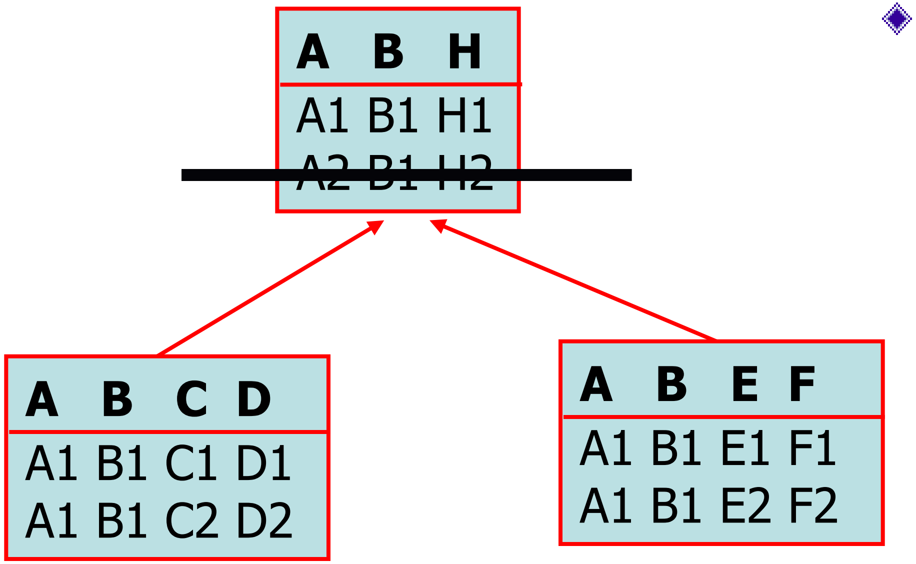
[Gottlob & Greco, EC'07]

CSOP: Tractability of Acyclic Instances

- Adapt the dynamic programming approach in **(Yannakakis'81)**

With a bottom-up computation:

- Filter the tuples that do not match



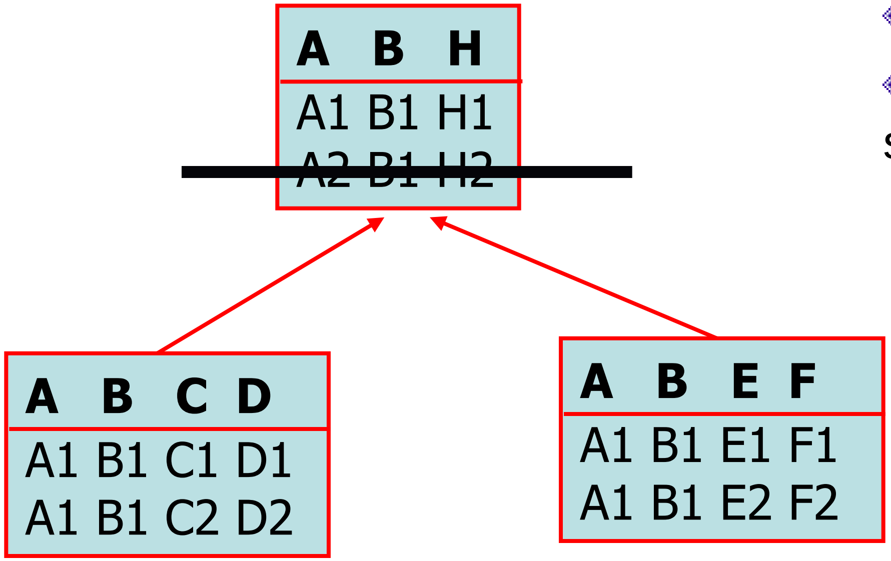
A	B	H
A1	B1	H1
~~A2~~	~~B1~~	~~H2~~

A	B	C	D
A1	B1	C1	D1
A1	B1	C2	D2

A	B	E	F
A1	B1	E1	F1
A1	B1	E2	F2

CSOP: Tractability of Acyclic Instances

- Adapt the dynamic programming approach in **(Yannakakis'81)**



A	B	H
A1	B1	H1
~~A2~~	~~B1~~	~~H2~~

A	B	C	D
A1	B1	C1	D1
A1	B1	C2	D2

A	B	E	F
A1	B1	E1	F1
A1	B1	E2	F2

With a bottom-up computation:

- Filter the tuples that do not match
- Compute the cost of the best partial solution, by looking at the children

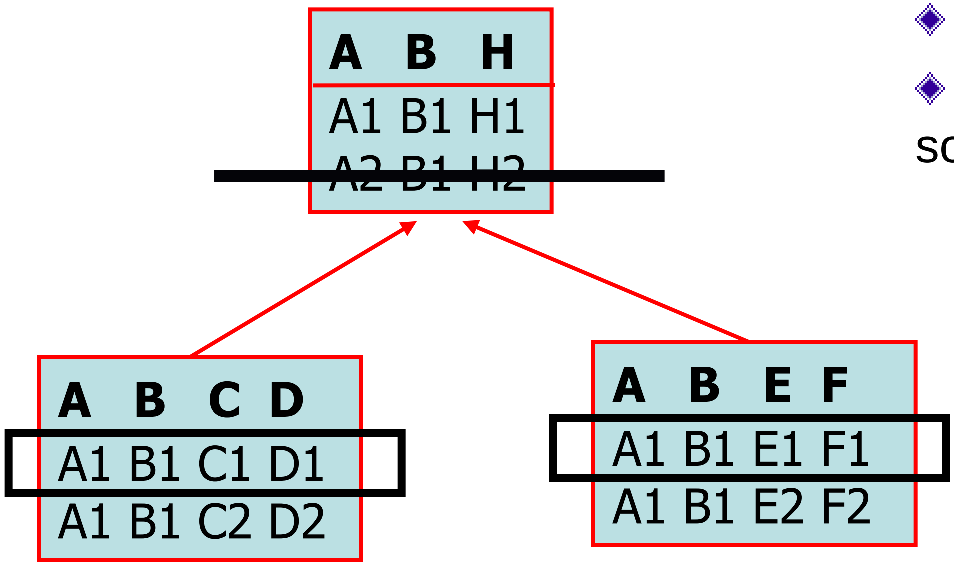
cost(C/C1)=cost(D/D1)=0
cost(C/C2)=cost(D/D2)=1
cost(E/E1)=cost(F/F1)=0
cost(E/E2)=cost(F/F2)=1

CSOP: Tractability of Acyclic Instances

- Adapt the dynamic programming approach in **(Yannakakis'81)**

With a bottom-up computation:

- Filter the tuples that do not match
- Compute the cost of the best partial solution, by looking at the children



A	**B**	**H**
A1	B1	H1
~~A2~~	~~B1~~	~~H2~~

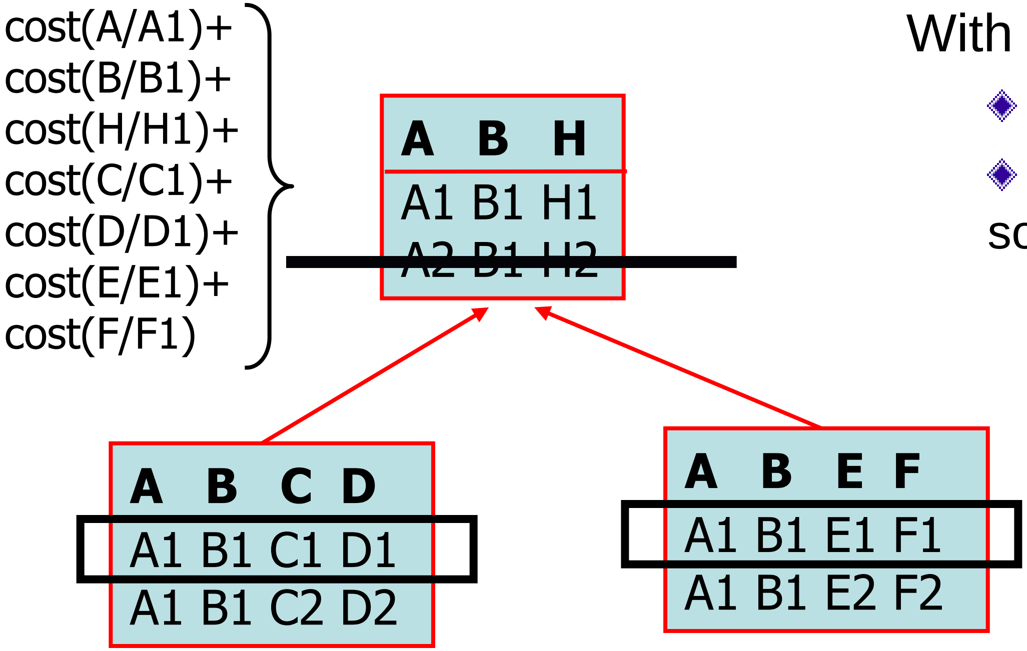
A	**B**	**C**	**D**
A1	B1	C1	D1
A1	B1	C2	D2

A	**B**	**E**	**F**
A1	B1	E1	F1
A1	B1	E2	F2

cost(C/C1)=cost(D/D1)=0
cost(C/C2)=cost(D/D2)=1
cost(E/E1)=cost(F/F1)=0
cost(E/E2)=cost(F/F2)=1

CSOP: Tractability of Acyclic Instances

- Adapt the dynamic programming approach in **(Yannakakis'81)**



cost(A/A1)+
cost(B/B1)+
cost(H/H1)+
cost(C/C1)+
cost(D/D1)+
cost(E/E1)+
cost(F/F1)

A	**B**	**H**
A1	B1	H1
~~A2~~	~~B1~~	~~H2~~

A	**B**	**C**	**D**
A1	B1	C1	D1
A1	B1	C2	D2

A	**B**	**E**	**F**
A1	B1	E1	F1
A1	B1	E2	F2

With a bottom-up computation:

- Filter the tuples that do not match
- Compute the cost of the best partial solution, by looking at the children

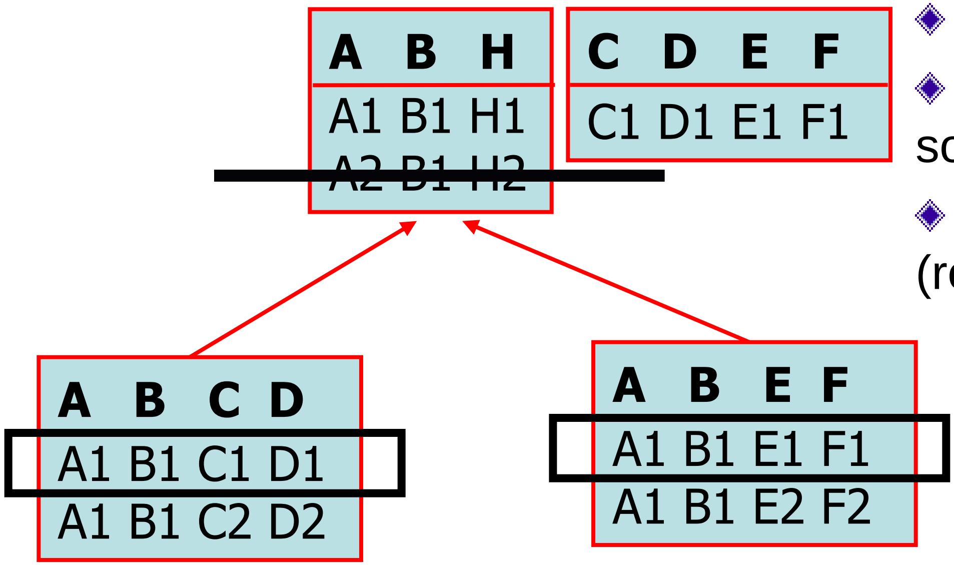
cost(C/C1)=cost(D/D1)=0
cost(C/C2)=cost(D/D2)=1
cost(E/E1)=cost(F/F1)=0
cost(E/E2)=cost(F/F2)=1

CSOP: Tractability of Acyclic Instances

- Adapt the dynamic programming approach in **(Yannakakis'81)**

With a bottom-up computation:

- Filter the tuples that do not match
- Compute the cost of the best partial solution, by looking at the children
- Propagate the best partial solution (resolving ties arbitrarily)



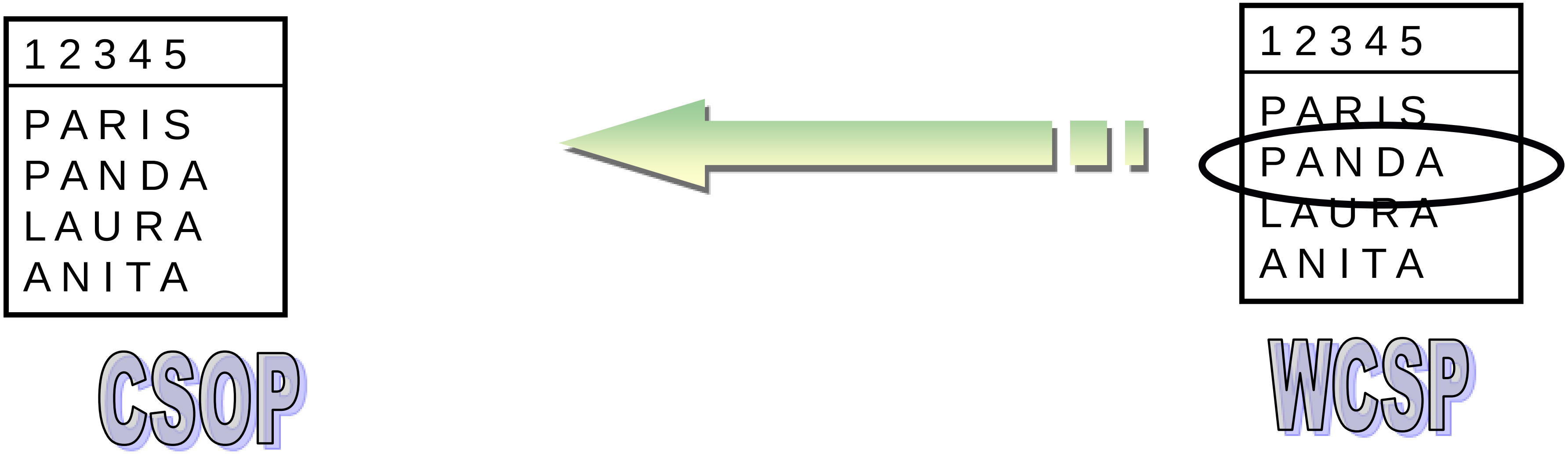
A	B	H
A1	B1	H1
~~A2~~	~~B1~~	~~H2~~

C	D	E	F
C1	D1	E1	F1

A	B	C	D
A1	B1	C1	D1
A1	B1	C2	D2

A	B	E	F
A1	B1	E1	F1
A1	B1	E2	F2

WCSP: Tractability of Acyclic Instances



12345

PARIS
PANDA
LAURA
ANITA

CSOP

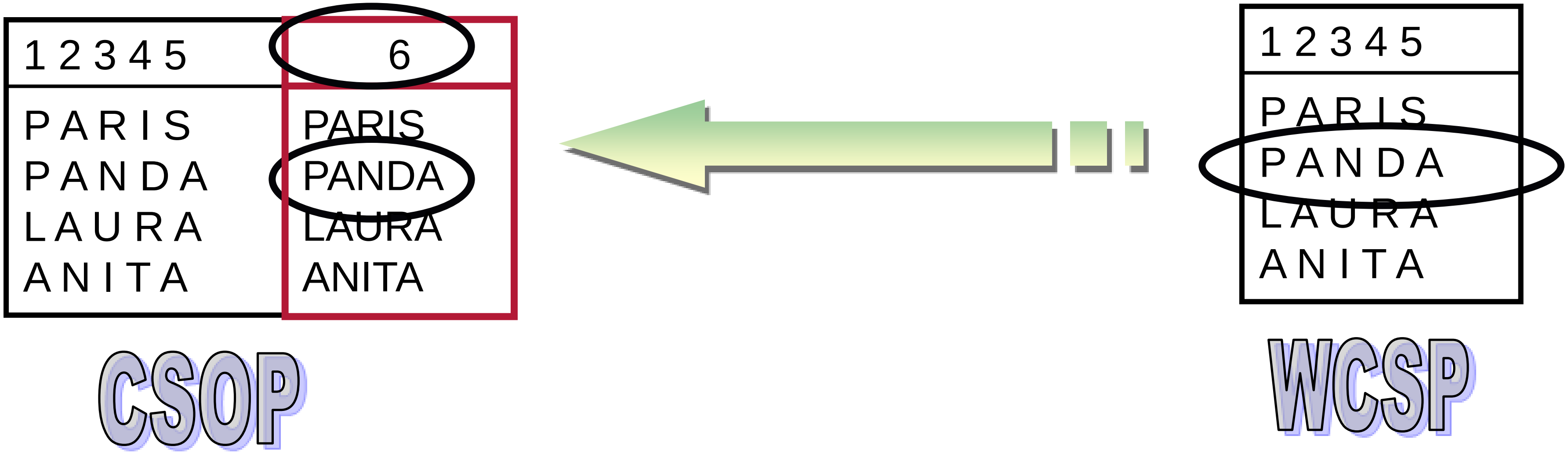
12345

PARIS
PANDA
LAURA
ANITA

WCSP

[Gottlob, Greco, and Scarcello, ICALP‘09]

WCSP: Tractability of Acyclic Instances



12345	6
PARIS	PARIS
PANDA	PANDA
LAURA	LAURA
ANITA	ANITA

CSOP

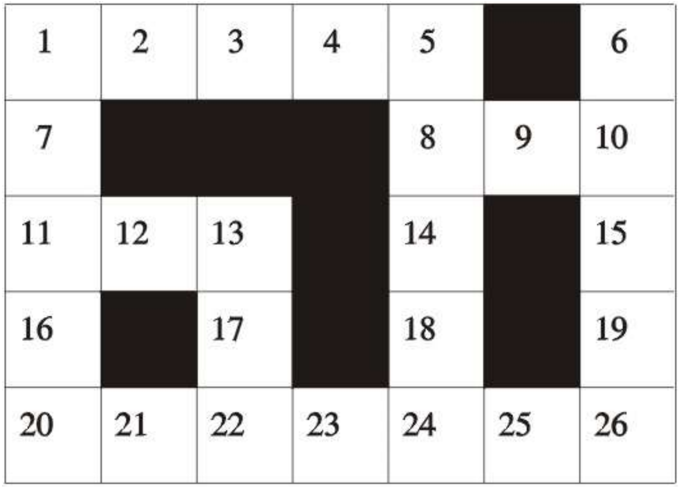
12345
PARIS
PANDA
LAURA
ANITA

WCSP

The mapping:

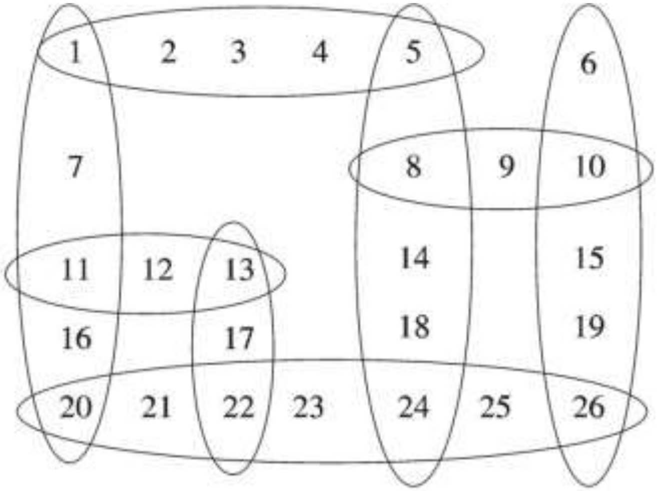
- Is feasible in linear time
- Preserves the solutions
- Preserves acyclicity

In-Tractability of MAX-CSP Instances



1	2	3	4	5	■	6
7	■	■	■	8	9	10
11	12	13	■	14	■	15
16	■	17	■	18	■	19
20	21	22	23	24	25	26

- Maximize the number of words placed in the puzzle

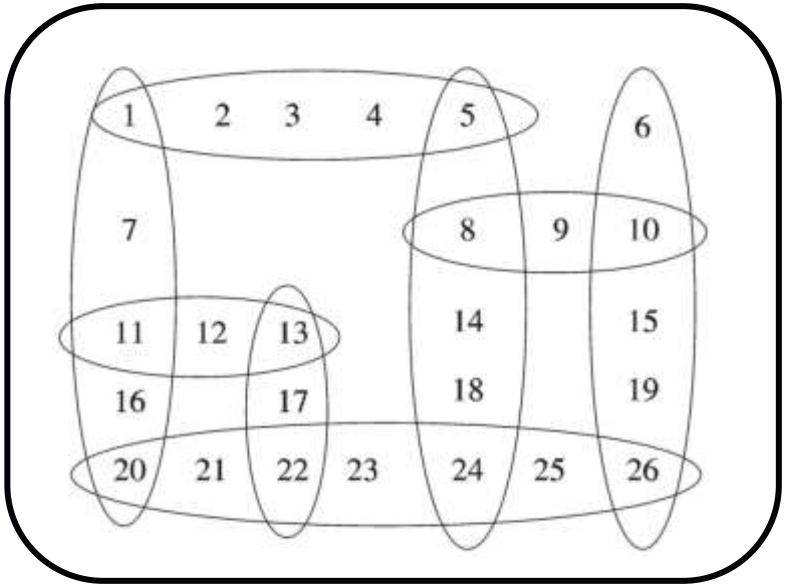


[Gottlob, Greco, and Scarcello, ICALP‘09]

In-Tractability of MAX-CSP Instances

1	2	3	4	5	■	6
7	■	■	■	8	9	10
11	12	13	■	14	■	15
16	■	17	■	18	■	19
20	21	22	23	24	25	26

- Maximize the number of words placed in the puzzle

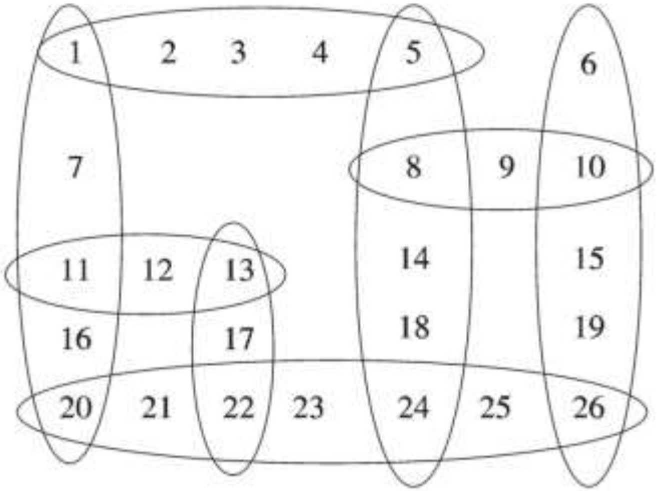


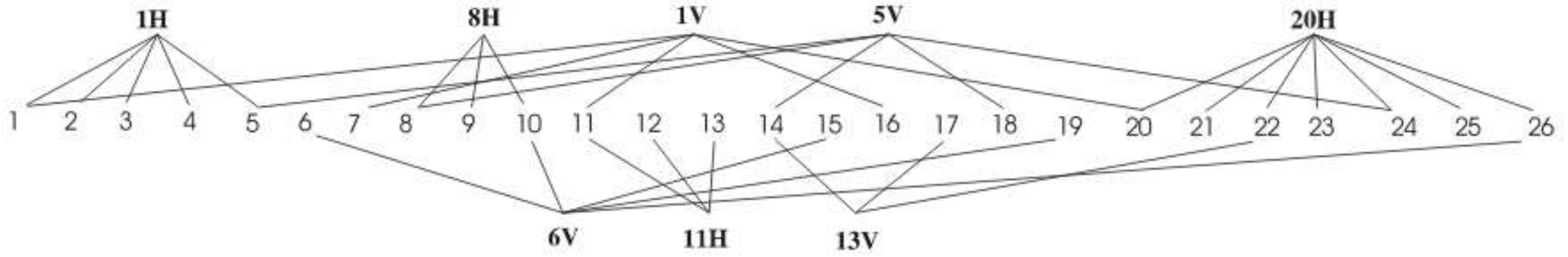
- Add a “big” constraint with no tuple

The puzzle is satisfiable ↔ exactly one constraint is violated in the **acyclic** MAX-CSP

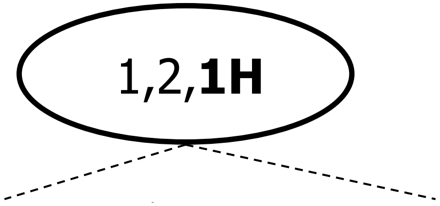
Tractability of MAX-CSP Instances

1. Consider the incidence graph
2. Compute a Tree Decomposition



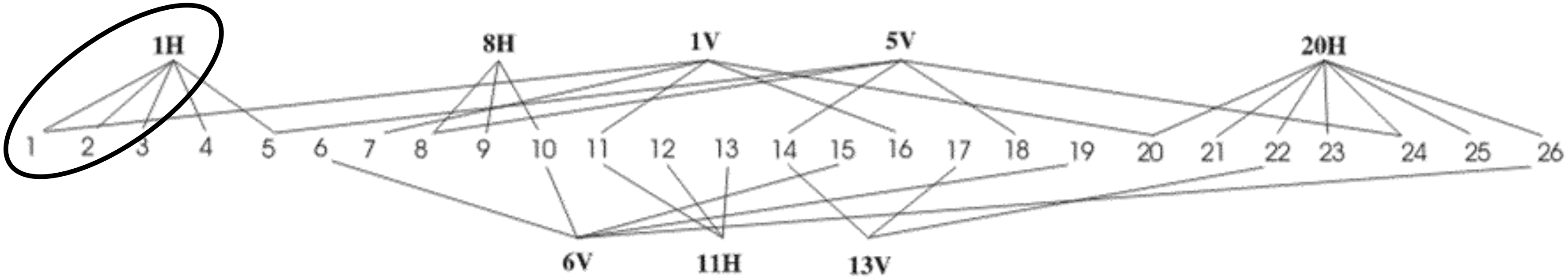


Tractability of MAX-CSP Instances



1,2,**1H**

MAX-CSP

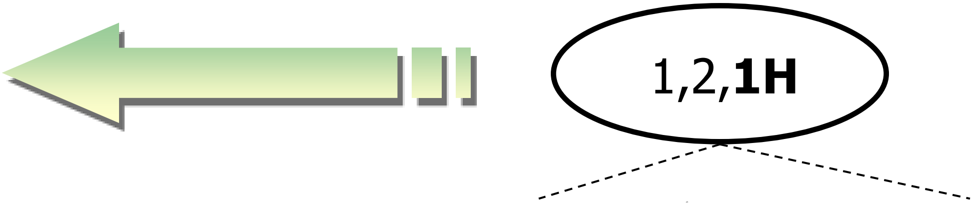


Tractability of MAX-CSP Instances

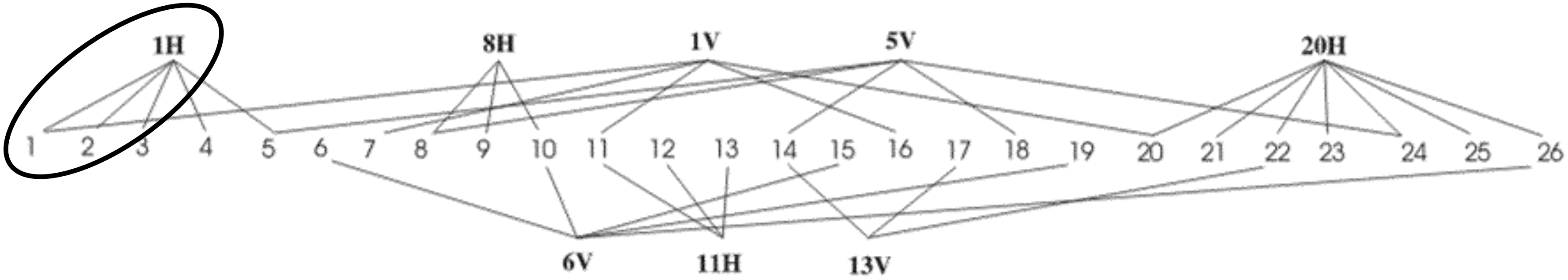
1 2	**1H**
P A	PARIS
P A	PANDA
L A	LAURA
A N	ANITA
A A	*unsat*
A B	*unsat*
…	*unsat*

Cost 1,
otherwise cost 0

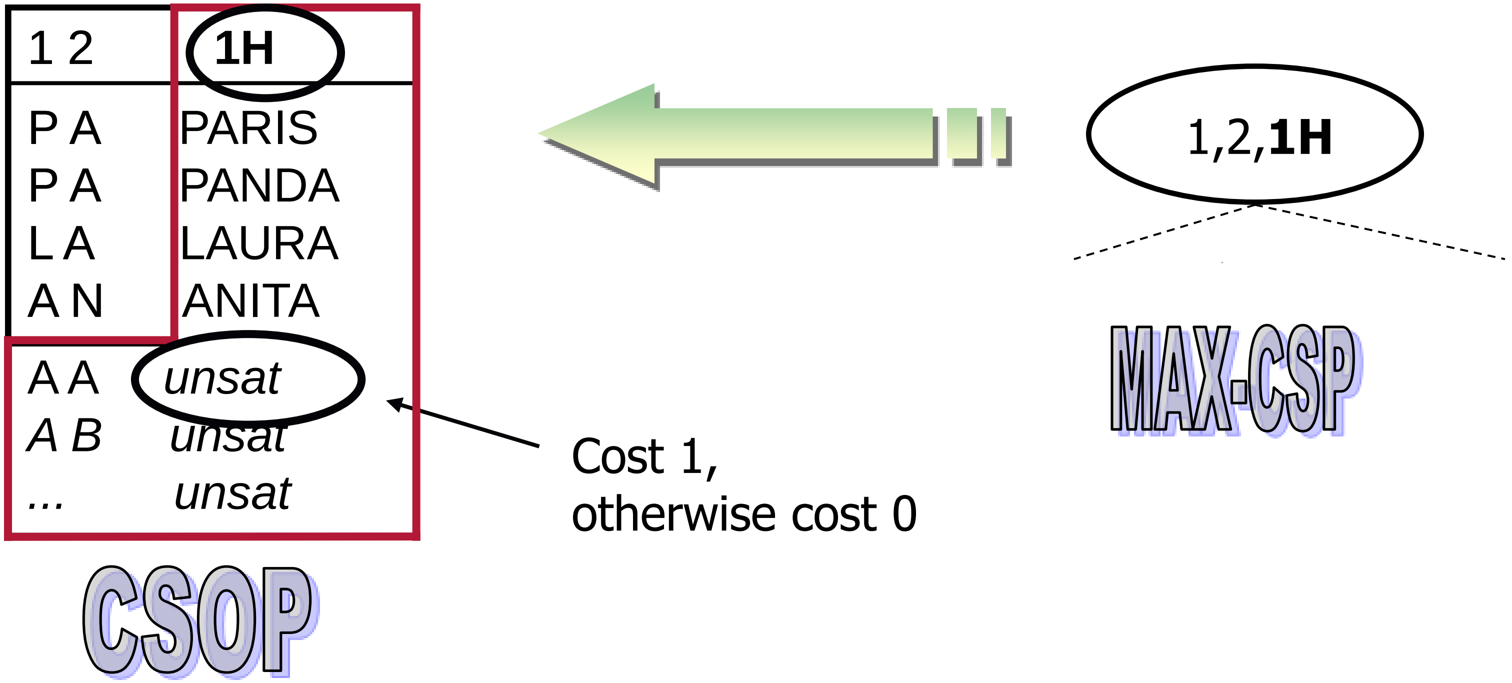
CSOP



MAX-CSP



In-Tractability of MAX-CSP Instances



1 2	**1H**
P A	PARIS
P A	PANDA
L A	LAURA
A N	ANITA
A A	*unsat*
A B	*unsat*
...	*unsat*

Cost 1,
otherwise cost 0

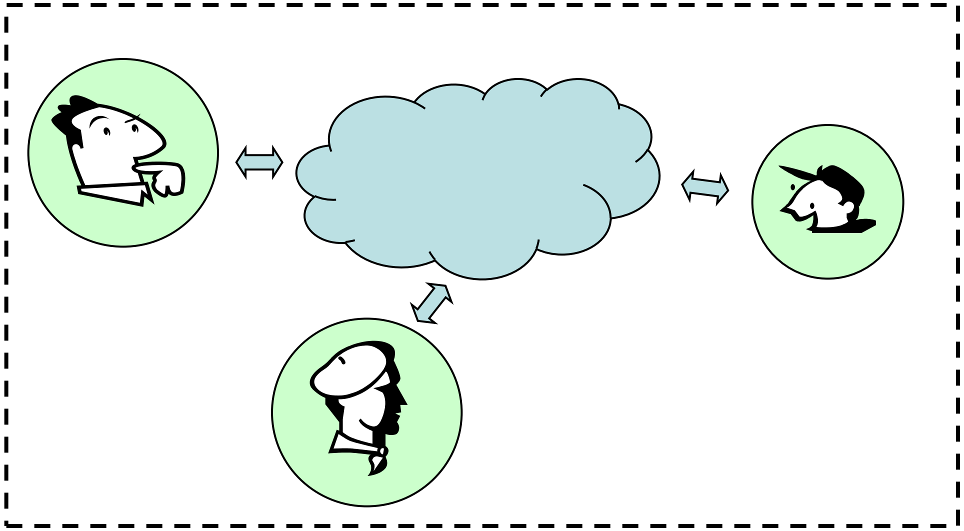
The mapping:

- Is feasible in time exponential in the width
- Preserves the solutions
- Leads to an Acyclic CSOP Instance

Outline of Part III

- Applications to Optimization Problems
- **Application: Nash Equilibria**
- Application: Coalitional Games
- Application: Combinatorial Auctions
- Appendix: Beyond Hypertree Width

Game Theory (in a Nutshell)

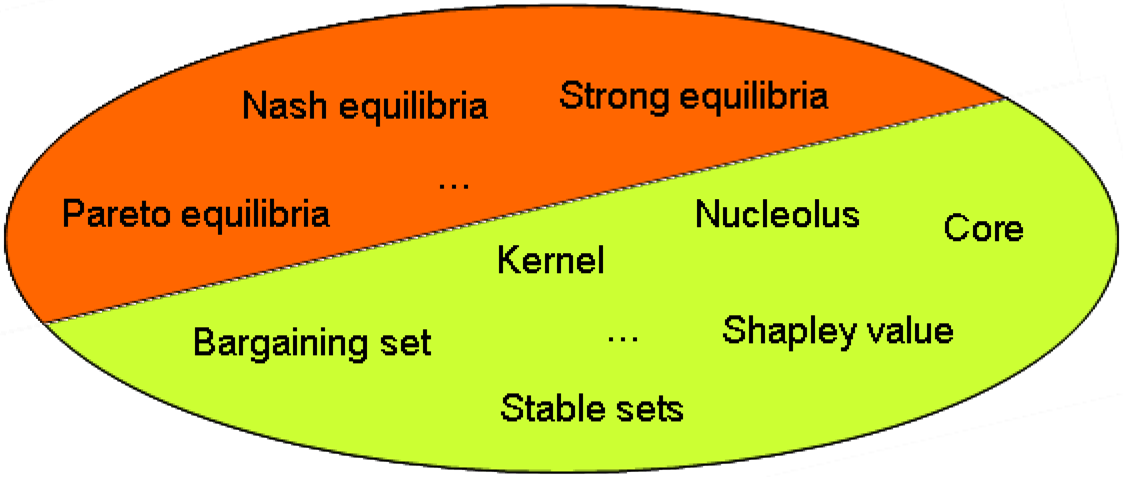


Each player:

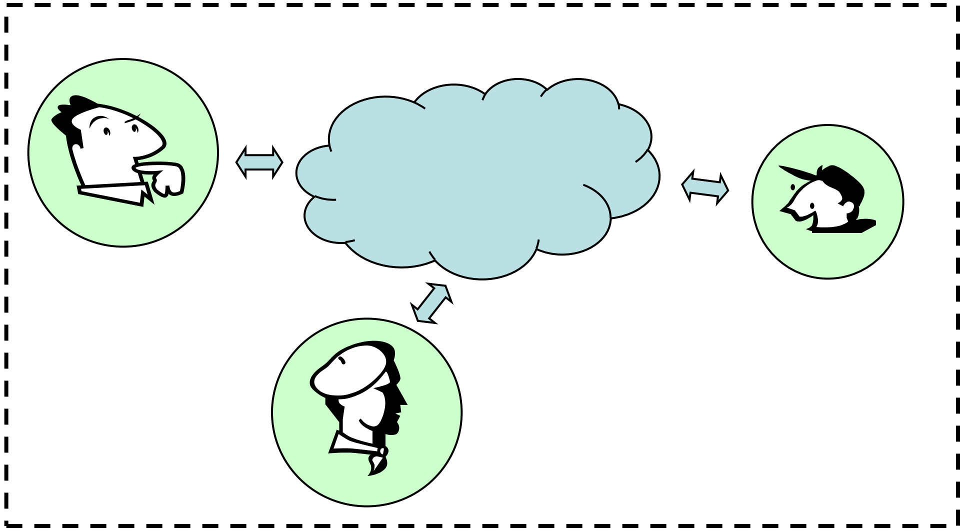
- Has a **goal** to be achieved
- Has a set of possible **actions**
- **Interacts** with other players
- Is **rational**

Which actions have to be performed?

Solution Concepts



Game Theory (in a Nutshell)

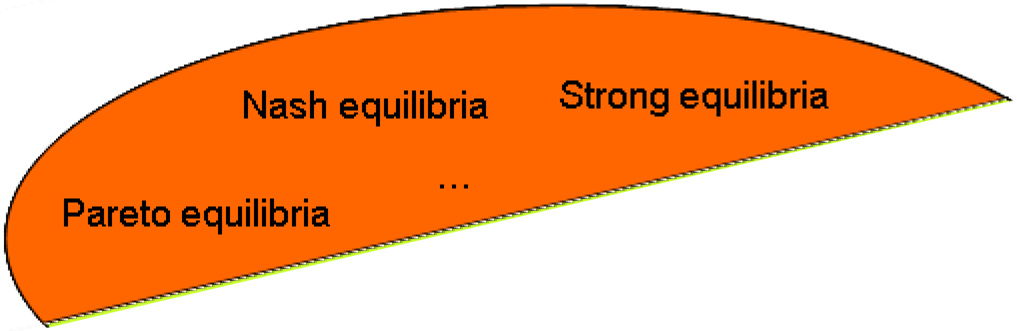


Each player:

- Has a **goal** to be achieved
- Has a set of possible **actions**
- **Interacts** with other players
- Is **rational**

Which actions have to be performed?

Solution Concepts



Non-Cooperative Games$_{(1/3)}$

Payoff maximization problem

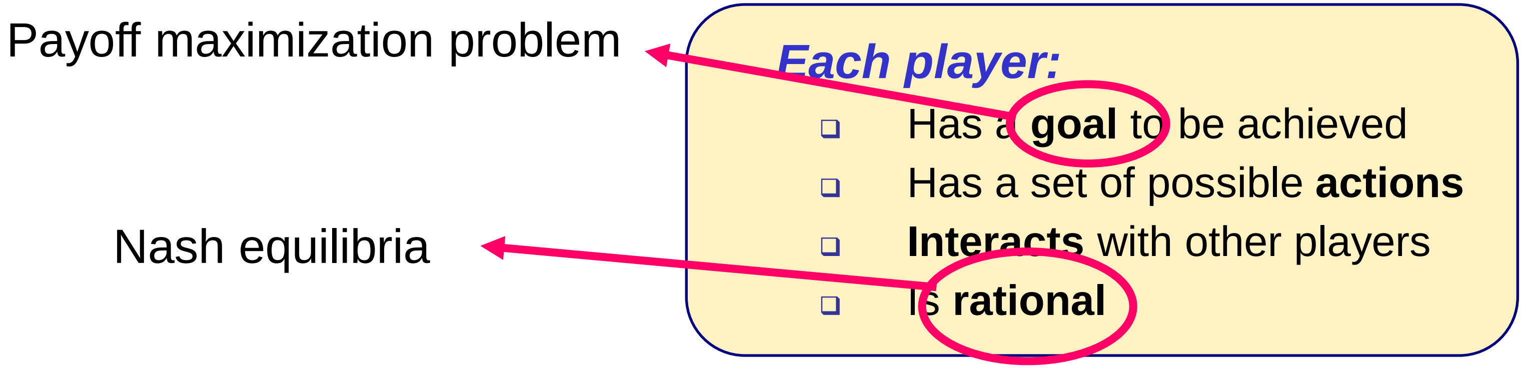
Each player:

- Has a **goal** to be achieved
- Has a set of possible **actions**
- **Interacts** with other players
- Is **rational**

Bob	John goes *out*	John stays at *home*
out	2	0
home	0	1

John	Bob goes *out*	Bob stays at *home*
out	1	1
home	0	0

Non-Cooperative Games$_{(2/3)}$



Payoff maximization problem

Nash equilibria

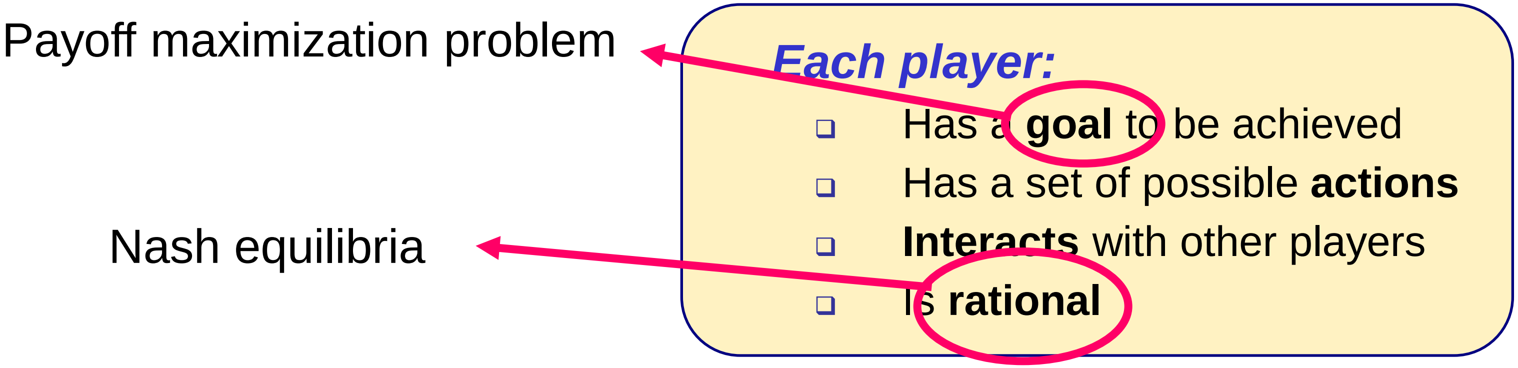
Each player:

- Has a **goal** to be achieved
- Has a set of possible **actions**
- **Interacts** with other players
- Is **rational**

Bob	John goes *out*	John stays at *home*
out	2	0
home	0	1

John	Bob goes *out*	Bob stays at *home*
out	1	1
home	0	0

Non-Cooperative Games$_{(2/3)}$



Payoff maximization problem

Nash equilibria

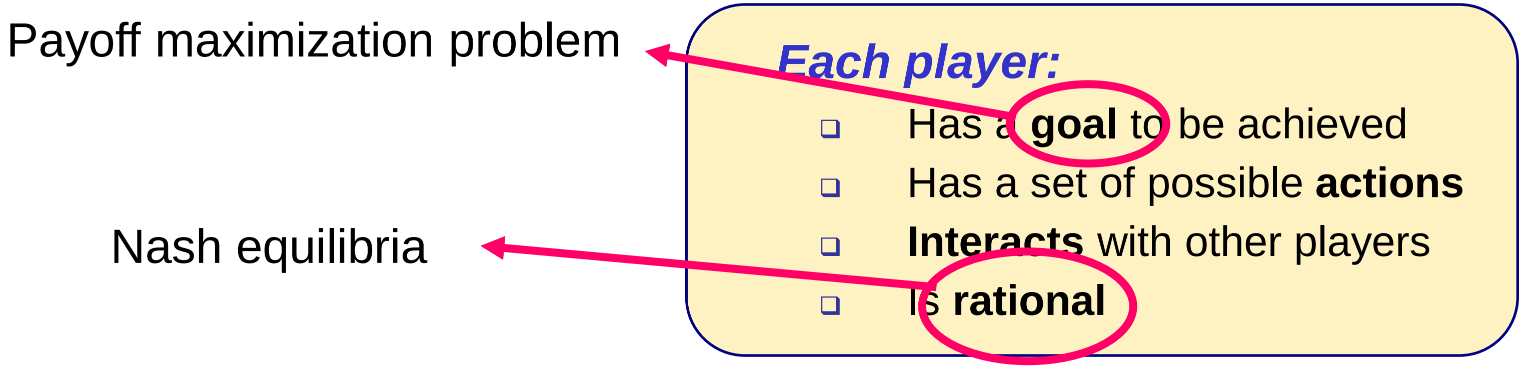
Each player:

- Has a **goal** to be achieved
- Has a set of possible **actions**
- **Interacts** with other players
- Is **rational**

Bob	John goes *out*	John stays at *home*
out	2	0
home	0	1

John	Bob goes *out*	Bob stays at *home*
out	1	1
home	0	0

Non-Cooperative Games$_{(2/3)}$



Payoff maximization problem

Nash equilibria

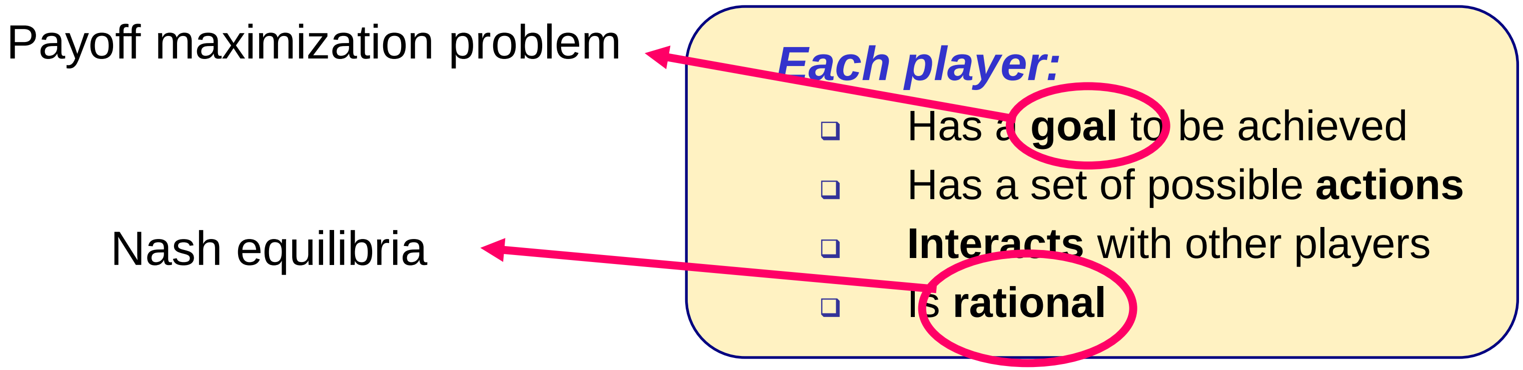
Each player:

- Has a **goal** to be achieved
- Has a set of possible **actions**
- **Interacts** with other players
- Is **rational**

Bob	John goes *out*	John stays at *home*
out	2	0
home	0	1

John	Bob goes *out*	Bob stays at *home*
out	1	1
home	0	0

Non-Cooperative Games$_{(2/3)}$



Payoff maximization problem

Nash equilibria

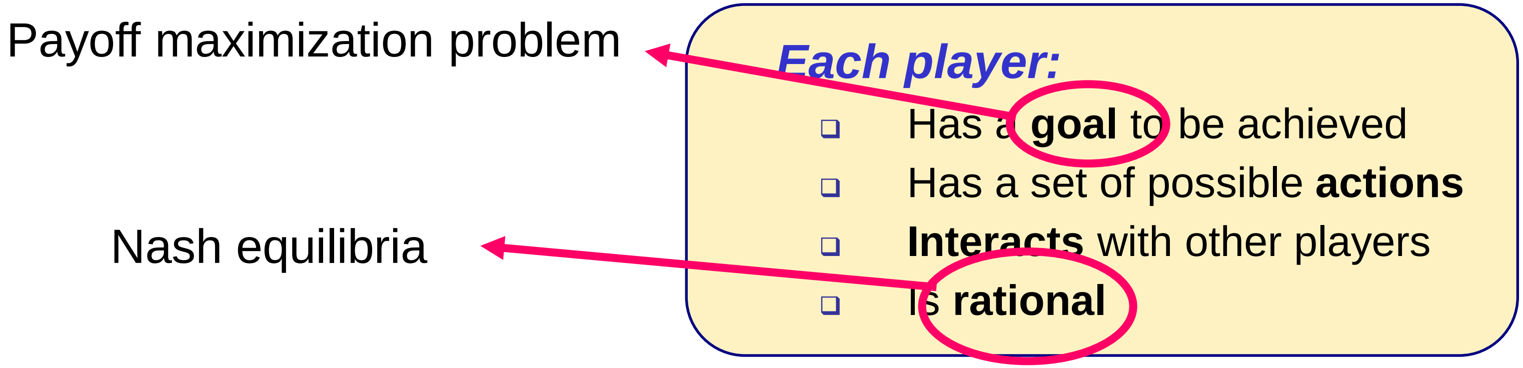
Each player:

- Has a **goal** to be achieved
- Has a set of possible **actions**
- **Interacts** with other players
- Is **rational**

Bob	John goes *out*	John stays at *home*
out	**2**	0
home	0	1

John	Bob goes *out*	Bob stays at *home*
out	**1**	1
home	0	0

Non-Cooperative Games$_{(3/3)}$



Payoff maximization problem

Nash equilibria

Each player:

- ❑ Has a **goal** to be achieved
- ❑ Has a set of possible **actions**
- ❑ **Interacts** with other players
- ❑ Is **rational**

***pure* Nash equilibria**

Every game admits a *mixed* Nash equilibrium,

- ❑ **where players chose their strategies according to probability distributions**

Succint Game Representations

- Players:
  - Maria, Francesco
- Choices:
  - movie, opera

If 2 players, then size = 2^2

Maria	Francesco, *movie*	Francesco, *opera*
movie	2	0
opera	0	1

Succint Game Representations

- Players:
  - Maria, Francesco, Paola
- Choices:
  - movie, opera

If 2 players, then size = 2^2

If 3 players, then size = 2^3

Maria	F_{movie} *and* P_{movie}	F_{movie} *and* P_{opera}	F_{opera} *and* P_{movie}	F_{opera} *and* P_{opera}
movie	2	0	2	1
opera	0	1	2	2

Succint Game Representations

- Players:
  - Maria, Francesco, Paola, Roberto, and Giorgio
- Choices:
  - movie, opera

If 2 players, then size = 2^2

If 3 players, then size = 2^3

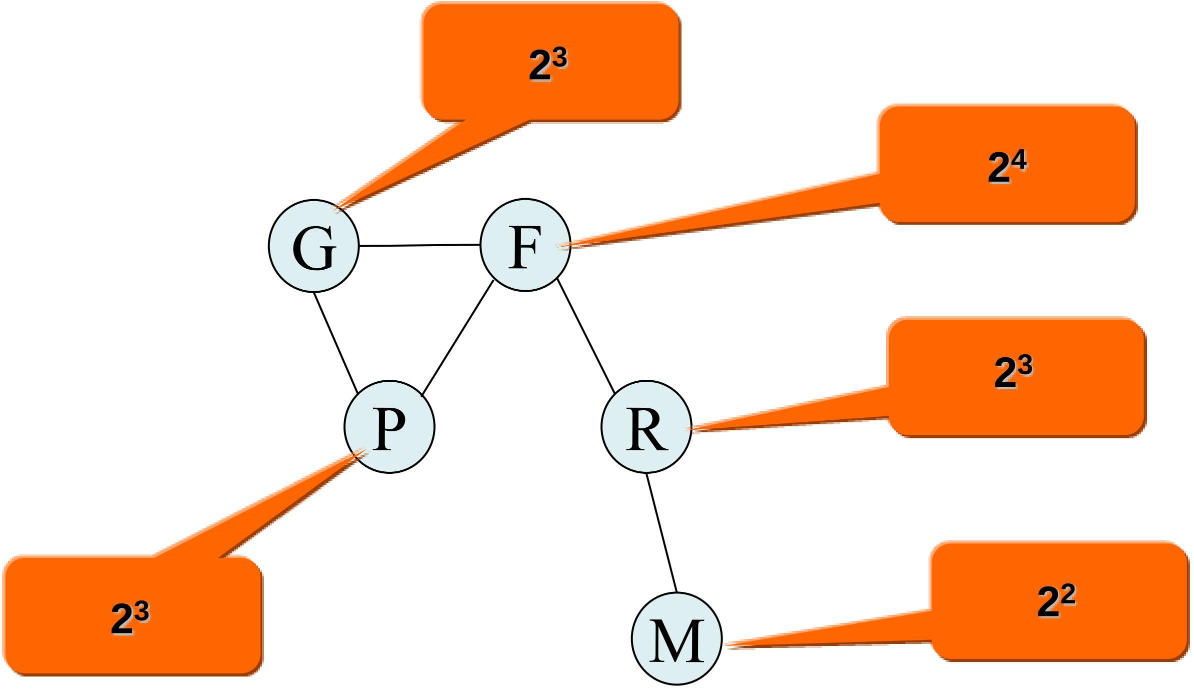
…

If N players, then size = 2^N

Maria	F_{movie} *and* P_{movie} *and* R_{movie} *and* G_{movie}			…………………………...
movie	2	……..	……..	……..
opera	0	……..	……..	……..

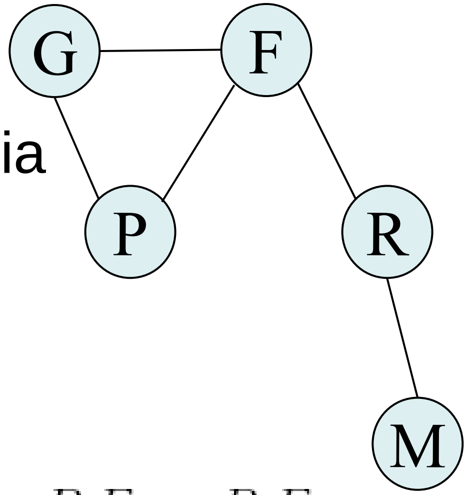
Succint Game Representations

- Players:
  - Francesco, Paola, Roberto, Giorgio, and Maria
- Choices:
  - movie, opera



Succinct Game Representations

- Players:
  - Francesco, Paola, Roberto, Giorgio, and Maria
- Choices:
  - movie, opera



F	P_mR_m	P_mR_o	P_oR_m	P_oR_o
m	2	2	1	0
o	0	2	1	2

G	P_mF_m	P_mF_o	P_oF_m	P_oF_o
m	2	0	0	1
o	2	0	0	1

R	F_m	F_o
m	0	1
o	2	0

P	F_m	F_o
m	2	0
o	0	1

M	R_m	R_o
m	1	0
o	0	2

Pure Equilibria

- Players:
  - Francesco, Paola, Roberto, Giorgio, and Maria
- Choices:
  - movie, opera

F	P_mR_m	P_mR_o	P_oR_m	P_oR_o
m	2	2	1	0
o	0	2	1	2

G	P_mF_m	P_mF_o	P_oF_m	P_oF_o
m	2	0	0	1
o	2	0	0	1

R	F_m	F_o
m	0	1
o	2	0

P	F_m	F_o
m	2	0
o	0	1

M	R_m	R_o
m	1	0
o	0	2

Pure Equilibria

- Players:
  - Francesco, Paola, Roberto, Giorgio, and Maria
- Choices:
  - movie, opera

F	P_mR_m	P_mR_o	P_oR_m	P_oR_o
m	2	[✓ 2]	1	0
o	0	2	1	2

G	P_mF_m	P_mF_o	P_oF_m	P_oF_o
m	[✓ 2]	0	0	1
o	2	0	0	1

R	F_m	F_o
m	0	1
o	[✓ 2]	0

P	F_m	F_o
m	[✓ 2]	0
o	0	1

M	R_m	R_o
m	1	0
o	0	[✓ 2]

Pure Equilibria

- Players:
  - Francesco, Paola, Roberto, Giorgio, and Maria
- Choices:
  - movie, opera

NP-hard !

F	P_mR_m	P_mR_o	P_oR_m	P_oR_o
m	2	2	1	0
o	0	2	1	2

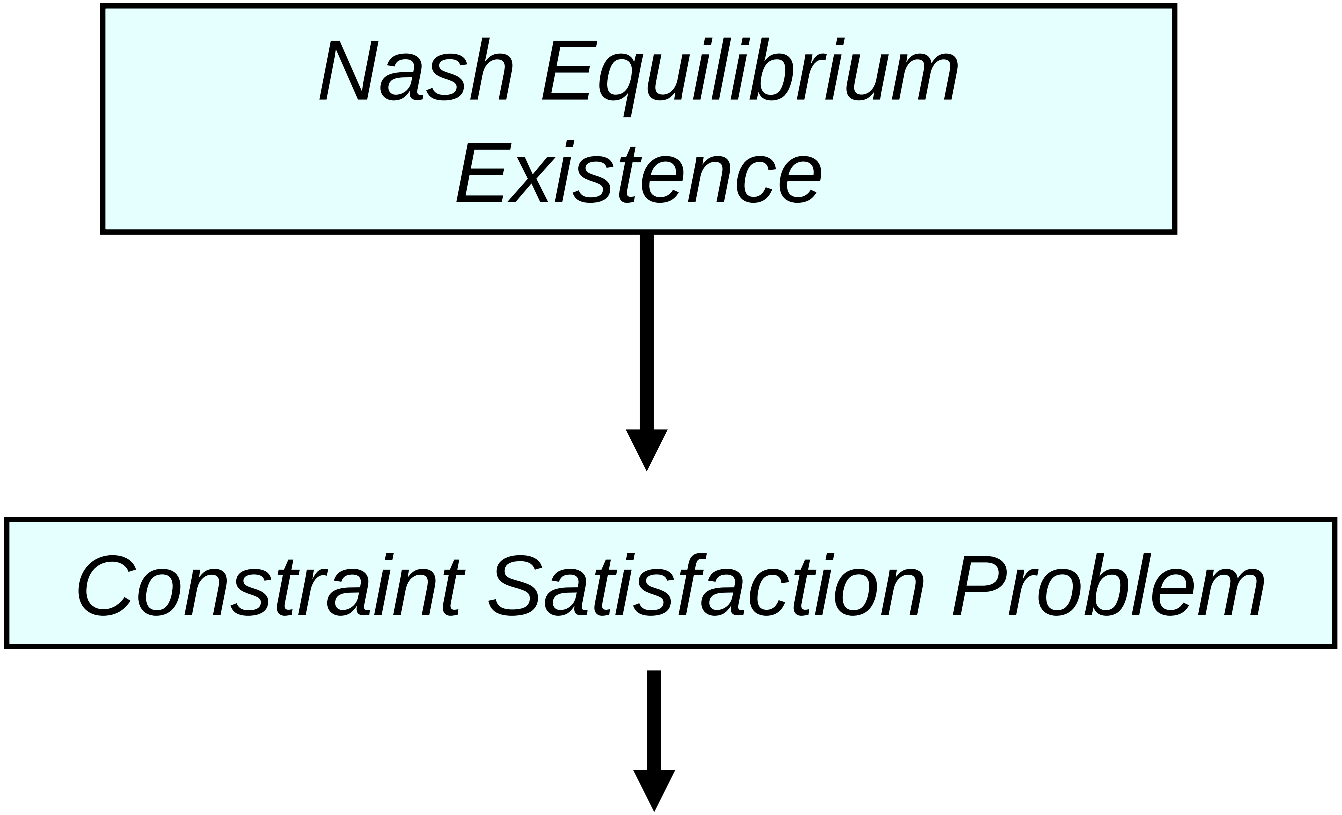
G	P_mF_m	P_mF_o	P_oF_m	P_oF_o
m	2	0	0	1
o	2	0	0	1

R	F_m	F_o
m	0	1
o	2	0

P	F_m	F_o
m	2	0
o	0	1

M	R_m	R_o
m	1	0
o	0	2

Pure Nash Equilibria and Easy Games



Nash Equilibrium Existence

Constraint Satisfaction Problem

Solve CSP in polynomial time using known methods

[Gottlob, Greco, and Scarcello, JAIR’05]

Encoding Games in CSPs

F	P_mR_m	P_mR_o	P_oR_m	P_oR_o
m	2	2	1	0
o	0	2	1	2

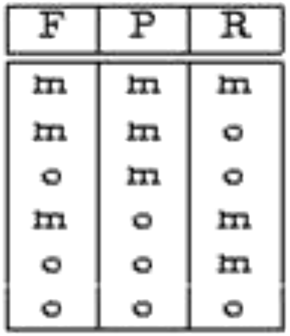
G	P_mF_m	P_mF_o	P_oF_m	P_oF_o
m	2	0	0	1
o	2	0	0	1

R	F_m	F_o
m	0	1
o	2	0

P	F_m	F_o
m	2	0
o	0	1

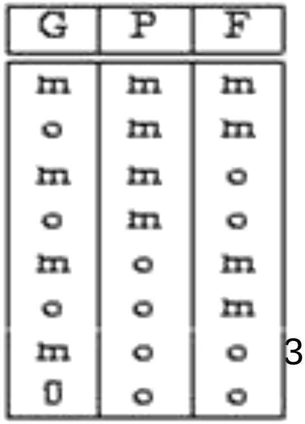
M	R_m	R_o
m	1	0
o	0	2

r_F :



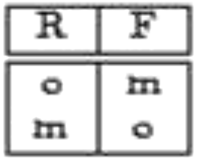
F	P	R
m	m	m
m	m	o
o	m	o
m	o	m
o	o	m
o	o	o

r_G :



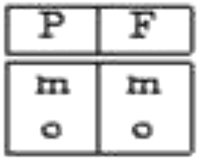
G	P	F
m	m	m
o	m	m
m	m	o
o	m	o
m	o	m
o	o	m
m	o	o
0	o	o

r_R :



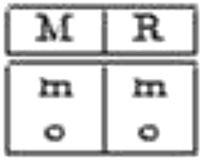
R	F
o	m
m	o

r_P :



P	F
m	m
o	o

r_M :



M	R
m	m
o	o

Encoding Games in CSPs

F	P_mR_m	P_mR_o	P_oR_m	P_oR_o
m	2	2	1	0
o	0	2	1	2

G	P_mF_m	P_mF_o	P_oF_m	P_oF_o
m	2	0	0	1
o	2	0	0	1

R	F_m	F_o
m	0	1
o	2	0

P	F_m	F_o
m	2	0
o	0	1

M	R_m	R_o
m	1	0
o	0	2

r_F :

F	P	R
m	m	m
m	m	o
o	m	o
m	o	m
o	o	m
o	o	o

r_G :

G	P	F
m	m	m
o	m	m
m	m	o
o	m	o
m	o	m
o	o	m
m	o	o
0	o	o

r_R :

R	F
o	m
m	o

r_P :

P	F
m	m
o	o

r_M :

M	R
m	m
o	o

Encoding Games in CSPs

F	P_mR_m	P_mR_o	P_oR_m	P_oR_o
m	2	2	1	0
o	0	2	1	2

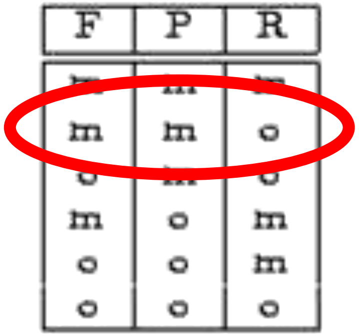
G	P_mF_m	P_mF_o	P_oF_m	P_oF_o
m	2	0	0	1
o	2	0	0	1

R	F_m	F_o
m	0	1
o	2	0

P	F_m	F_o
m	2	0
o	0	1

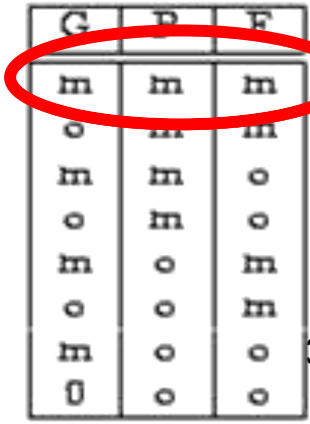
M	R_m	R_o
m	1	0
o	0	2

r_F :



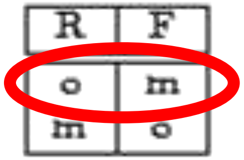
F	P	R
m	m	m
m	m	o
o	m	o
m	o	m
o	o	m
o	o	o

r_G :



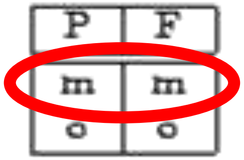
G	P	F
m	m	m
o	m	m
m	m	o
o	m	o
m	o	m
o	o	m
m	o	o
0	o	o

r_R :



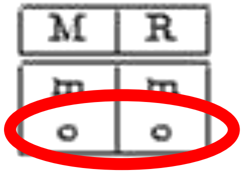
R	F
o	m
m	o

r_P :



P	F
m	m
o	o

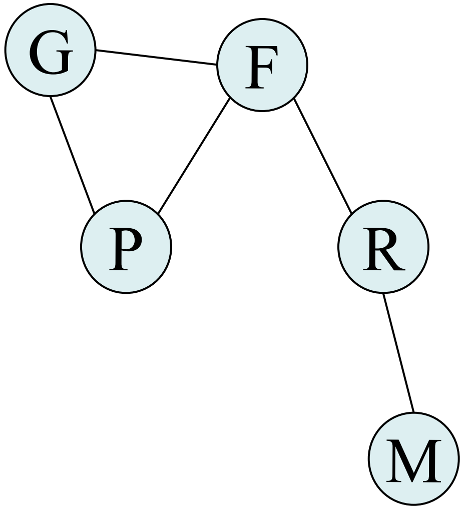
r_M :



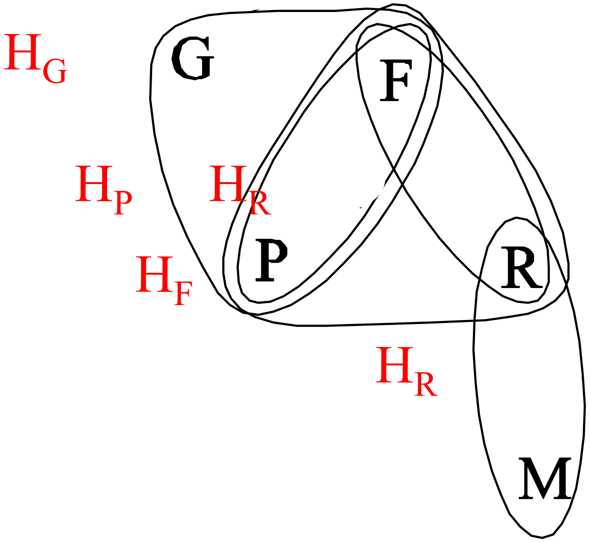
M	R
m	m
o	o

Interaction Among Players: Friends

- The interaction structure of a game G can be represented by:
  - the dependency graph G(G) according to Neigh(G)
  - a hypergraph H(G) with edges: H(p)=Neigh(p) ∪ {p}



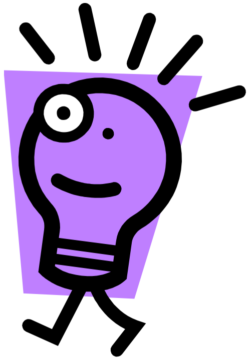
G(*FRIENDS*)

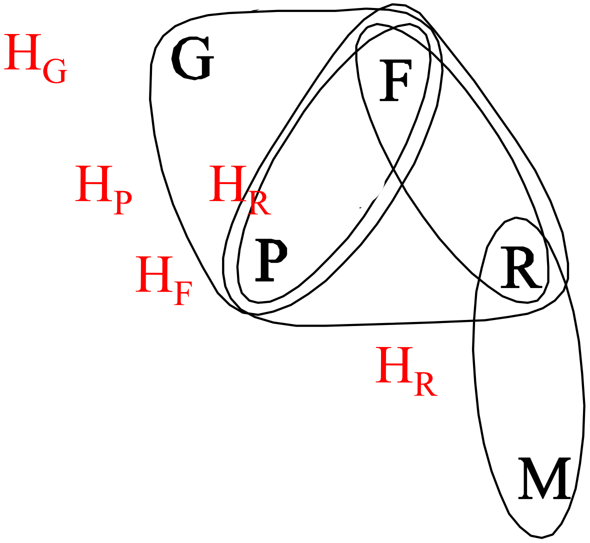


H(*FRIENDS*)

Interaction Among Players: Friends

This is the same structure as the one of the associated CSP



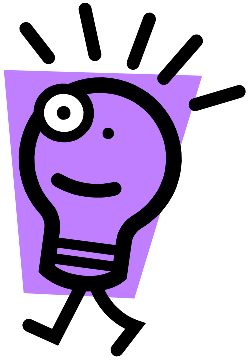


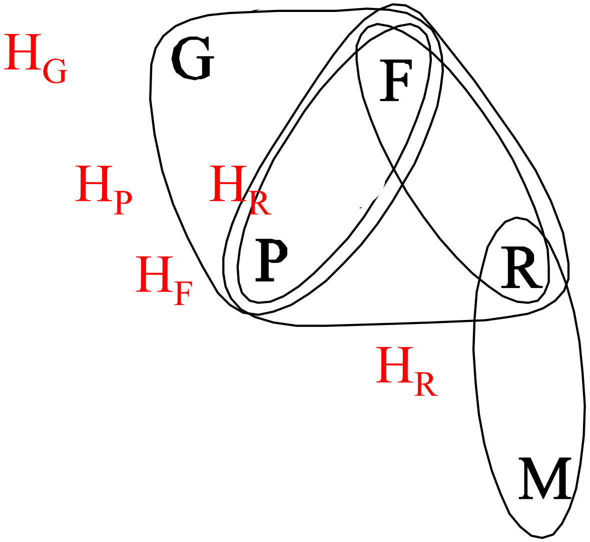
H(*FRIENDS*)

Interaction Among Players: Friends

This is the same structure as the one of the associated CSP

On (nearly)-Acyclic Instances, Nash equilibria are easy



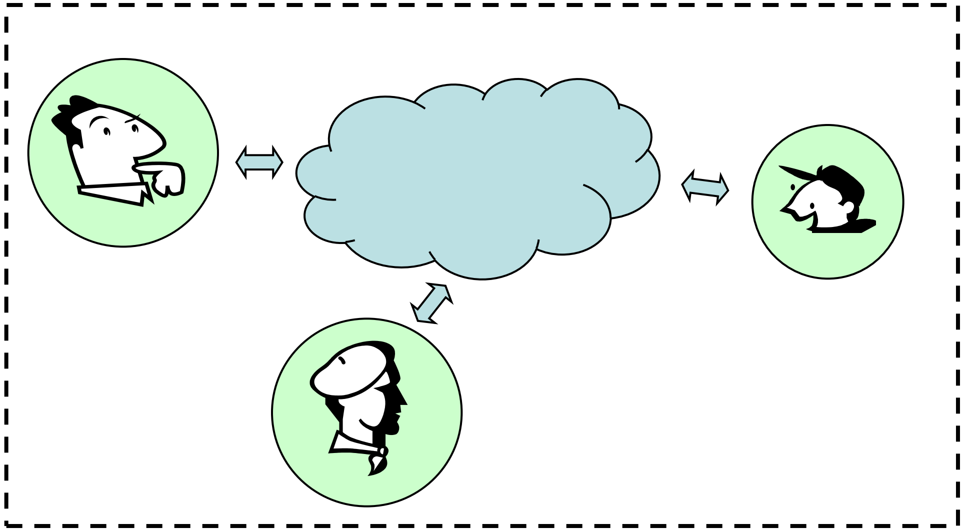


H(*FRIENDS*)

Outline of Part III

- Applications to Optimization Problems
- Application: Nash Equilibria
- **Application: Coalitional Games**
- Application: Combinatorial Auctions
- Appendix: Beyond Hypertree Width

Game Theory (in a Nutshell)

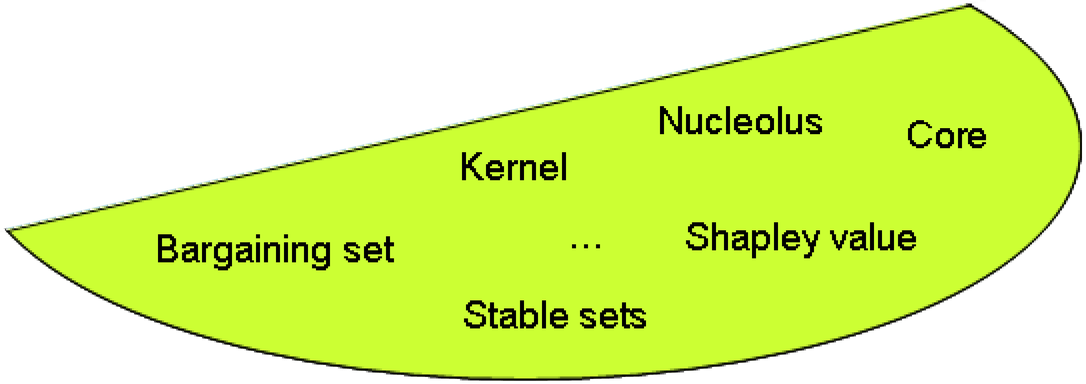


Each player:

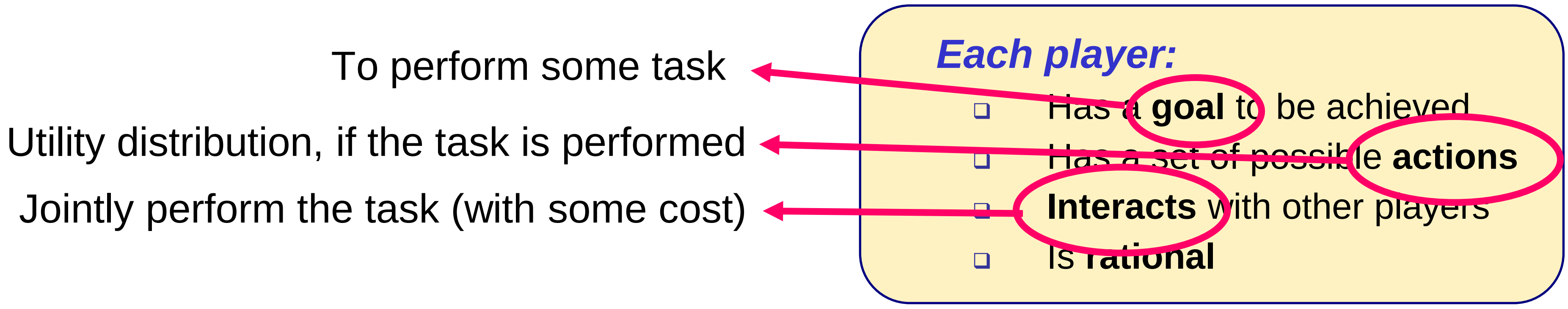
- □ Has a **goal** to be achieved
- □ Has a set of possible **actions**
- □ **Interacts** with other players
- □ Is **rational**

Which actions have to be performed?

Solution Concepts



Cooperative Game Theory$_{(1/2)}$



To perform some task

Utility distribution, if the task is performed

Jointly perform the task (with some cost)

Each player:

- □ Has a **goal** to be achieved
- □ Has a set of possible **actions**
- □ **Interacts** with other players
- □ Is **rational**

Cooperative Game Theory$_{(1/2)}$

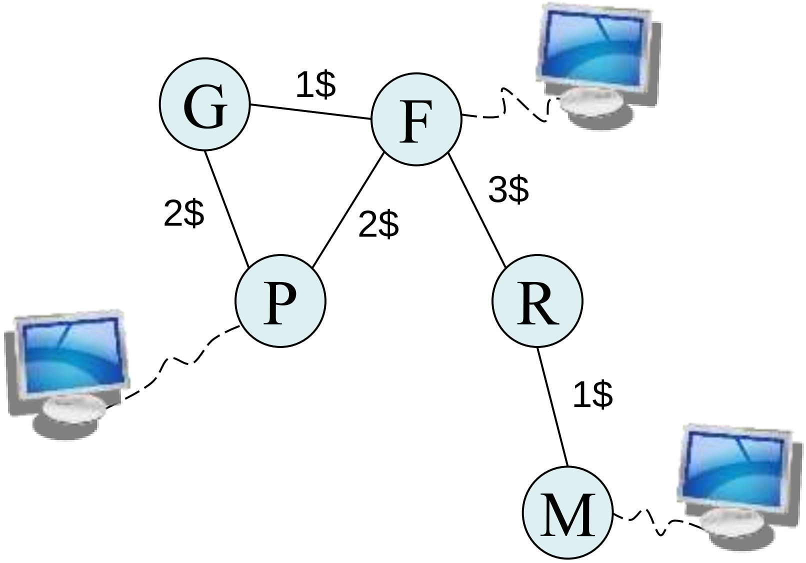
To perform some task

Utility distribution, if the task is performed

Jointly perform the task (with some cost)

Each player:

- □ Has a **goal** to be achieved
- □ Has a set of possible **actions**
- □ **Interacts** with other players
- □ Is **rational**



- ➢ Players get **9$**, if they enforce connectivity
- ➢ Enforcing connectivity over an edge as a cost

Cooperative Game Theory$_{(1/2)}$

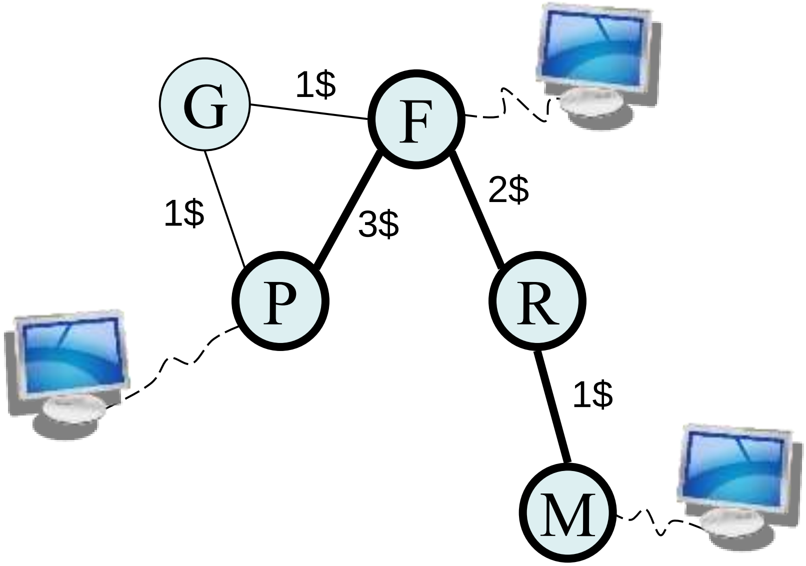
To perform some task

Utility distribution, if the task is performed

Jointly perform the task (with some cost)

Each player:

- ❑ Has a **goal** to be achieved
- ❑ Has a set of possible **actions**
- ❑ **Interacts** with other players
- ❑ Is **rational**



- ➢ Players get **9$**, if they enforce connectivity
- ➢ Enforcing connectivity over an edge as a cost

Coalition {F,P,R,M} gets **9$**, and pays **6$**

worth v({F,P,R,M}) = 9$ - 6$

Cooperative Game Theory$_{(1/2)}$

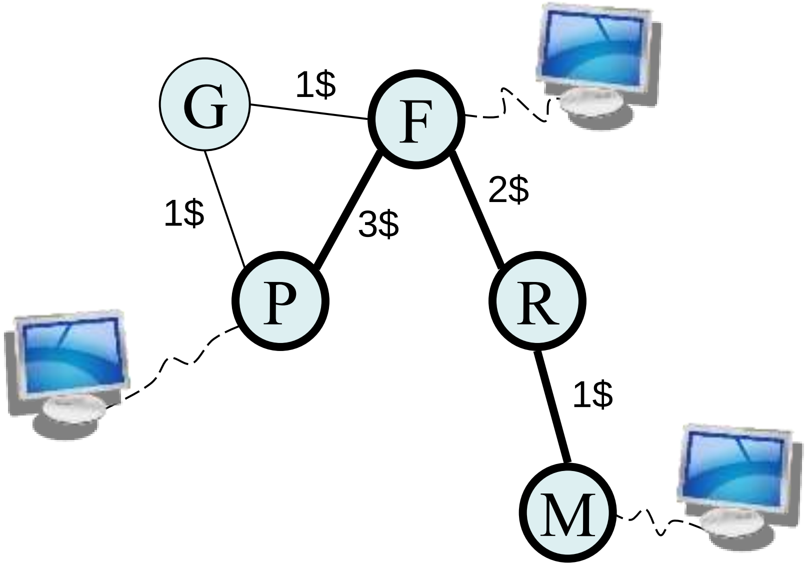
To perform some task

Utility distribution, if the task is performed

Jointly perform the task (with some cost)

Each player:

- □ Has a **goal** to be achieved
- □ Has a set of possible **actions**
- □ **Interacts** with other players
- □ Is **rational**



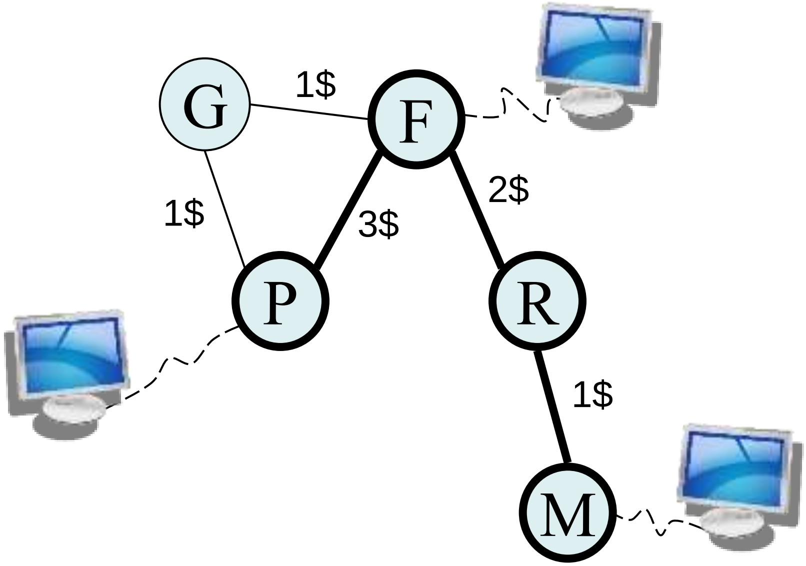
coalition	worth
{F}	0
…	0
{G,P,R,M}	0
{F,P,R,M}	**3**
{G,F,P,R,M}	4

How to distribute 4$, based on such worths?

Cooperative Game Theory$_{(2/2)}$

Each player:

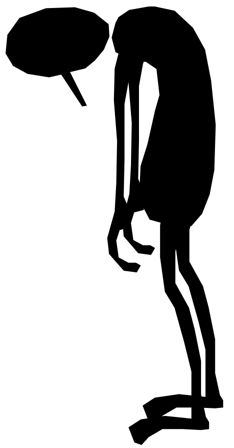
- Has a **goal** to be achieved
- Has a set of possible **actions**
- **Interacts** with other players
- Is **rational**



coalition	worth
{F}	0
…	0
{G,P,R,M}	0
{F,P,R,M}	**3**
{G,F,P,R,M}	4

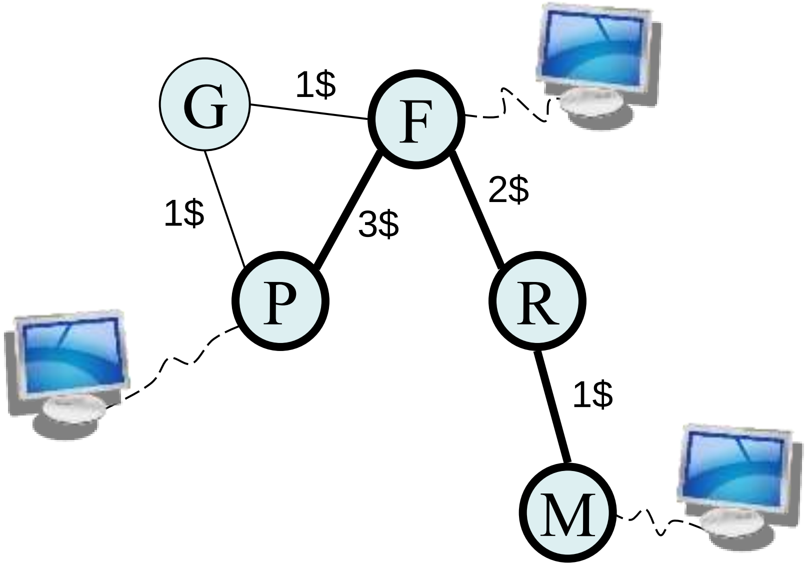
How to distribute 4$, based on such worths?

Cooperative Game Theory(2/2)



Each player:

- Has a **goal** to be achieved
- Has a set of possible **actions**
- **Interacts** with other players
- Is **rational**



G ←4$
P,F,R,M ←0$

coalition	worth	value	excess
{F}	0	0	0
…	0	…	…
{G,P,R,M}	0	4	-4
{F,P,R,M}	**3**	**0**	**3**
{G,F,P,R,M}	4	4	0

How to distribute 4$, based on such worths?

Cooperative Game Theory (2/2)

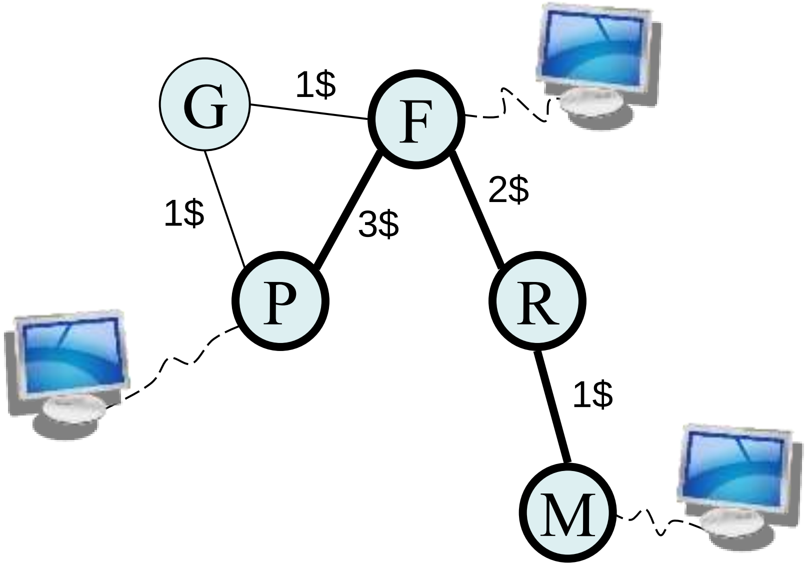
Find the distribution(s) that:

- Each coalition has a non-positive excess
- Lexicographically minimize the excess vector
- Is immune against deviating coalitions
- …

core

nucleolus

bargaing



coalition	worth
{F}	0
…	0
{G,P,R,M}	0
{F,P,R,M}	**3**
{G,F,P,R,M}	4

G ←4$
P,F,R,M ←0$

value	excess
0	0
…	…
4	-4
0	**3**
4	0

How to distribute 4$, based on such worths?

The Model

- Players form *coalitions*
- Each coalition is associated with a *worth*
- A *total worth* has to be distributed

$$\boxed{\mathcal{G} = \langle N, v \rangle, v : 2^N \mapsto \mathbb{R}}$$

- Outcomes belong to the imputation set $X(\mathcal{G})$

$x \in X(\mathcal{G})$

- *Efficiency*

$$x(N) = v(N)$$

- *Individual Rationality*

$$x_i \geq v(\{i\}), \quad \forall i \in N$$

The Model

- Players form *coalitions*
- Each coalition is associated with a *worth*
- A *total worth* has to be distributed

$$\mathcal{G} = \langle N, v \rangle,\ v : 2^N \mapsto \mathbb{R}$$

- **Solution Concepts** characterize outcomes in terms of
  - Fairness
  - Stability

The Model

- Players form *coalitions*
- Each coalition is associated with a *worth*
- A *total worth* has to be distributed

$$\mathcal{G} = \langle N, v \rangle,\ v : 2^N \mapsto \mathbb{R}$$

- **Solution Concepts** characterize outcomes in terms of
  - Fairness
  - Stability

$$0 \geq e(S, x) = v(S) - \sum_{i \in S} x_i$$

$$\uparrow$$

The Core: $\forall S \subseteq N, x(S) \geq v(S);$

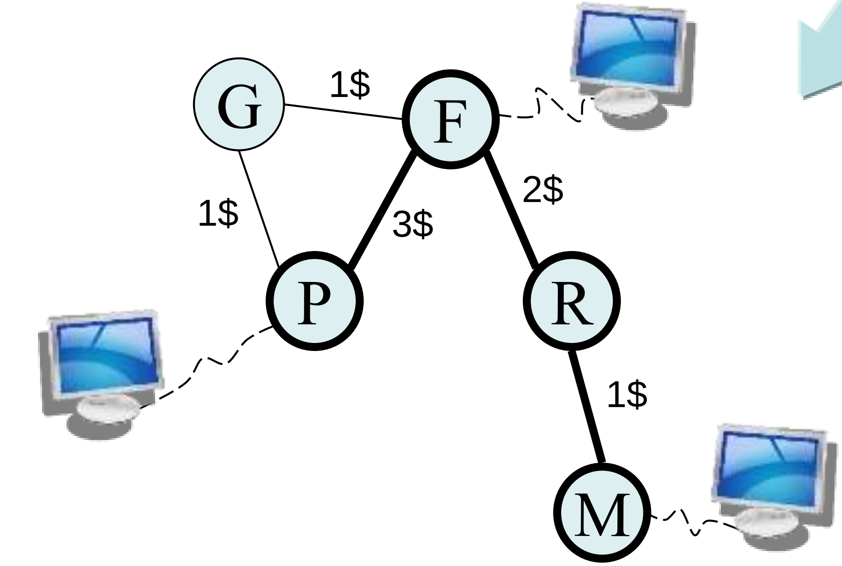
$$x(N) = v(N)$$

Complexity of Solution Concepts

- Nucleolus
- Kernel
- Bargaining Set
- Stable Sets

Graph games:

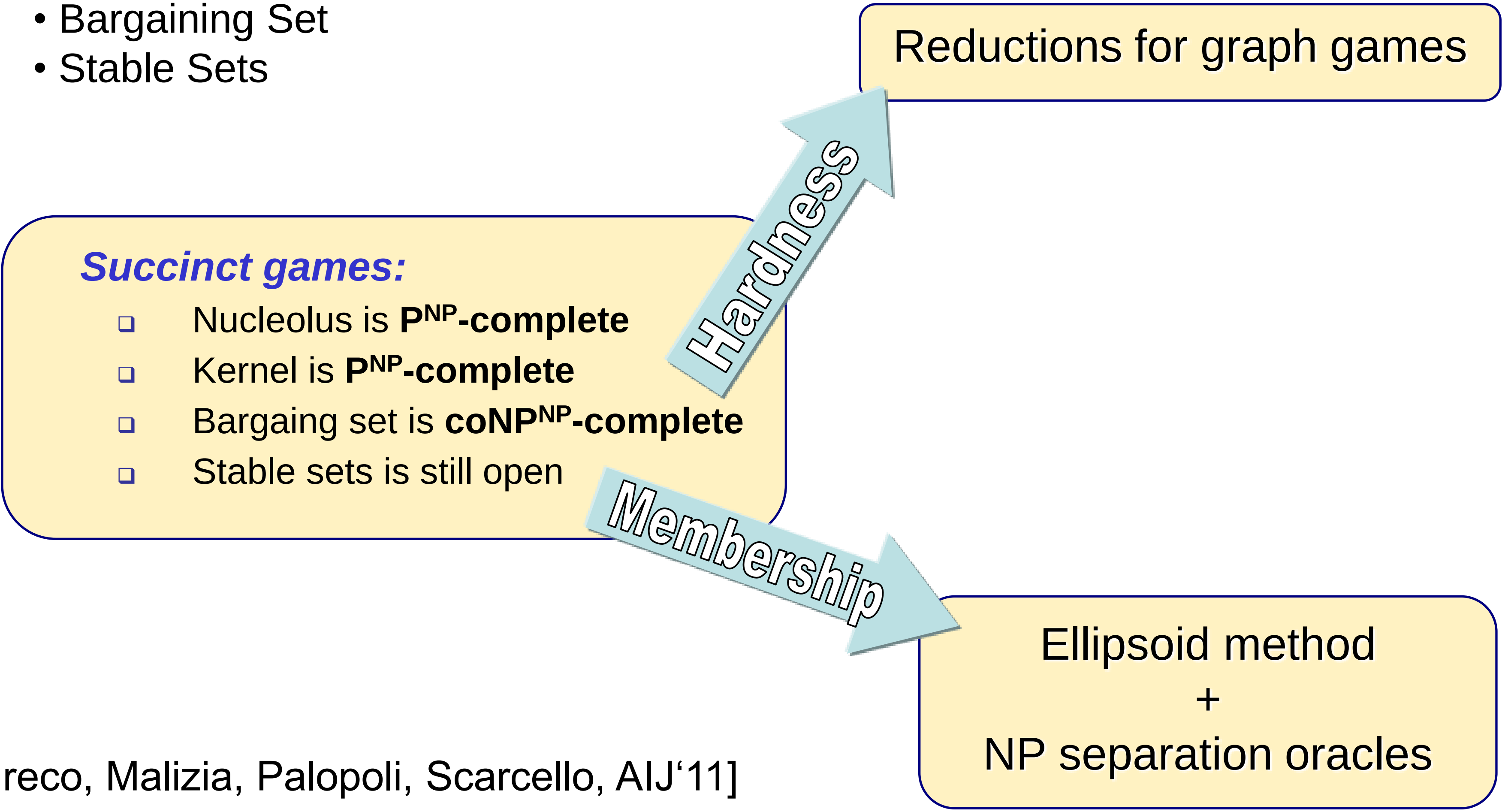
- ❑ Succinct specification
- ❑ Core existence is coNP-complete



coalition	worth
{F}	0
…	0
{G,P,R,M}	0
{F,P,R,M}	**3**
{G,F,P,R,M}	4

Complexity of Solution Concepts

- Nucleolus
- Kernel
- Bargaining Set
- Stable Sets



Succinct games:

- ❑ Nucleolus is **P^{NP}-complete**
- ❑ Kernel is **P^{NP}-complete**
- ❑ Bargaing set is **coNP^{NP}-complete**
- ❑ Stable sets is still open

Reductions for graph games

Ellipsoid method
+
NP separation oracles

[Greco, Malizia, Palopoli, Scarcello, AIJ‘11]

Membership in the Core on Graph Games

The Core: $\forall S \subseteq N, x(S) \geq v(S);$

$$x(N) = v(N)$$

Consider the sentence,

over the graph where *N is the set of nodes and E the set of edges* :

$$proj(X,Y) \equiv X \subseteq N \wedge$$
$$\forall c, c' \big(Y(c,c') \rightarrow X(c) \wedge X(c') \big) \wedge$$
$$\forall c, c' \big(X(c) \wedge X(c') \wedge E(c,c') \rightarrow Y(c,c') \big)$$

Membership in the Core on Graph Games

The Core: $\forall S \subseteq N, x(S) \geq v(S);$

$$x(N) = v(N)$$

Consider the sentence,

over the graph where *N is the set of nodes and E the set of edges* :

$$
\begin{aligned}
proj(X,Y) \equiv X \subseteq N \wedge \\
\forall c,c'\big(\,Y(c,c') \rightarrow X(c) \wedge X(c')\big) \wedge \\
\forall c,c'\big(\,X(c) \wedge X(c') \wedge E(c,c') \rightarrow Y(c,c')\big)
\end{aligned}
$$

…it tells that Y is the set of edges covered by the nodes in X

Membership in the Core on Graph Games

The Core: $\forall S \subseteq N, x(S) \geq v(S);$

$$x(N) = v(N)$$

Let $proj(X, Y)$ be the formula stating that Y is the set of edges covered by the nodes in X

Define the following weights: $w_E(c, c') = -w(c, c');$ $w_N(c) = x_c$

Value of the edge (negated)

Value at the imputation

Membership in the Core on Graph Games

The Core: $\forall S \subseteq N, x(S) \geq v(S);$

$$x(N) = v(N)$$

Let $proj(X, Y)$ be the formula stating that Y is the set of edges covered by the nodes in X

Define the following weights: $w_E(c, c') = -w(c, c');$ $\quad w_N(c) = x_c$

$\uparrow$ Value of the edge (negated)

$\uparrow$ Value at the imputation

Find the "minimum-weight" X and Y such that $proj(X, Y)$ **holds**

Membership in the Core on Graph Games

$0 \geq e(S,x) = v(S) - \sum_{i \in S} x_i$

The Core: $\forall S \subseteq N, x(S) \geq v(S);$
$x(N) = v(N)$

Let $proj(X,Y)$ be the formula stating that Y is the set of edges covered by the nodes in X

Define the following weights: $w_E(c,c') = -w(c,c');$ $w_N(c) = x_c$

Value of the edge (negated) Value at the imputation

Find the "minimum-weight" X and Y such that $proj(X,Y)$ holds

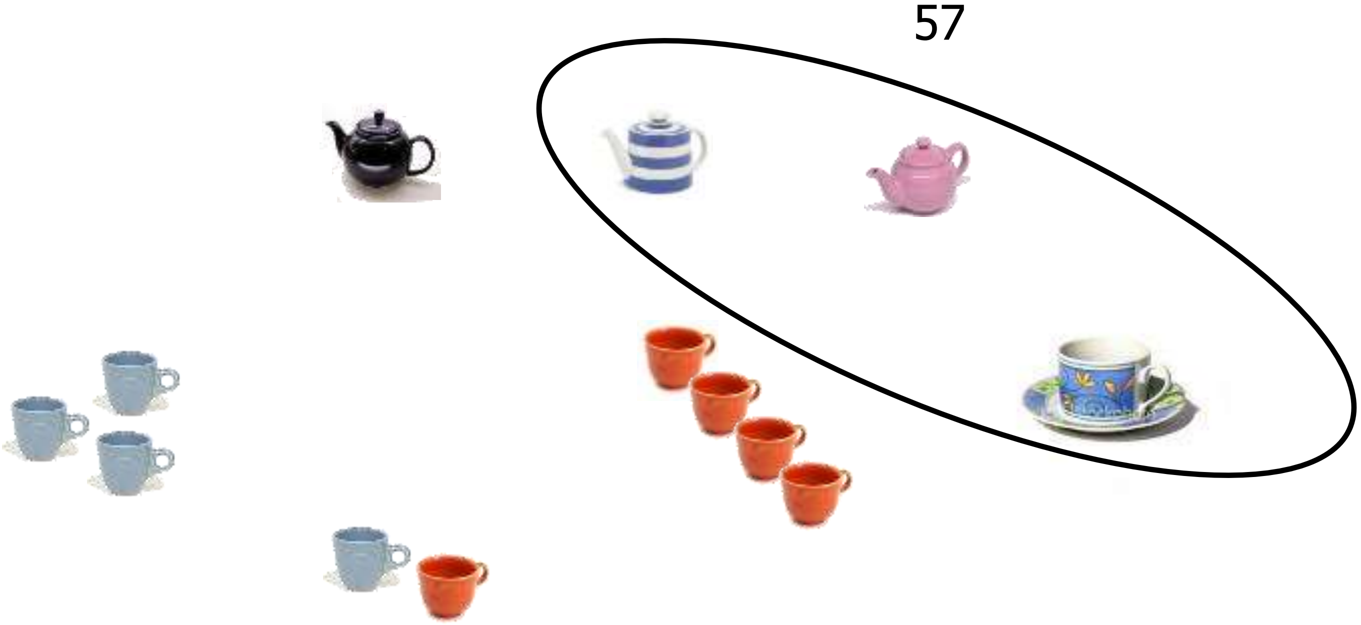
Max (value of edges – value of the imputation), i.e., $max_{S \subseteq N} e(S,x)$

Outline of Part III

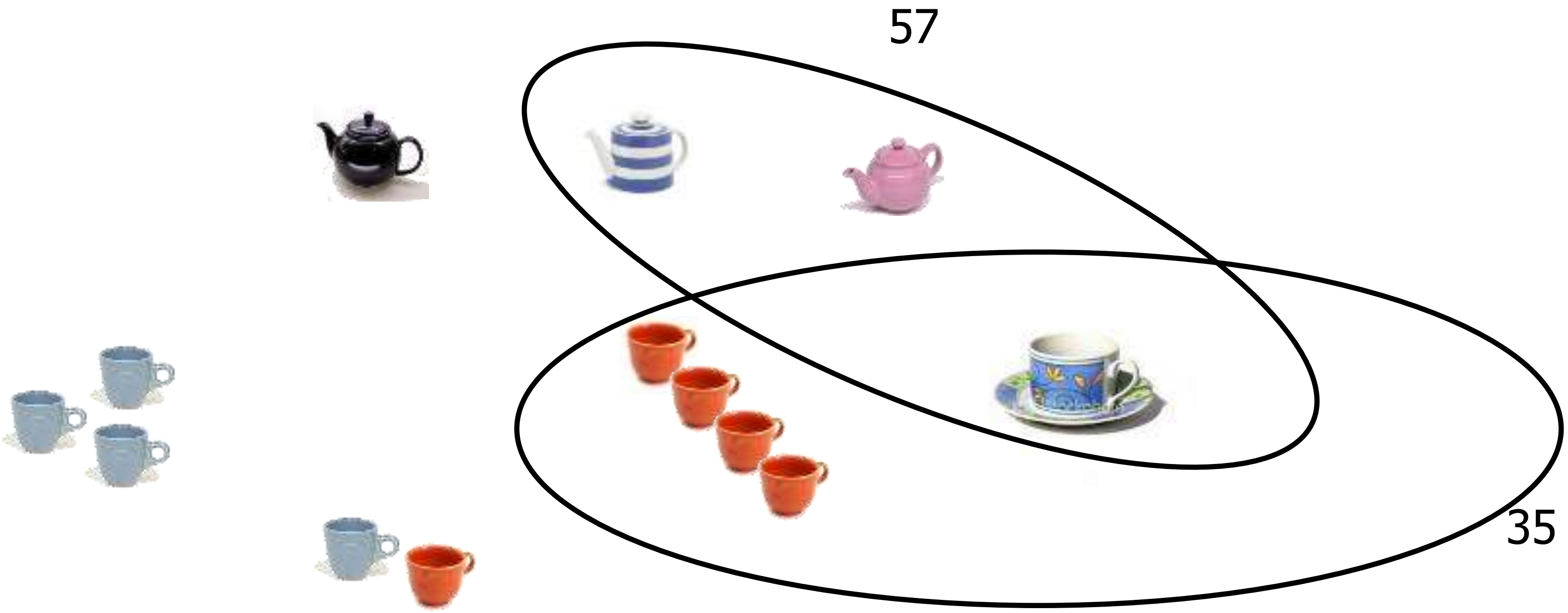
- Applications to Optimization Problems
- Application: Nash Equilibria
- Application: Coalitional Games
- **Application: Combinatorial Auctions**
- Appendix: Beyond Hypertree Width

Example: Combinatorial Auctions

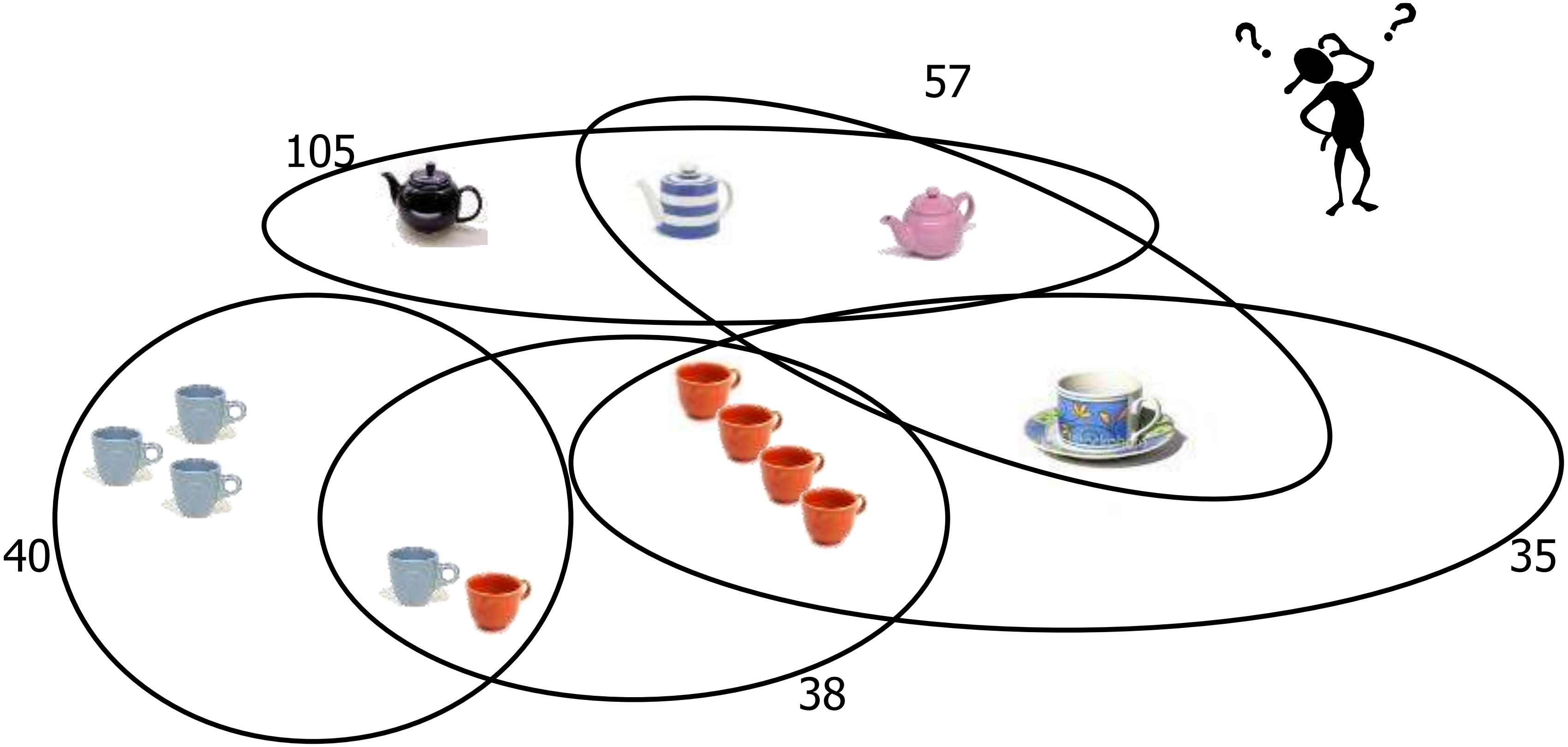
Example: Combinatorial Auctions



Example: Combinatorial Auctions



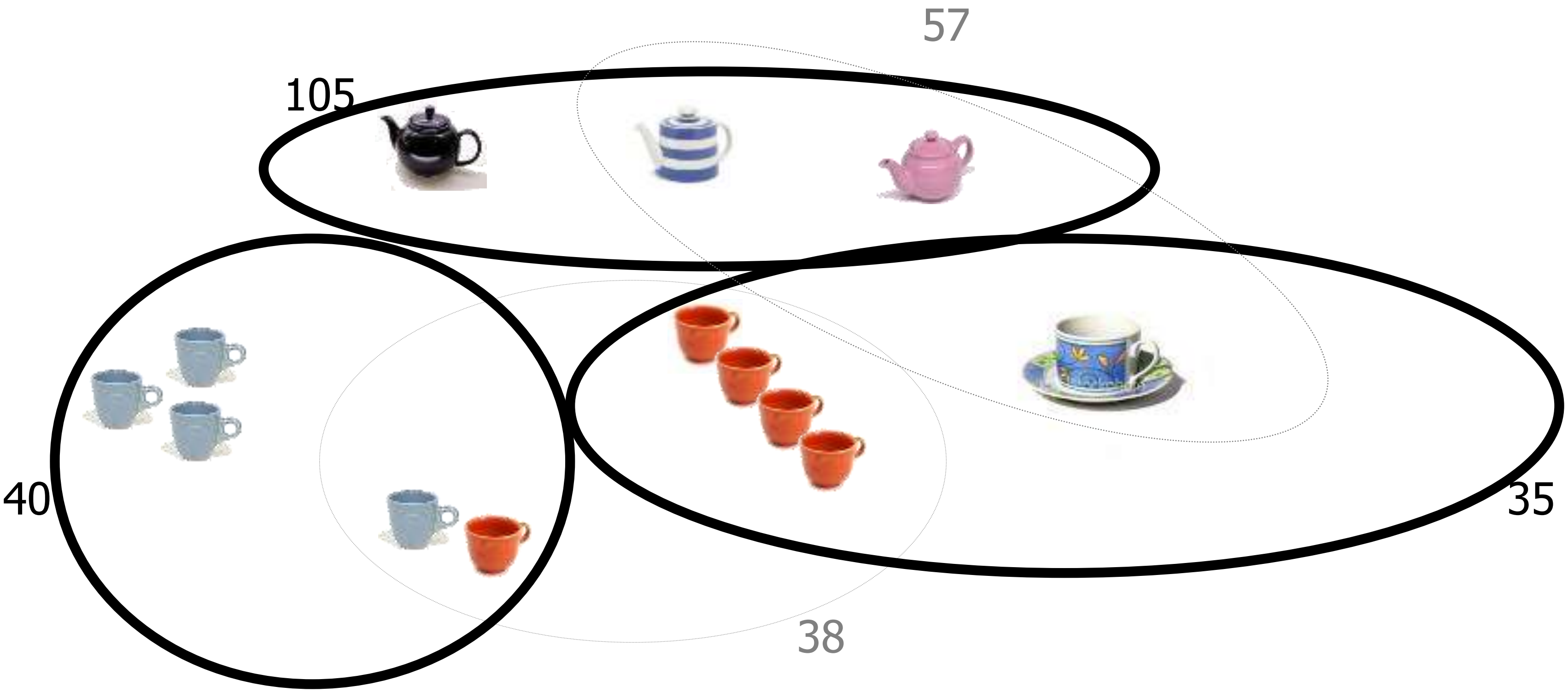
Example: Combinatorial Auctions



Winner Determination Problem

- Determine the outcome that maximizes the sum of accepted bid prices

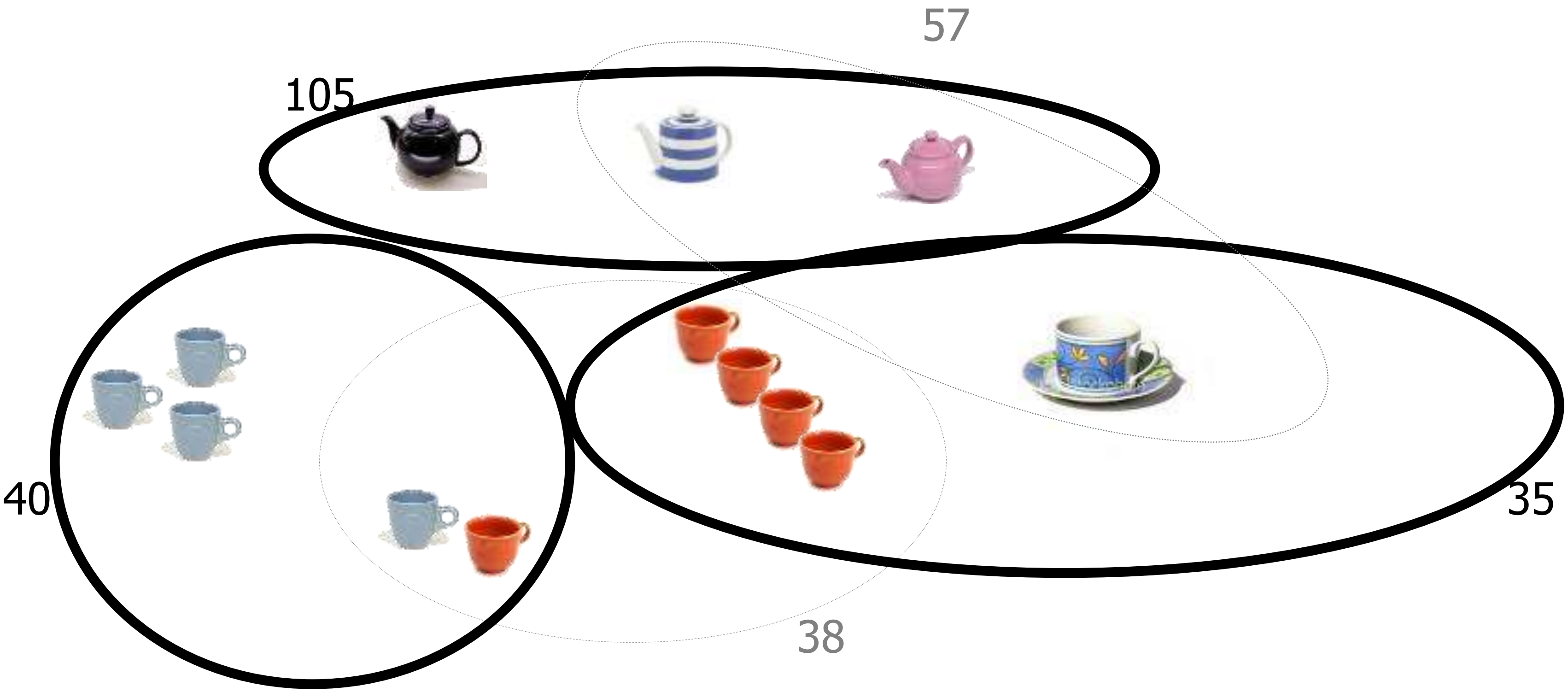
Example: Combinatorial Auctions



Winner Determination Problem ⇒ **180**

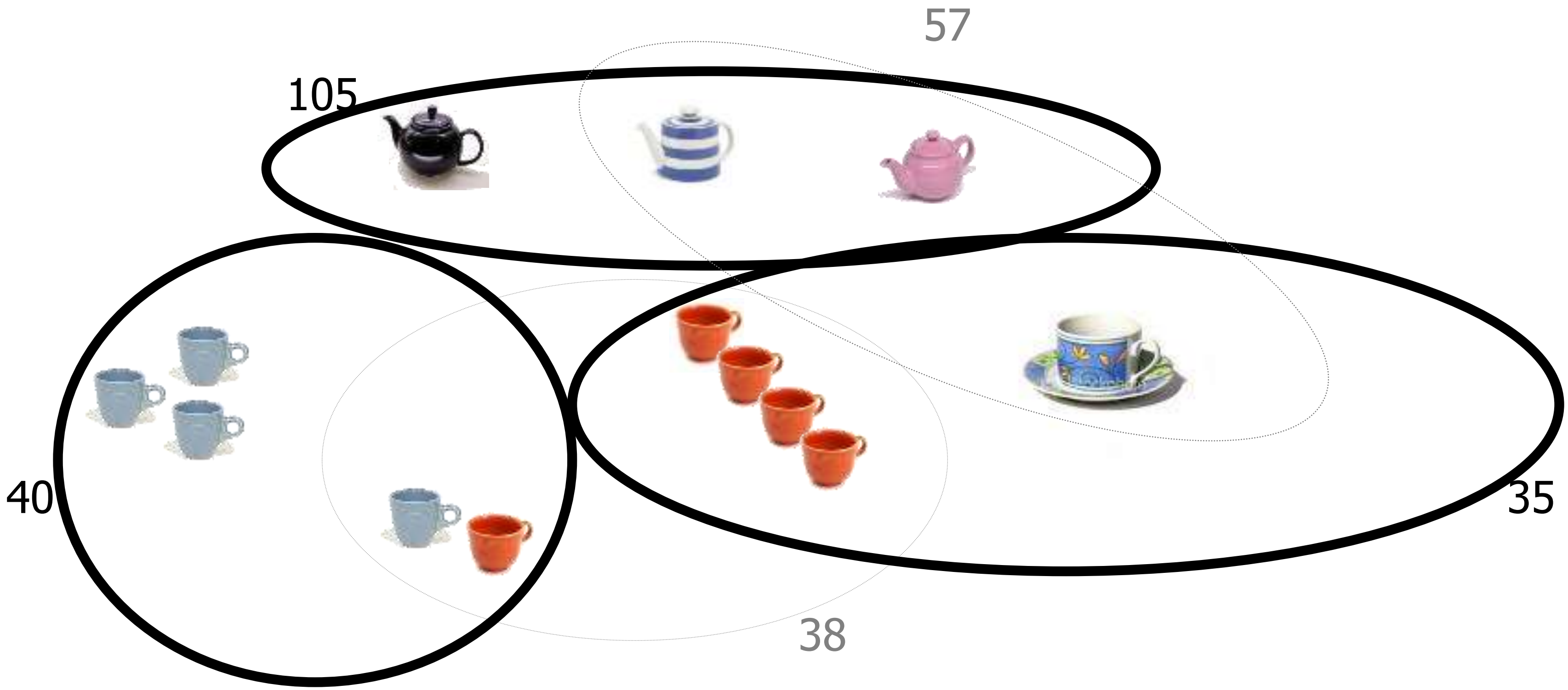
- Determine the outcome that maximizes the sum of accepted bid prices

Example: Combinatorial Auctions



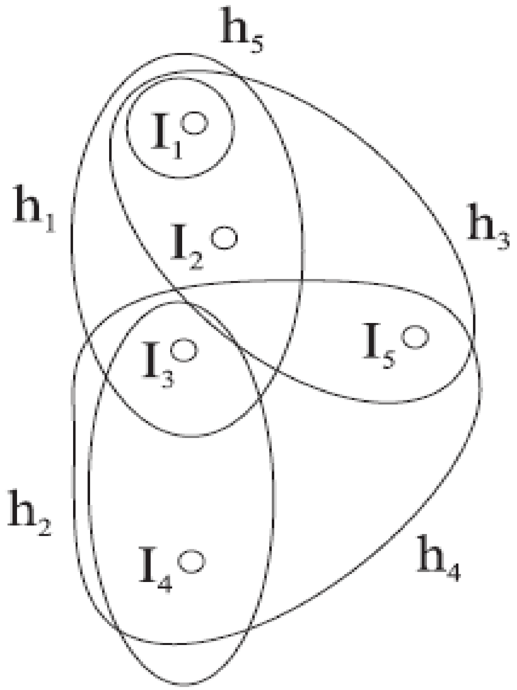
- Other applications [Cramton, Shoham, and Steinberg, ‘06]
  - airport runway access
  - trucking
  - bus routes
  - industrial procurement

Example: Combinatorial Auctions



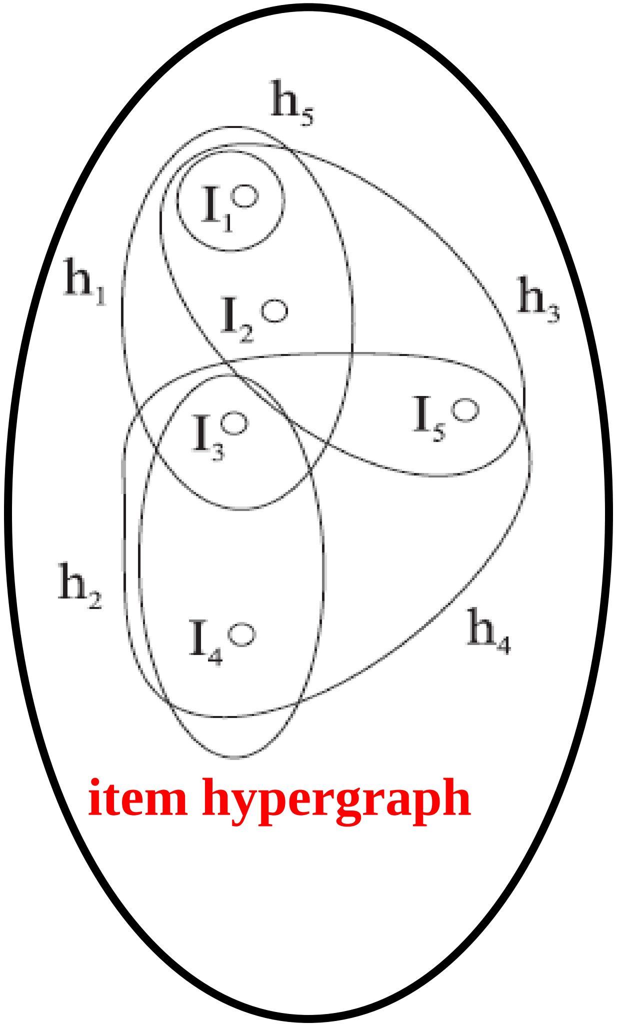
Winner Determination is NP-hard

Structural Properties



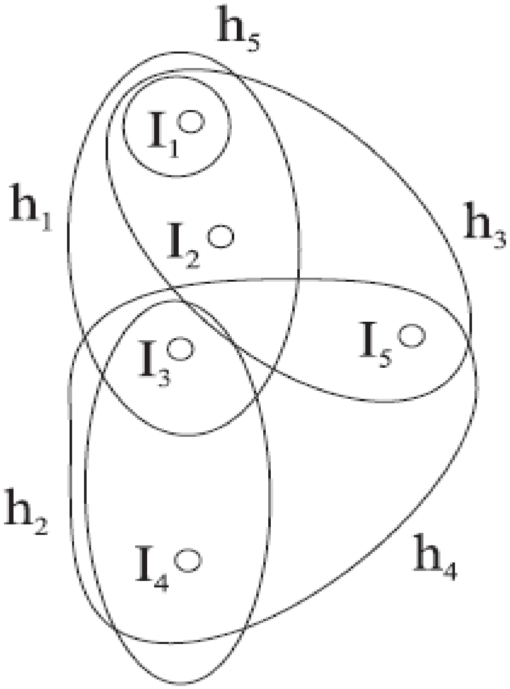
item hypergraph

Structural Properties



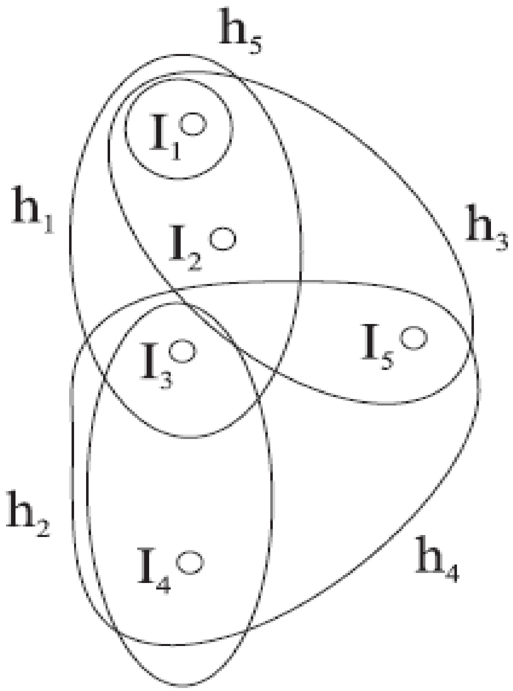
The Winner Determination Problem remains NP-hard even in case of acyclic hypergraphs

Dual Hypergraph



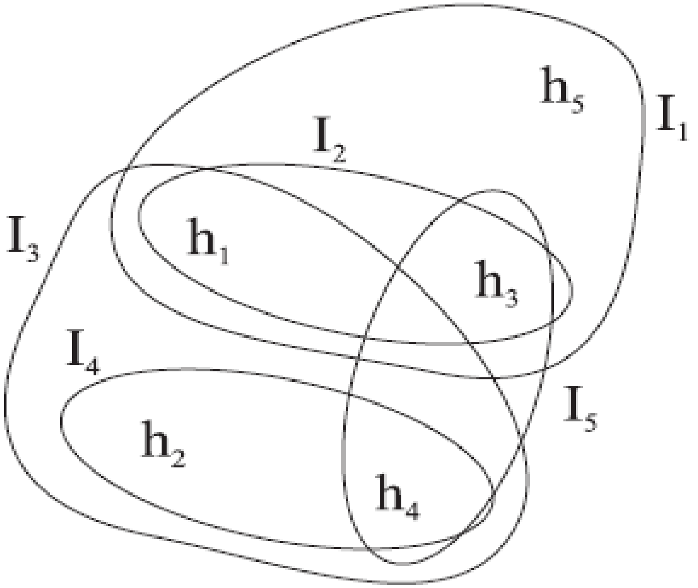
item hypergraph

Dual Hypergraph



item hypergraph

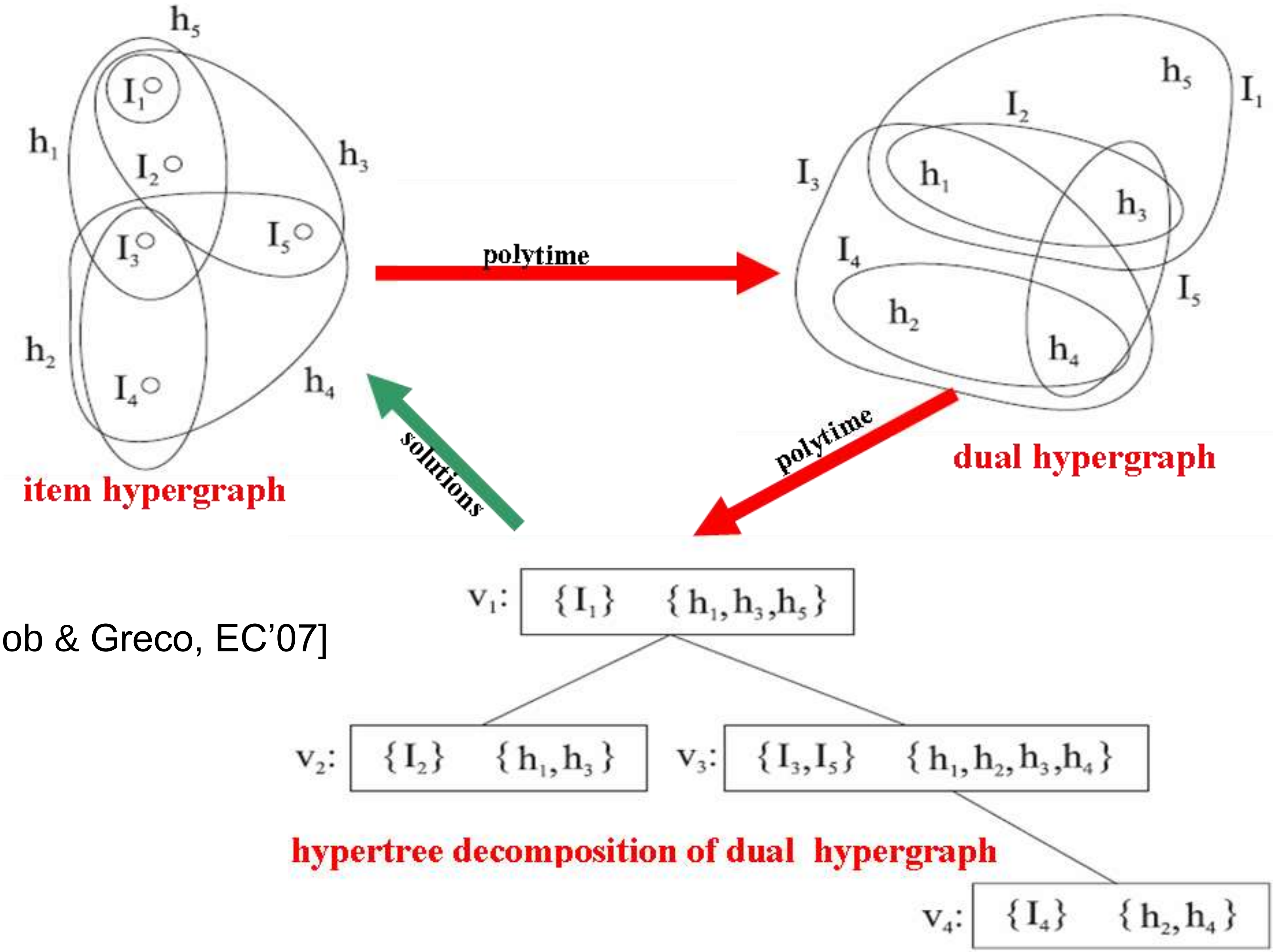
polytime



dual hypergraph

I_1	h_1	h_3	h_5
	0	0	0
	1	0	0
	0	1	0
	0	0	1

The Approach


h_5
I_1
h_1
I_2
h_3
I_3
I_5
h_2
I_4
h_4
item hypergraph
polytime
h_5
I_1
I_2
I_3
h_1
h_3
I_4
I_5
h_2
h_4
dual hypergraph
polytime
solutions
v_1: $\{I_1\}$ $\{h_1, h_3, h_5\}$
v_2: $\{I_2\}$ $\{h_1, h_3\}$
v_3: $\{I_3, I_5\}$ $\{h_1, h_2, h_3, h_4\}$
v_4: $\{I_4\}$ $\{h_2, h_4\}$
hypertree decomposition of dual hypergraph

[Gottlob & Greco, EC’07]

Outline of Part III

Applications to Optimization Problems

Application: Nash Equilibria

Application: Coalitional Games

Application: Combinatorial Auctions

Appendix: Beyond Hypertree Width

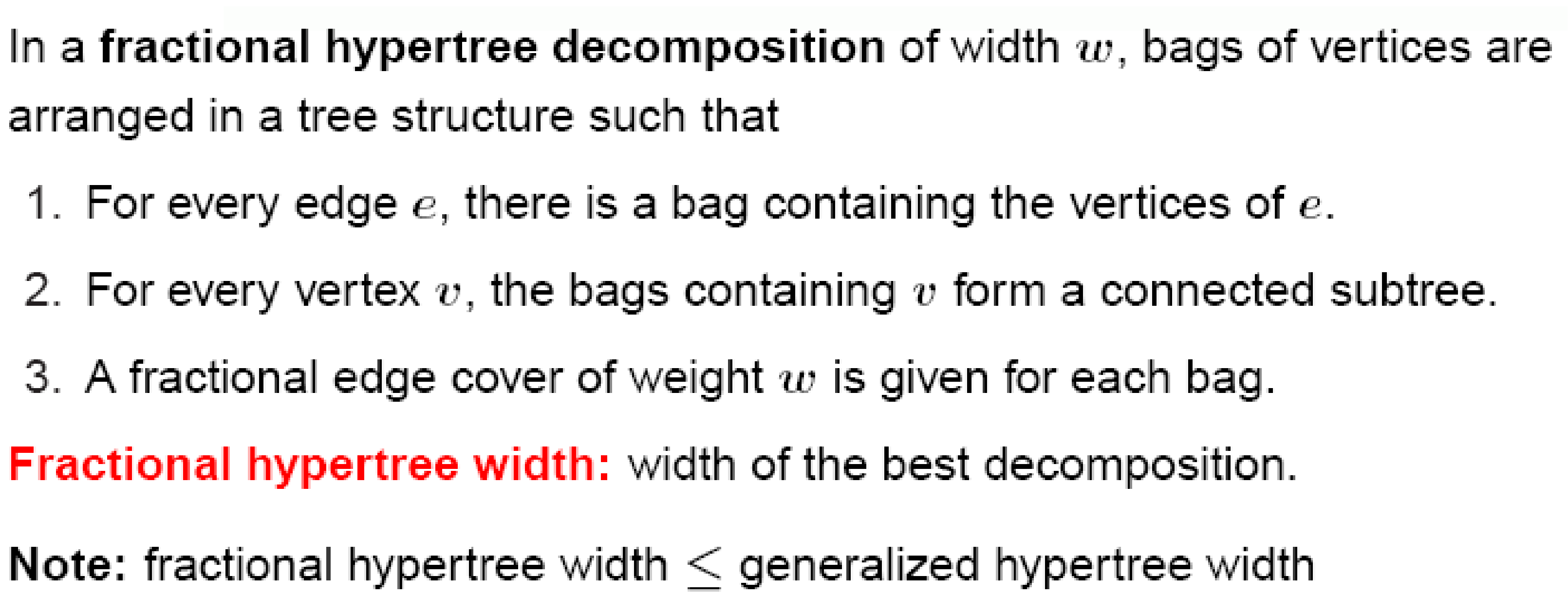
Going Beyond…

- Treewidth and Hypertree width are based on tree-like aggregations of subproblems that are efficiently solvable
- k variables (resp. k atoms) ➔ $||I||^k$ solutions (per subproblem)
- Is there some more general property that makes the number of solutions in any bag polynomial?

- YES!

  [Grohe & Marx ’06]

Fractional Hypertree Decompositions



In a **fractional hypertree decomposition** of width w, bags of vertices are arranged in a tree structure such that

1. For every edge e, there is a bag containing the vertices of e.
2. For every vertex v, the bags containing v form a connected subtree.
3. A fractional edge cover of weight w is given for each bag.

Fractional hypertree width: width of the best decomposition.

Note: fractional hypertree width $\leq$ generalized hypertree width

[Grohe & Marx '06]

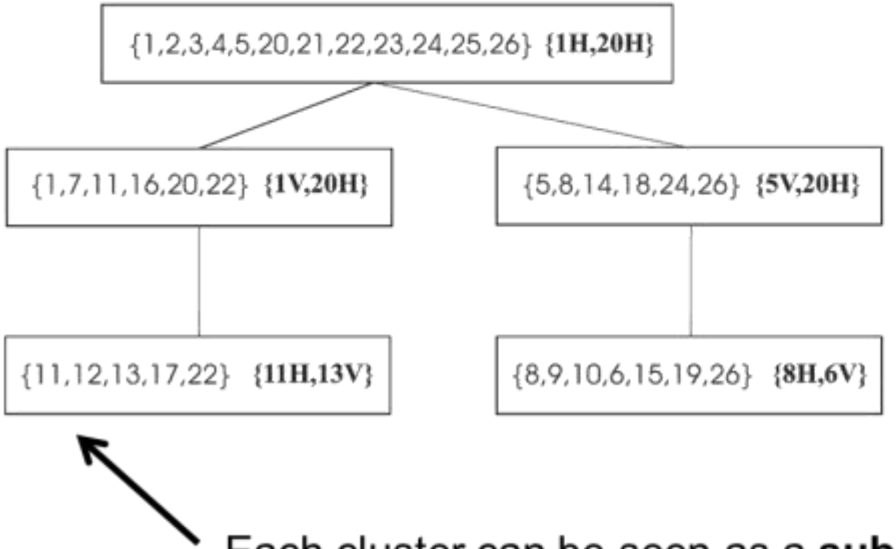
- A query may be solved efficiently, if a fractional hypertree decomposition is given
- FHDs are approximable: If the the width is ≤ *w*, a decomposition of width $O(w^3)$ may be computed in polynomial time [Marx '09]

More Beyond?

- A new notion: the submodular width
- Bounded submodular width is a necessary and sufficient condition for fixed-parameter tractability
  (under a technical complexity assumption)

[Marx ‘10]

Revisiting Decomposition Methods



{1,2,3,4,5,20,21,22,23,24,25,26} **{1H,20H}**

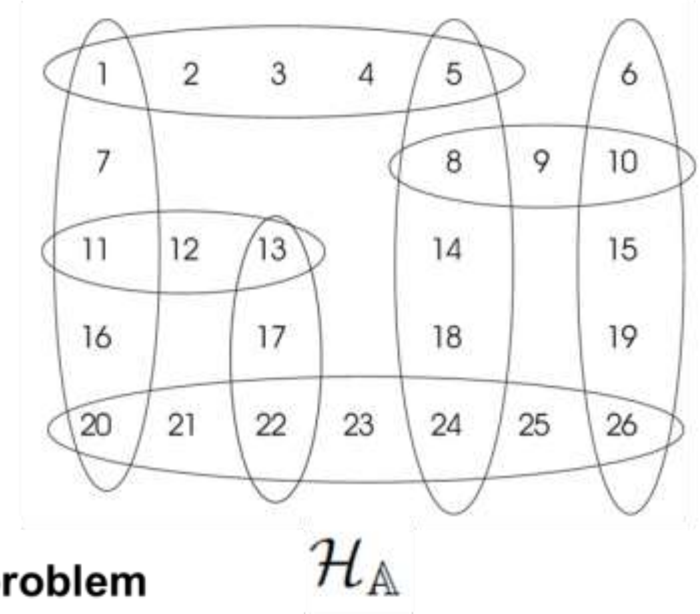
{1,7,11,16,20,22} **{1V,20H}**

{5,8,14,18,24,26} **{5V,20H}**

{11,12,13,17,22} **{11H,13V}**

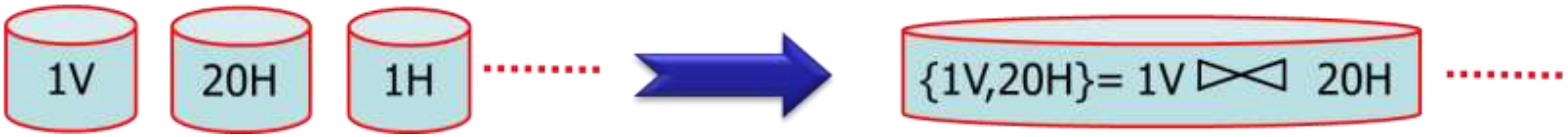
{8,9,10,6,15,19,26} **{8H,6V}**

Each cluster can be seen as a **subproblem**



Relations:

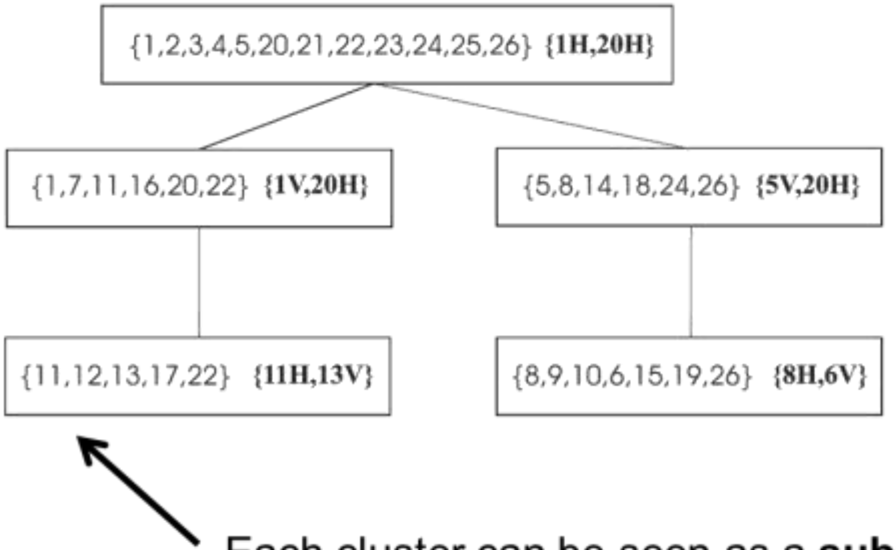
Relations:



1V 20H 1H ....

{1V,20H}= 1V ⋈ 20H ....

Revisiting Decomposition Methods



{1,2,3,4,5,20,21,22,23,24,25,26} {1H,20H}

{1,7,11,16,20,22} {1V,20H}

{5,8,14,18,24,26} {5V,20H}

{11,12,13,17,22} {11H,13V}

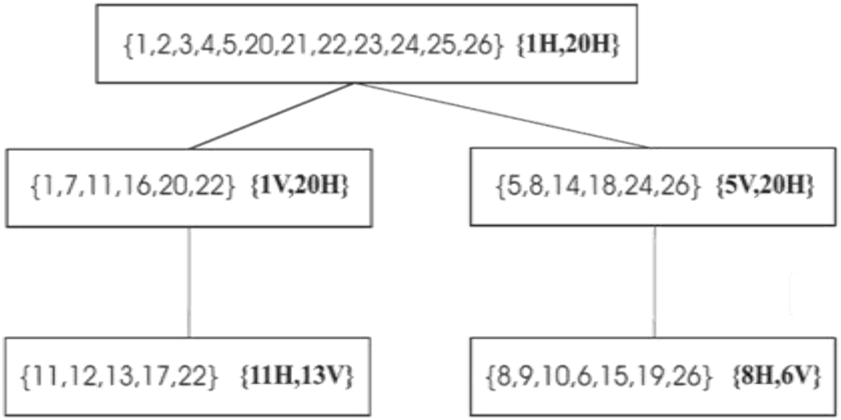
{8,9,10,6,15,19,26} {8H,6V}

Each cluster can be seen as a **subproblem**

Relations:

$\{1V,20H\}= 1V \bowtie 20H$ ....

Revisiting Decomposition Methods



{1,2,3,4,5,20,21,22,23,24,25,26} {1H,20H}

{1,7,11,16,20,22} {1V,20H}

{5,8,14,18,24,26} {5V,20H}

{11,12,13,17,22} {11H,13V}

{8,9,10,6,15,19,26} {8H,6V}

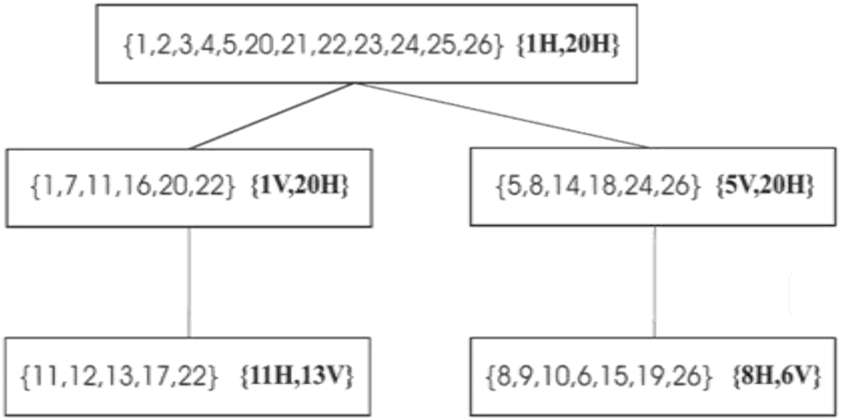
Relations:

$\{1V,20H\}= 1V \bowtie 20H$

Revisiting Decomposition Methods

CSP instance $(\mathbb{A}, \mathbb{B})$

$$\mathbb{A}_{\mathcal{V}} = \ell\text{-DM}(\mathbb{A}) \quad \mathbb{B}_{\mathcal{V}} = r\text{-DM}(\mathbb{A}, \mathbb{B})$$



{1,2,3,4,5,20,21,22,23,24,25,26} **{1H,20H}**

{1,7,11,16,20,22} **{1V,20H}**

{5,8,14,18,24,26} **{5V,20H}**

{11,12,13,17,22} **{11H,13V}**

{8,9,10,6,15,19,26} **{8H,6V}**

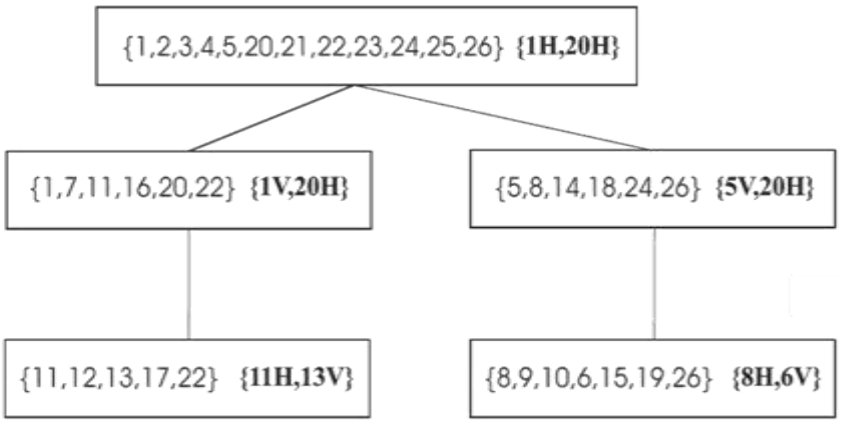
Relations:

{1V,20H}= 1V $\bowtie$ 20H ………

Revisiting Decomposition Methods

CSP instance $(\mathbb{A}, \mathbb{B})$

$$\underbrace{\underbrace{\mathbb{A}_{\mathcal{V}} = \ell\text{-DM}(\mathbb{A})}_{\text{Scopes}} \;\Big|\; \underbrace{\mathbb{B}_{\mathcal{V}} = r\text{-DM}(\mathbb{A}, \mathbb{B})}_{\text{Solutions}}}_{\text{Work on subproblems}}$$


{1,2,3,4,5,20,21,22,23,24,25,26} {1H,20H}
{1,7,11,16,20,22} {1V,20H}
{5,8,14,18,24,26} {5V,20H}
{11,12,13,17,22} {11H,13V}
{8,9,10,6,15,19,26} {8H,6V}

Relations:

$\{1V,20H\} = 1V \bowtie 20H$ ....

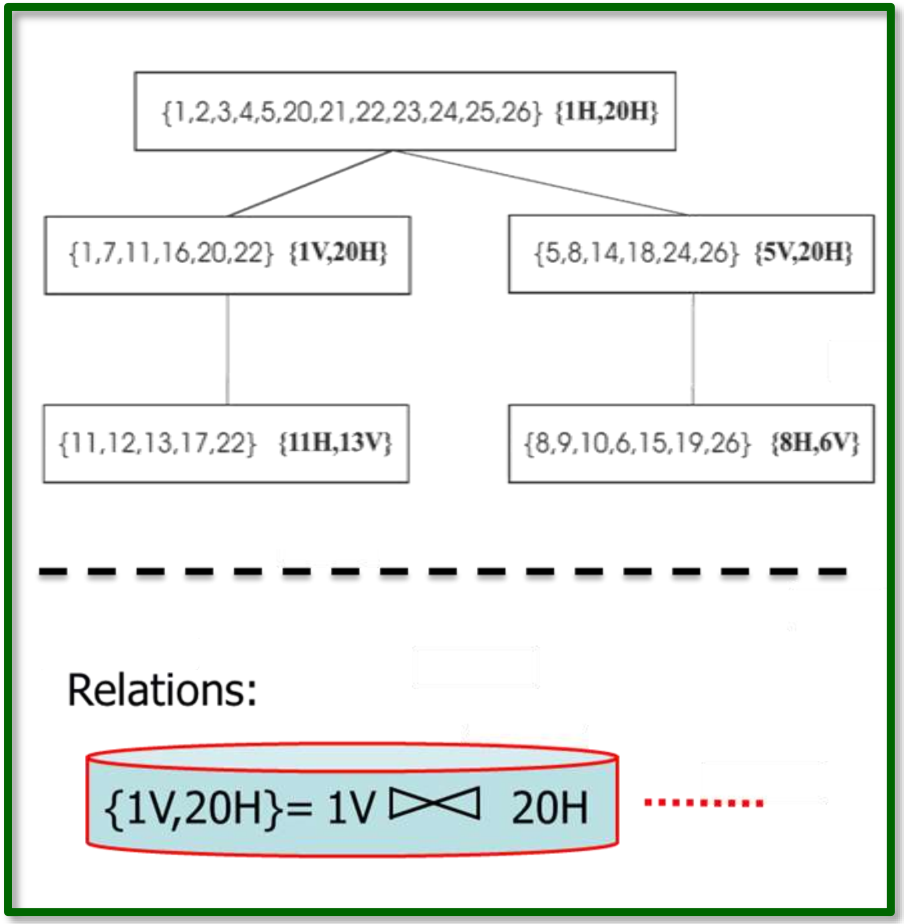
Revisiting Decomposition Methods

CSP instance $(\mathbb{A}, \mathbb{B})$

$$\underbrace{\mathbb{A}_{\mathcal{V}} = \ell\text{-DM}(\mathbb{A})}_{\text{Scopes}} \;\Big|\; \underbrace{\mathbb{B}_{\mathcal{V}} = r\text{-DM}(\mathbb{A}, \mathbb{B})}_{\text{Solutions}}$$

Work on subproblems

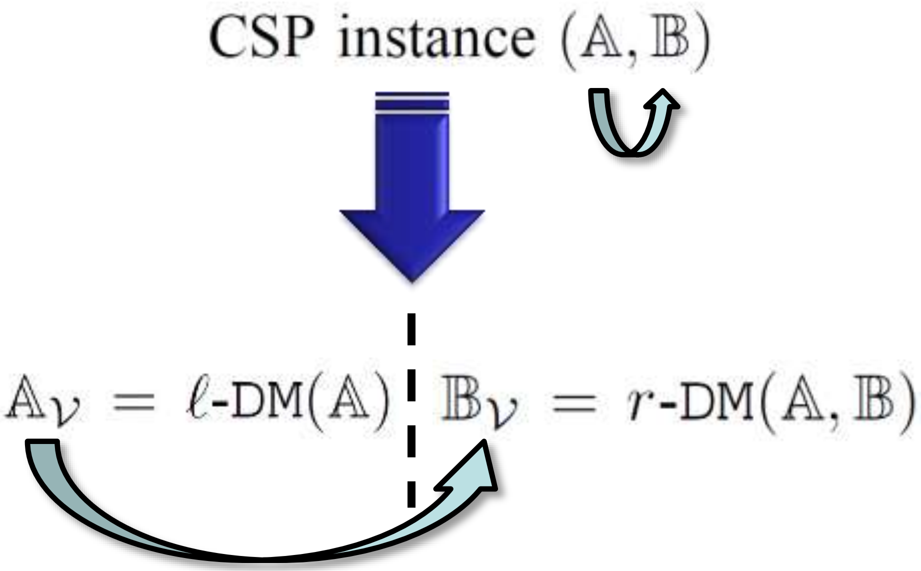
- *Generalized hypertree width*: take all views that can be computed by joining at most k atoms (k query views)


{1,2,3,4,5,20,21,22,23,24,25,26} {1H,20H}
{1,7,11,16,20,22} {1V,20H}
{5,8,14,18,24,26} {5V,20H}
{11,12,13,17,22} {11H,13V}
{8,9,10,6,15,19,26} {8H,6V}

Relations:

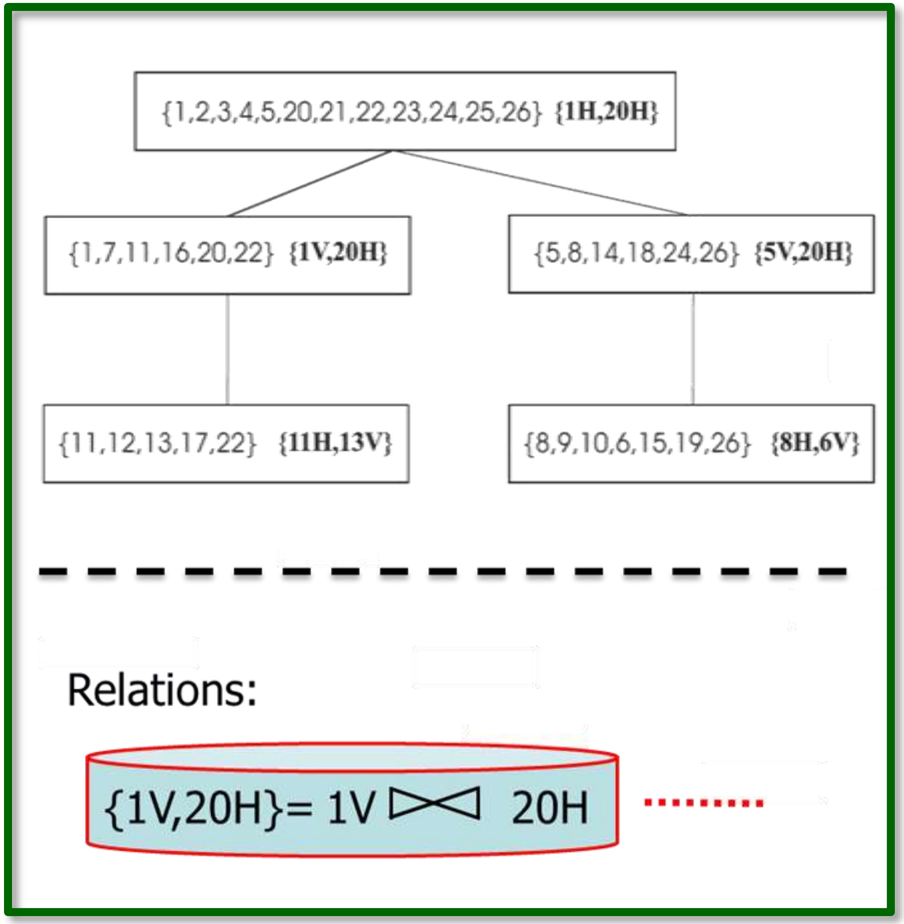
{1V,20H}= 1V ⋈ 20H ....

Revisiting Decomposition Methods



CSP instance $(\mathbb{A},\mathbb{B})$

$\mathbb{A}_\mathcal{V} = \ell\text{-DM}(\mathbb{A})$ $\mathbb{B}_\mathcal{V} = r\text{-DM}(\mathbb{A},\mathbb{B})$



{1,2,3,4,5,20,21,22,23,24,25,26} **{1H,20H}**

{1,7,11,16,20,22} **{1V,20H}**

{5,8,14,18,24,26} **{5V,20H}**

{11,12,13,17,22} **{11H,13V}**

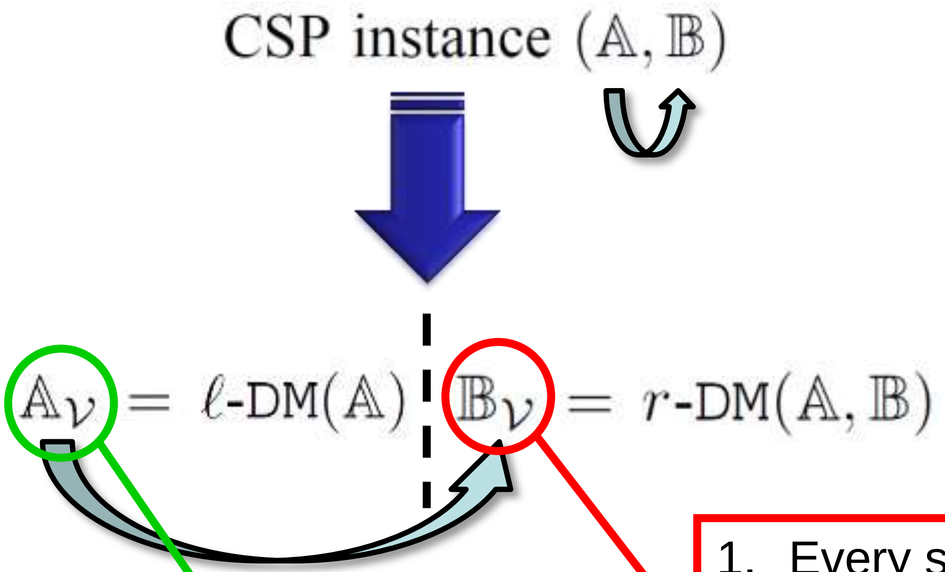
{8,9,10,6,15,19,26} **{8H,6V}**

Relations:

$\{1V,20H\}= 1V \bowtie 20H$ ....

- *Generalized hypertree width*: take all views that can be computed by joining at most k atoms (k query views)

Requirements on Subproblem Definition



CSP instance $(\mathbb{A}, \mathbb{B})$

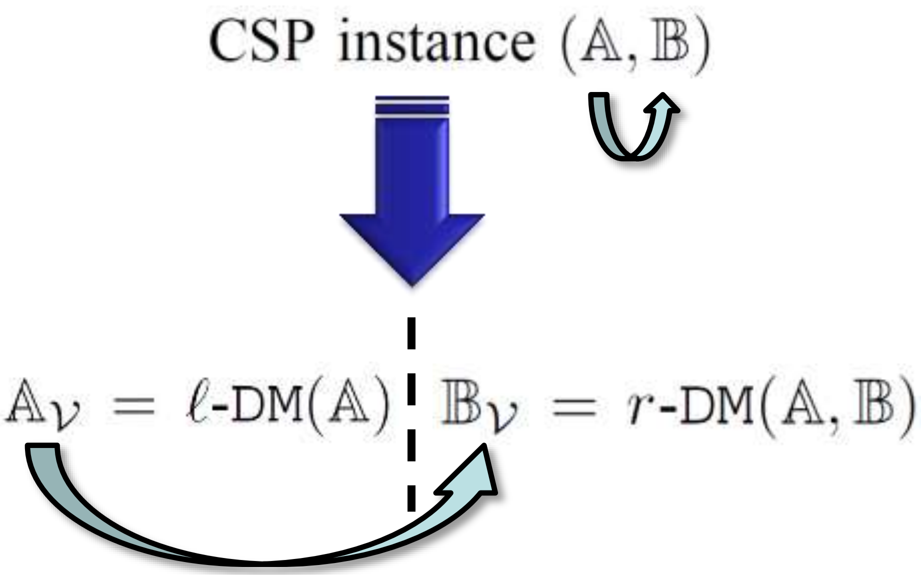
$\mathbb{A}_{\mathcal{V}} = \ell\text{-DM}(\mathbb{A})$

$\mathbb{B}_{\mathcal{V}} = r\text{-DM}(\mathbb{A}, \mathbb{B})$

1. Every subproblem is not more restrictive than the full problem
2. Every base subproblem is at least restrictive as the corresponding constraint

1. Every constraint is associated with a base subproblem
2. Further subproblems can be defined

Acyclicity in Decomposition Methods

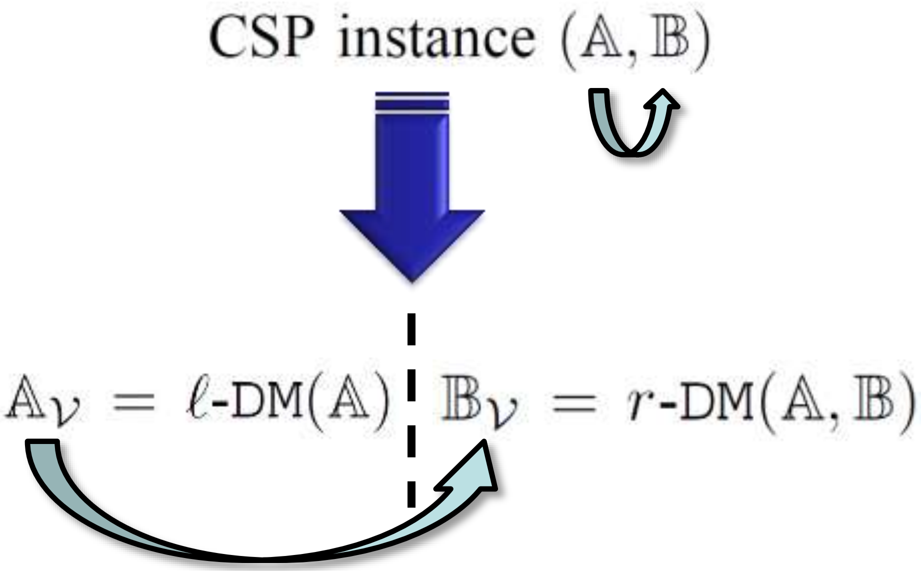


CSP instance $(\mathbb{A}, \mathbb{B})$

$\mathbb{A}_{\mathcal{V}} = \ell\text{-DM}(\mathbb{A})$ $\mathbb{B}_{\mathcal{V}} = r\text{-DM}(\mathbb{A}, \mathbb{B})$

Working on subproblems is not necessarily beneficial…

Acyclicity in Decomposition Methods



CSP instance $(\mathbb{A}, \mathbb{B})$

$\mathbb{A}_{\mathcal{V}} = \ell\text{-DM}(\mathbb{A})$ $\mathbb{B}_{\mathcal{V}} = r\text{-DM}(\mathbb{A}, \mathbb{B})$

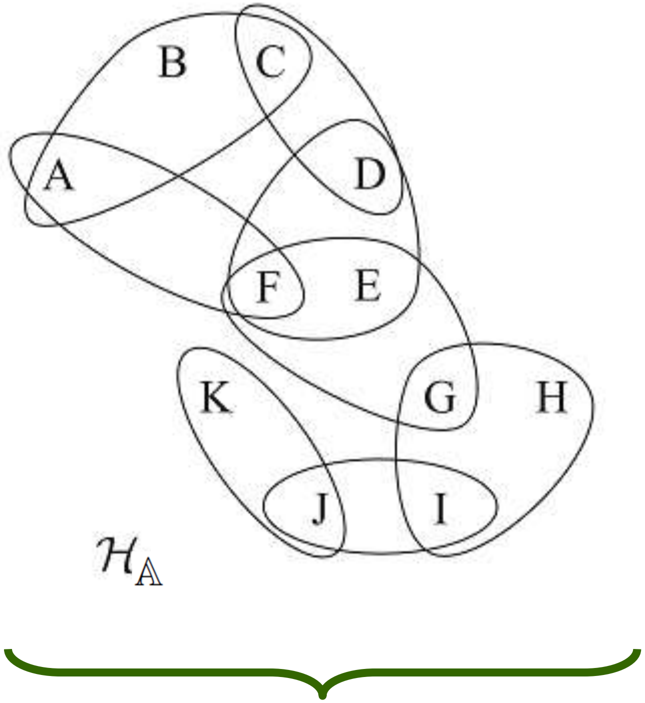
Working on subproblems is not necessarily beneficial…

Can some and/or portions of them be selected such that:

- They still cover $\mathbb{A}$, and
- They can be arranged as a tree

Tree Projections (by Example)

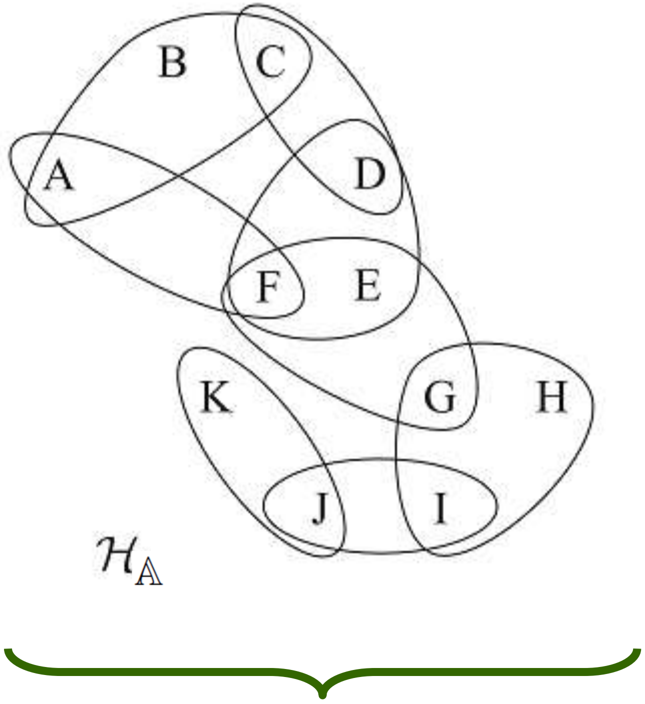
$$\mathbb{A}: \quad r_1(A,B,C) \quad r_2(A,F) \quad r_3(C,D) \quad r_4(D,E,F)$$
$$r_5(E,F,G) \quad r_6(G,H,I) \quad r_7(I,J) \quad r_8(J,K)$$



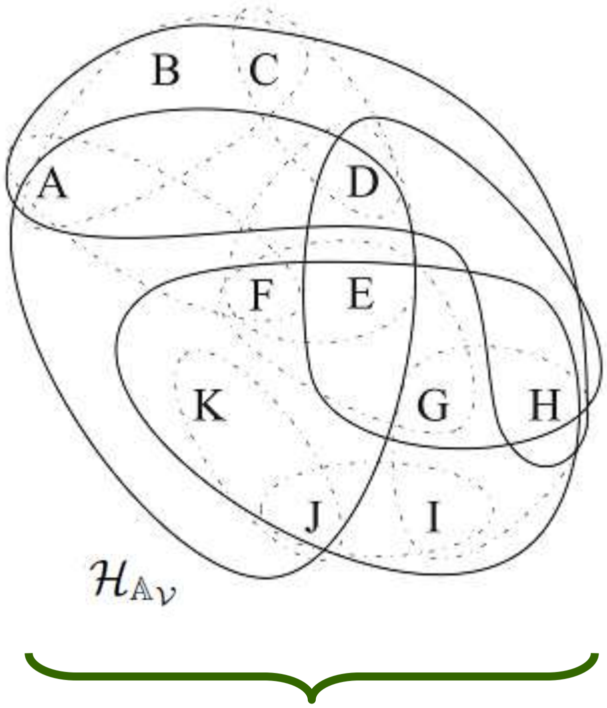
Structure of the CSP

Tree Projections (by Example)

$\mathbb{A}$: $r_1(A,B,C)$ $r_2(A,F)$ $r_3(C,D)$ $r_4(D,E,F)$
$r_5(E,F,G)$ $r_6(G,H,I)$ $r_7(I,J)$ $r_8(J,K)$



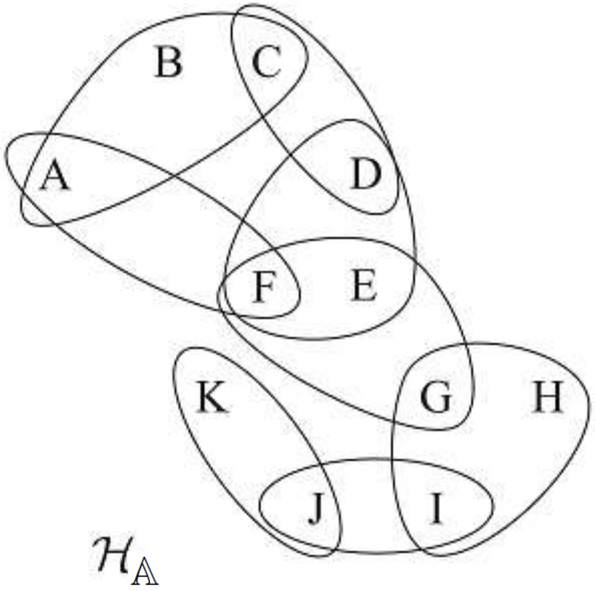
Structure of the CSP

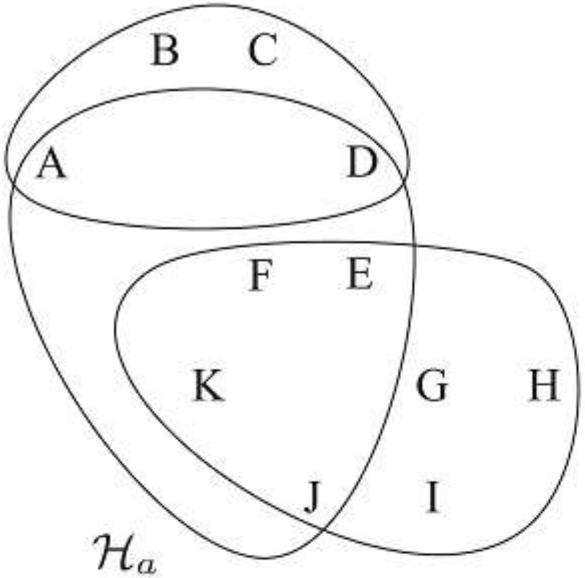


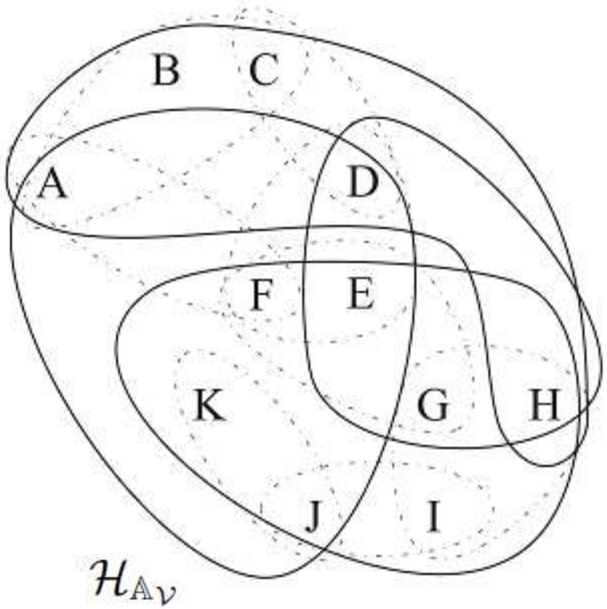
Available Views

Tree Projections (by Example)

$$\mathbb{A}: \quad r_1(A,B,C) \quad r_2(A,F) \quad r_3(C,D) \quad r_4(D,E,F)$$
$$r_5(E,F,G) \quad r_6(G,H,I) \quad r_7(I,J) \quad r_8(J,K)$$







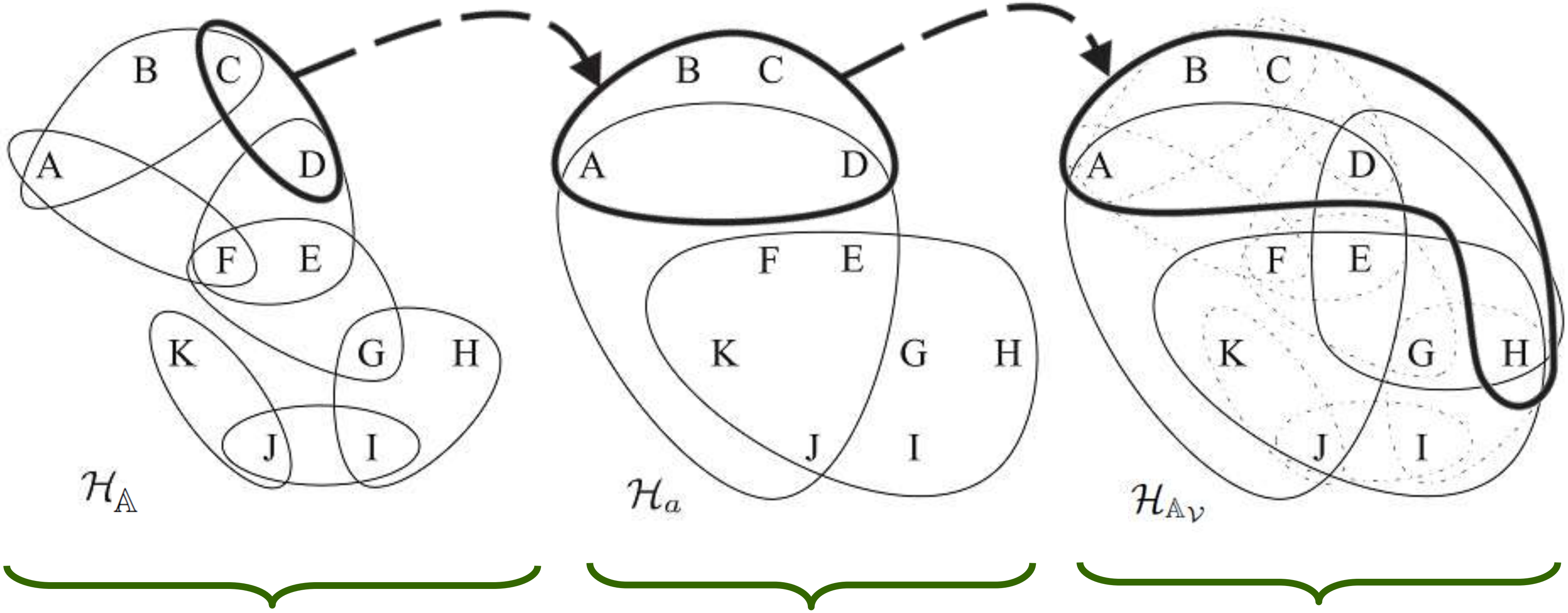
Structure of the CSP

Tree Projection

Available Views

Tree Projections (by Example)

$\mathbb{A}: \quad r_1(A,B,C) \quad r_2(A,F) \quad r_3(C,D) \quad r_4(D,E,F)$
$\quad r_5(E,F,G) \quad r_6(G,H,I) \quad r_7(I,J) \quad r_8(J,K)$

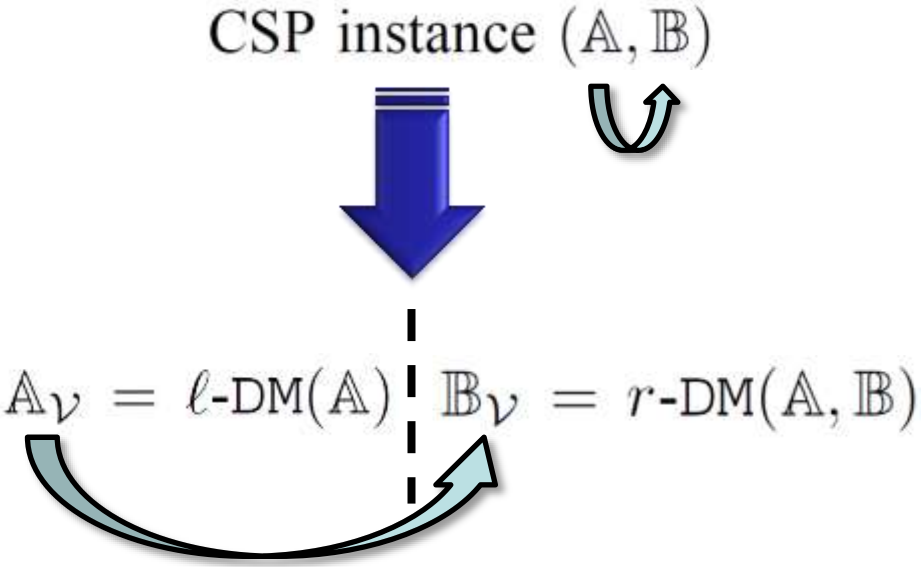

B
C
A
D
F
E
K
G
H
J
I
$\mathcal{H}_{\mathbb{A}}$
B
C
A
D
F
E
K
G
H
J
I
$\mathcal{H}_a$
B
C
A
D
F
E
K
G
H
J
I
$\mathcal{H}_{\mathbb{A}_{\mathcal{V}}}$

Structure of the CSP

Tree Projection

Available Views

(Noticeable) Examples

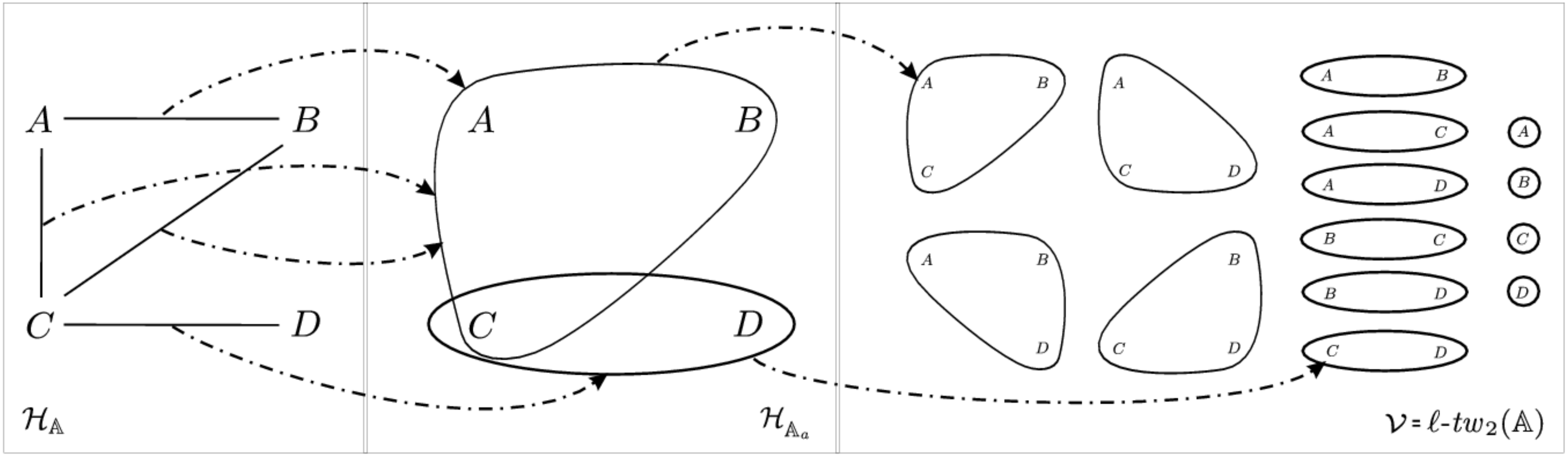


CSP instance $(\mathbb{A}, \mathbb{B})$

$\mathbb{A}_{\mathcal{V}} = \ell\text{-DM}(\mathbb{A})$ $\mathbb{B}_{\mathcal{V}} = r\text{-DM}(\mathbb{A}, \mathbb{B})$

- *Treewidth*: take all views that can be computed with at most k variables
- *Generalized hypertree width*: take all views that can be computed by joining at most k atoms (k query views)
- *Fractional hypertree width*: take all views that can be computed through subproblems having fractional cover at most k (or use Marx’s O(k^3) approximation to have polynomially many views)

Tree Decomposition


A
B
C
D
$\mathcal{H}_{\mathbb{A}}$
A
B
C
D
$\mathcal{H}_{\mathbb{A}_a}$
A B C
A C D
A B D
B C D
A B
A C
A D
B C
B D
C D
A
B
C
D
$\mathcal{V} = \ell\text{-}tw_2(\mathbb{A})$

A General Framework, but

- Decide the existence of a tree projection is **NP-hard**

[Gottlob, Miklos, and Schwentick, JACM‘09]

A General Framework, but

- Decide the existence of a tree projection is **NP-hard**

Hold on generalized hypertree width too.

[Gottlob, Miklos, and Schwentick, JACM‘09]

A Source of Complexity: The Core

The core of a query *Q* is a query *Q’* s.t.:

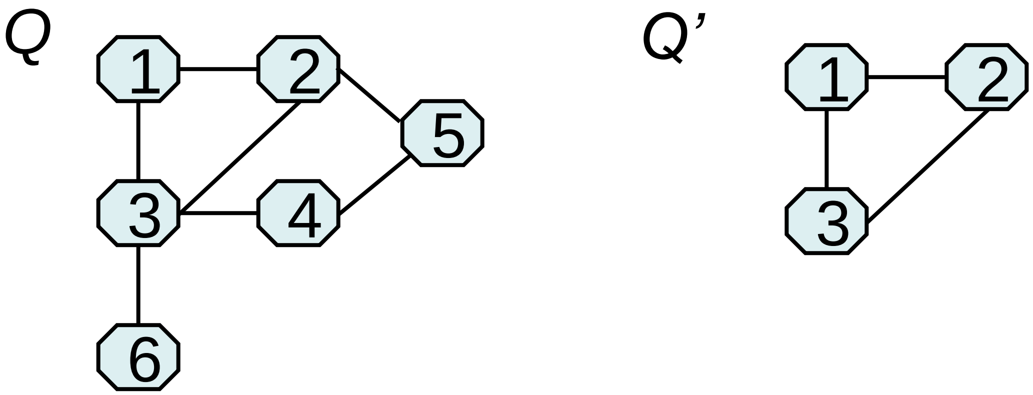
1. $atoms(Q') \subseteq atoms(Q)$
2. There is a mapping $h: var(Q) \rightarrow var(Q')$ s.t., $\forall\, r(\mathbf{X}) \in atoms(Q),\ \ r(h(\mathbf{X})) \in atoms(Q')$
3. There is no query Q’’ satisfying 1 and 2 and such that $atoms(Q'') \subset atoms(Q')$

A Source of Complexity: The Core

The core of a query *Q* is a query *Q'* s.t.:

1. $atoms(Q') \subseteq atoms(Q)$
2. There is a mapping $h: var(Q) \rightarrow var(Q')$ s.t., $\forall\, r(\mathbf{X}) \in atoms(Q),\ \ r(h(\mathbf{X})) \in atoms(Q')$
3. There is no query Q'' satisfying 1 and 2 and such that atoms(Q'') $\subset$ atoms(Q')

Example:

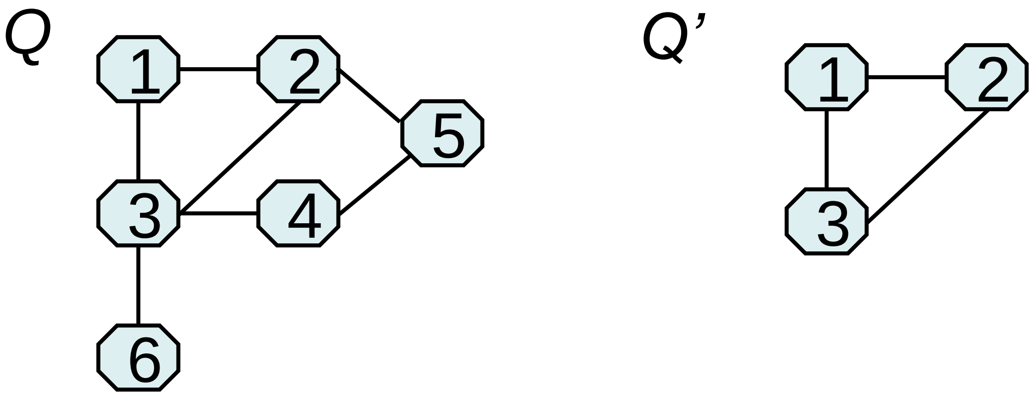


A Source of Complexity: The Core

Cores are isomorphic → The “Core”

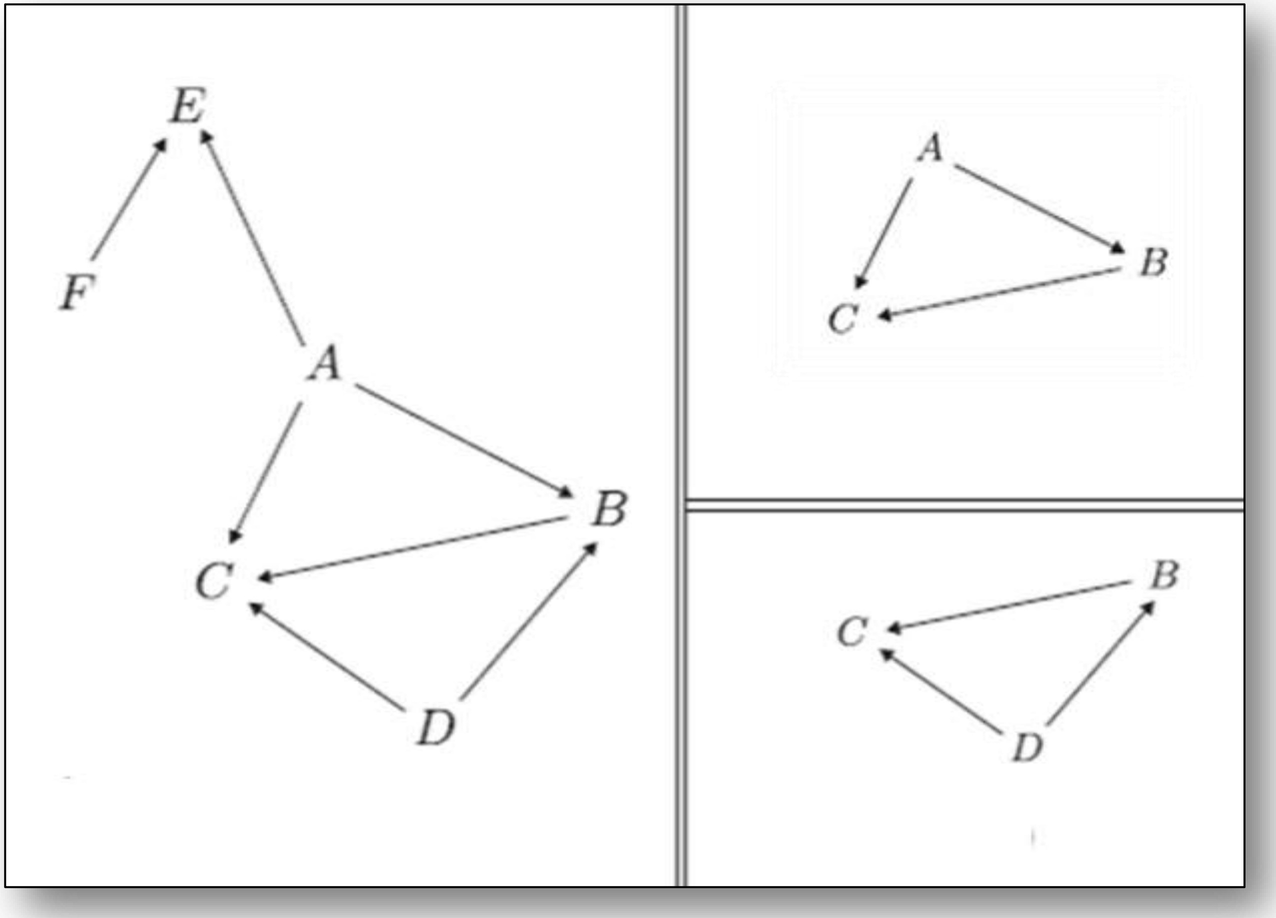
Cores are equivalent to the query

Example:



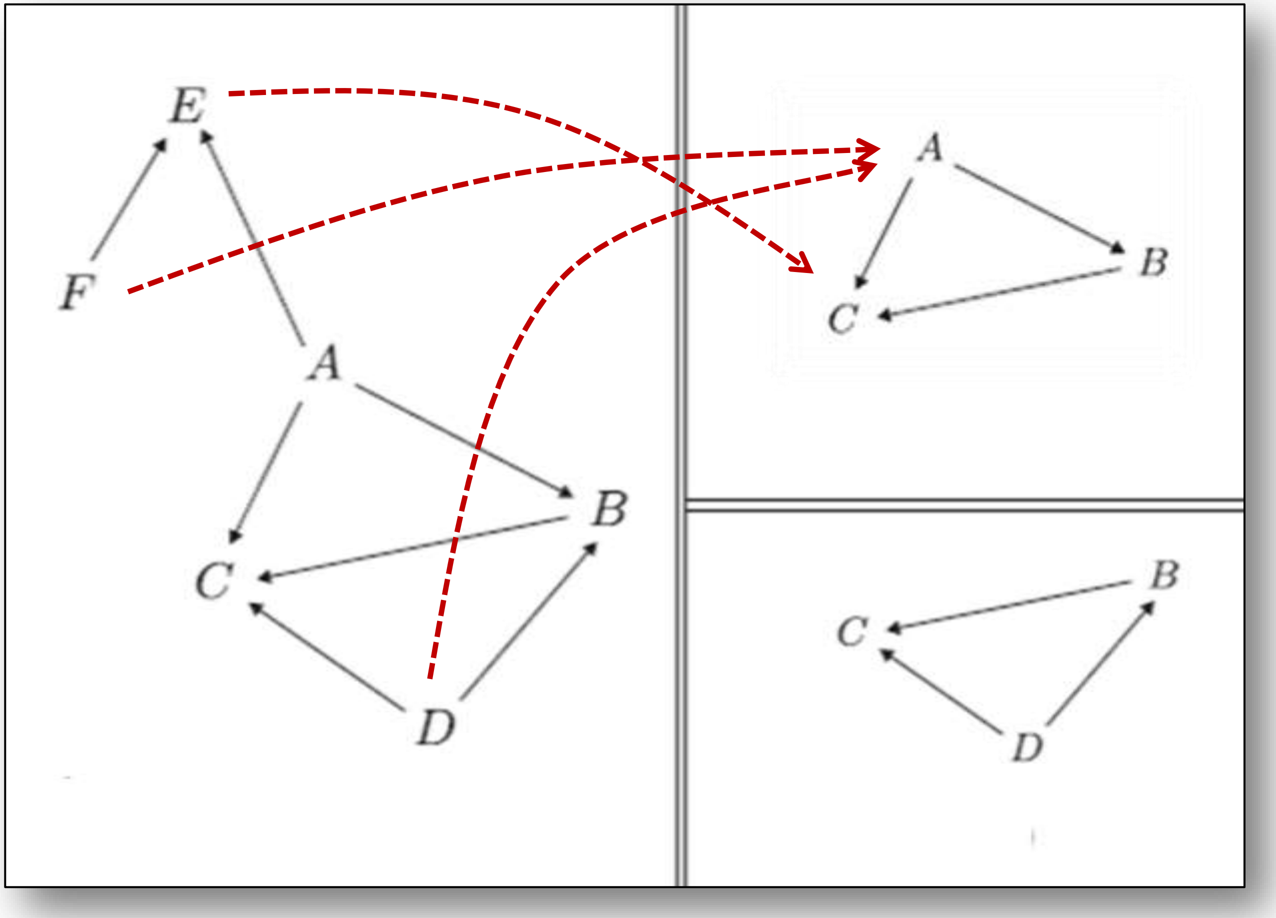
Example

$$Q \ : \quad r(A,B) \wedge r(B,C) \wedge r(A,C) \wedge r(D,C) \wedge \\ r(D,B) \wedge r(A,E) \wedge r(F,E),$$

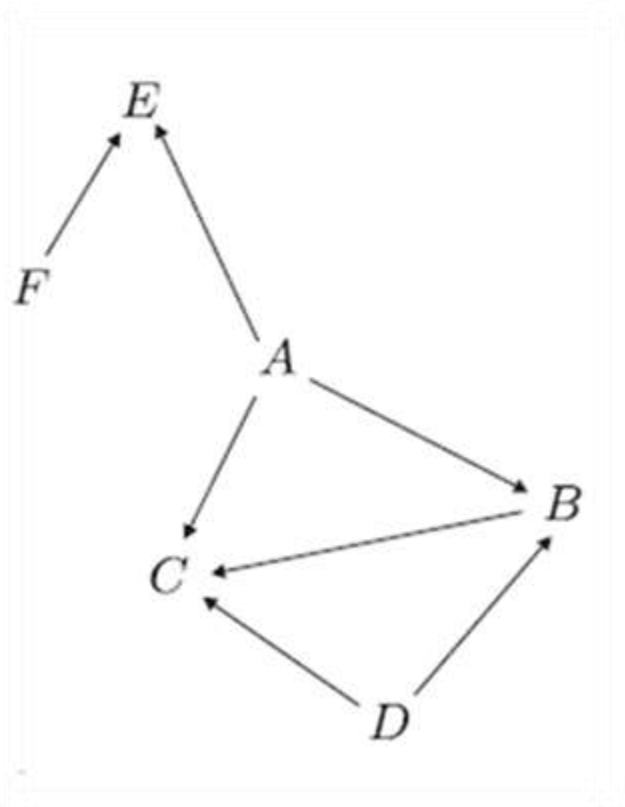


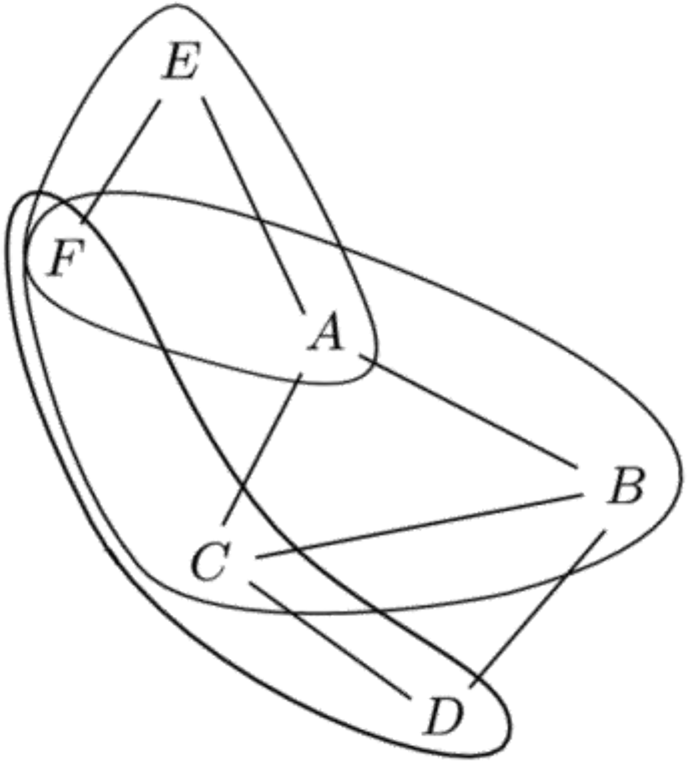
Example

$$Q \;:\; r(A,B) \wedge r(B,C) \wedge r(A,C) \wedge r(D,C) \wedge$$
$$r(D,B) \wedge r(A,E) \wedge r(F,E),$$



Cores and Tree Projections



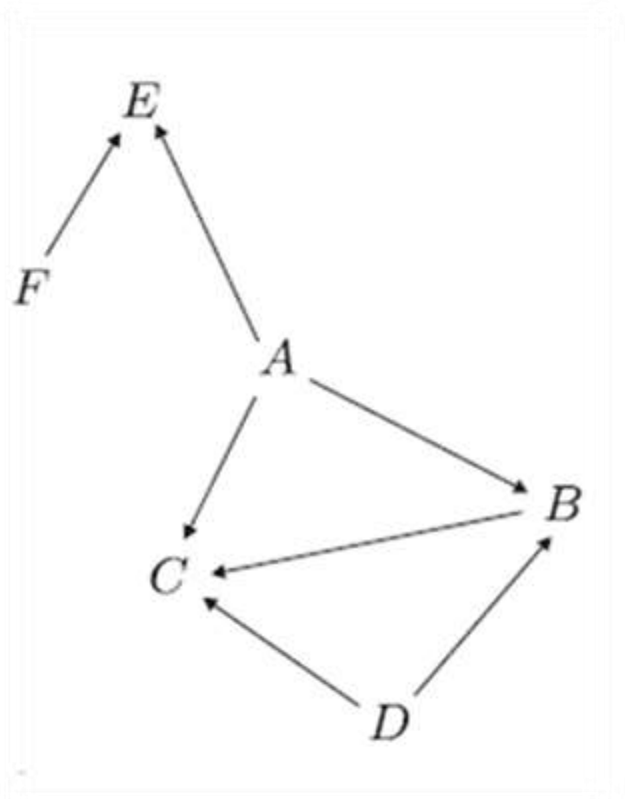


Structure of the CSP

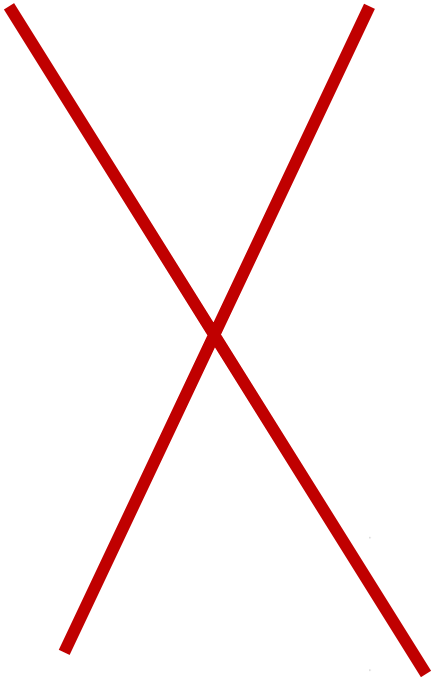
Tree Projection

Available Views

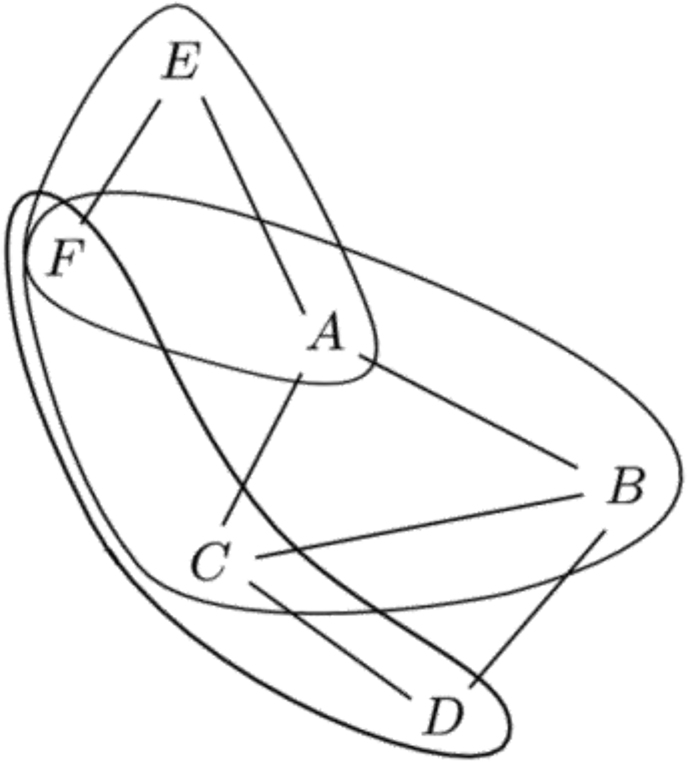
Cores and Tree Projections



Structure of the CSP

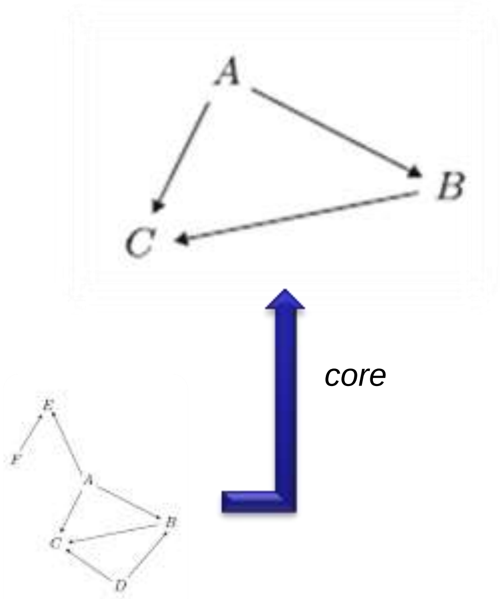


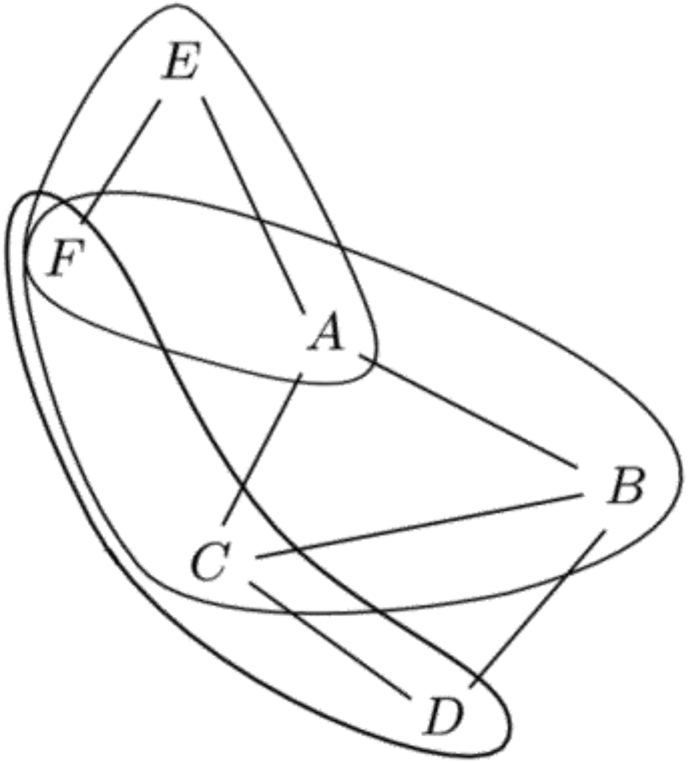
Tree Projection



Available Views

Cores and Tree Projections



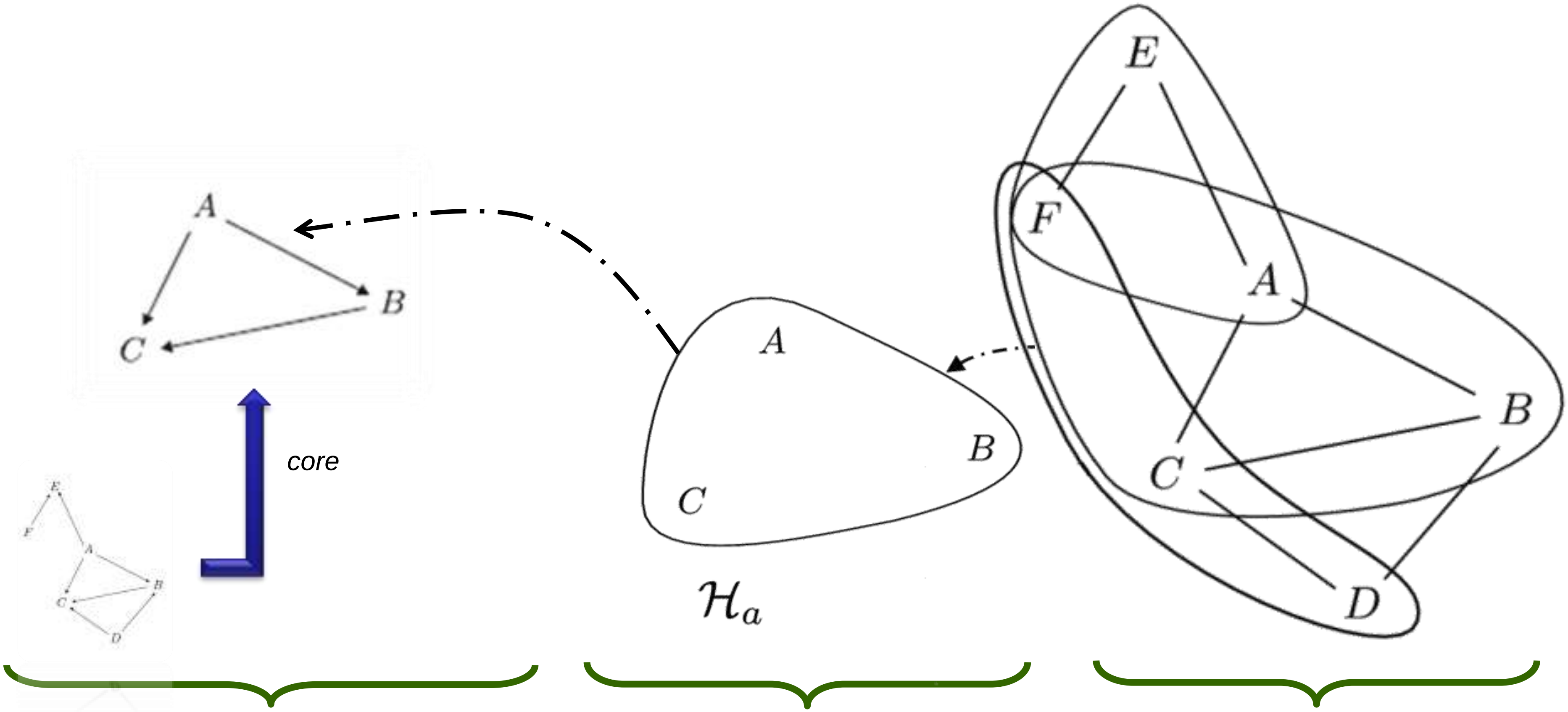


Structure of the CSP

Tree Projection

Available Views

Cores and Tree Projections



Structure of the CSP

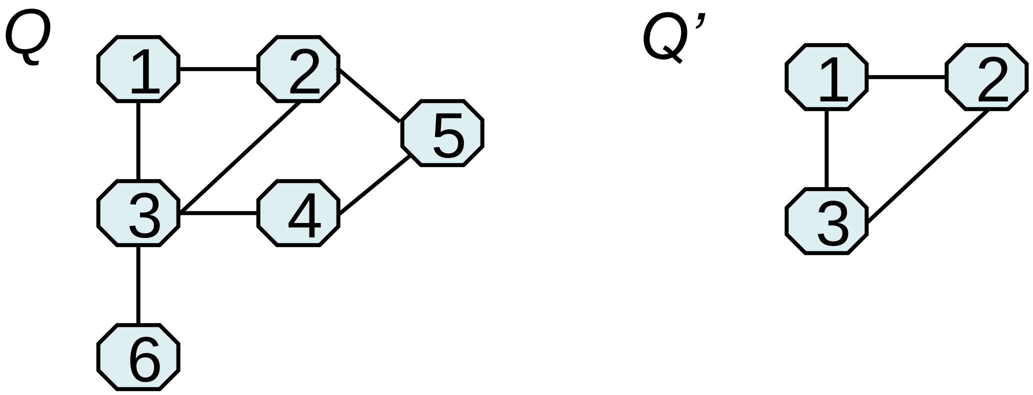
Tree Projection

Available Views

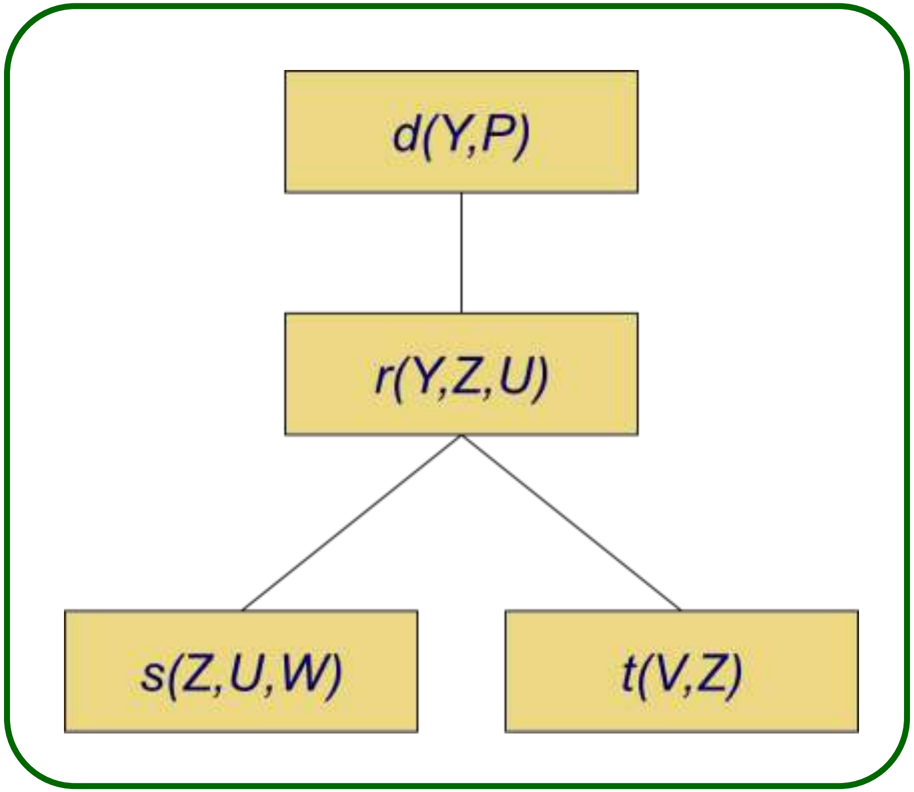
CORE is NP-hard

- Deciding whether Q’ is the core of Q is NP-hard
- For instance, let 3COL be the class of all 3-colourable graphs containing a triangle
- Clearly, deciding whether G$\in$3COL is NP-hard
- It is easy to see that G$\in$3COL $\Leftrightarrow$ K_3 is the core of G

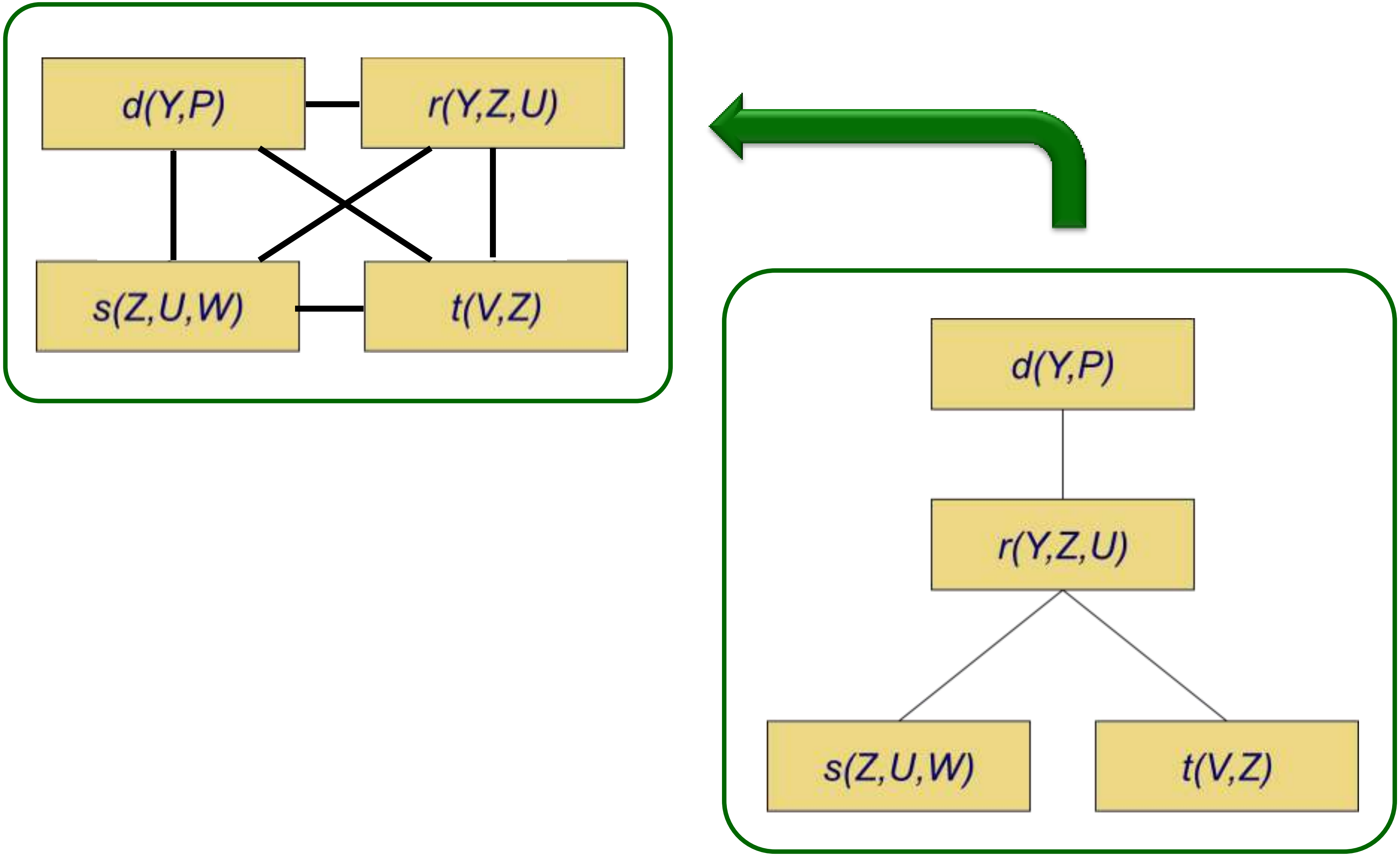
Example:



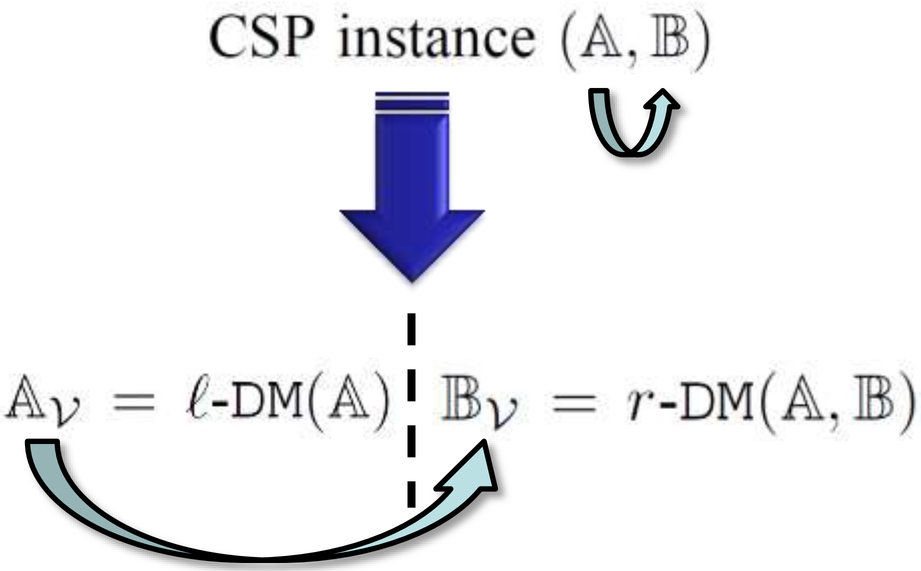
Enforcing Local Consistency (Acyclic)



Enforcing Local Consistency (Acyclic)



Enforcing Local Consistency (Decomposition)

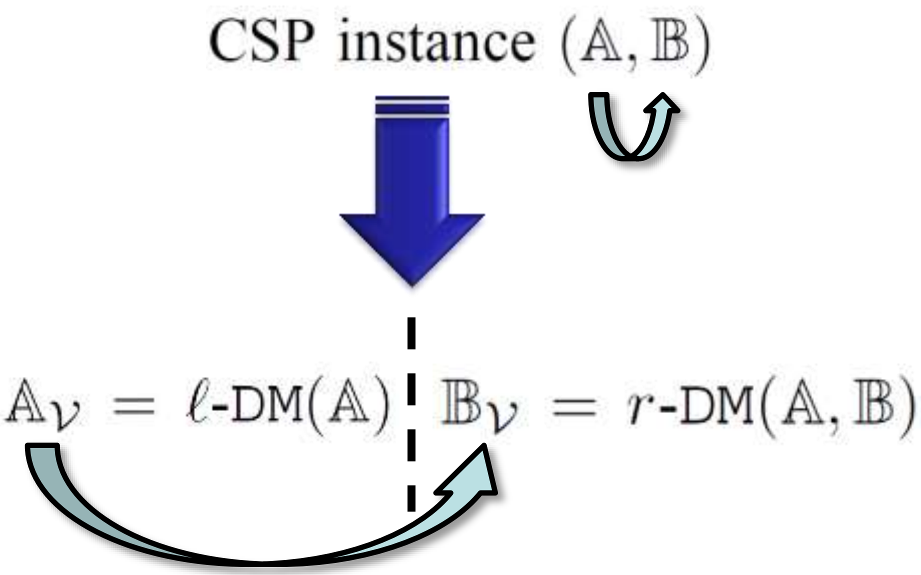


CSP instance $(\mathbb{A}, \mathbb{B})$

$\mathbb{A}_{\mathcal{V}} = \ell\text{-DM}(\mathbb{A})$

$\mathbb{B}_{\mathcal{V}} = r\text{-DM}(\mathbb{A}, \mathbb{B})$

Enforcing Local Consistency



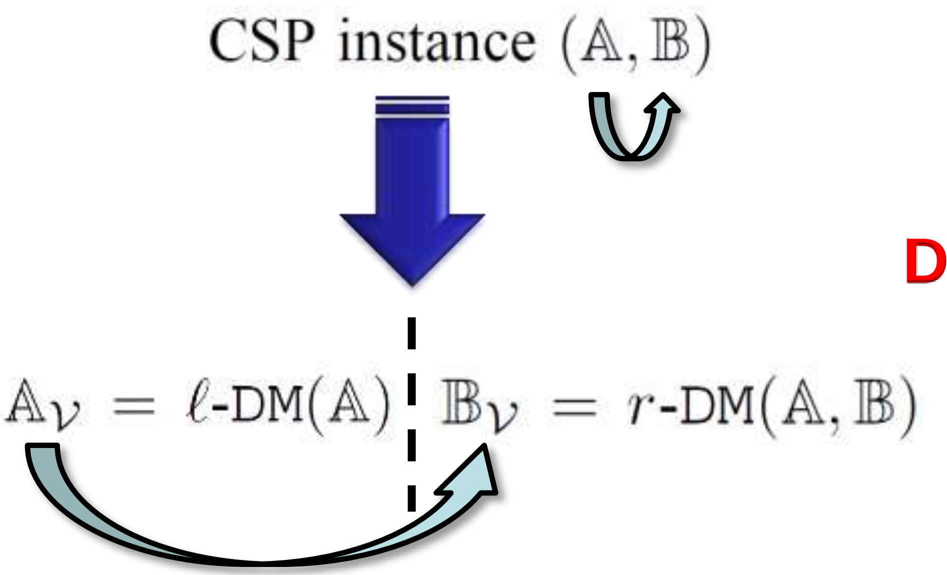
CSP instance $(\mathbb{A}, \mathbb{B})$

$\mathbb{A}_{\mathcal{V}} = \ell\text{-DM}(\mathbb{A})$ $\mathbb{B}_{\mathcal{V}} = r\text{-DM}(\mathbb{A}, \mathbb{B})$

If there is a tree projection, then
enforcing local consistency over the views solves the decision problem

[Sagiv & Smueli, ‘93]

Enforcing Local Consistency



CSP instance $(\mathbb{A}, \mathbb{B})$

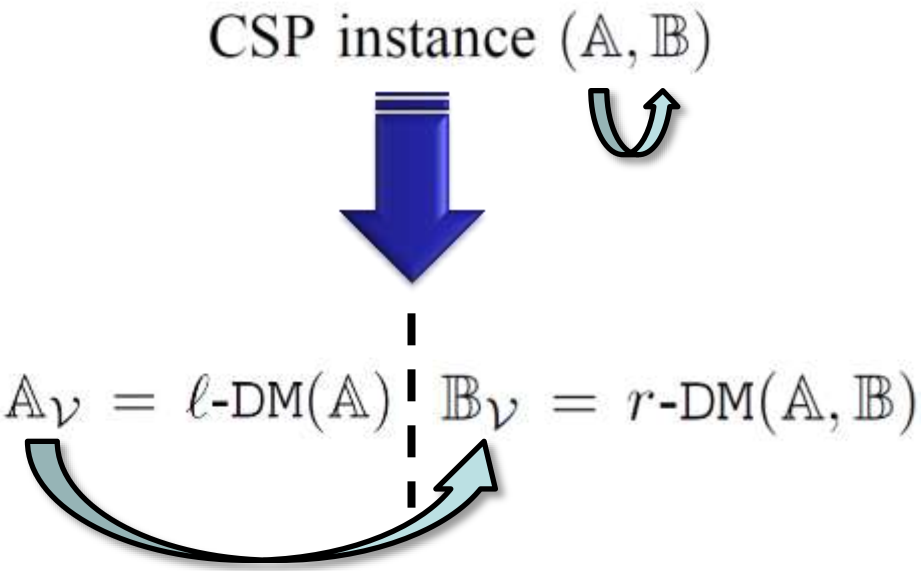
$\mathbb{A}_{\mathcal{V}} = \ell\text{-DM}(\mathbb{A})$ $\mathbb{B}_{\mathcal{V}} = r\text{-DM}(\mathbb{A}, \mathbb{B})$

Does not need to be computed

If there is a tree projection, then
enforcing local consistency over the views solves the decision problem

[Sagiv & Smueli, ‘93]

Even Better



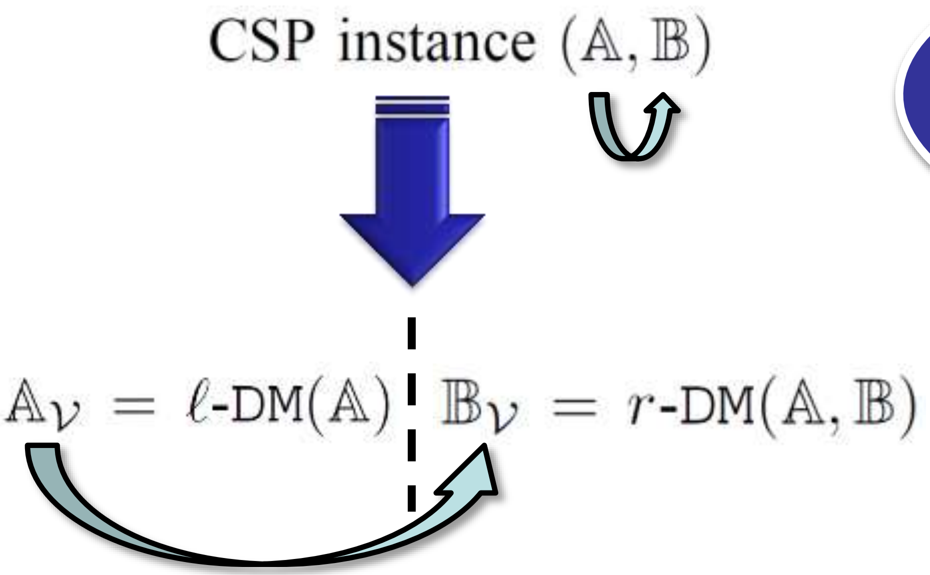
CSP instance $(\mathbb{A}, \mathbb{B})$

$\mathbb{A}_{\mathcal{V}} = \ell\text{-DM}(\mathbb{A})$ $\mathbb{B}_{\mathcal{V}} = r\text{-DM}(\mathbb{A}, \mathbb{B})$

There is a polynomial-time algorithm that:

- either returns that there is no tree projection,
- or solves the decision problem

Even Better



CSP instance $(\mathbb{A}, \mathbb{B})$

$\mathbb{A}_{\mathcal{V}} = \ell\text{-DM}(\mathbb{A})$ $\mathbb{B}_{\mathcal{V}} = r\text{-DM}(\mathbb{A}, \mathbb{B})$

just check the given solution

There is a polynomial-time algorithm that:

- either returns that there is no tree projection,
- or solves the decision problem

The Precise Power of Local Consistency

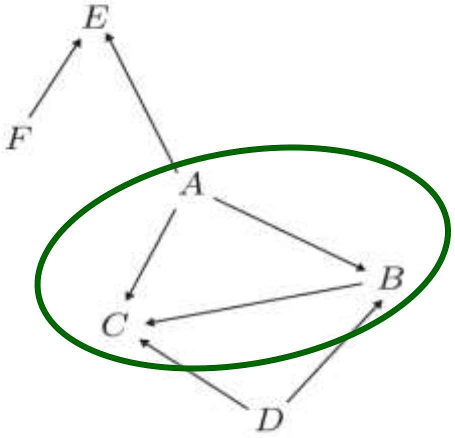
- The followings are equivalent:
  - Local consistency solves the decision problem
  - There is *a core* of the query having a tree projection

[Greco & Scarcello, PODS‘10]

The Precise Power of Local Consistency

- The followings are equivalent
  - Local consistency solves the decision problem
  - There is *a core* of the query having a tree projection

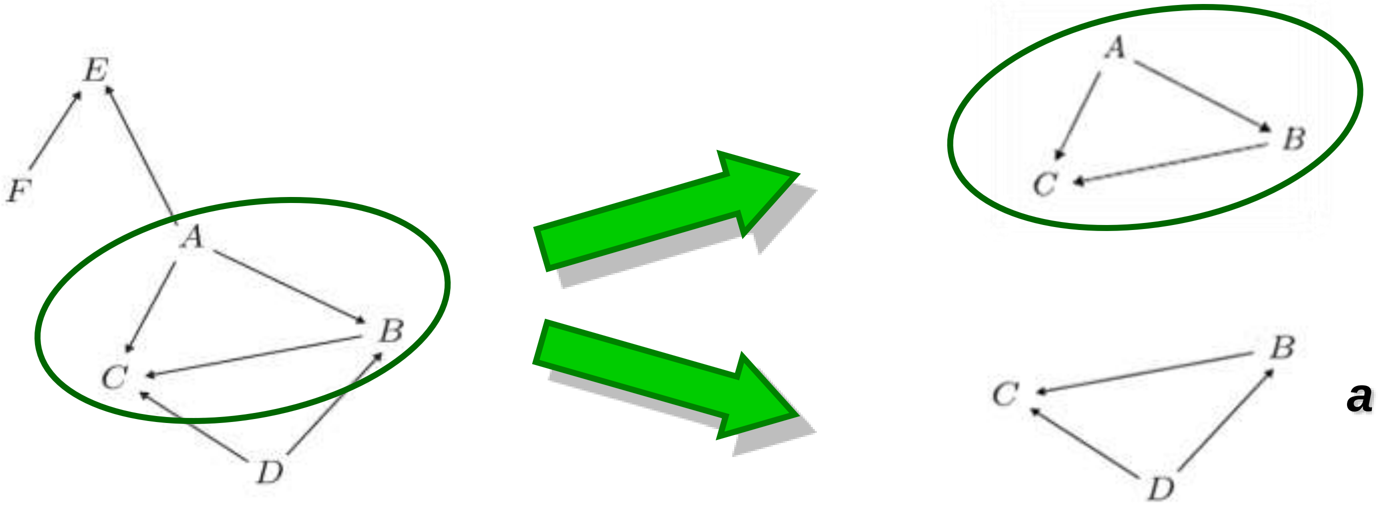
$$Q \; : \; r(A,B) \wedge r(B,C) \wedge r(A,C) \wedge r(D,C) \wedge \\ r(D,B) \wedge r(A,E) \wedge r(F,E),$$



The Precise Power of Local Consistency

- The followings are equivalent
  - Local consistency solves the decision problem
  - There is *a core* of the query having a tree projection

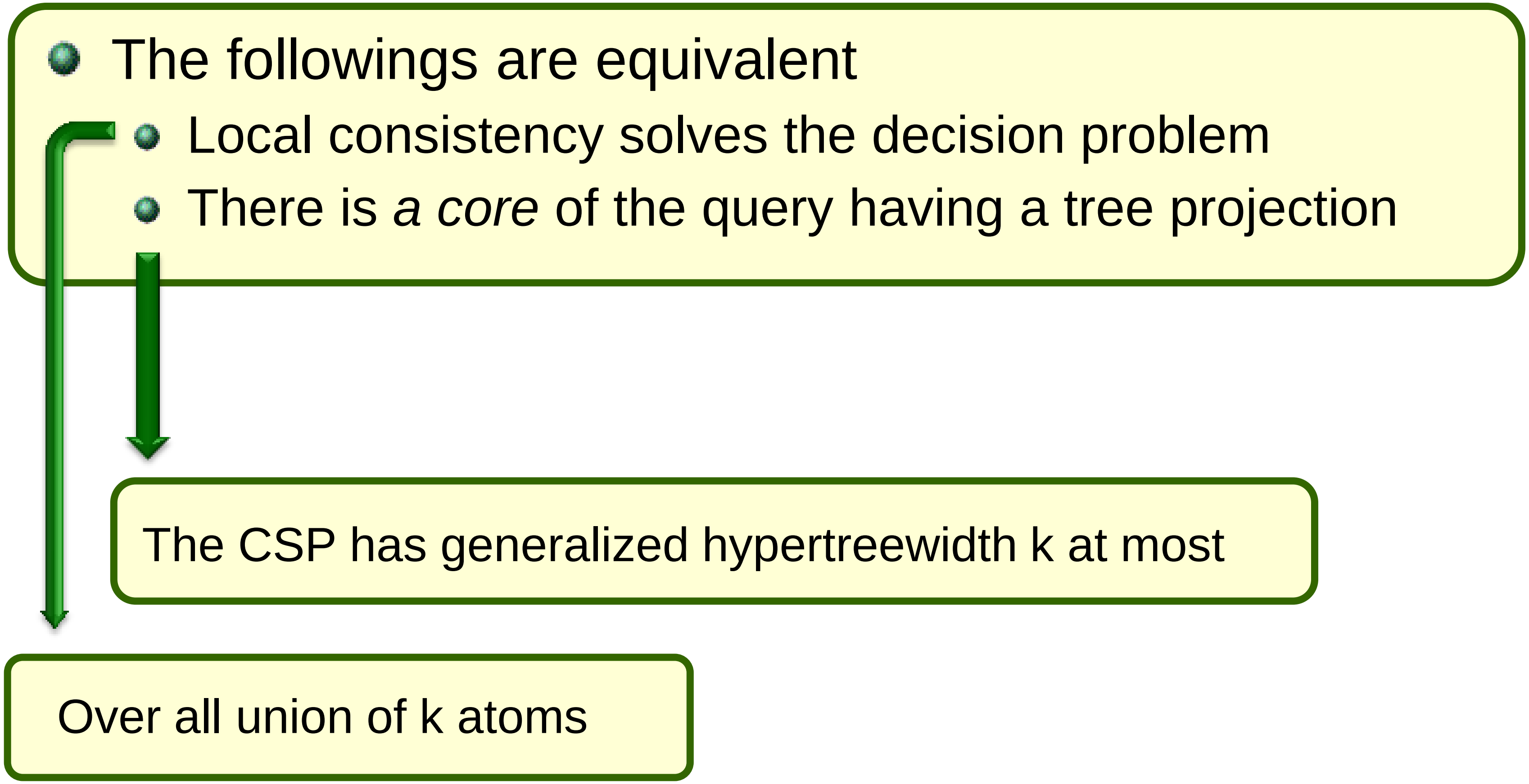
$$Q \ : \ r(A,B) \wedge r(B,C) \wedge r(A,C) \wedge r(D,C) \wedge \\ r(D,B) \wedge r(A,E) \wedge r(F,E),$$



a core *with* *TP*

a core *without* *TP*

A Relevant Specialization (not immediate)



- The followings are equivalent
  - Local consistency solves the decision problem
  - There is *a core* of the query having a tree projection

The CSP has generalized hypertreewidth k at most

Over all union of k atoms

[Greco & Scarcello, CP‘11]

Back on the Result

- The followings are equivalent
  - Local consistency solves the decision problem
  - There is *a core* of the query having a tree projection

«Promise» tractability

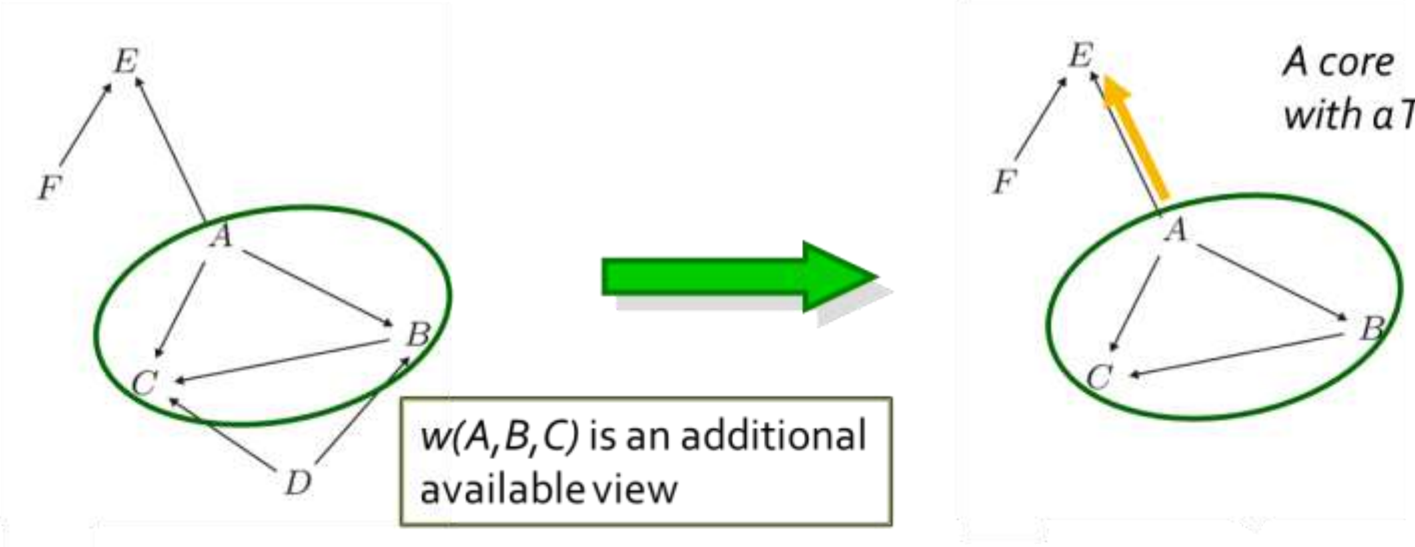
- There is no polynomial time algorithm that
  - either solves the decision problem
  - or disproves the promise

Local consistency for computing solutions

- The followings are equivalent
  - Local consistency entails «views containing variables O are correct»
  - The set of variables O is tp-covered in a tree projection

$$Q : r(A,B) \wedge r(B,C) \wedge r(A,C) \wedge r(D,C) \wedge r(D,B) \wedge r(A,E) \wedge r(F,E), \wedge \text{atoms}(\{A,E\})$$

{A,E} is tp-covered



w(A,B,C) is an additional available view

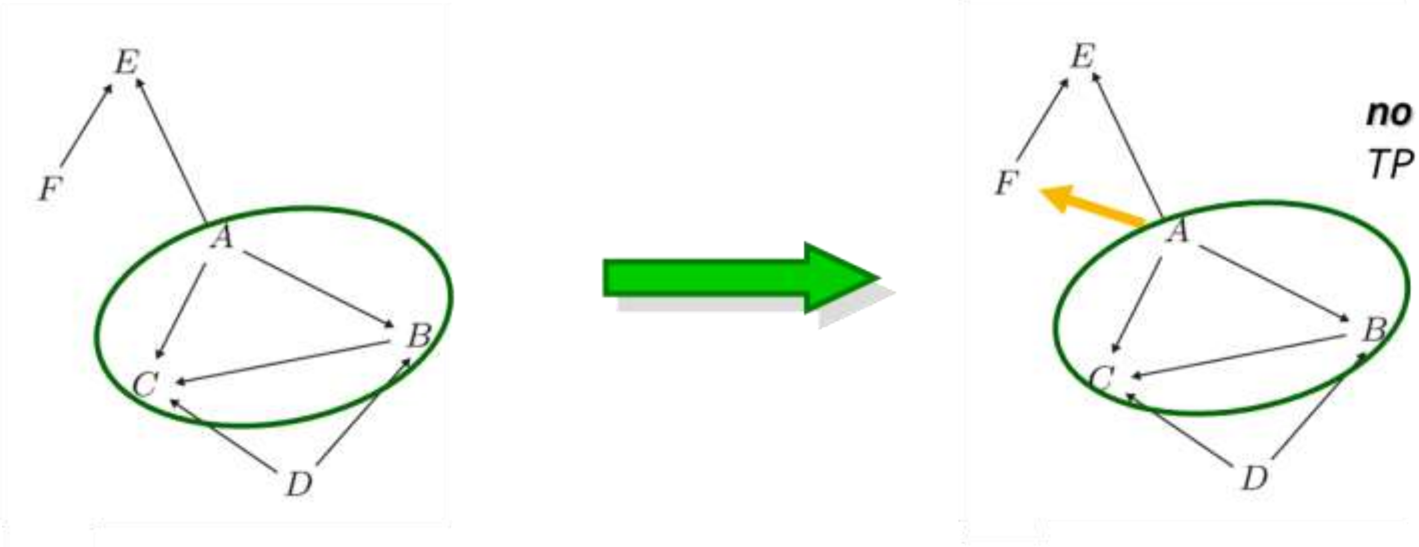
A core with a TP

Local consistency for computing solutions

- The followings are equivalent
  - Local consistency entails «views containing variables O are correct»
  - The set of variables O is tp-covered in a tree projection

$Q \; : \; r(A,B) \wedge r(B,C) \wedge r(A,C) \wedge r(D,C) \wedge$
$r(D,B) \wedge r(A,E) \wedge r(F,E),$ $\wedge$ atoms({A,F})

{A,F} is not tp-covered

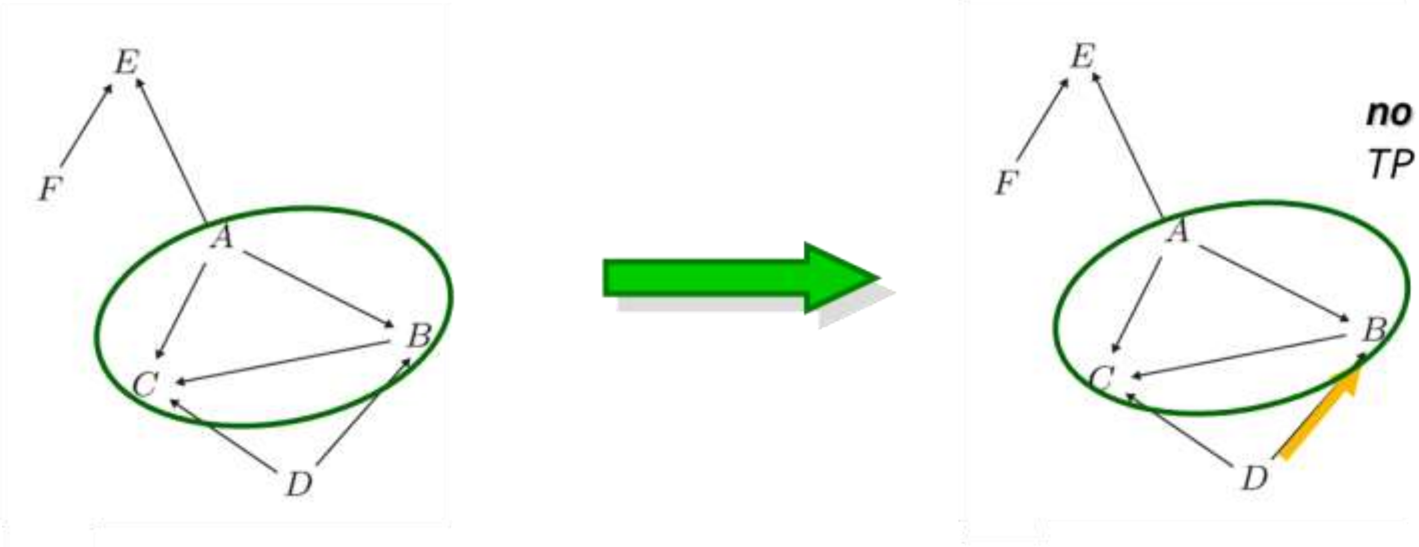


Local and global consistency

- The followings are equivalent
  - Local consistency entails global consistency
  - Every query atom/constraint is tp-covered in a tree projection

$$Q \ : \ r(A,B) \wedge r(B,C) \wedge r(A,C) \wedge r(D,C) \wedge r(D,B) \wedge r(A,E) \wedge r(F,E), \wedge \text{atoms}(\{D,B\})$$

{*D*,*B*} is not tp-covered



Thank you!