Structural decomposition methods and applications to game theory

Gianluigi Greco

Outline of PART I

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

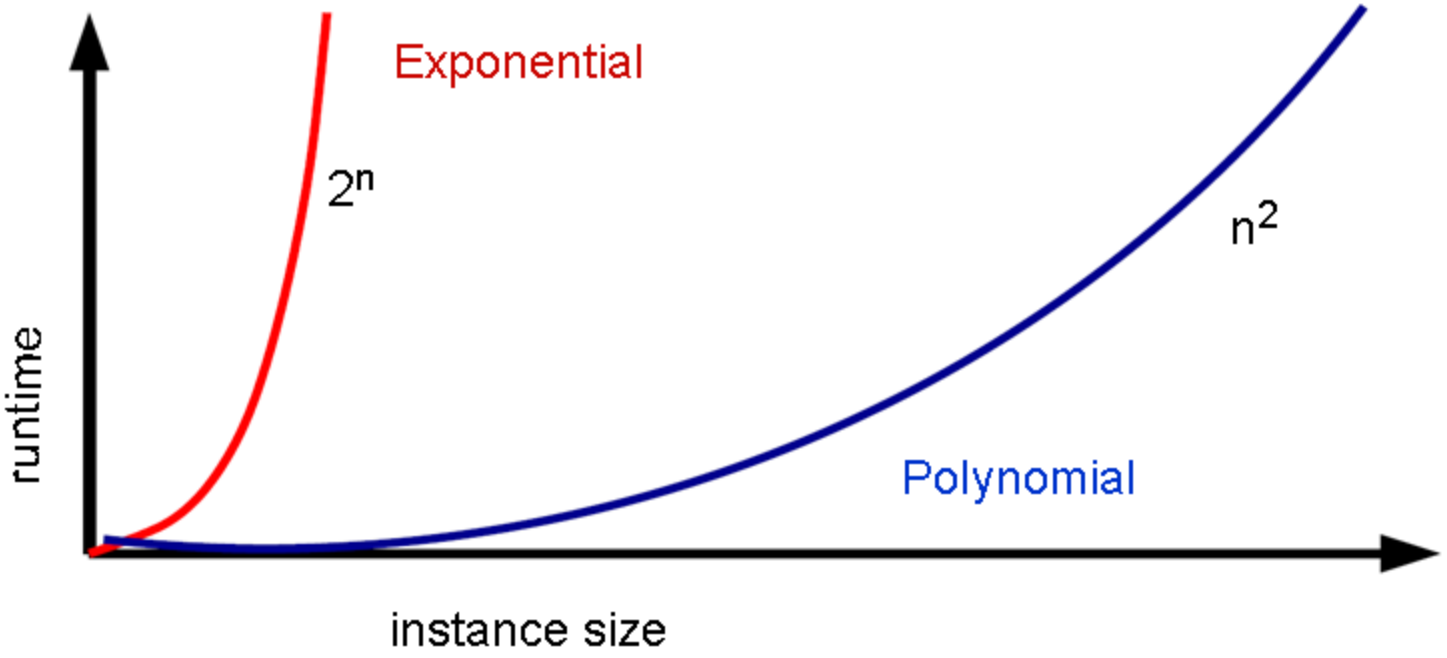
Inherent Problem Complexity

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

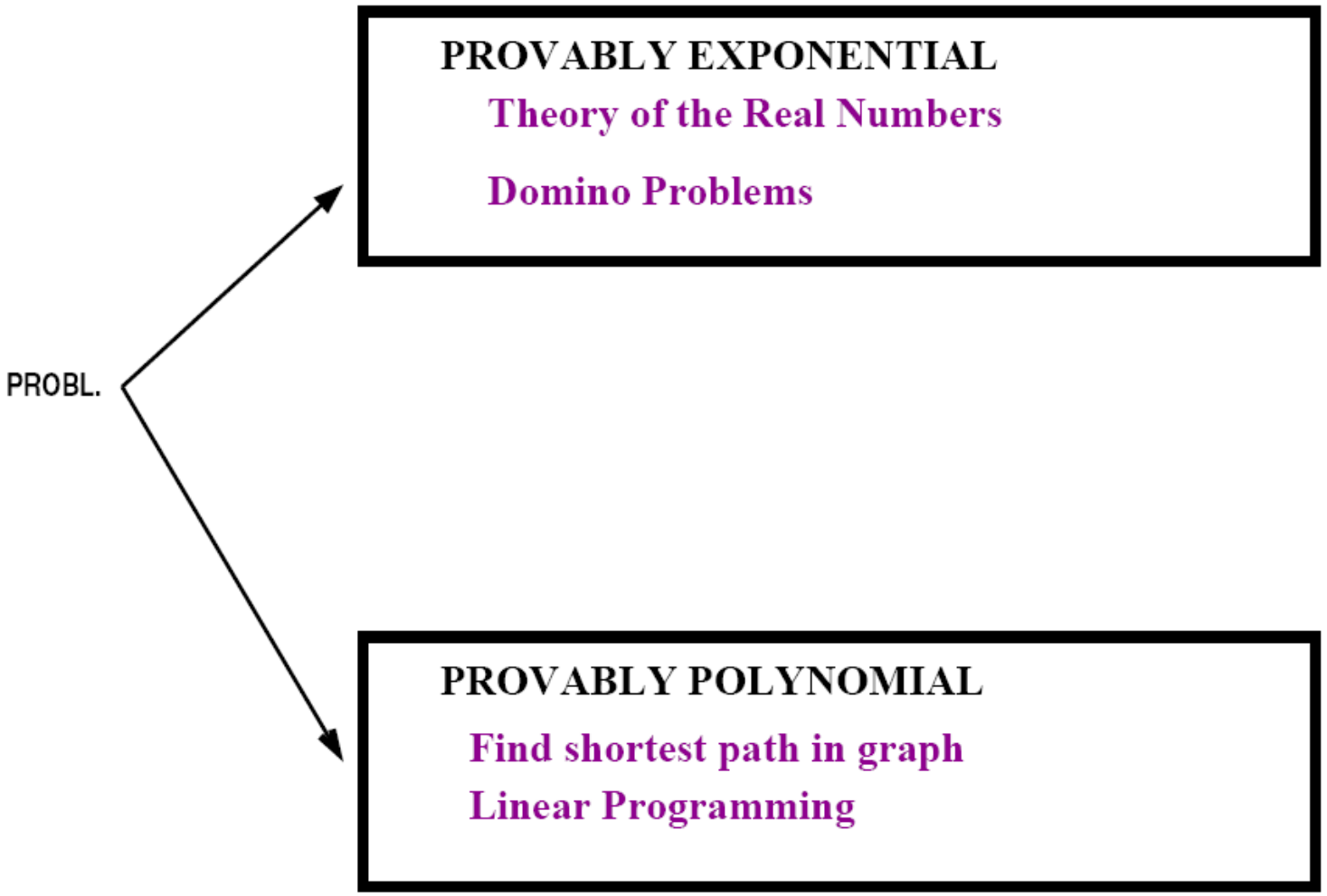
Inherent Problem Complexity

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

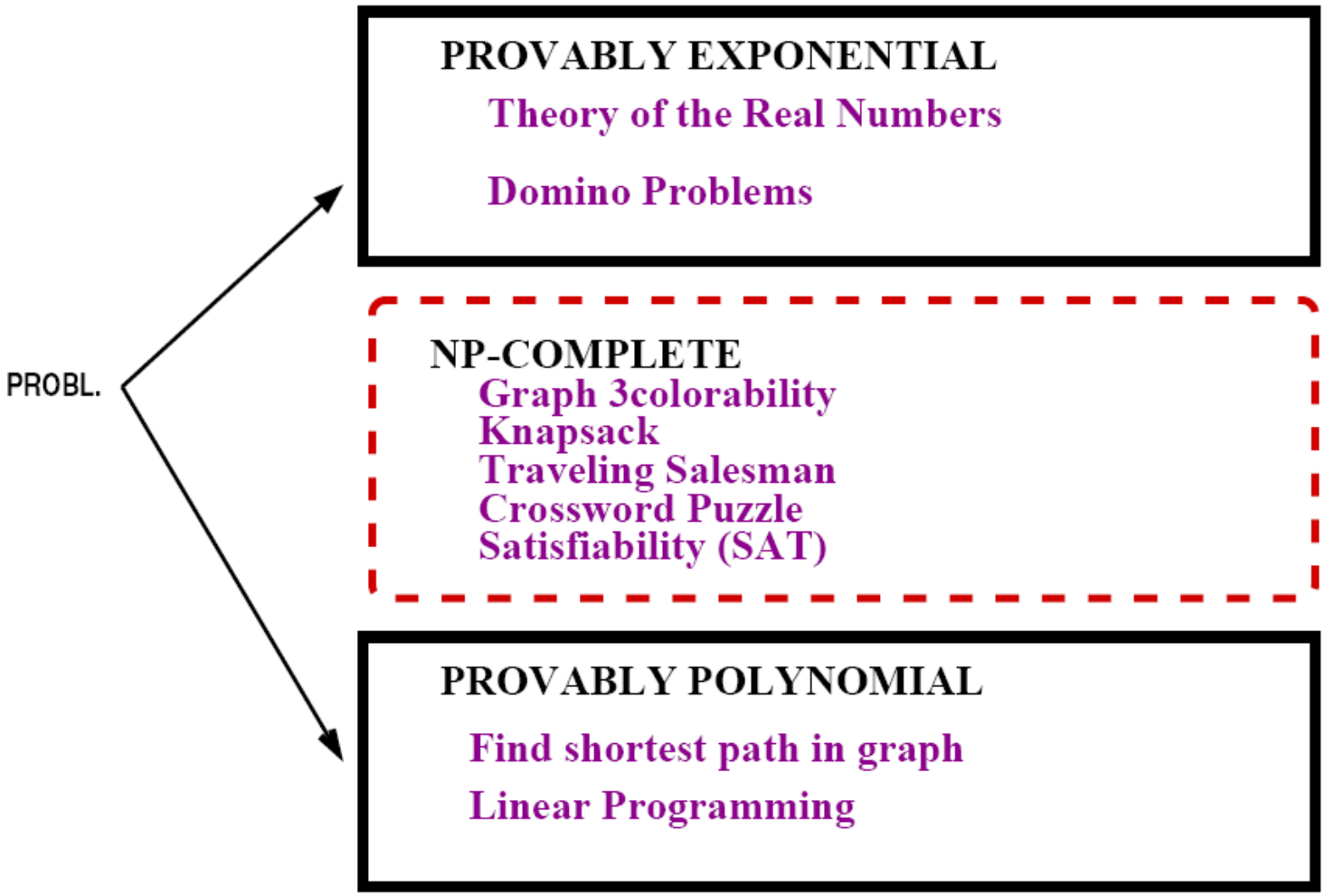
Number of steps it takes for input of size n



Time Complexity



Time Complexity

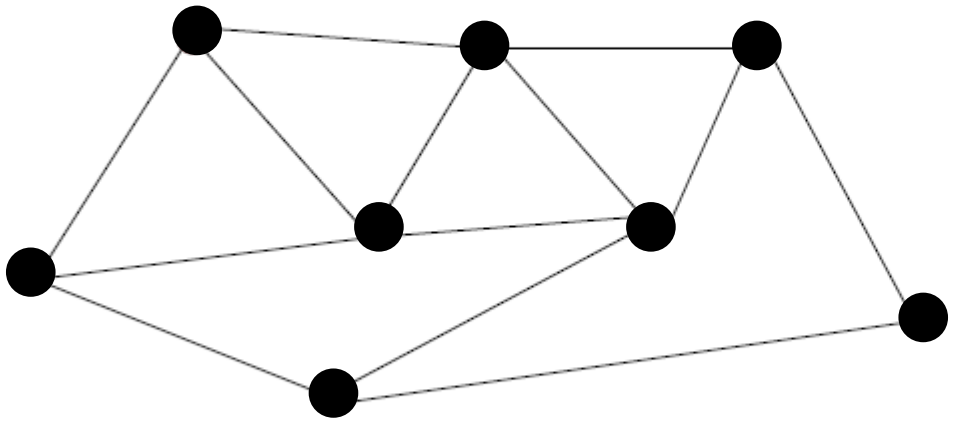


Graph Three-colorability

Instance: A graph G.

Question: Is G 3-colorable?

Examples of instances:

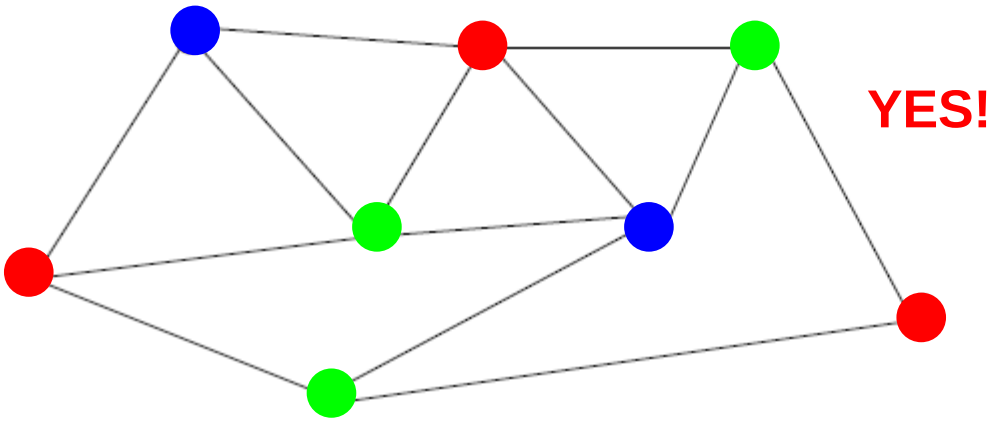


Graph Three-colorability

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,…

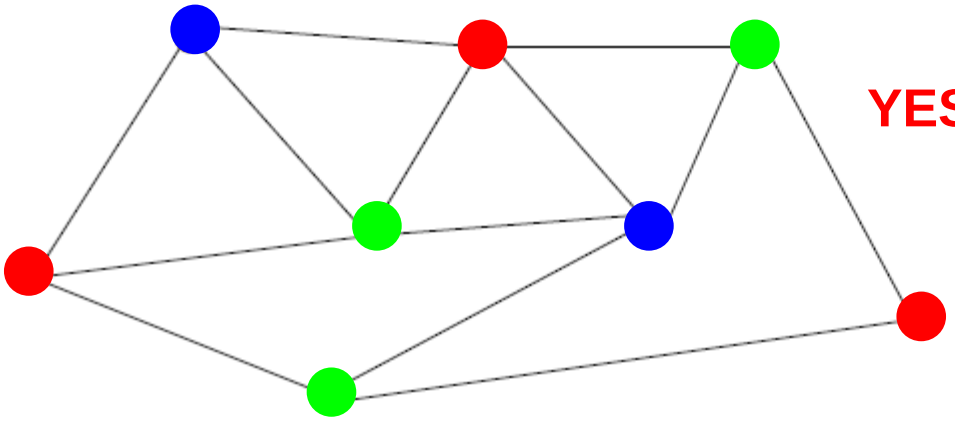
  Exploit bounded degree of cyclicity.

Graph Three-colorability

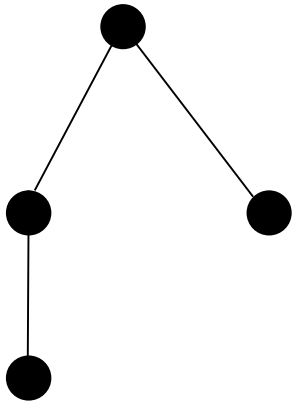
Instance: A graph G.

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 ($\rightarrow$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*.

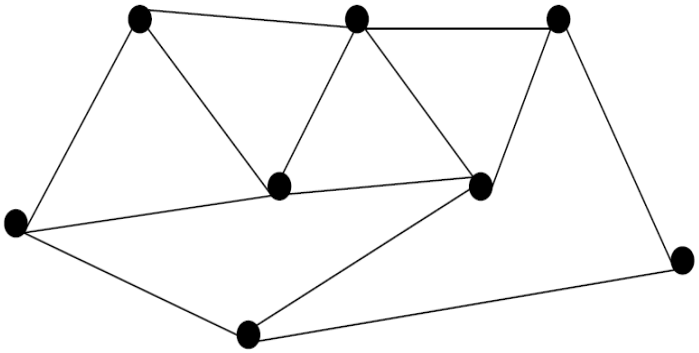
  Problems restricted to *acyclic* graphs are often trivially solvable ($\rightarrow$3COL).

- 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?

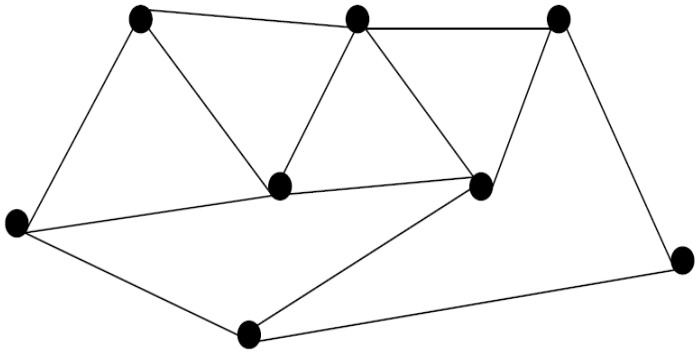
- Suggest a measure of distance from an acyclic graph



Three Early Approaches

Feedback vertex set

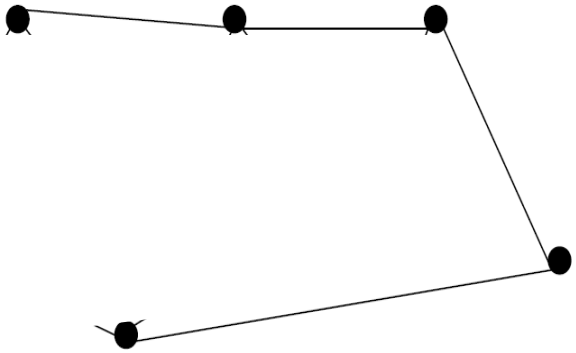
Set of vertices whose deletion makes the graph acyclic



The feedback vertex number

Feedback vertex number

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

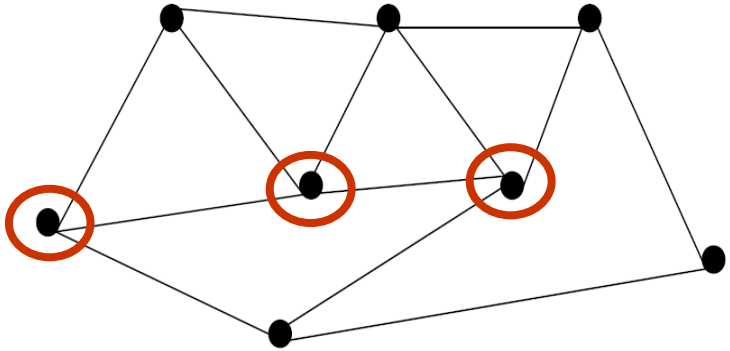


fwn(G)=3

FVN: Properties

Feedback vertex number

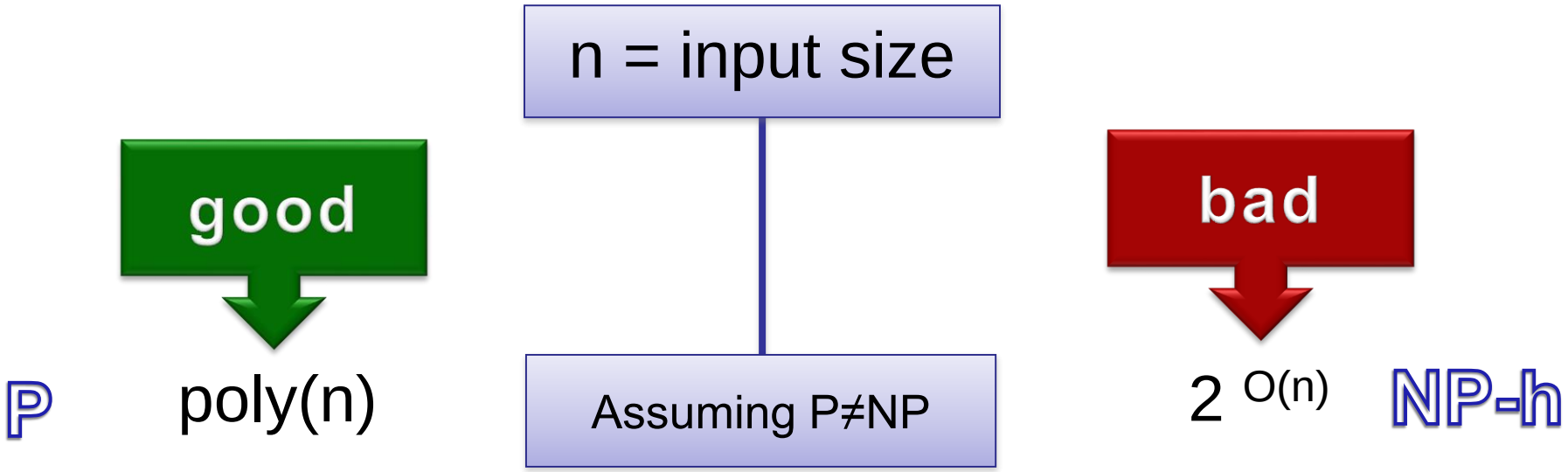
Min. number of vertices I need to eliminate to make the graph acyclic



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

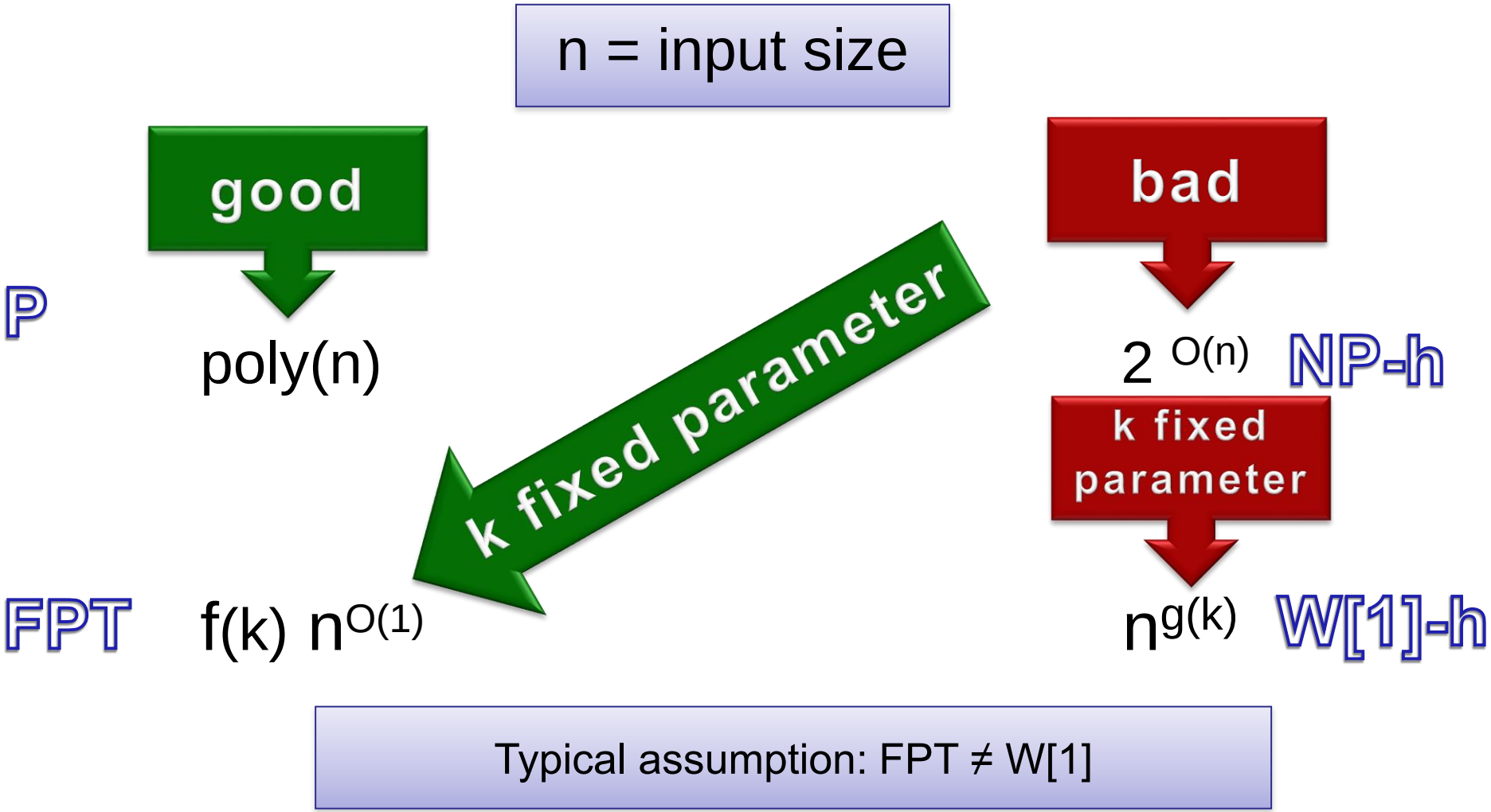


But…

- 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



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

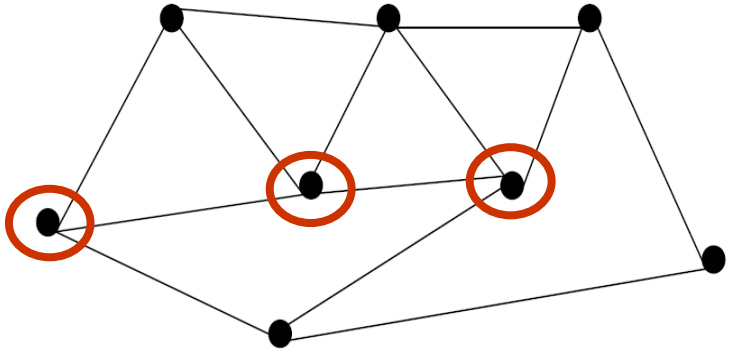
- Does G has a feedback vertex set of k vertices?

- 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

Feedback vertex number

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

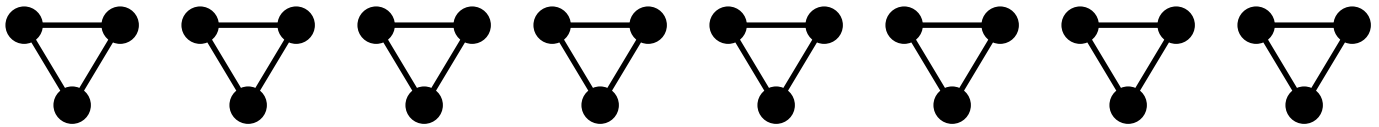


fwn(G)=3

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

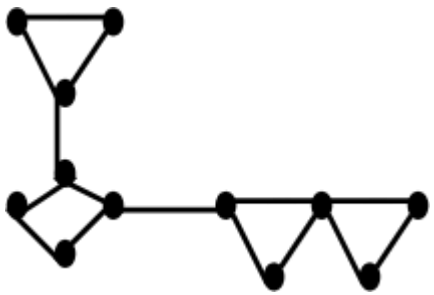
Pro: For fixed k we can check in quadratic time if fwn(G)=k (FPT) .

Con: Very simple graphs can have large FVN:

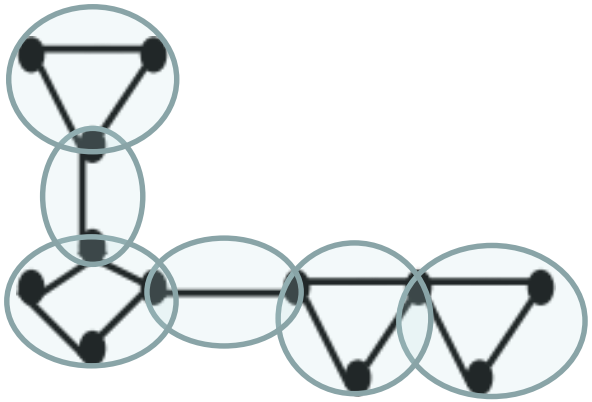


Feedback edge number

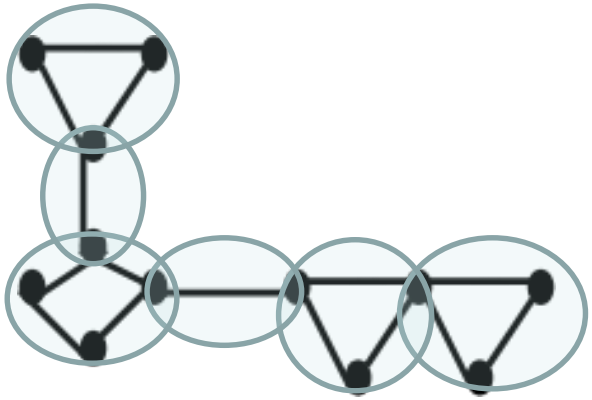
(2) **Feedback edge number → same problem.**



Any idea for further techniques?



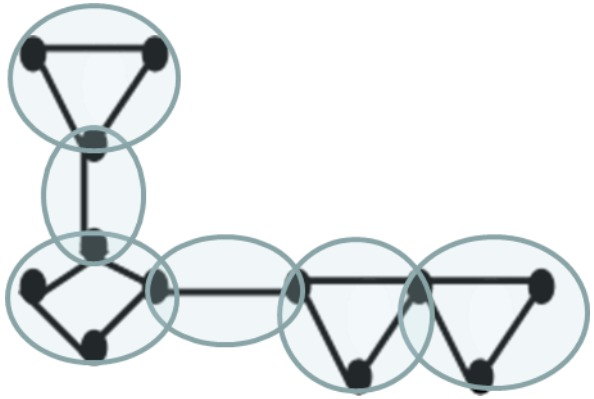
Yes! A tree of clusters (subproblems)



- 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**

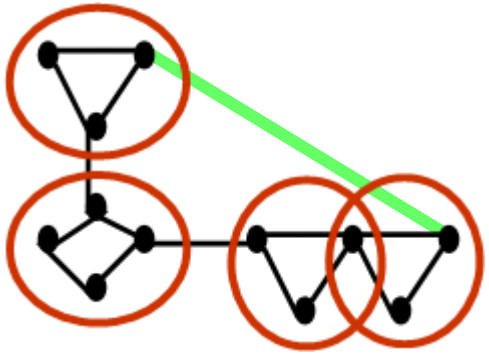


bcw(G)=4

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

Drawbacks of BiComp

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



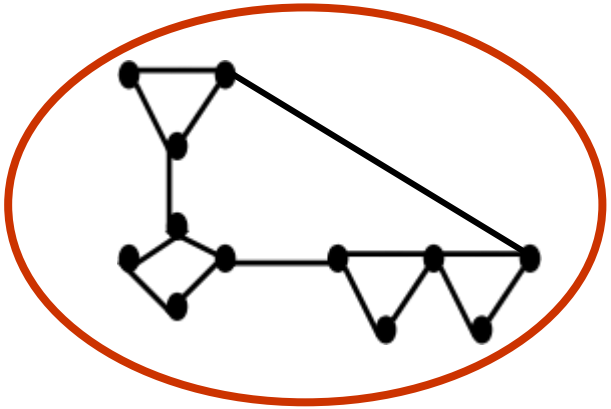
bcw(G)=4

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)?

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?

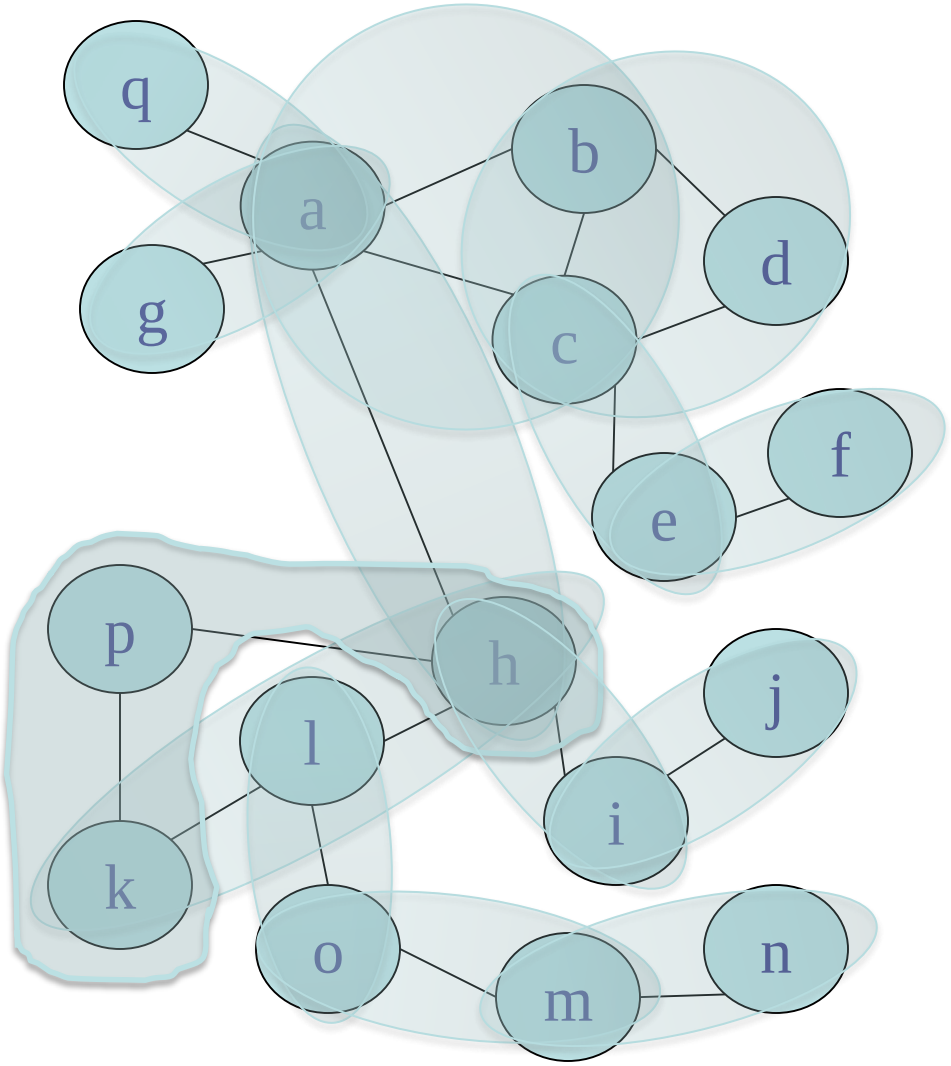
Outline of PART I

Introduction to Decomposition Methods

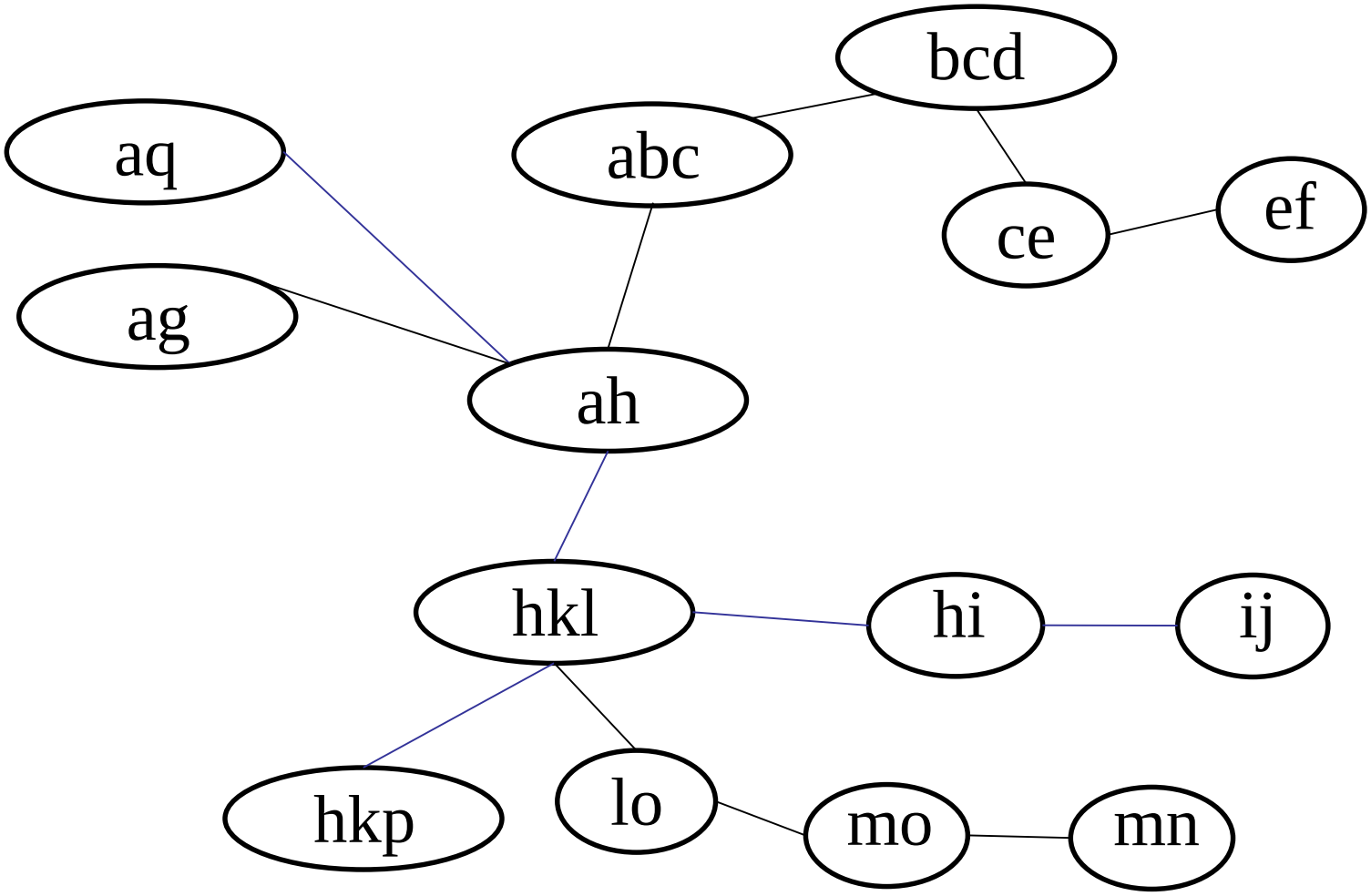
Tree Decompositions

Applications of Tree Decompositions

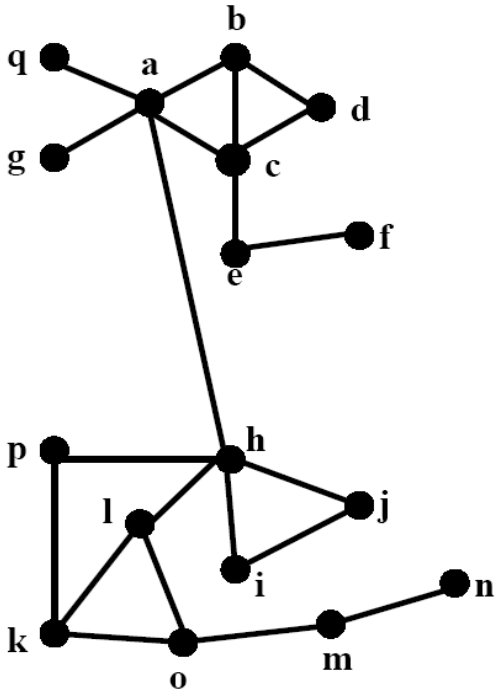
Tree Decompositions [Robertson & Seymour ‘86]



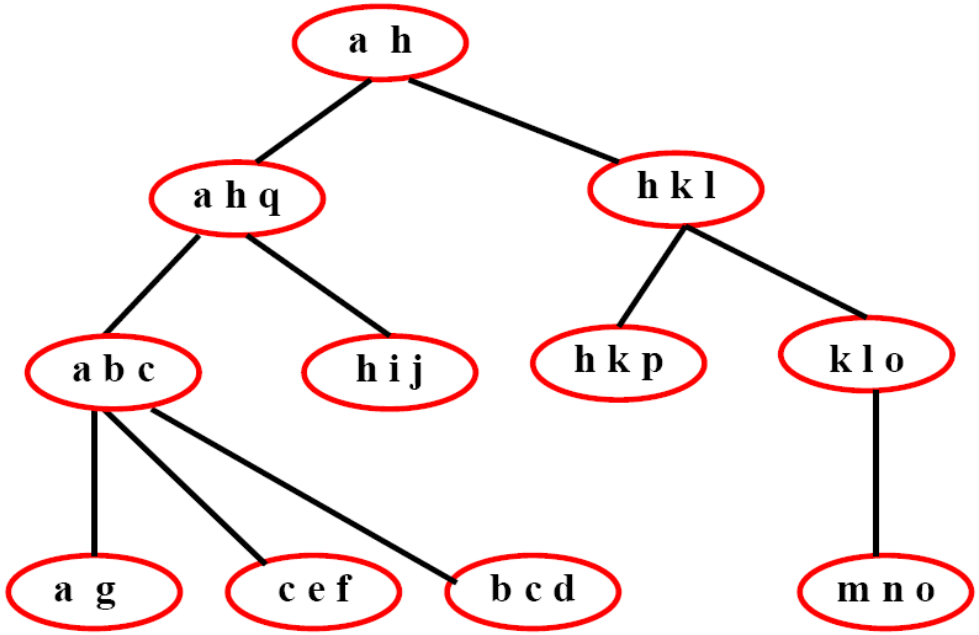
Tree Decompositions [Robertson & Seymour ‘86]



Tree Decompositions [Robertson & Seymour ‘86]

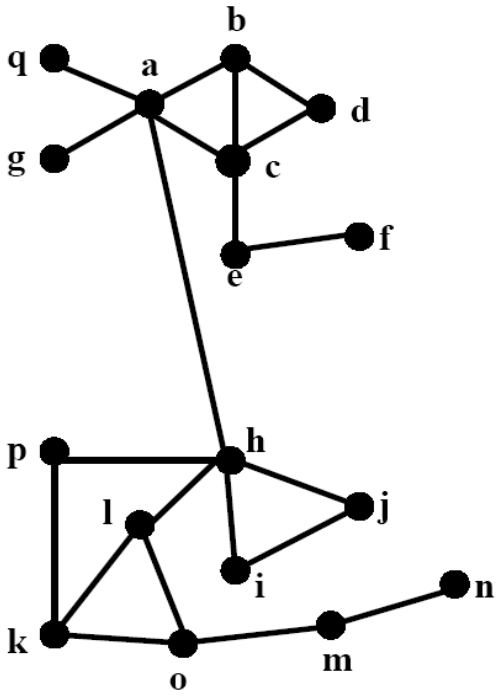


Graph G

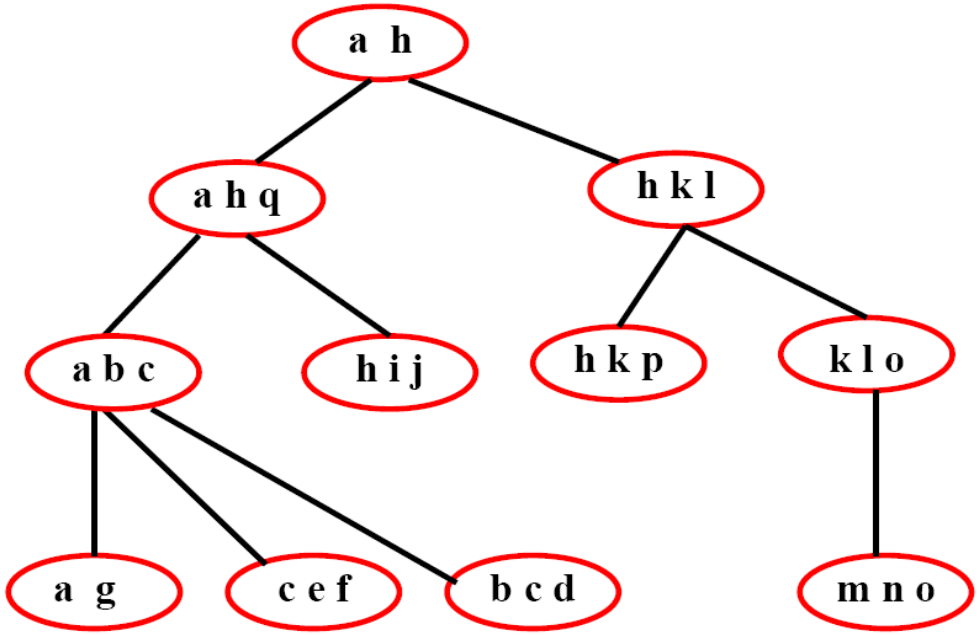


Tree decomposition of width 2 of G

Tree Decompositions [Robertson & Seymour '86]



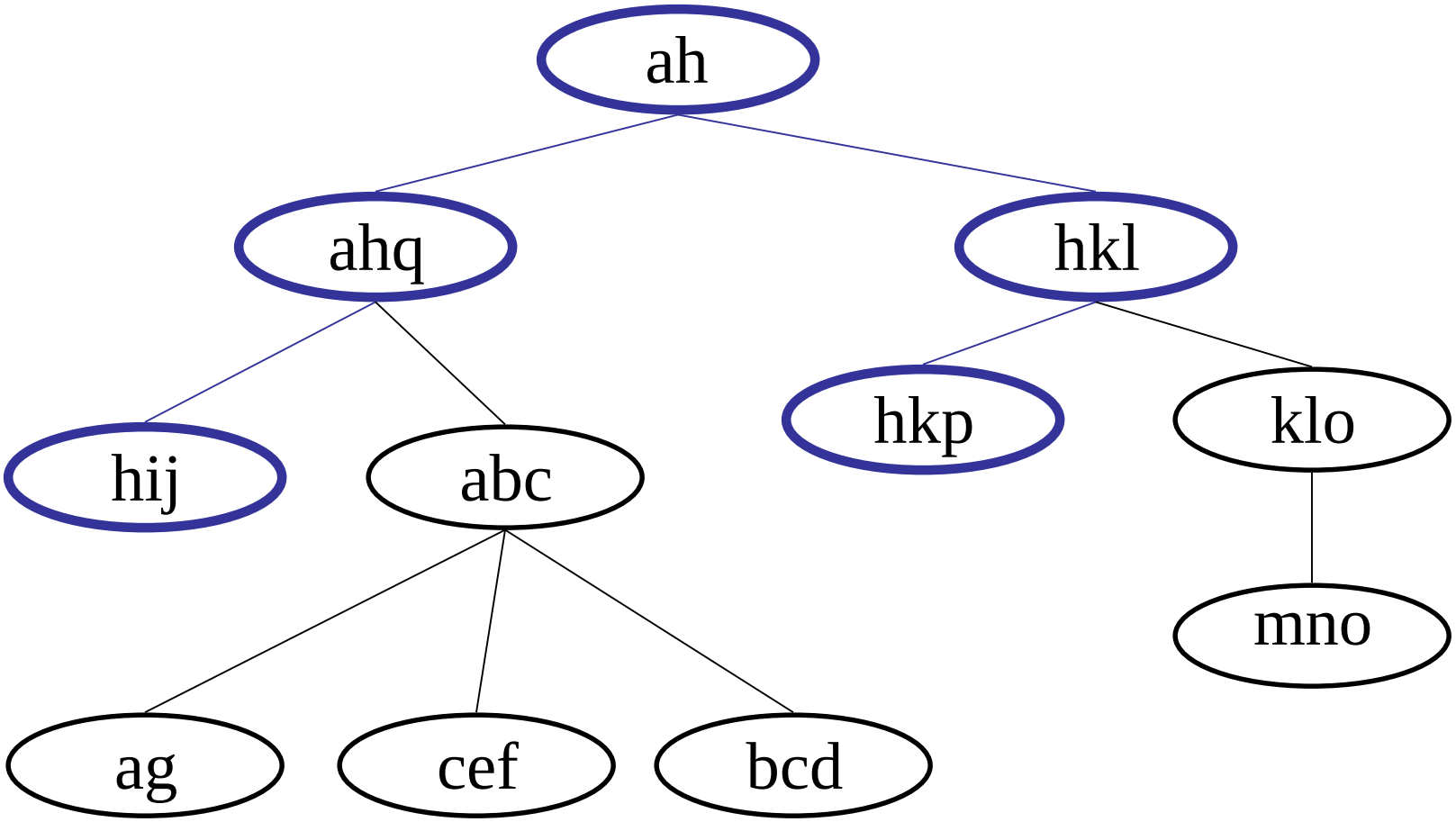
Graph G



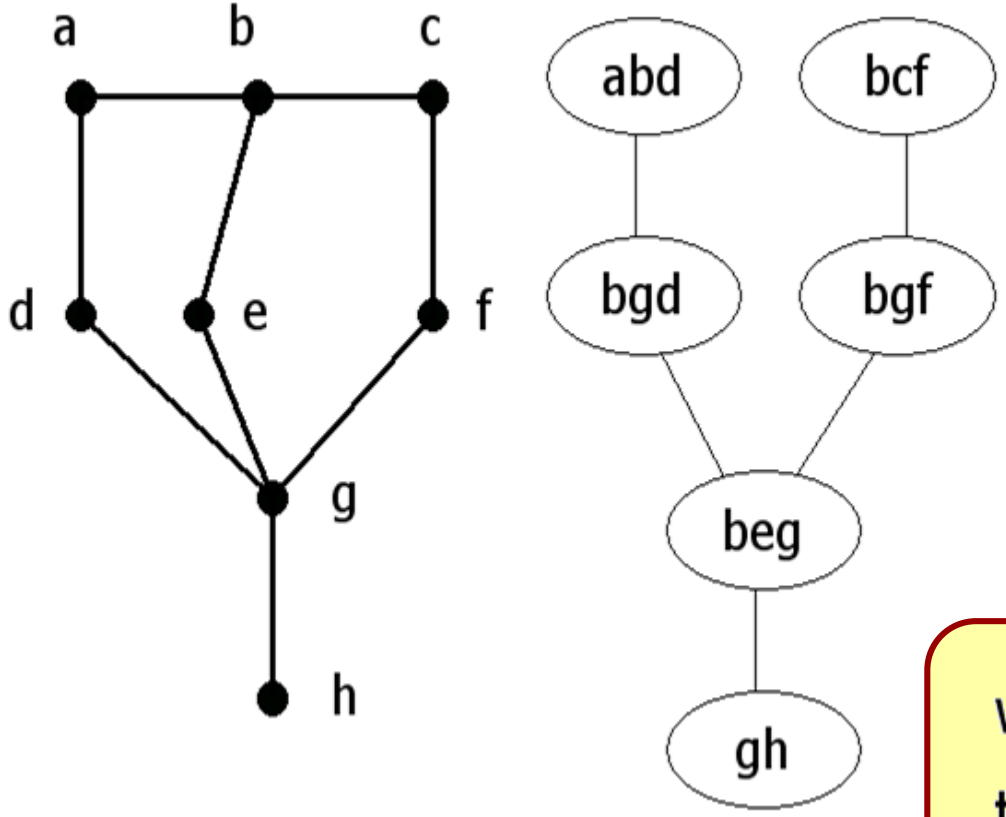
Tree decomposition of width 2 of G

- Every edge realized in some bag
- Connectedness condition

Connectedness condition for *h*



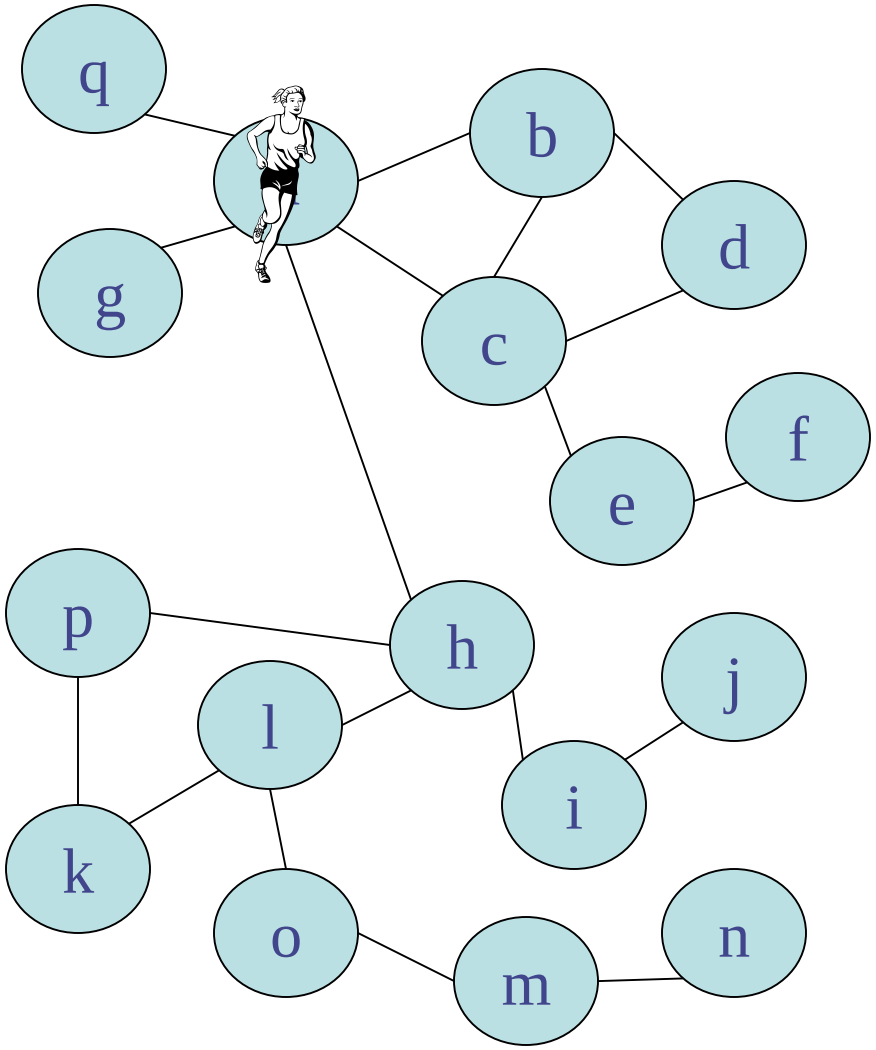
Tree Decompositions and Treewidth



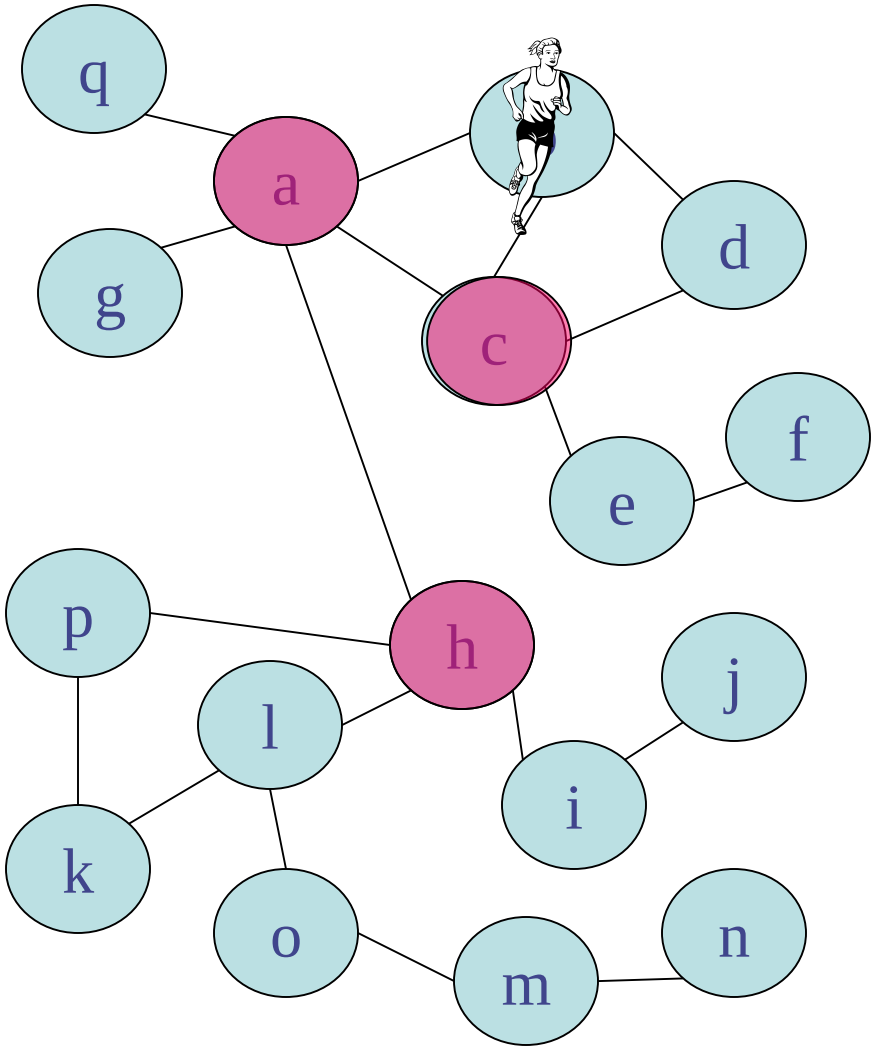
$$\text{width}(T, X_i) = \max |X_i| - 1$$

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

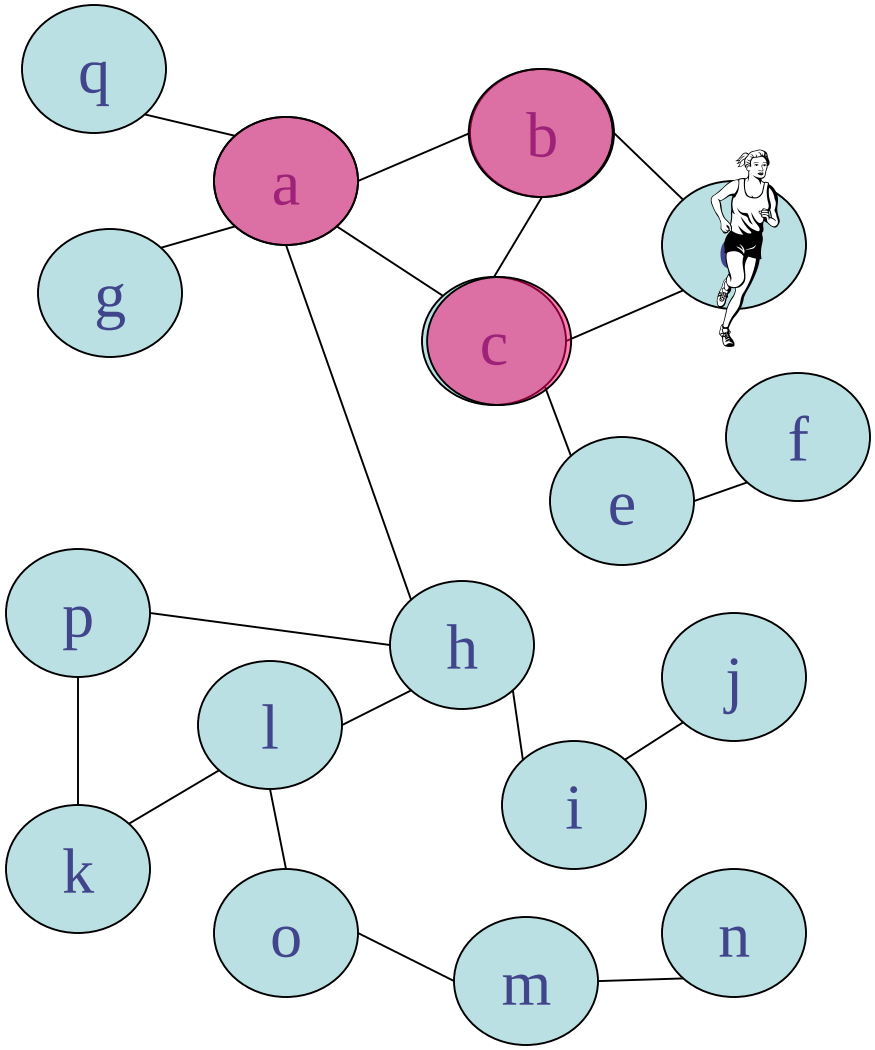
Playing the Robber & Cops Game



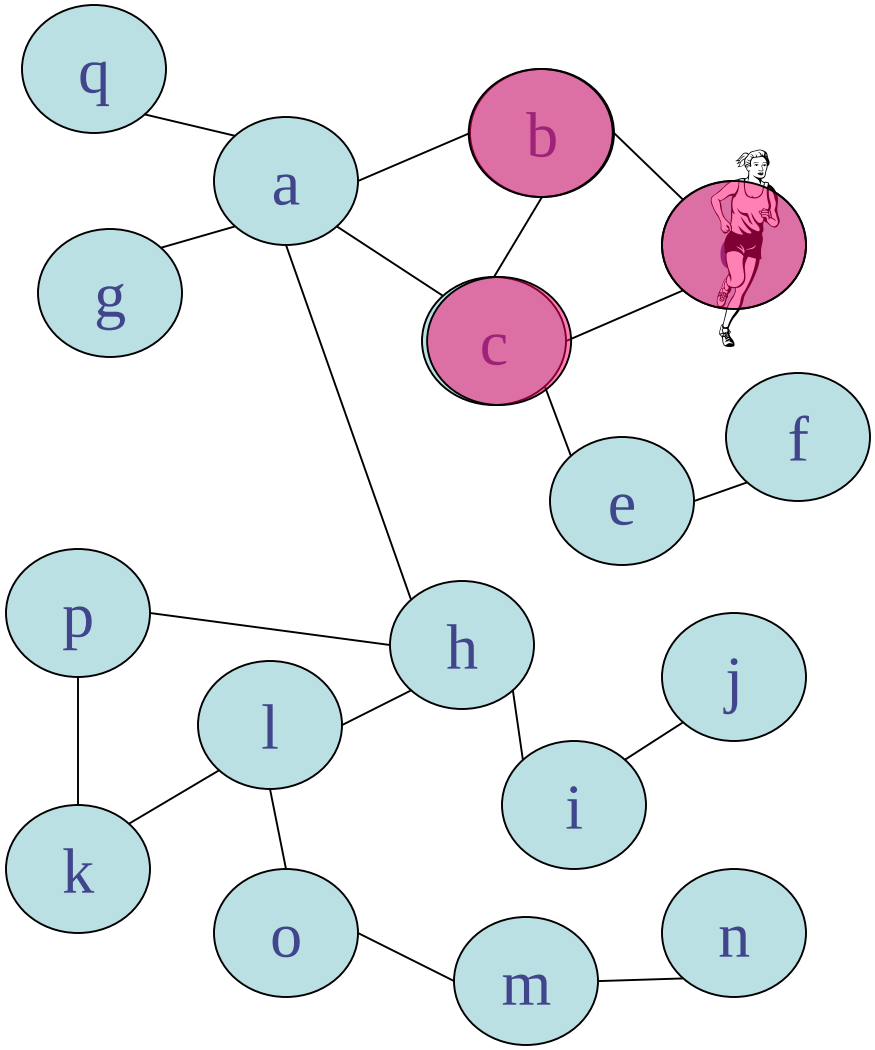
Playing the Robber & Cops Game



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

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

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

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

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

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

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

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.

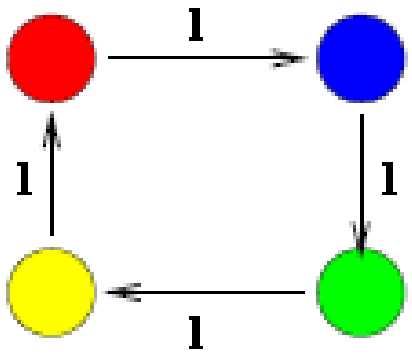
Proof Idea

Small Treewidth (≤c)

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

example

k=4

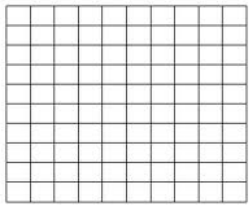


→Courcelle's Theorem

(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).

Outline of PART II

- Beyond Tree Decompositions
- Applications to Databases and CSPs
- Structural and Consistency Properties

Outline of PART II

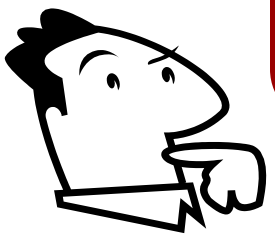
- **Beyond Tree Decompositions**
- Applications to Databases and CSPs
- Structural and Consistency Properties

Beyond Treewidth

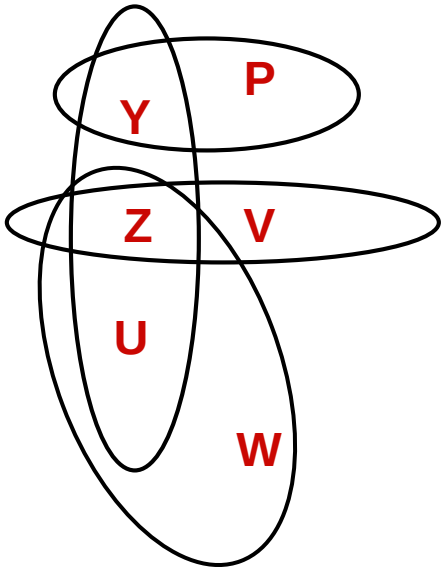
- Treewidth is currently the most successful measure of graph cyclicity. It subsumes most other methods.
- However, there are “simple” graphs that are heavily cyclic. For example, a clique.

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

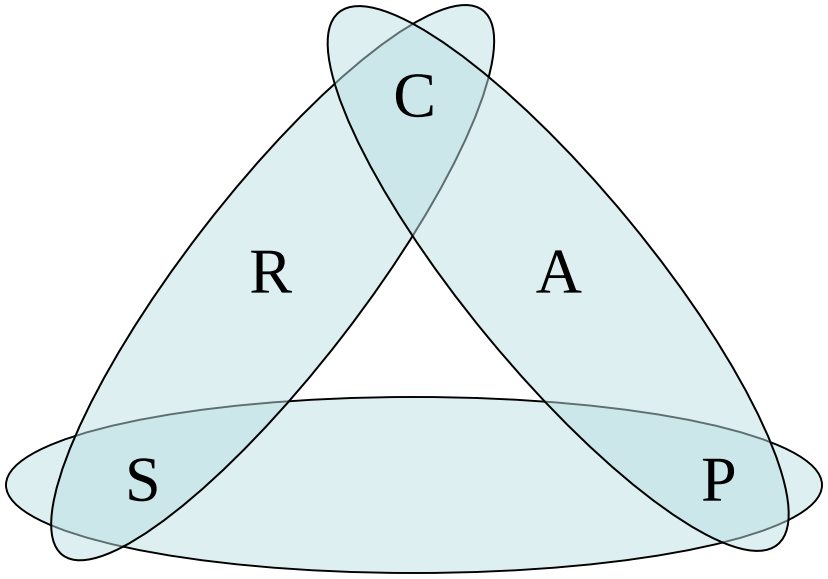
- Database schema (scopes):
  - *Enrolled (Pers#, Course, Reg-Date)*
  - *Teaches (Pers#, Course, Assigned)*
  - *Parent (Pers1, Pers2)*

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

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

Queries and Hypergraphs

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

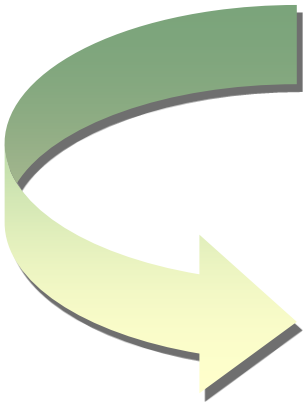


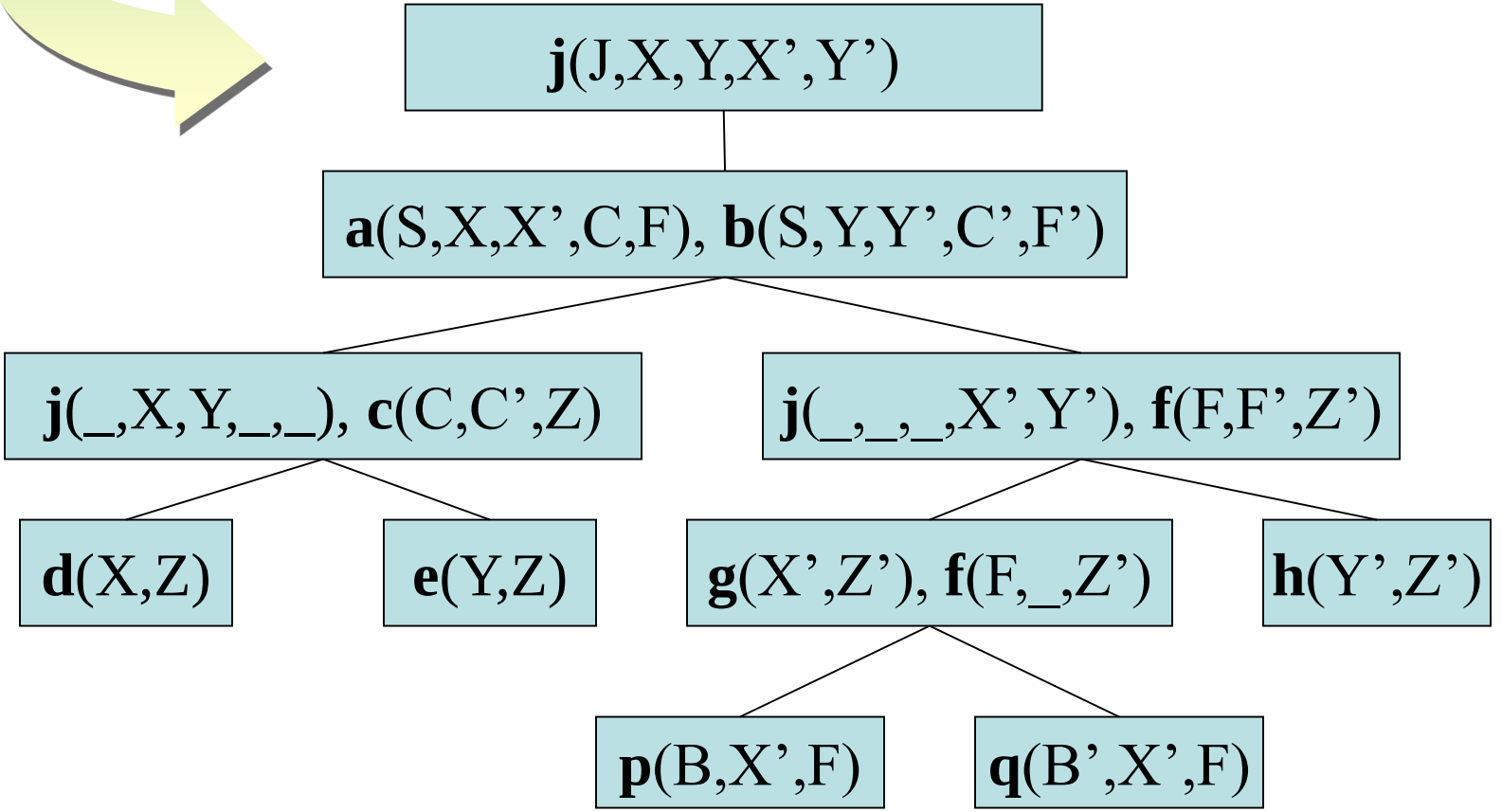
Generalized Hypertree Decompositions

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

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

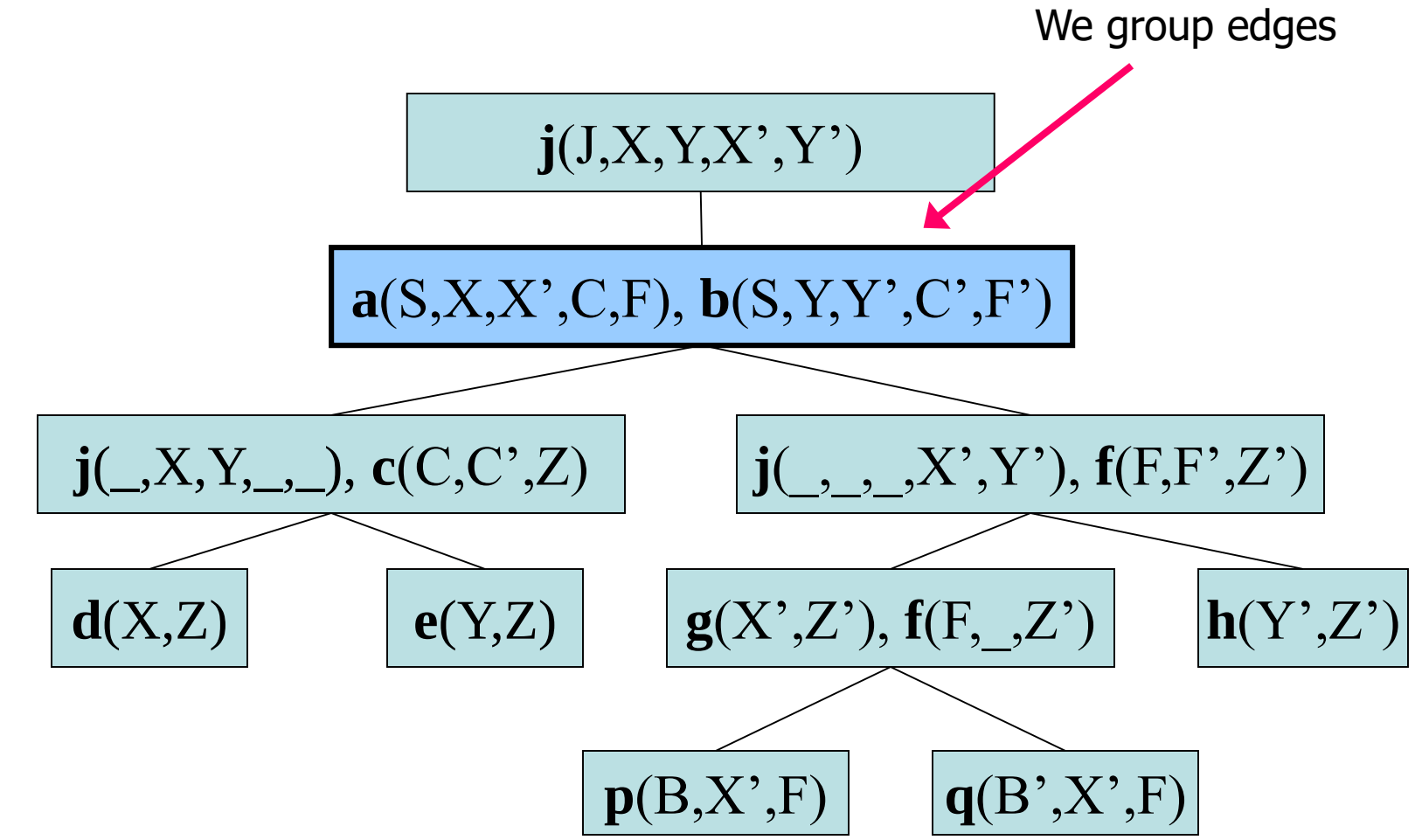
$j(J,X,Y,X',Y')$ $p(B,X',F)$ $q(B',X',F)$



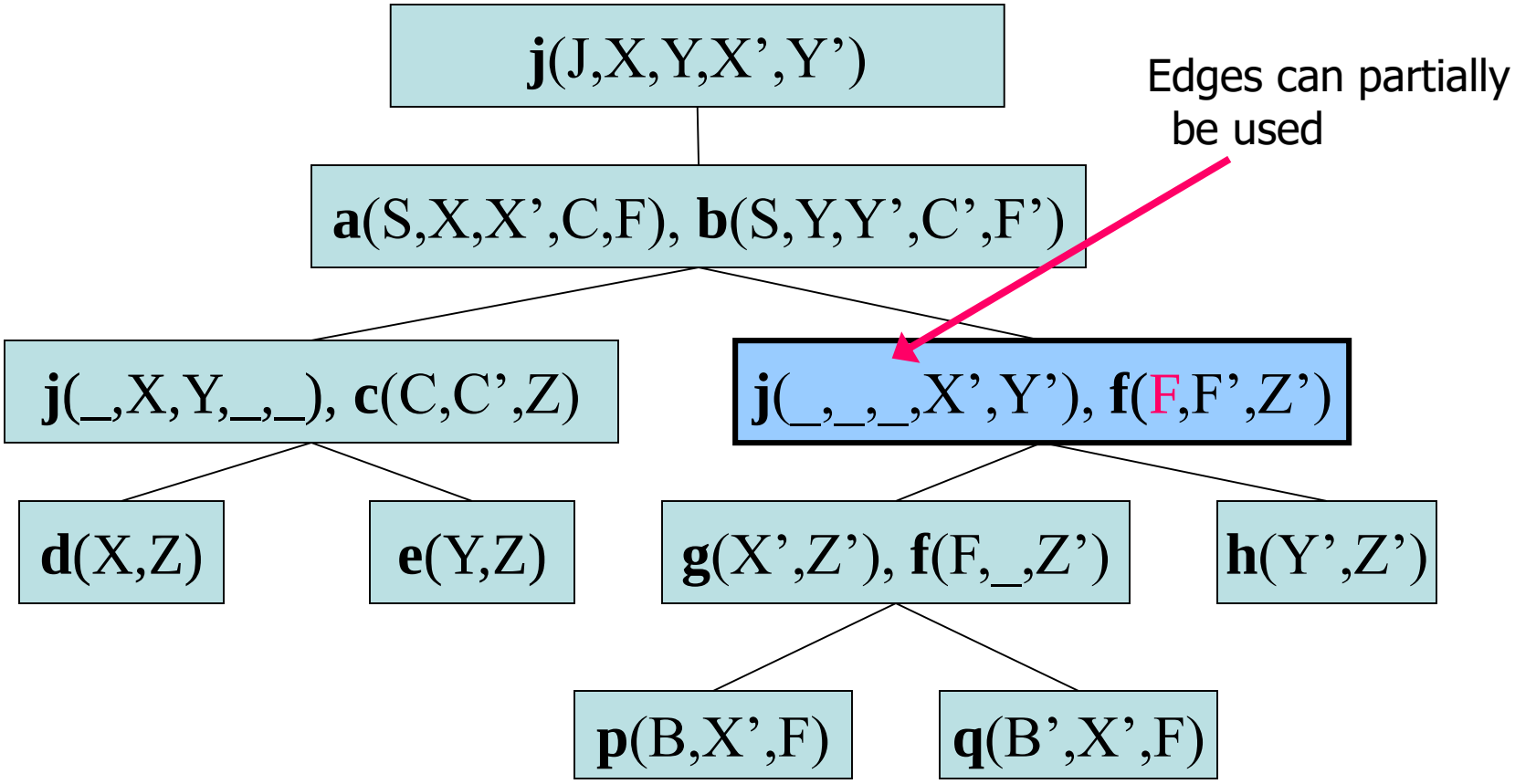


Basic Conditions(1/3)

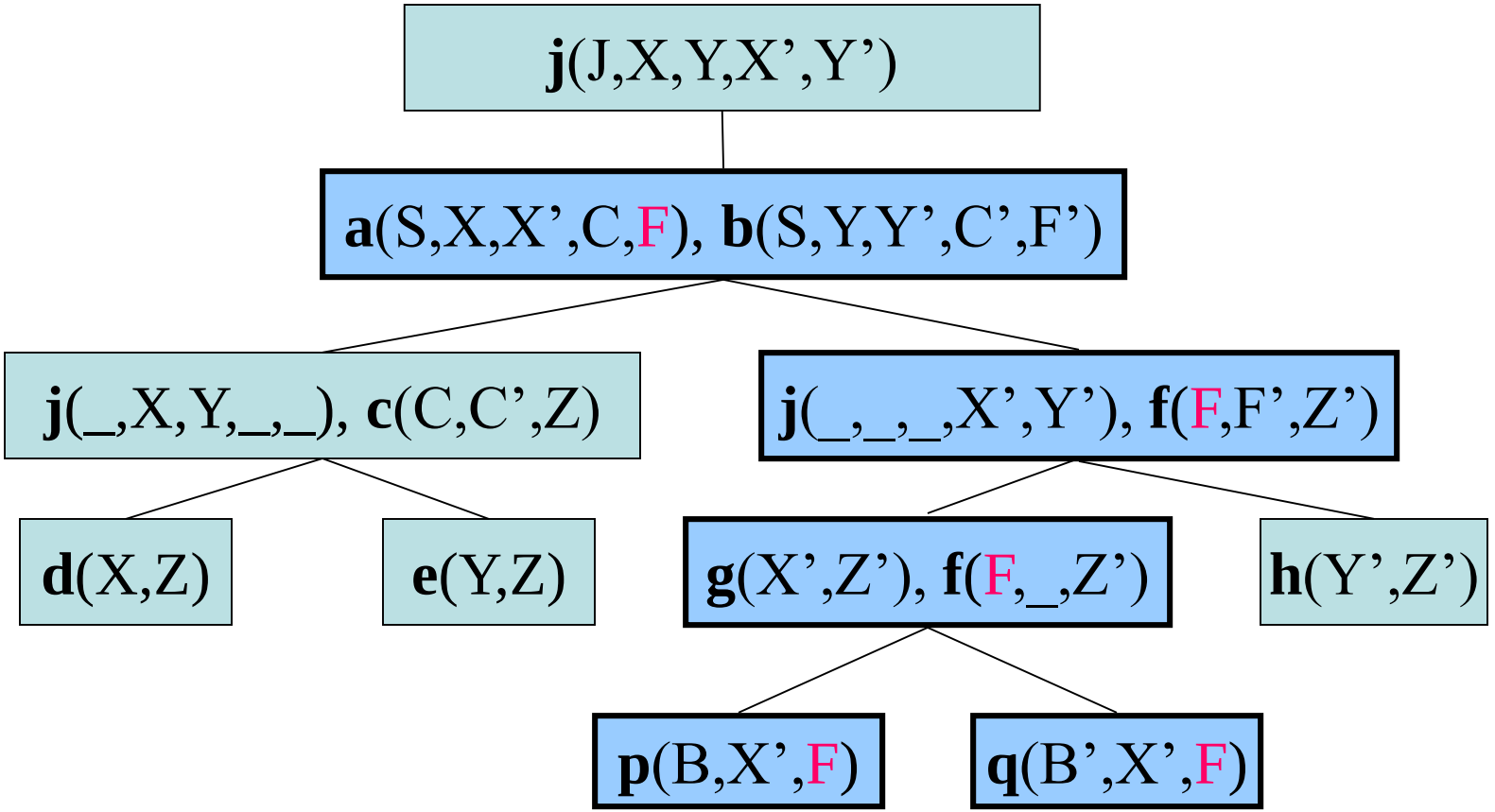
Original (direct) definition



Basic Conditions(2/3)



Connectedness Condition(3/3)

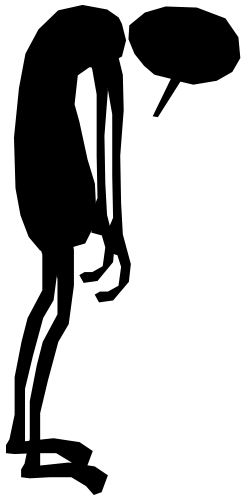


Computational Question

- 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 ?



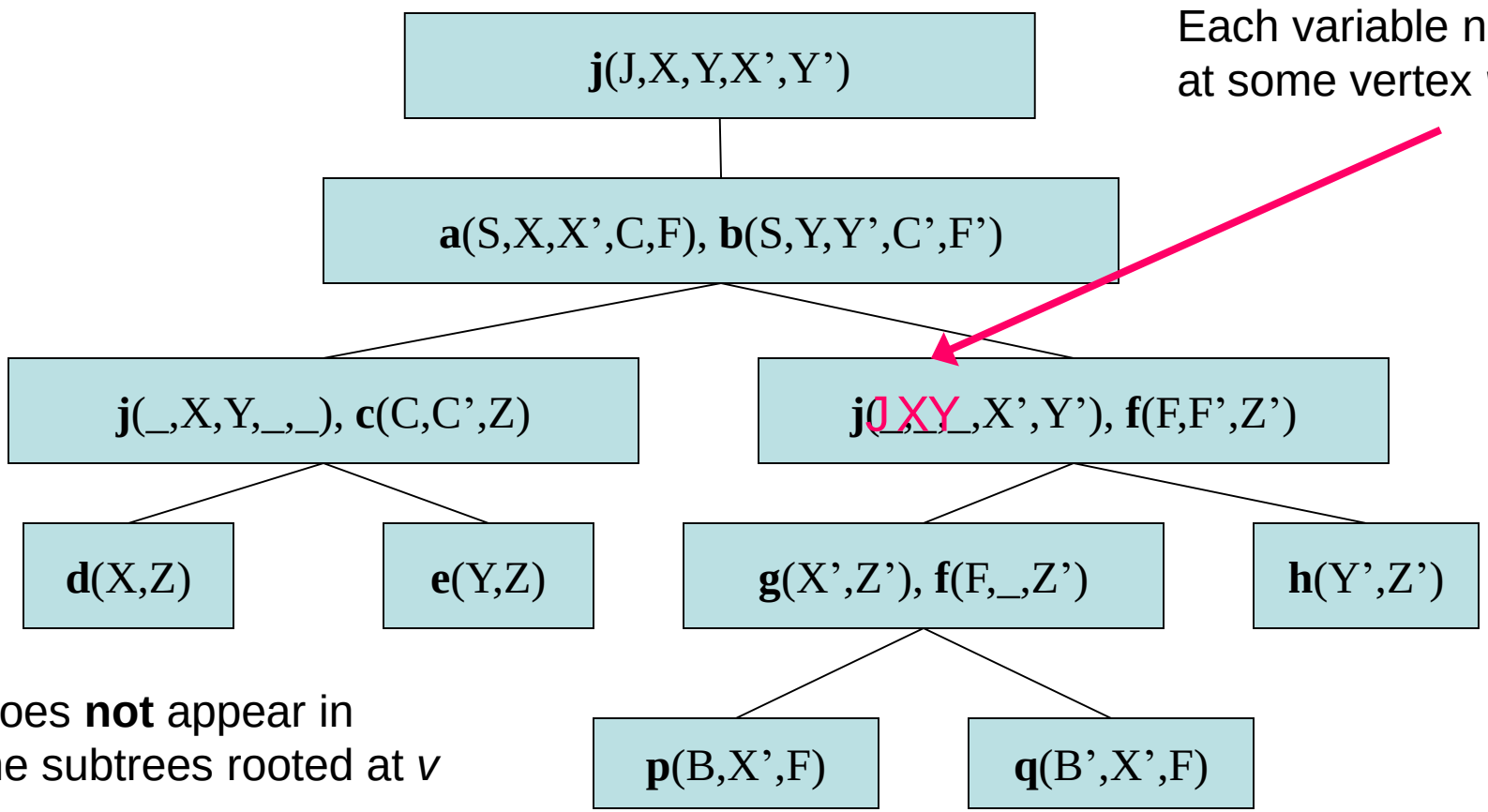
Bad news: ghw(H) < 4? NP-complete

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

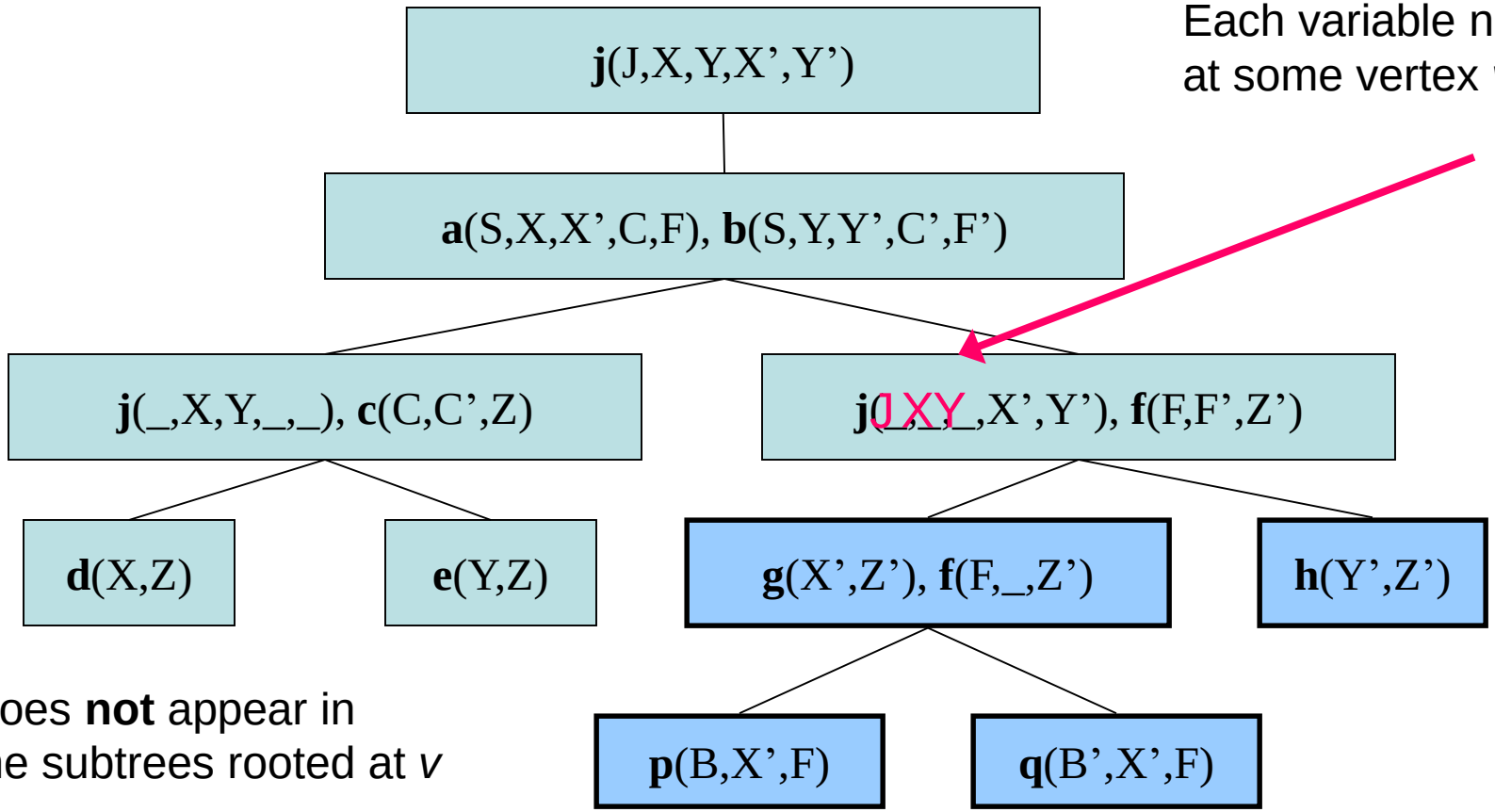
Hypertree Decomposition (HTD)

HTD = Generalized HTD +Special Condition

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



Special Condition

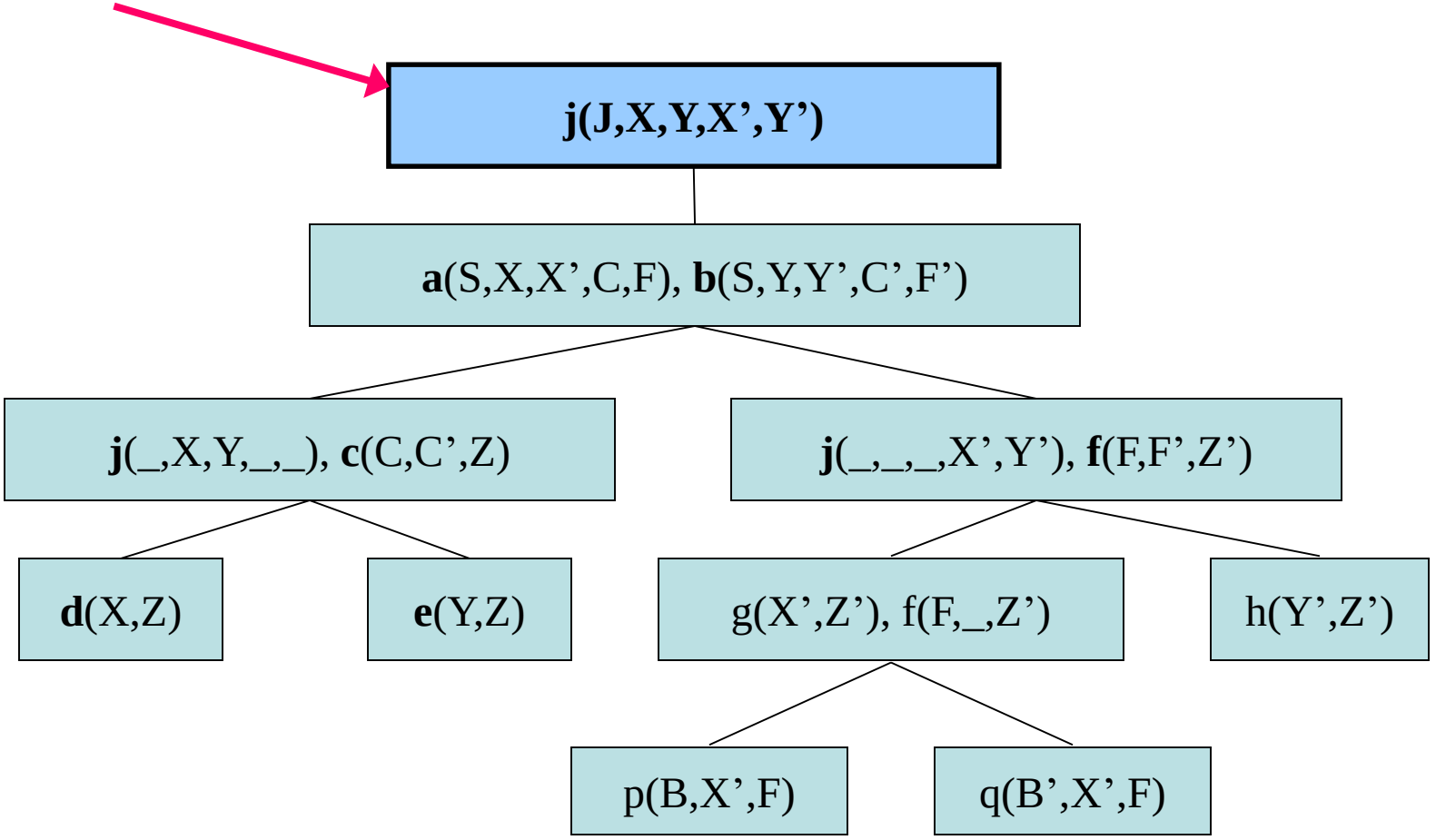


Each variable not used at some vertex *v*

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

Special Condition

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



Positive Results on Hypertree Decompositions

- For each query Q, $hw(Q) \leq qw(Q)$
- In some cases, $hw(Q) < qw(Q)$
- For fixed k, deciding whether $hw(Q) \leq k$ 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] :

$ghw(H) <= 3hw(H)+1$

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

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

$$\mathbb{A} = (U, R_1, R_2, \ldots, R_k)$$
$$\mathbb{B} = (V, S_1, S_2, \ldots, S_k)$$

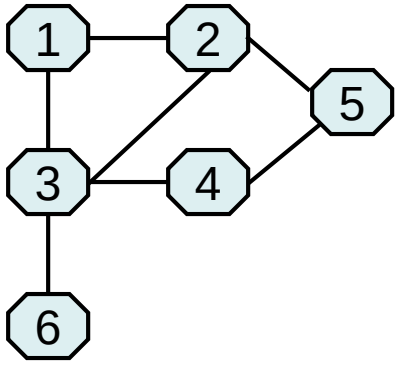
- Decide whether there exists a *homomorphism* ***h*** from $\mathbb{A}$ to $\mathbb{B}$

$$h: U \longrightarrow V$$

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

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

Example: graph colorability



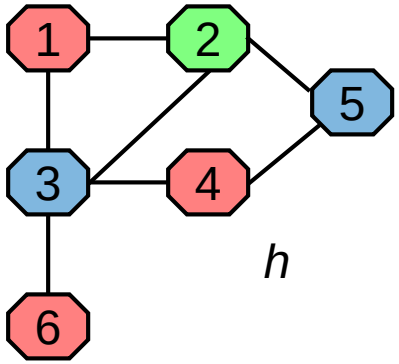
$\mathbb{A}$

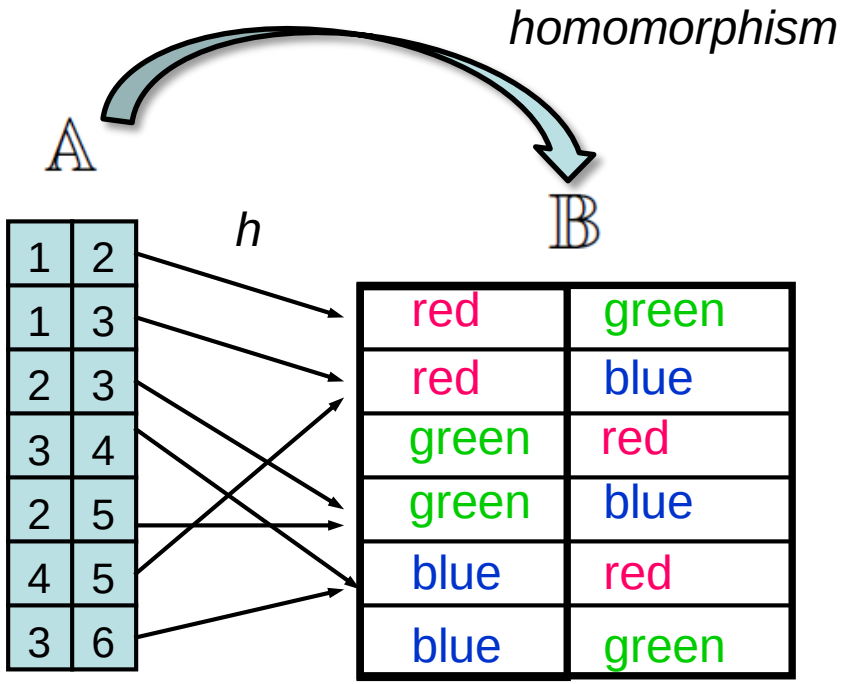
1	2
1	3
2	3
3	4
2	5
4	5
3	6

$\mathbb{B}$

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

Example: graph colorability



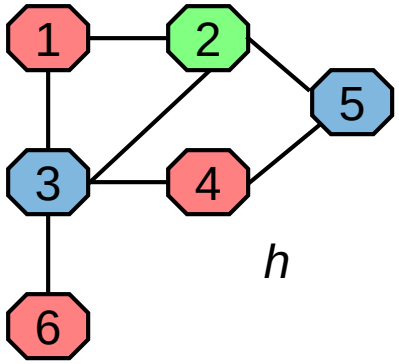


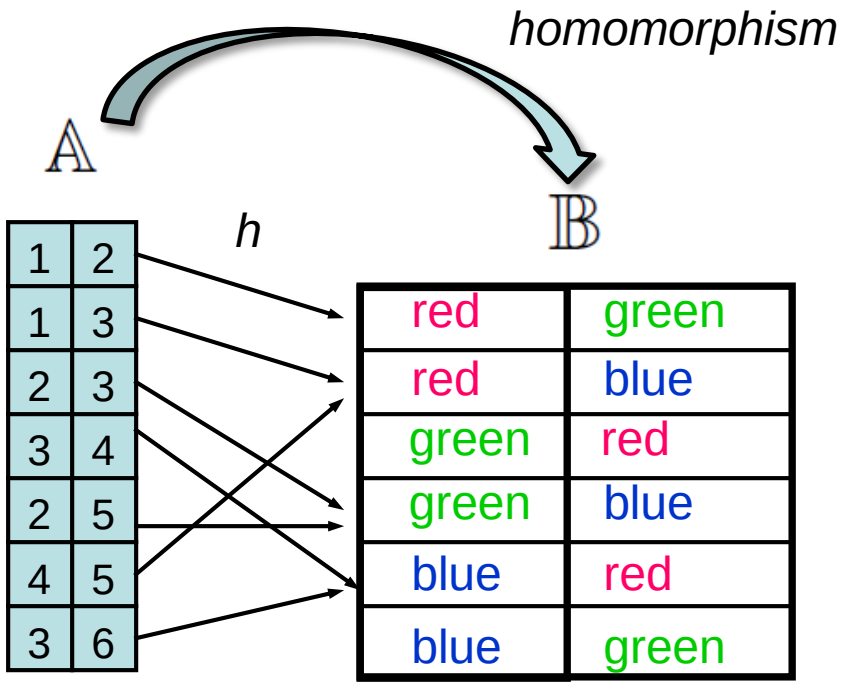
Complexity: HOM is NP-complete

(well-known, independently proved in various contexts)

Membership: Obvious, guess *h.*

Hardness: Transformation from 3COL.

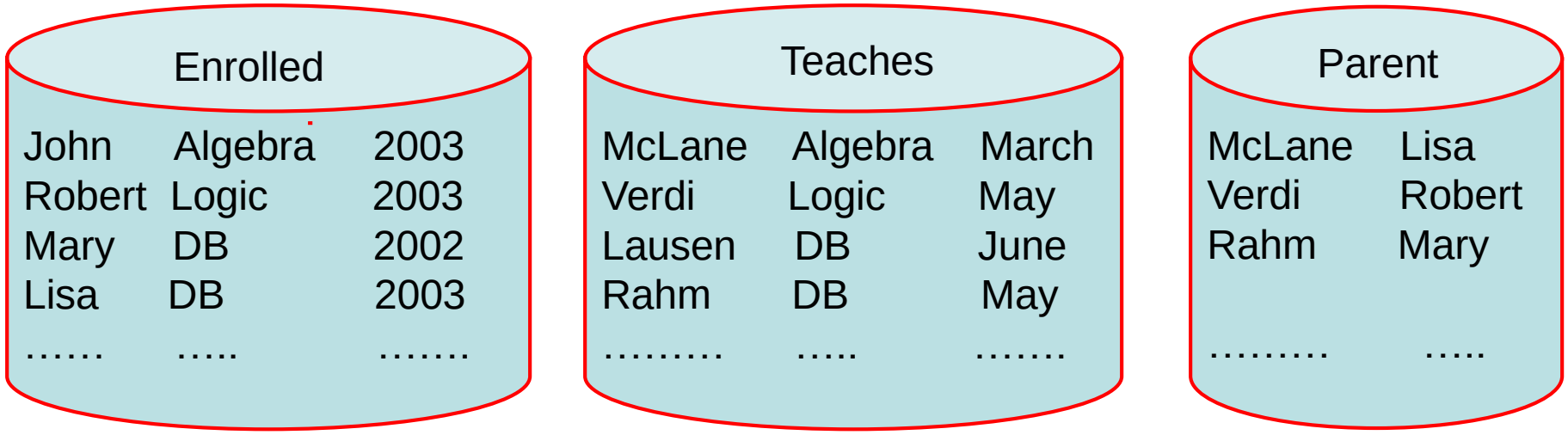




Graph 3-colourable *iff* HOM(*A*,*B*) yes-instance.

Conjunctive Database Queries

DATABASE:



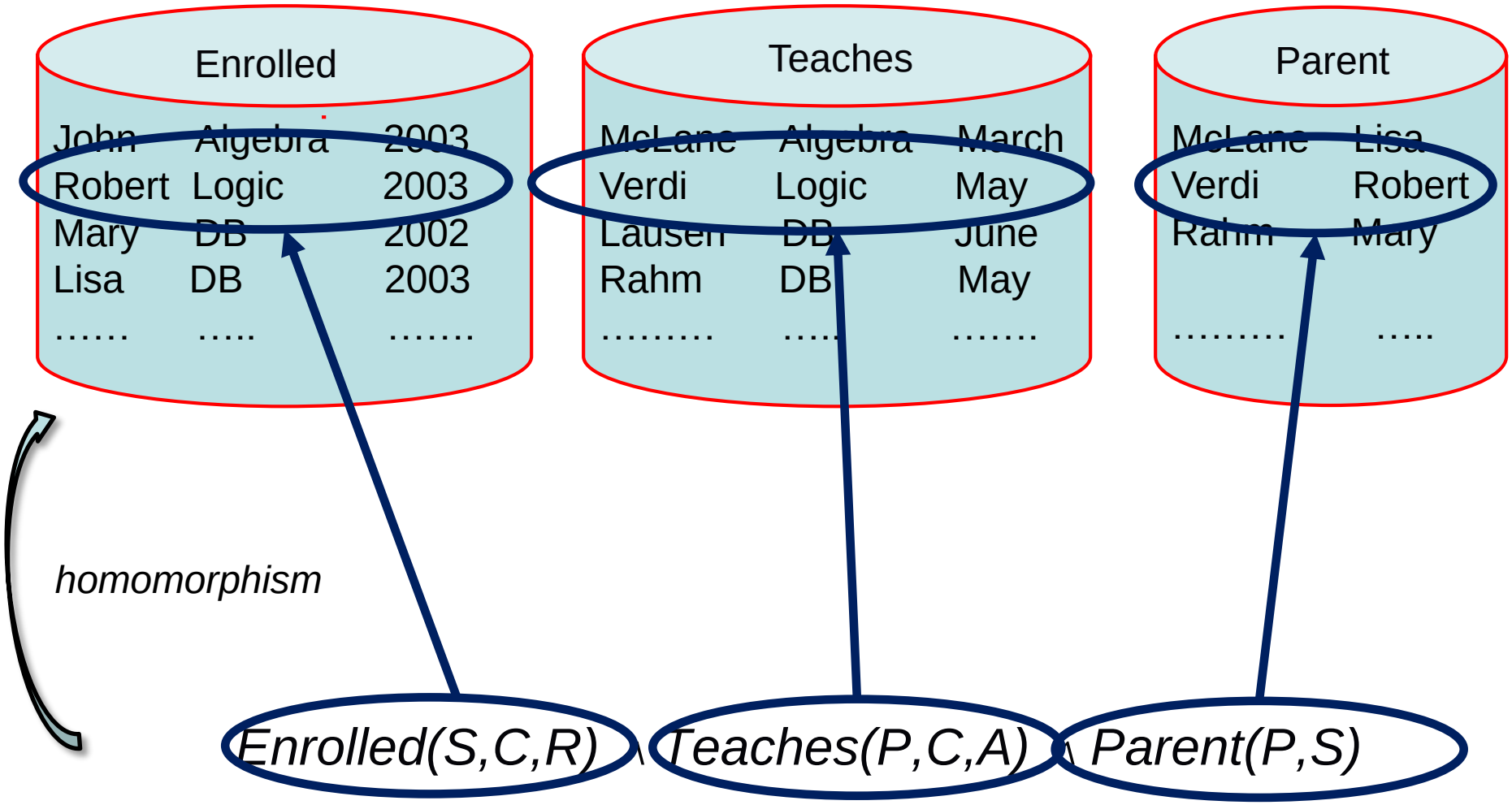
QUERY:

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

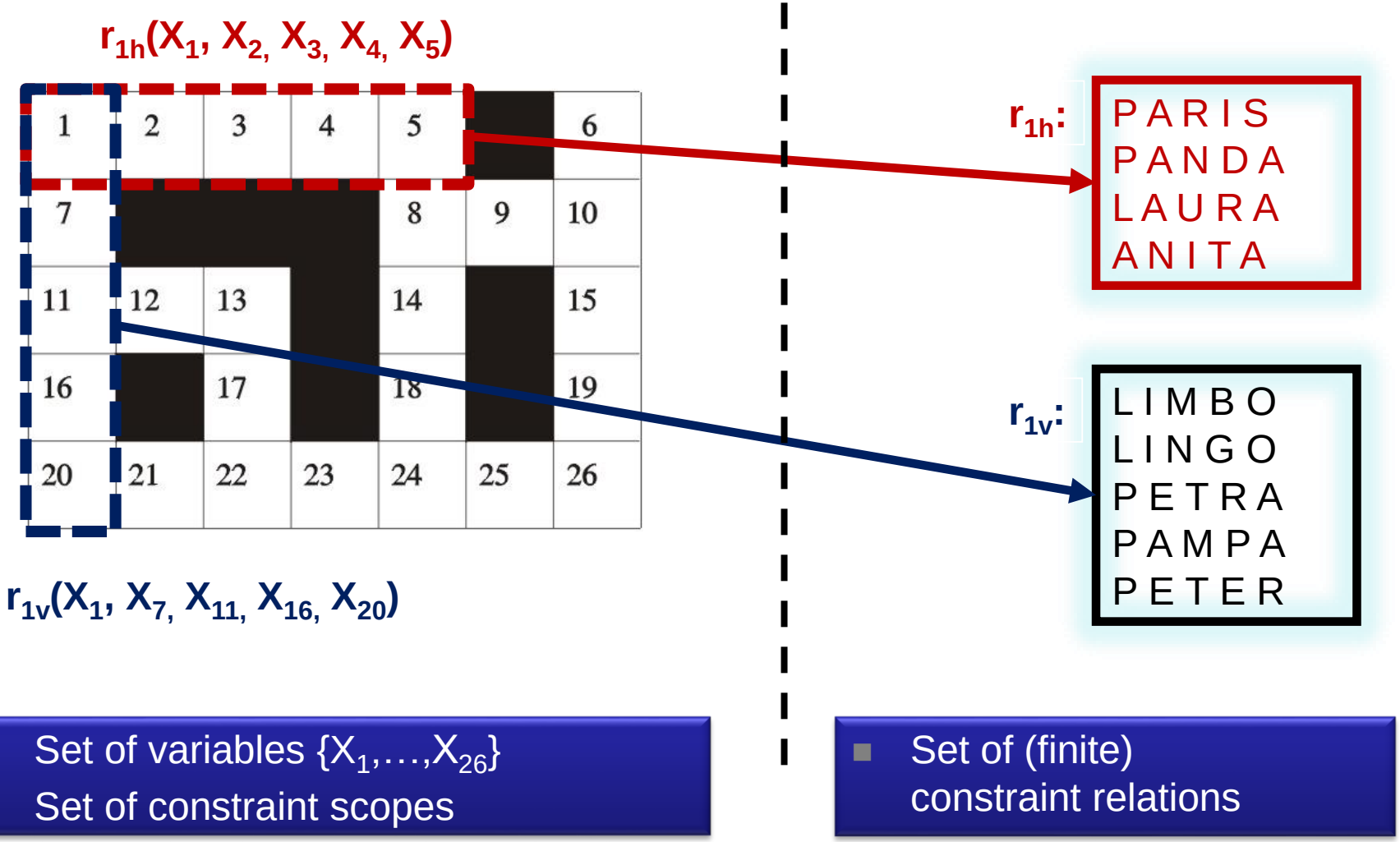
$$ans \leftarrow Enrolled(S,C,R) \wedge Teaches(P,C,A) \wedge Parent(P,S)$$

Conjunctive Database Queries

DATABASE:



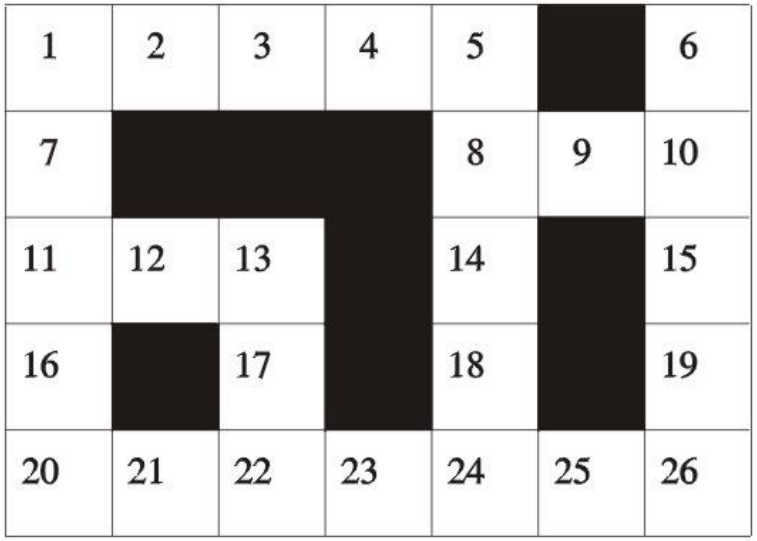
CSPs as Homomorphism Problems

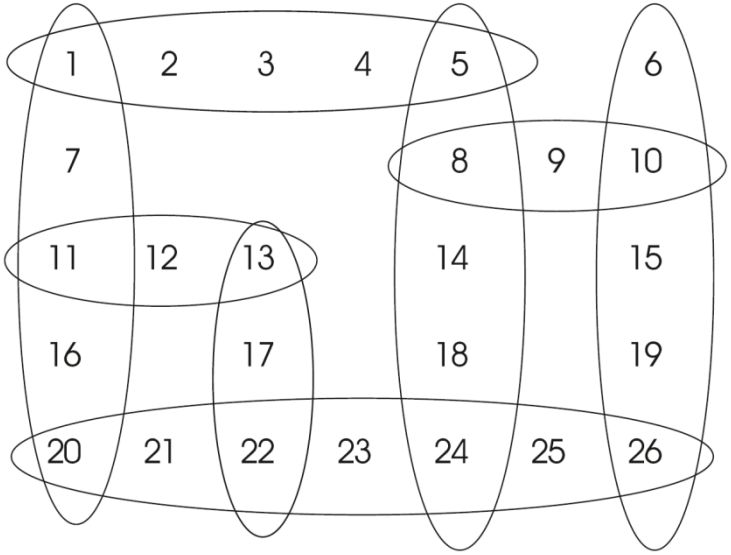


- Set of variables $\{X_1,\ldots,X_{26}\}$
- Set of constraint scopes

- Set of (finite) constraint relations

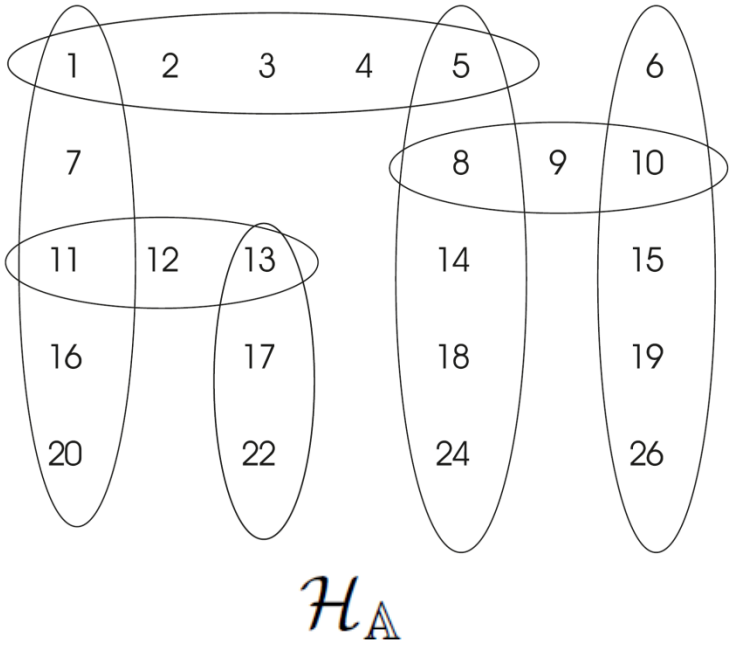
CSPs and Hypergraphs





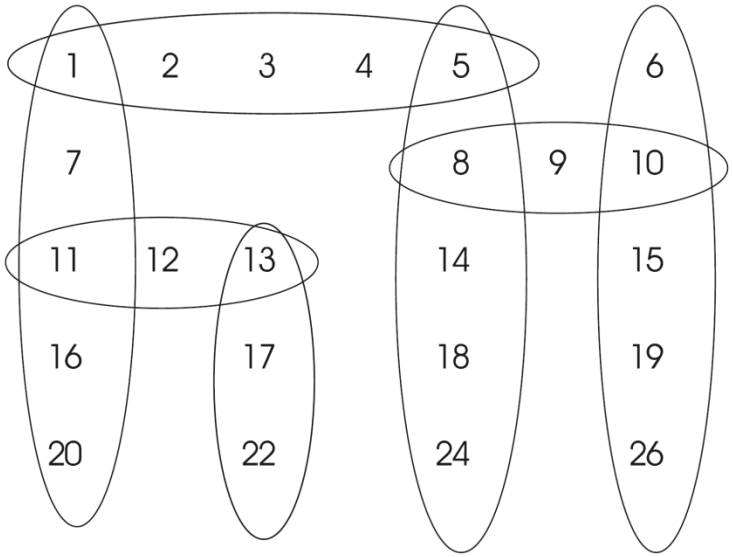
$\mathcal{H}_{\mathbb{A}}$

Structurally Restricted CSPs



Structurally Restricted CSPs

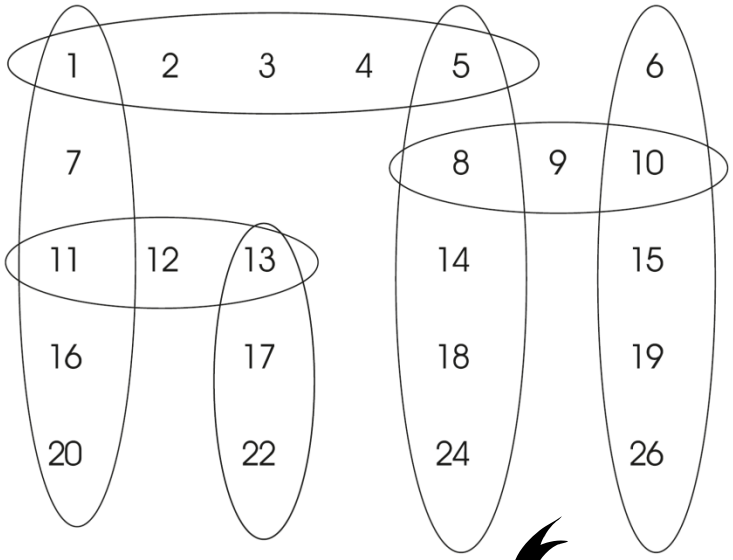
The hypergraph is acyclic



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

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(n^2 x log n)
- 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(n^2 x log n)
- 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(n^2 x log n)
- LOGCFL-complete

[Gottlob, Leone, Scarcello, J.ACM'00]

A Polynomial-time Algorithm

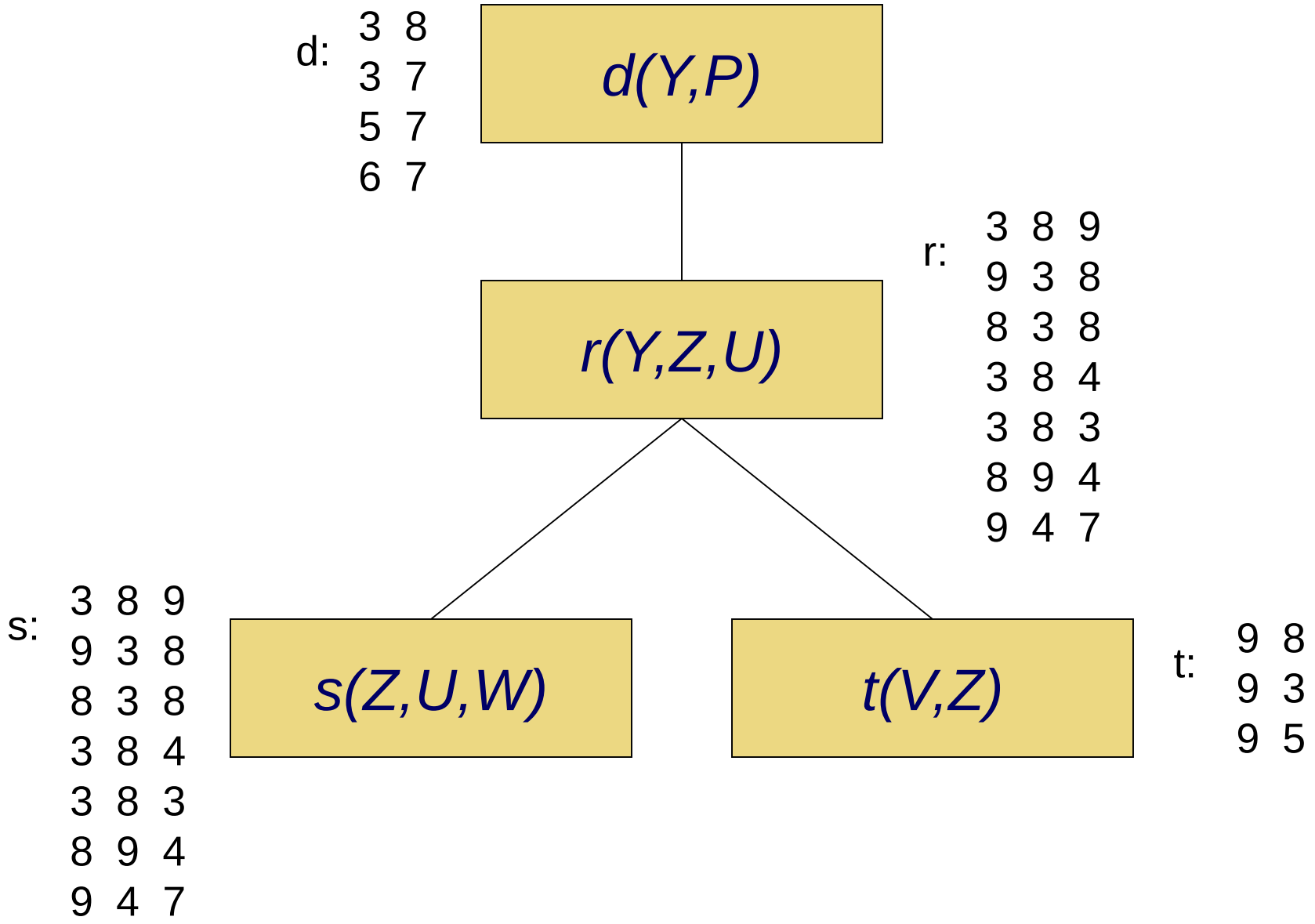
HOM: The homomorphism problem

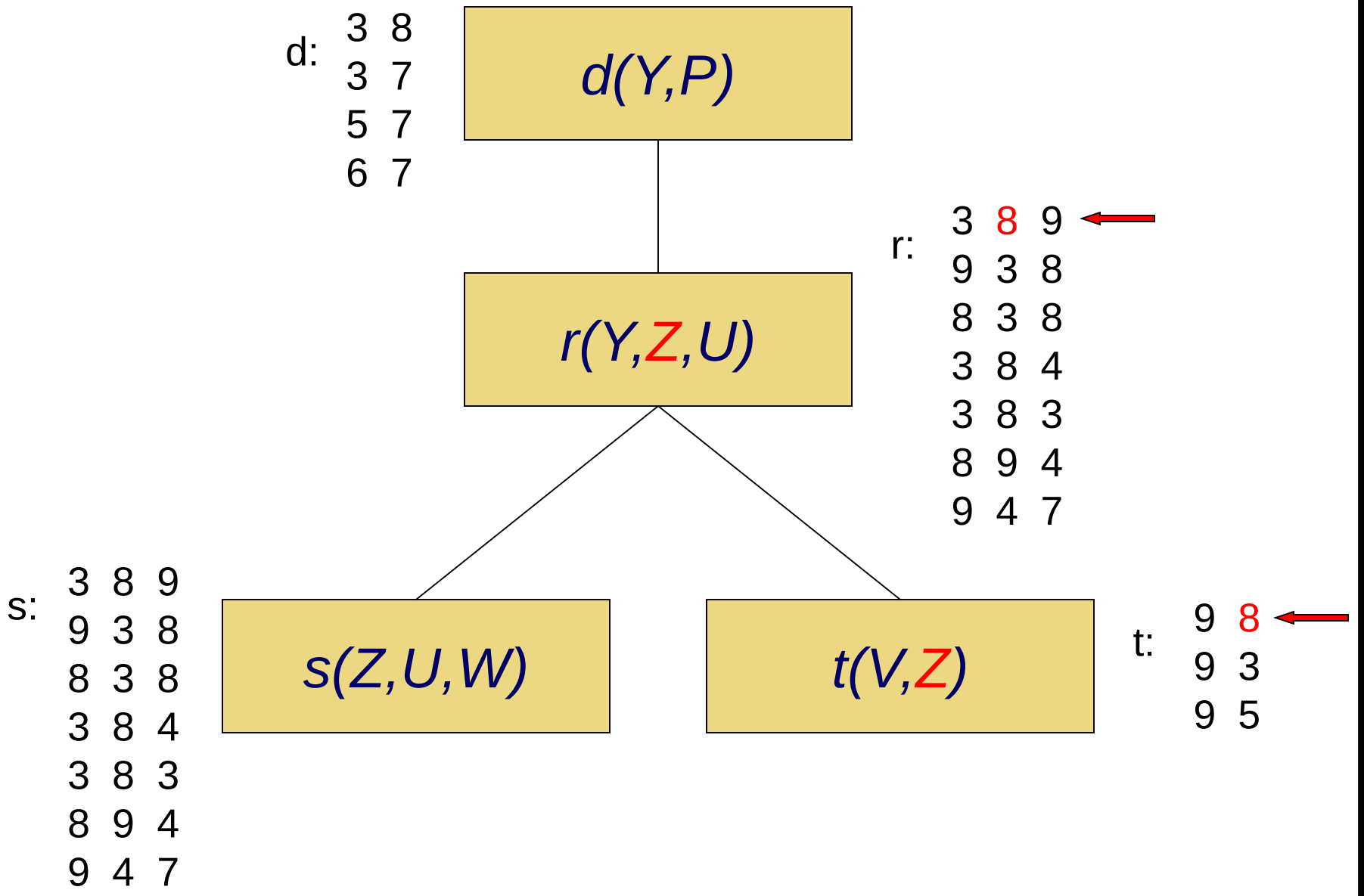
BCQ: Boolean conjunctive query evaluation

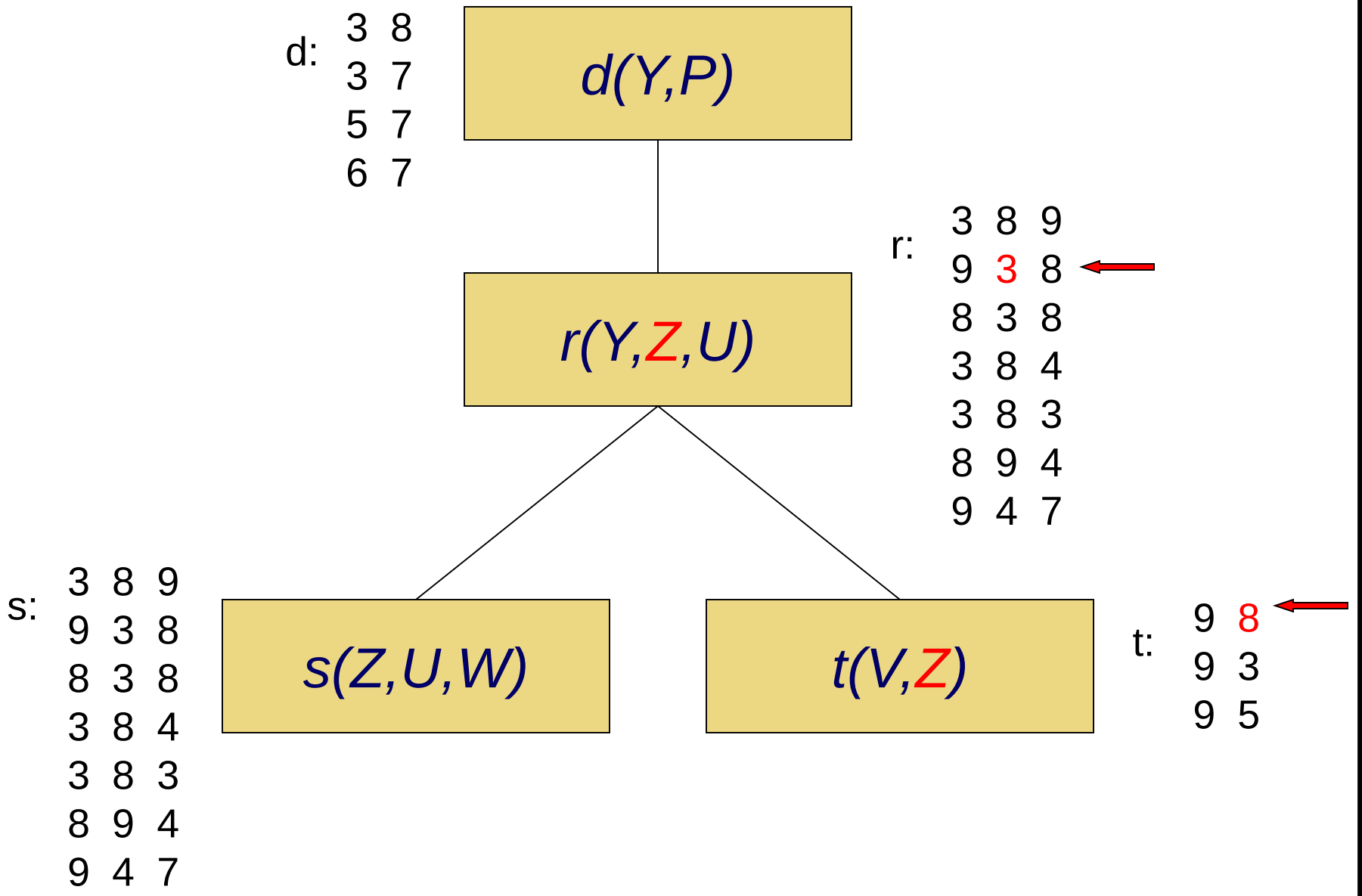
CSP: Constraint satisfaction problem

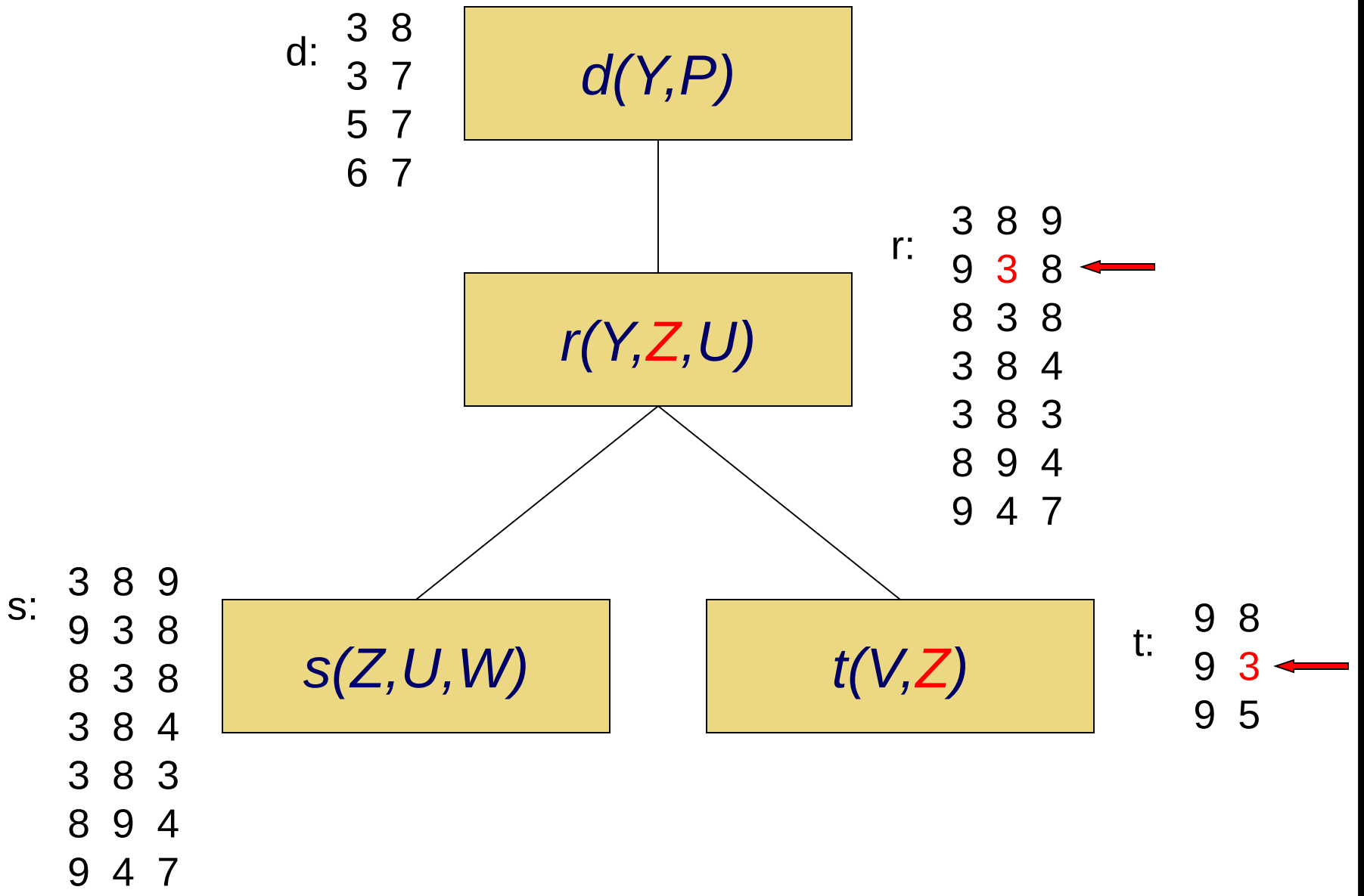
Yannakakis's Algorithm (Acyclic structures):

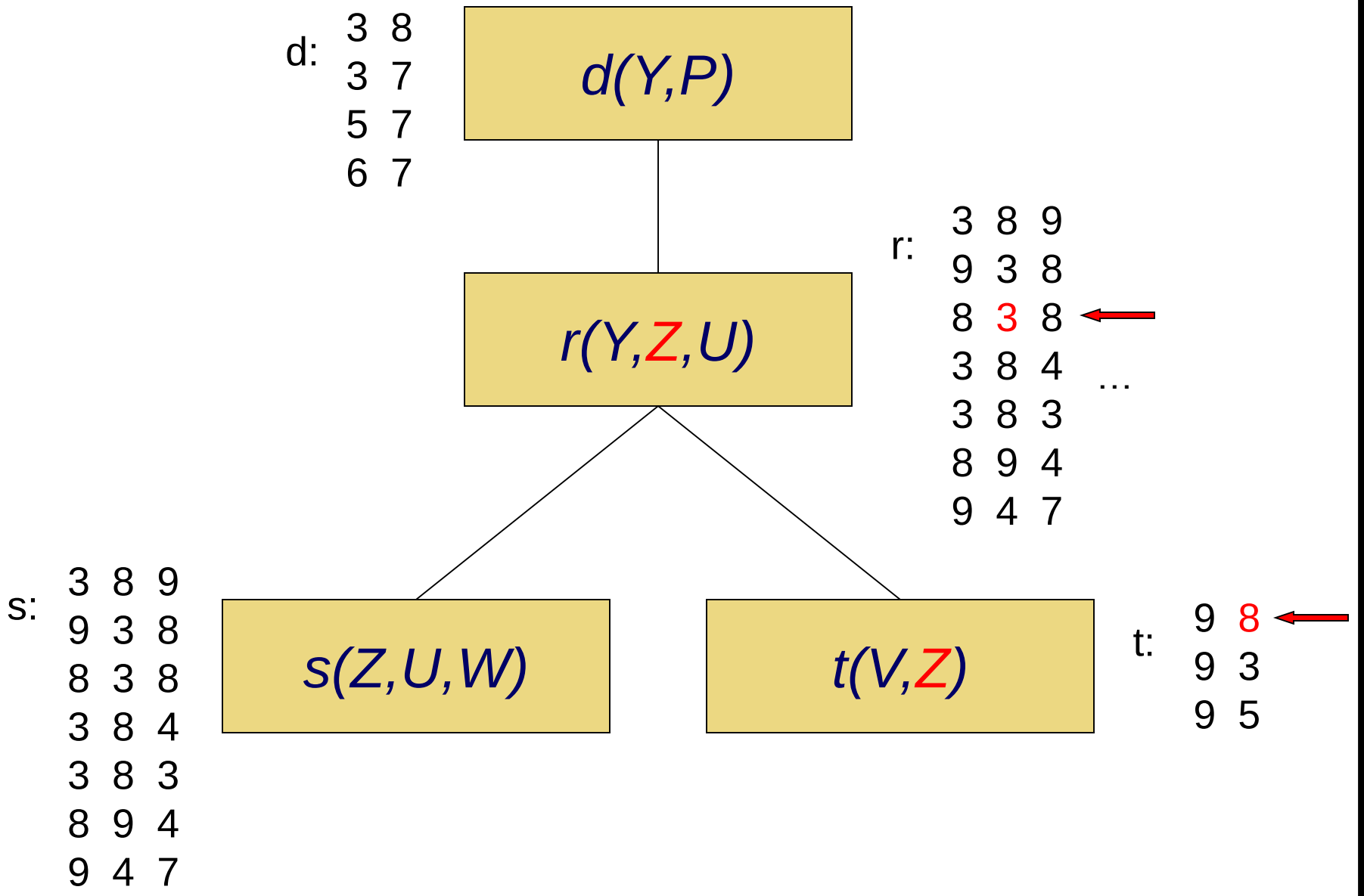
- Dynamic Programming over a Join Tree, where each vertex contains the relation associated with the corresponding hyperedge
- Therefore, if there are more constraints over the same relation, it may occur (as a copy) at different vertices

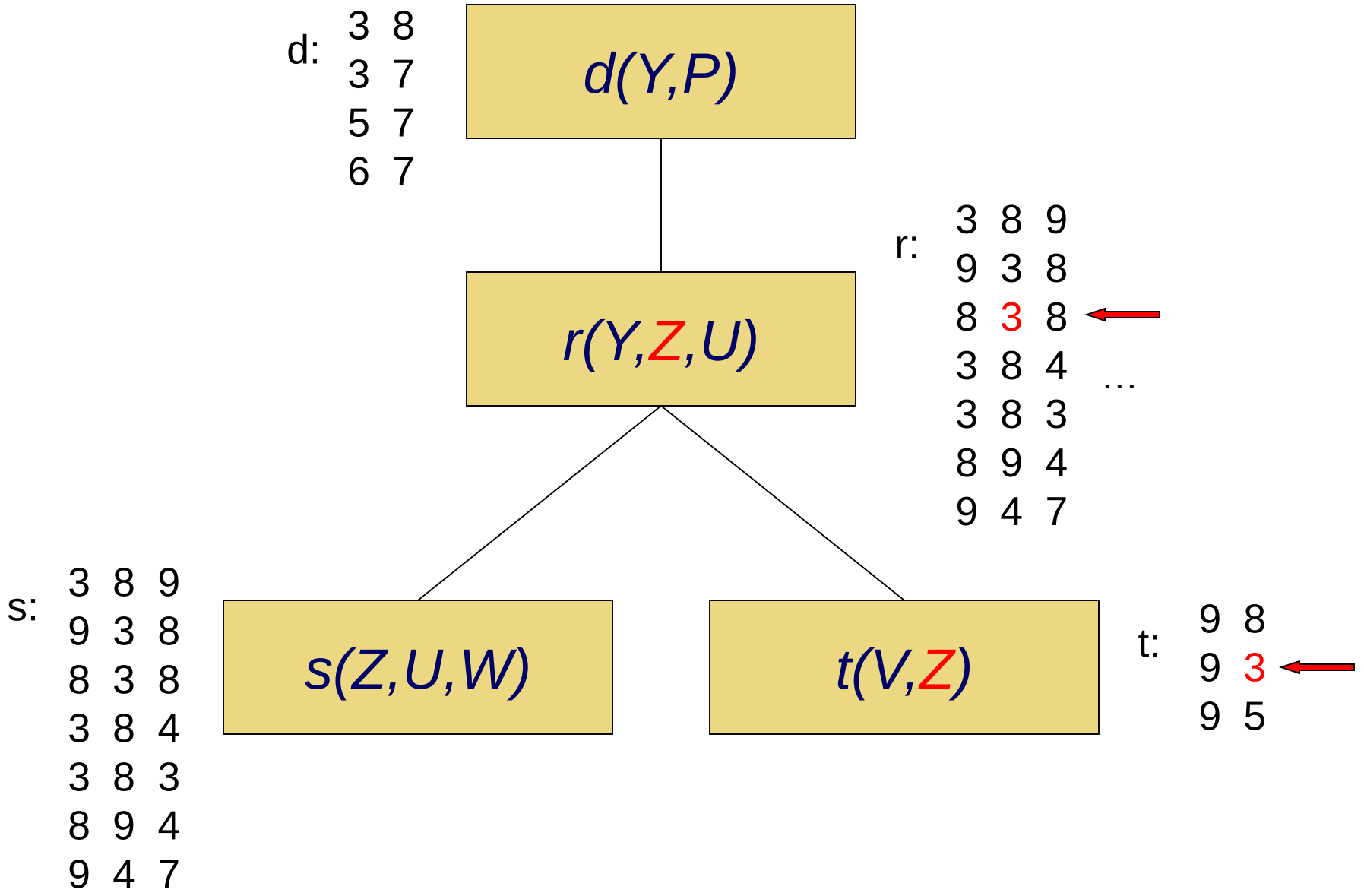

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

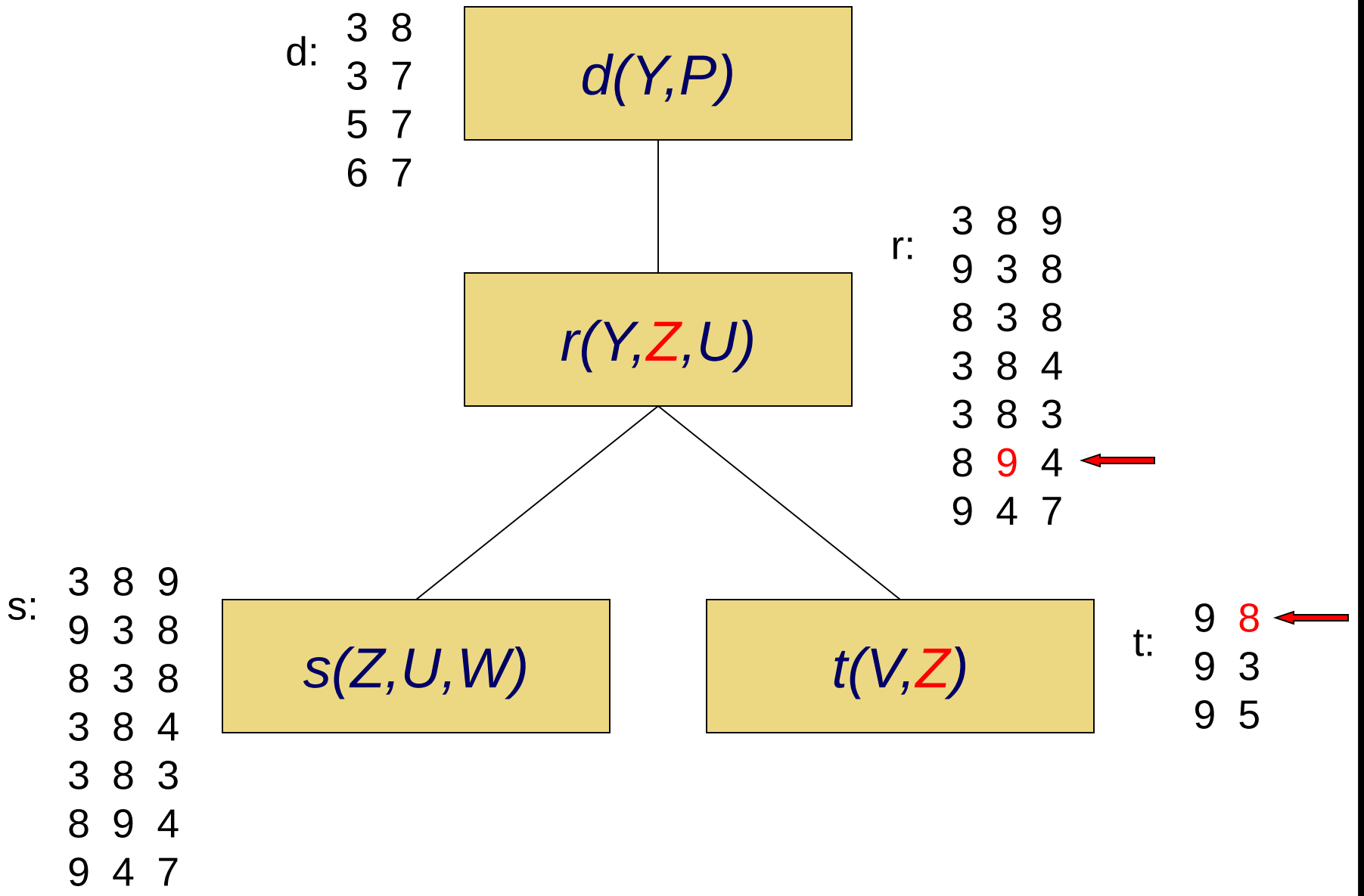

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

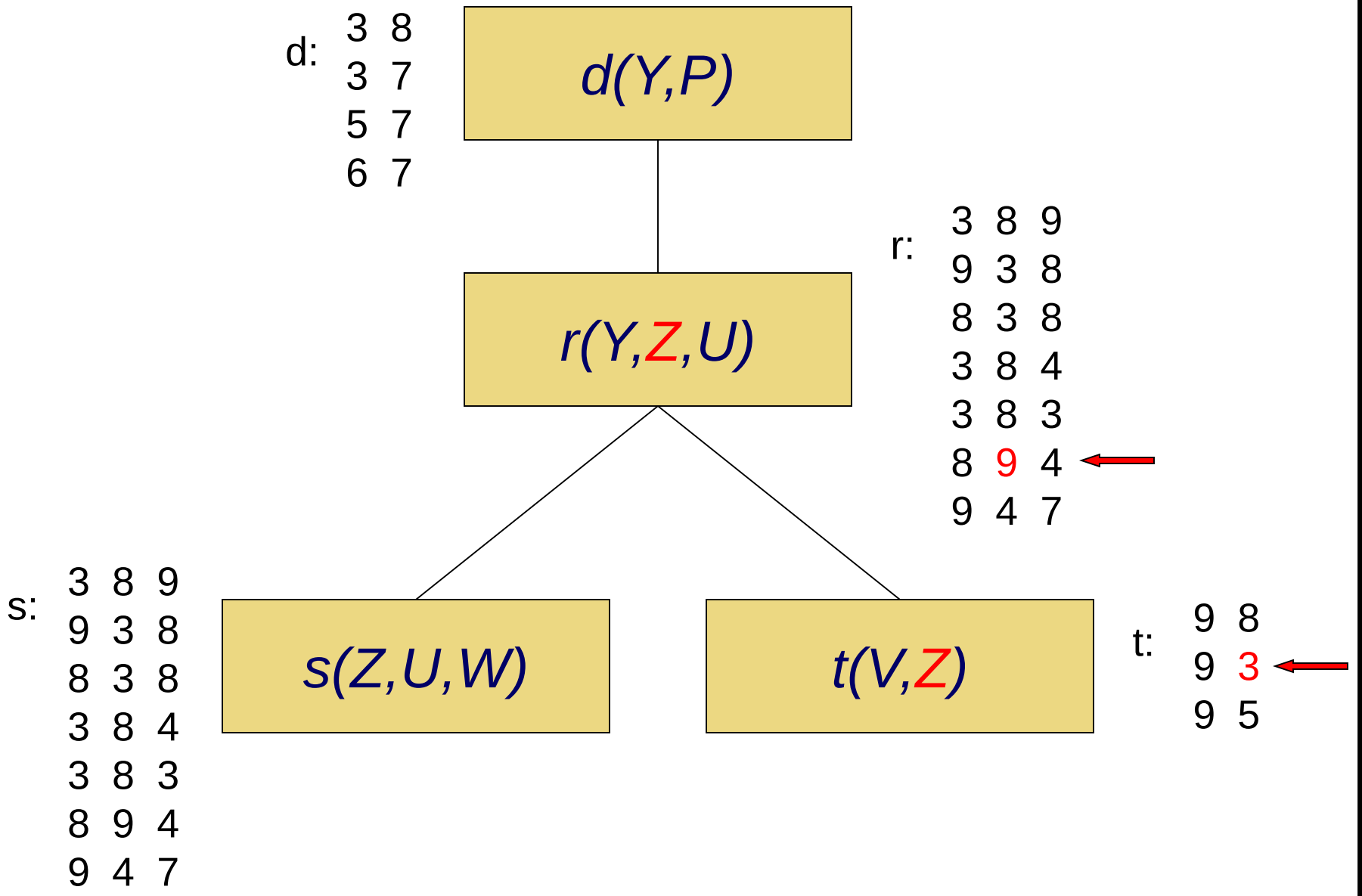

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

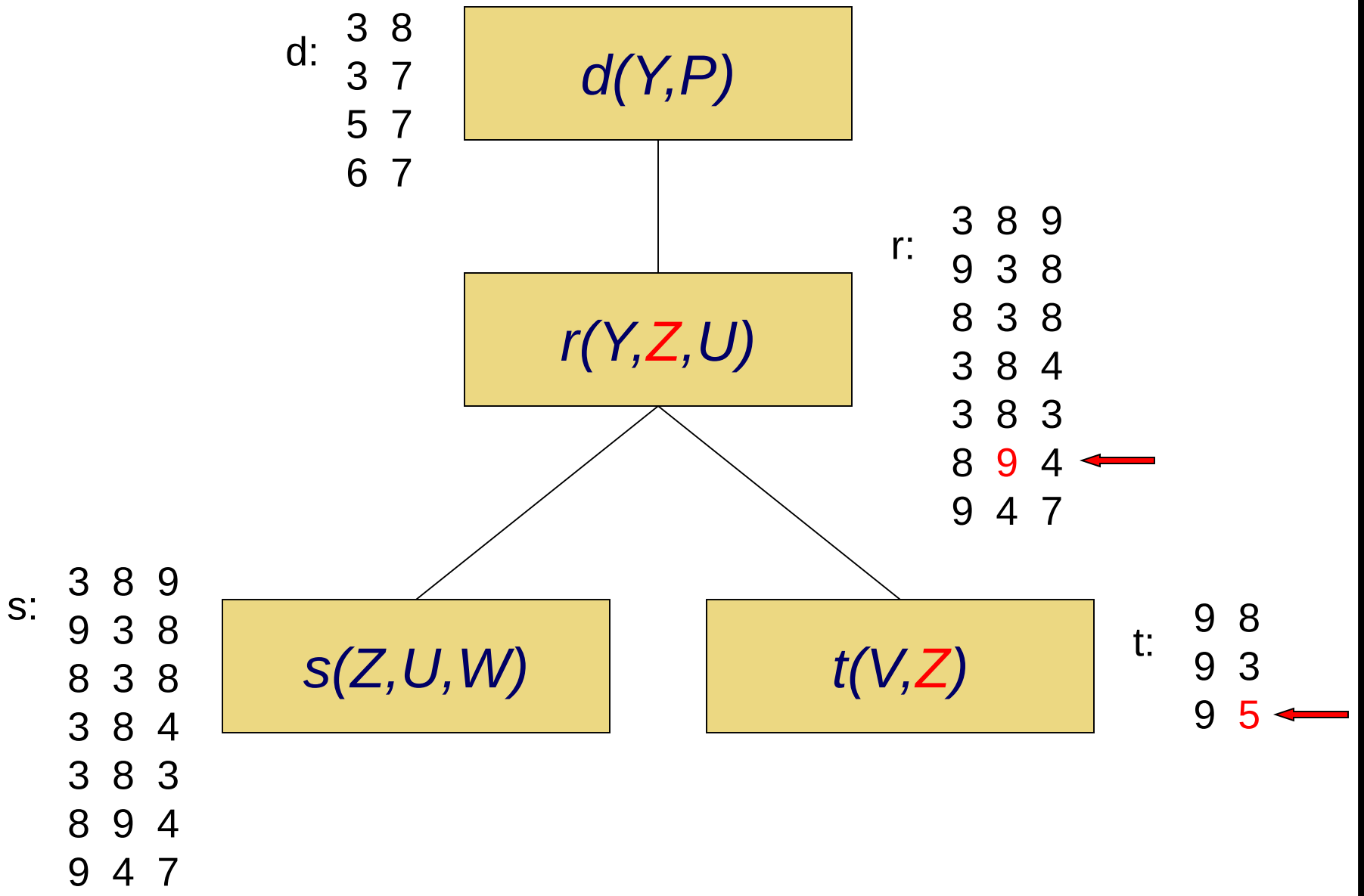

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

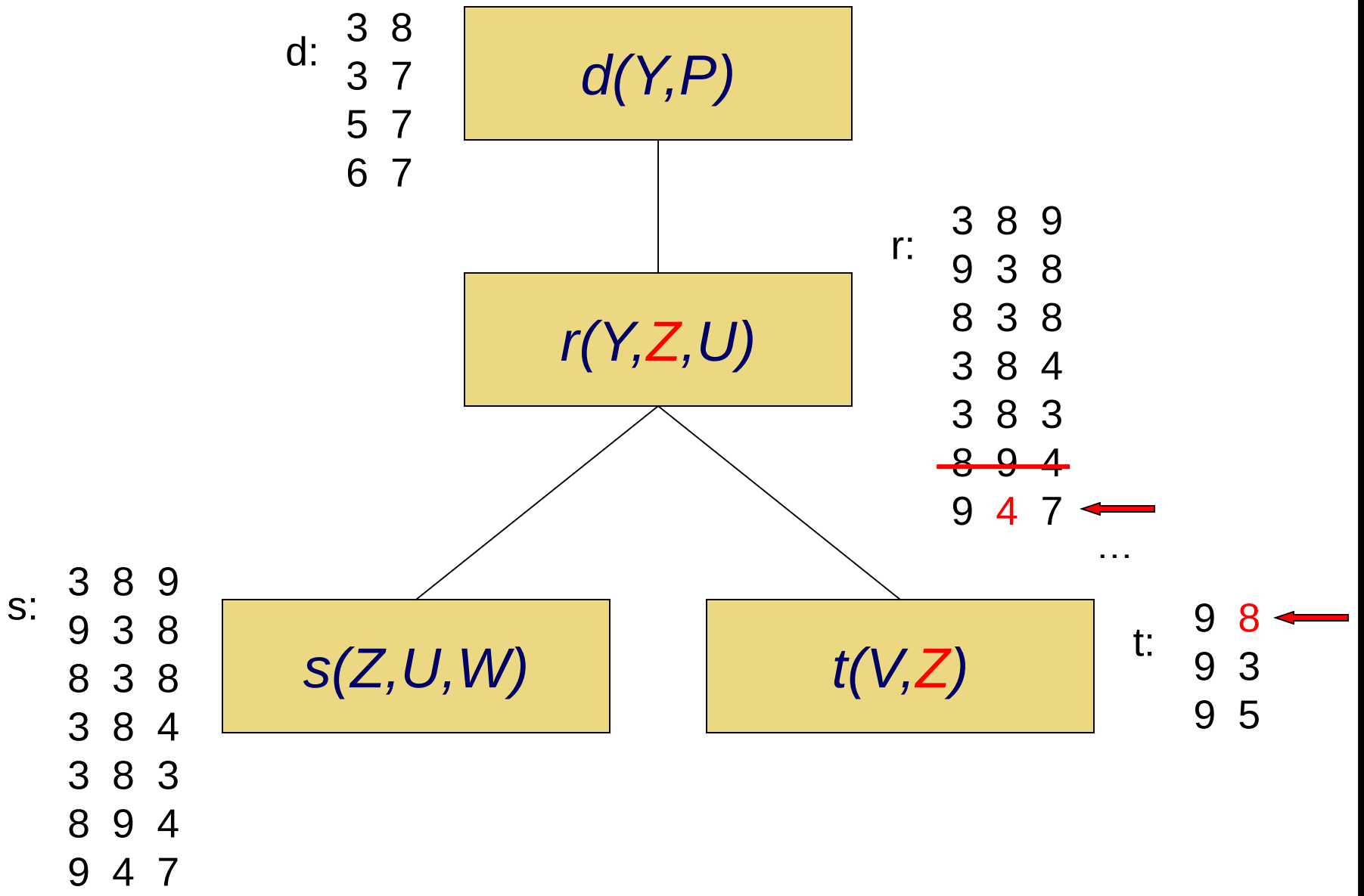

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
...
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

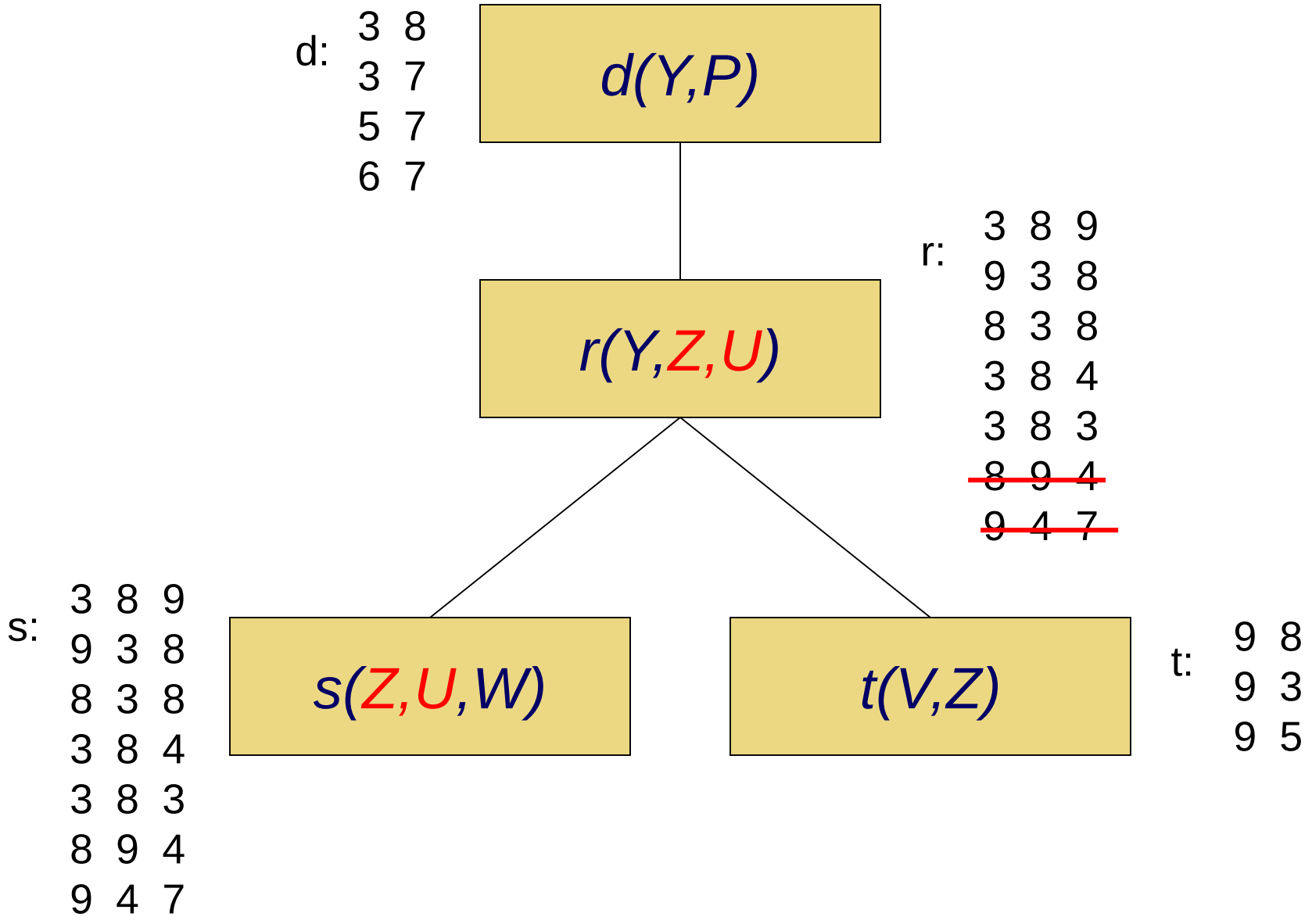

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
...
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

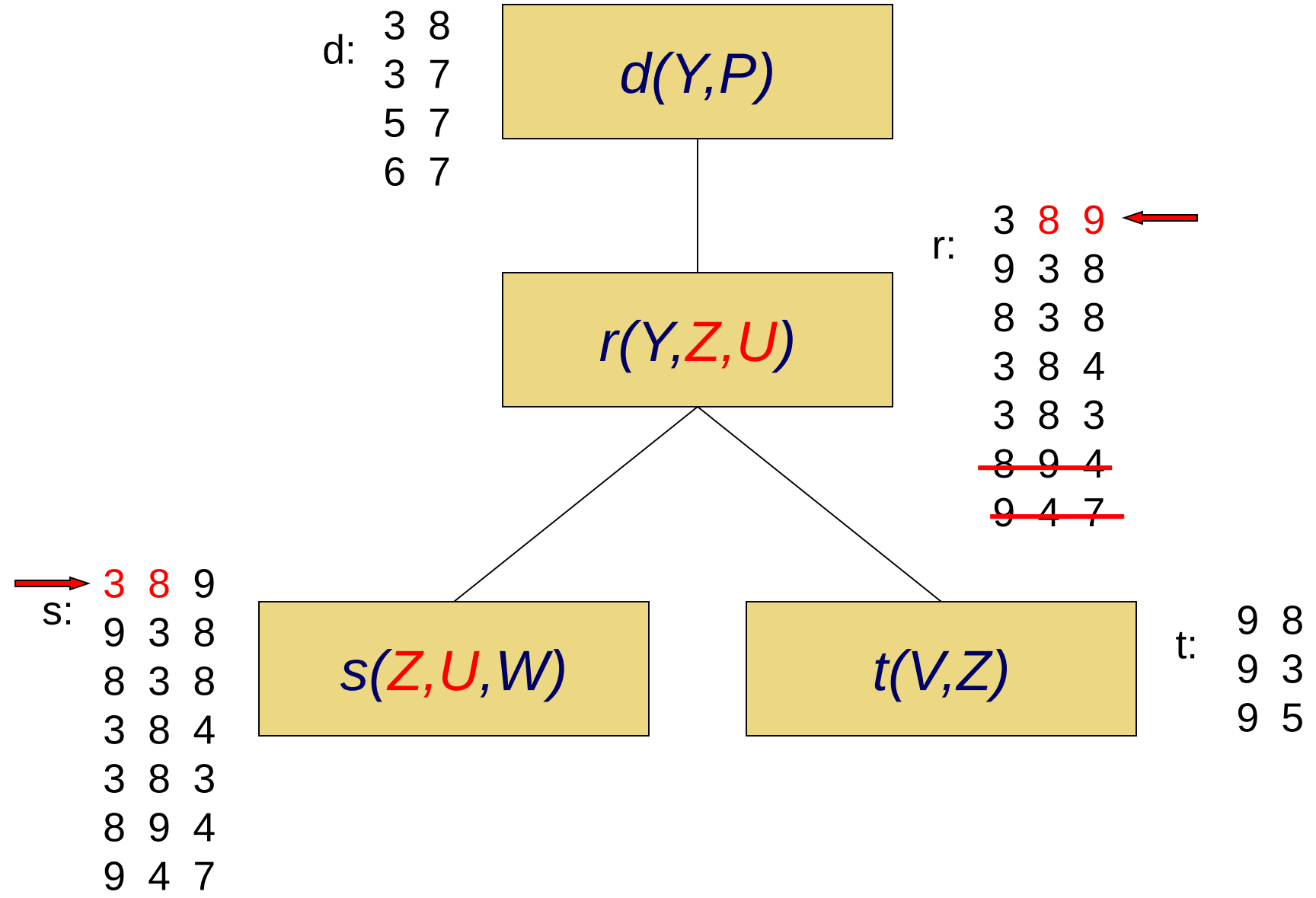

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

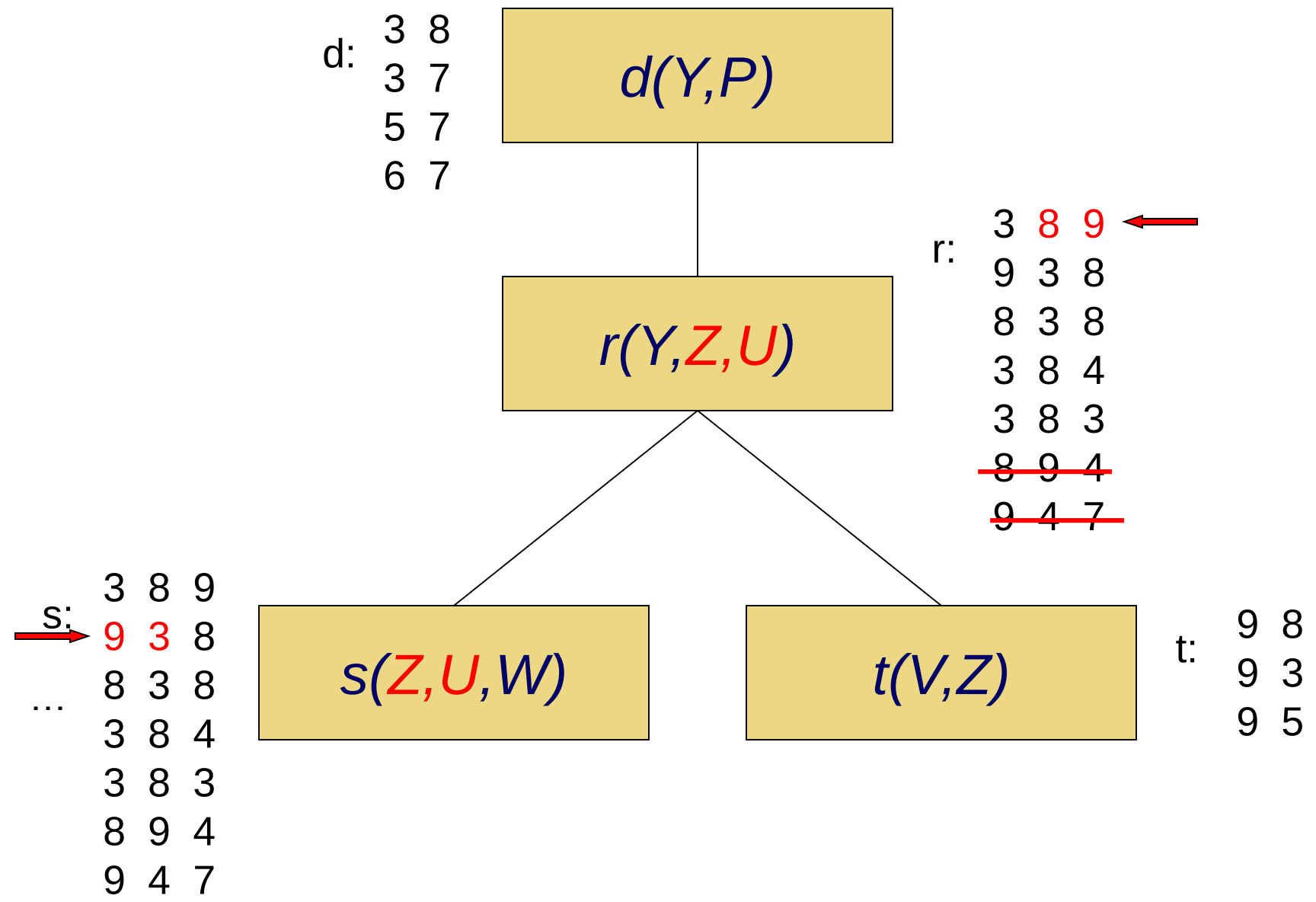

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

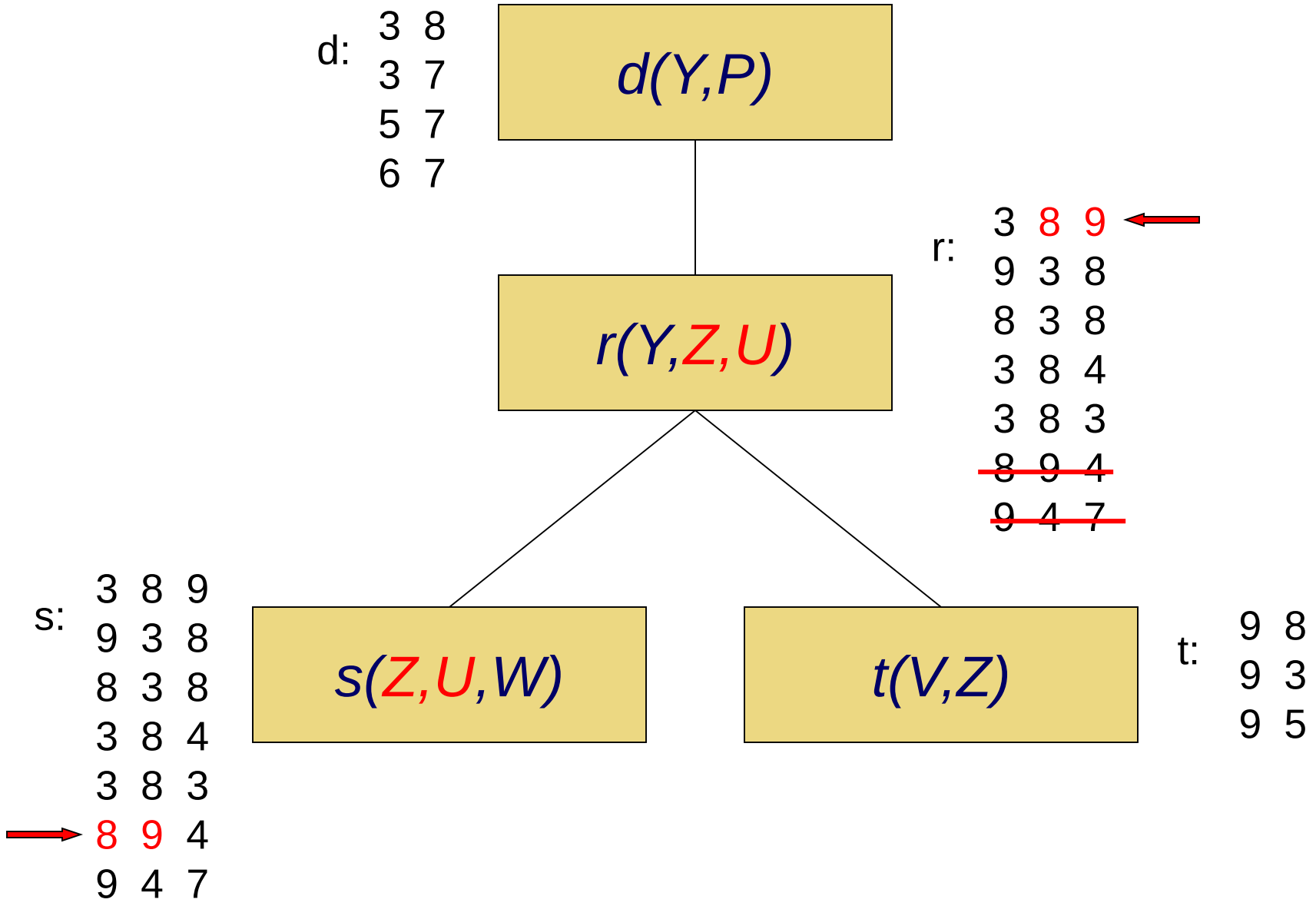

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

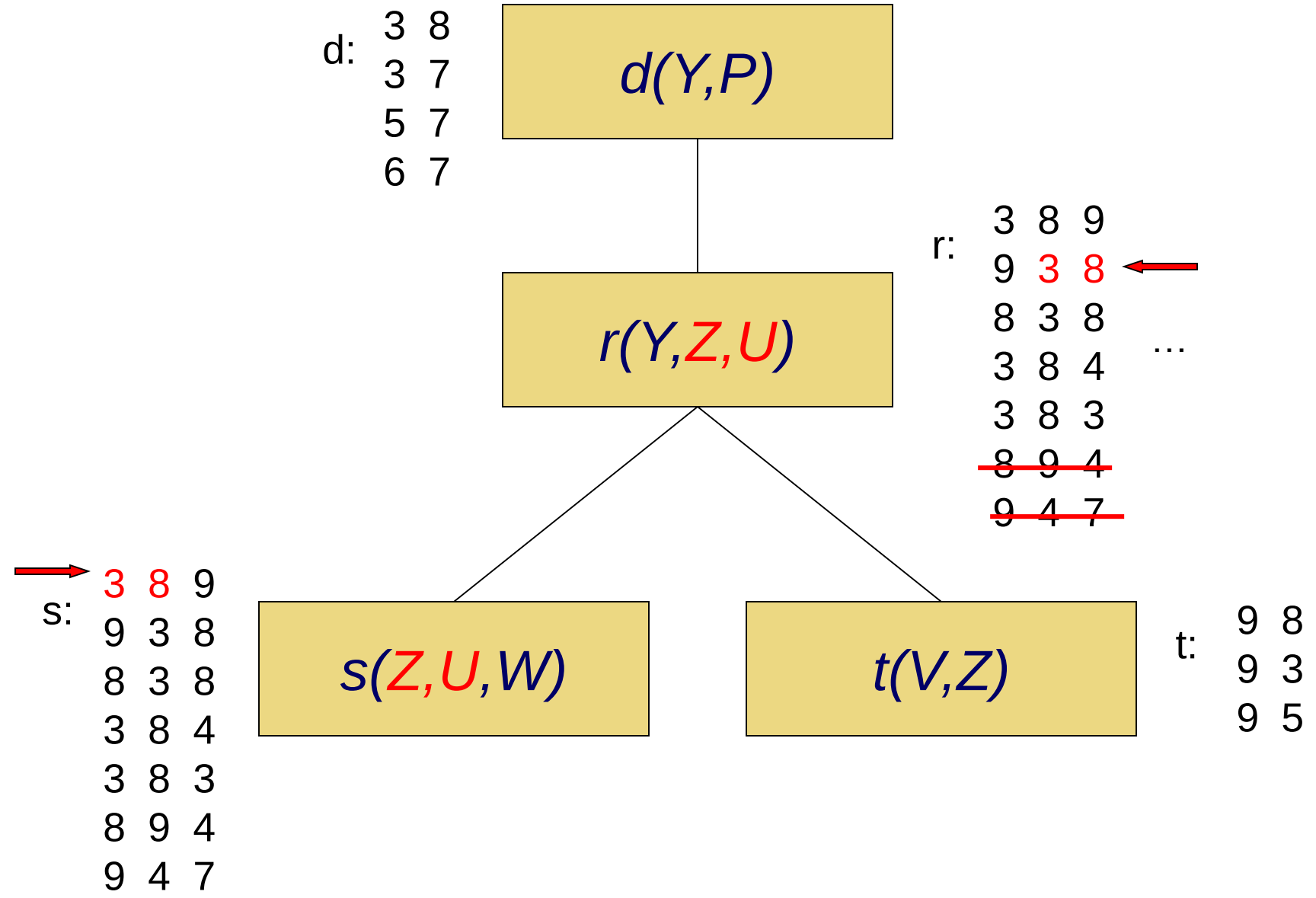

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
…
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

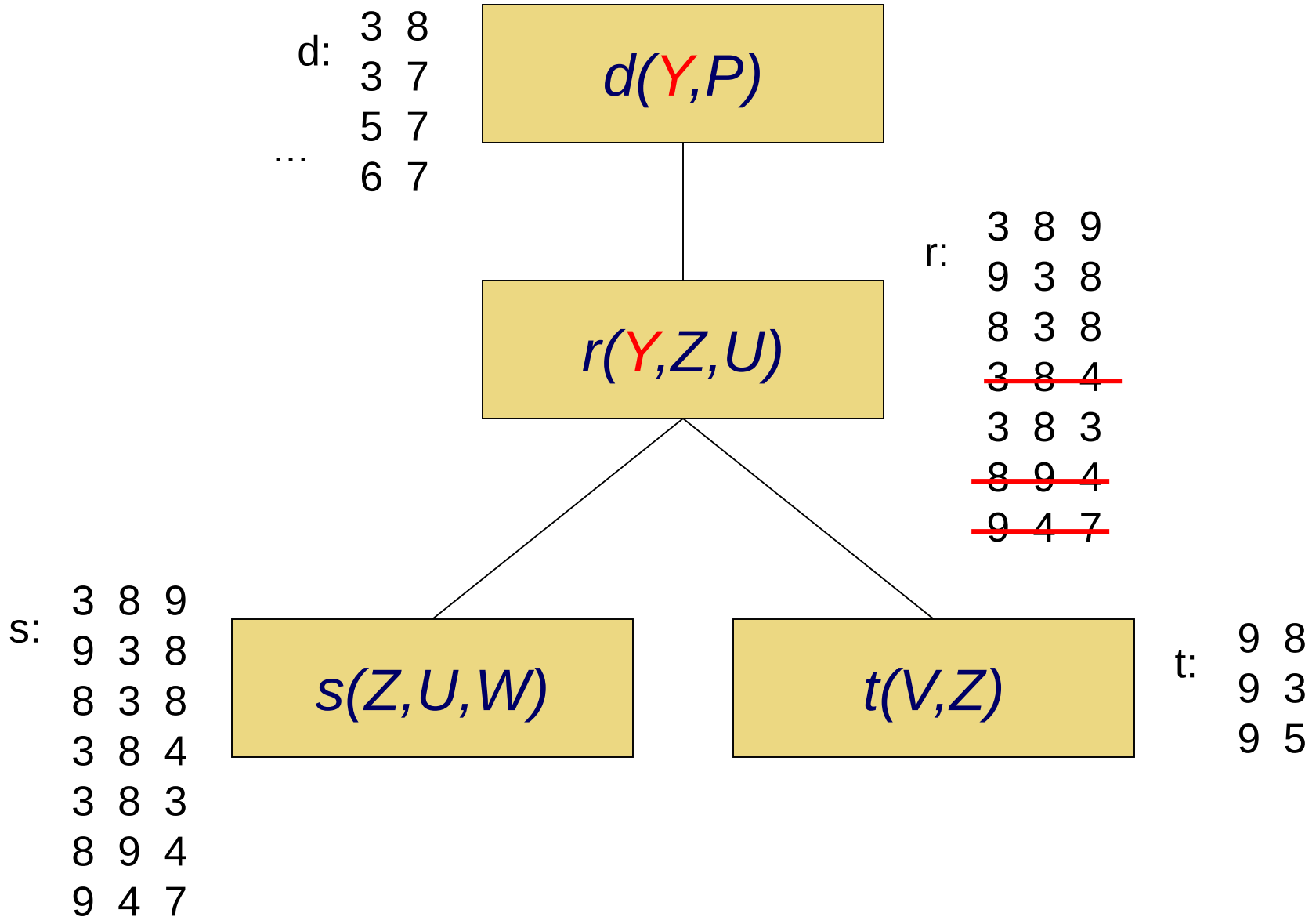

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

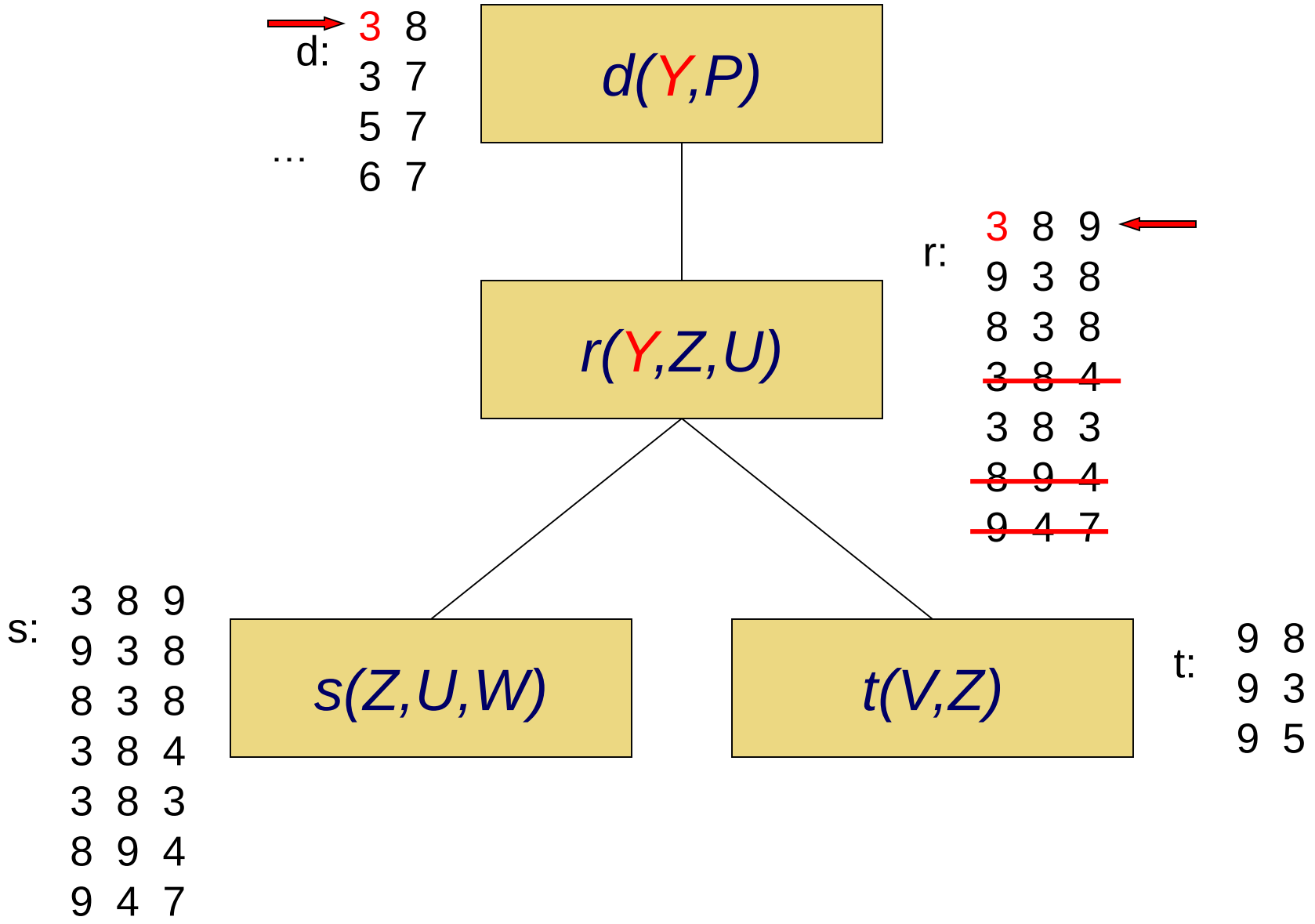

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

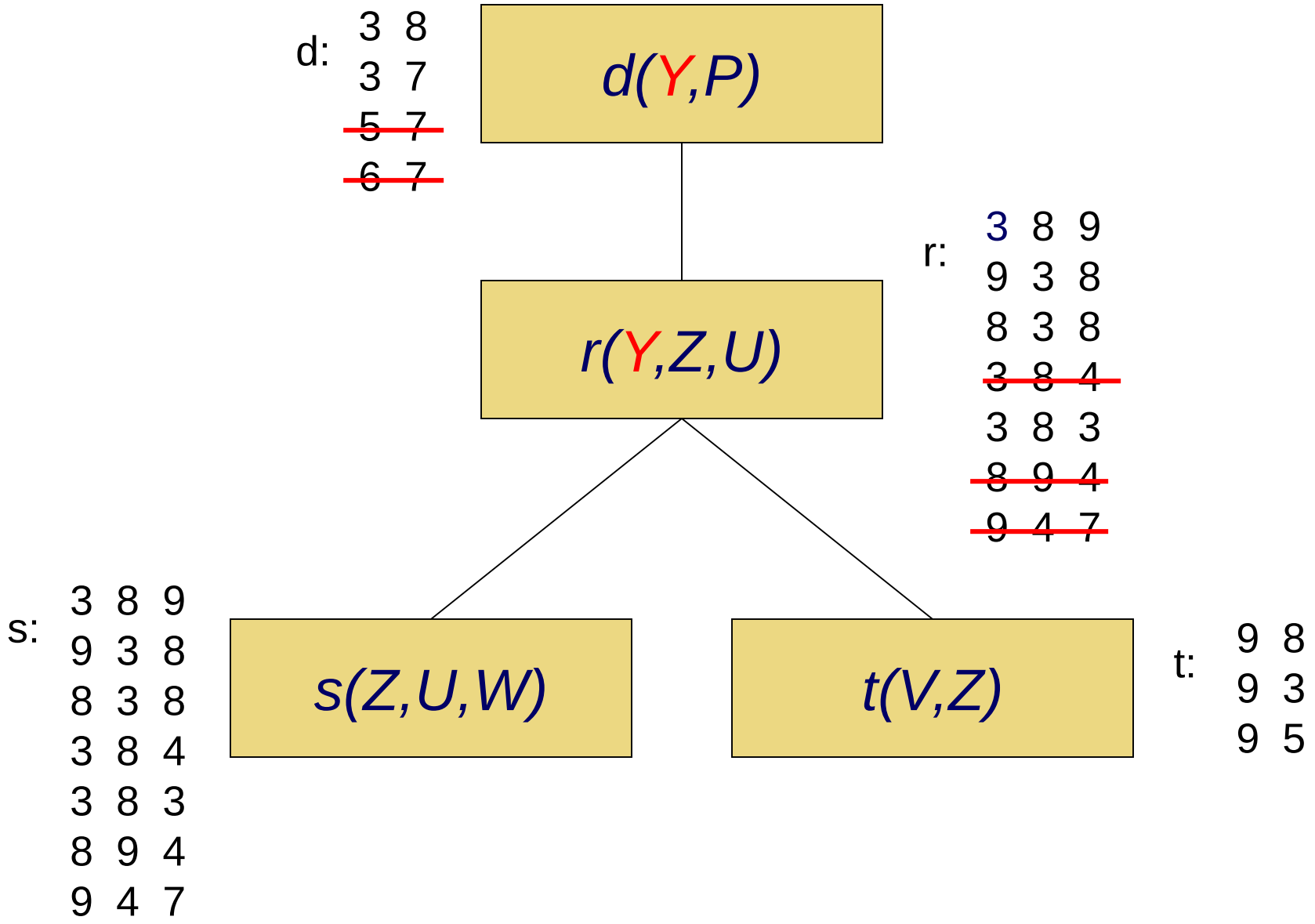

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
…
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5


d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5


d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
...
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5


d:
3 8
3 7
5 7
6 7
…
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5


d:
3 8
3 7
5 7
6 7
...
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5


d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

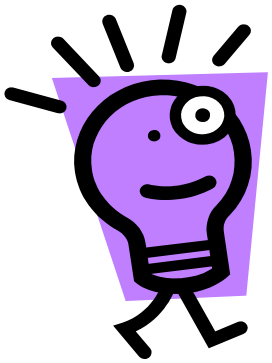
«Answering» Acyclic Instances

HOM: The homomorphism problem

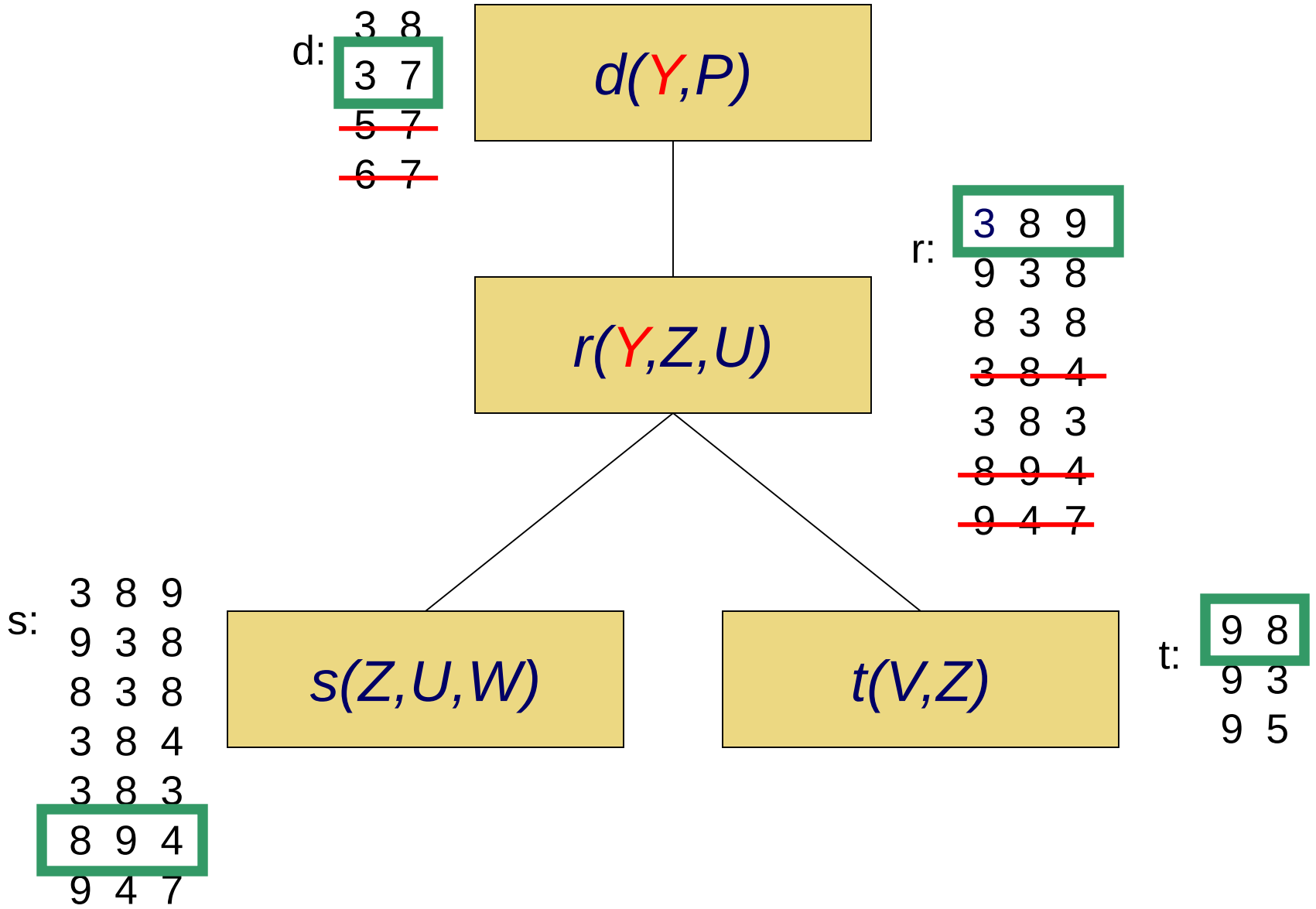
BCQ: Boolean conjunctive query evaluation

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



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

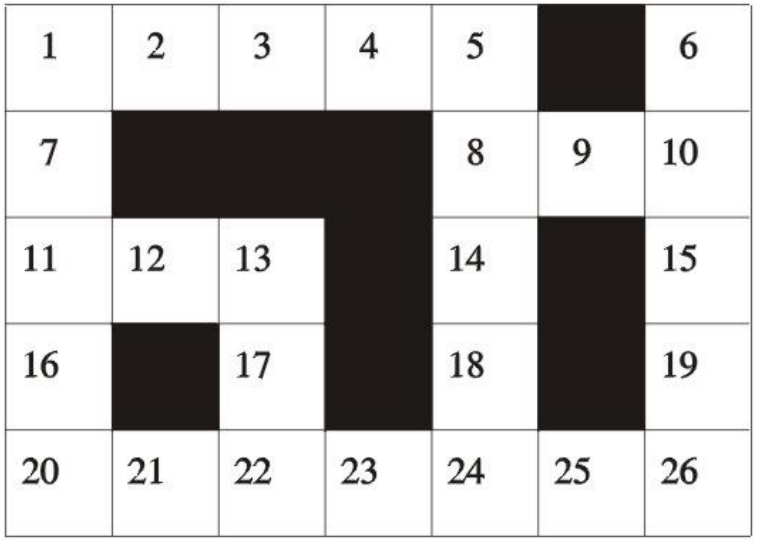
Outline of PART II

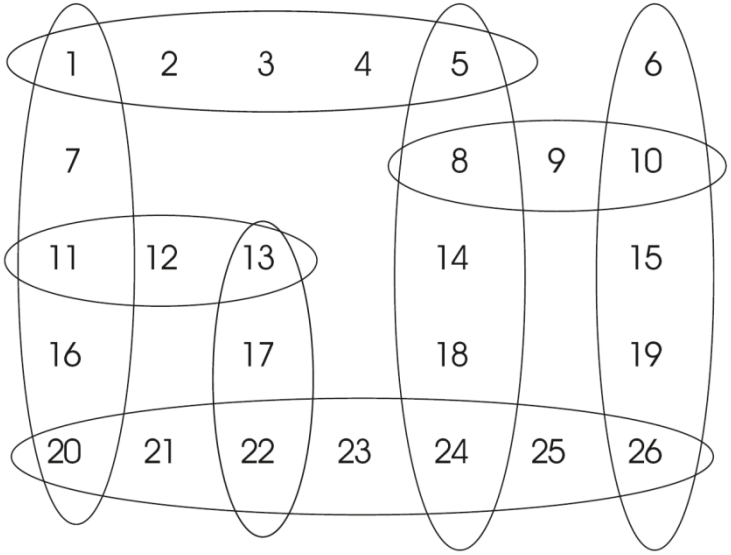
Beyond Tree Decompositions

Applications to Databases and CSPs

Structural and Consistency Properties

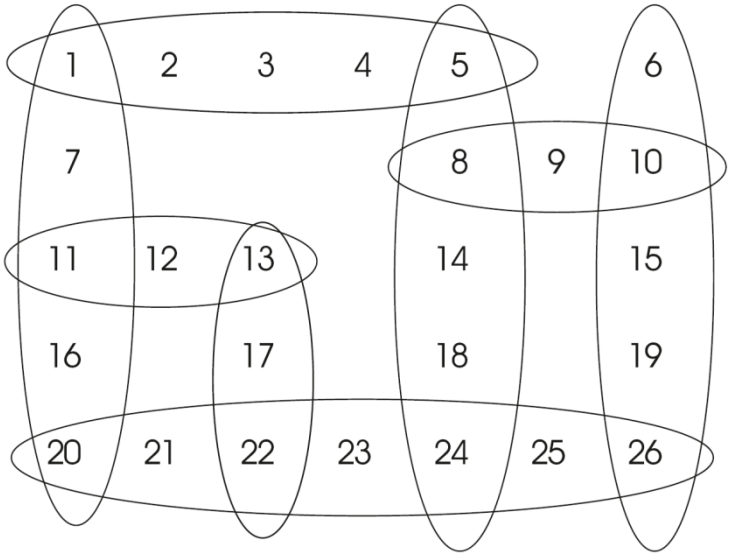
Decomposition Methods





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

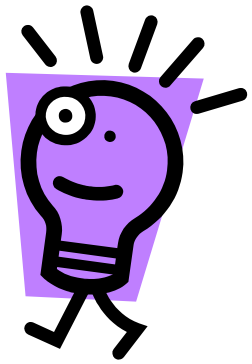
Decomposition Methods



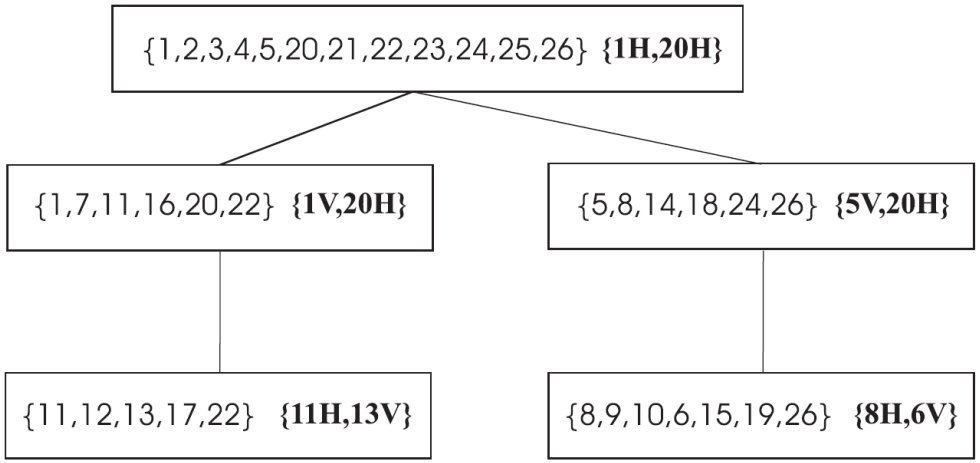
$\mathcal{H}_{\mathbb{A}}$

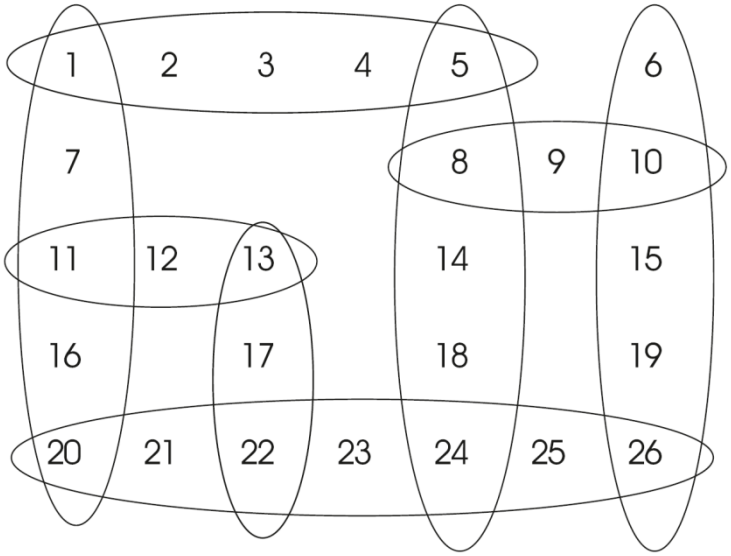
Transform the hypergraph into an acyclic one:

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



(Generalized) Hypertree Decompositions

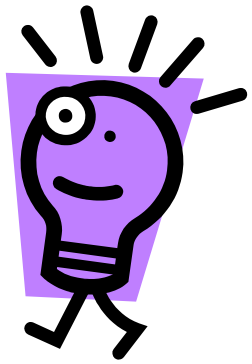




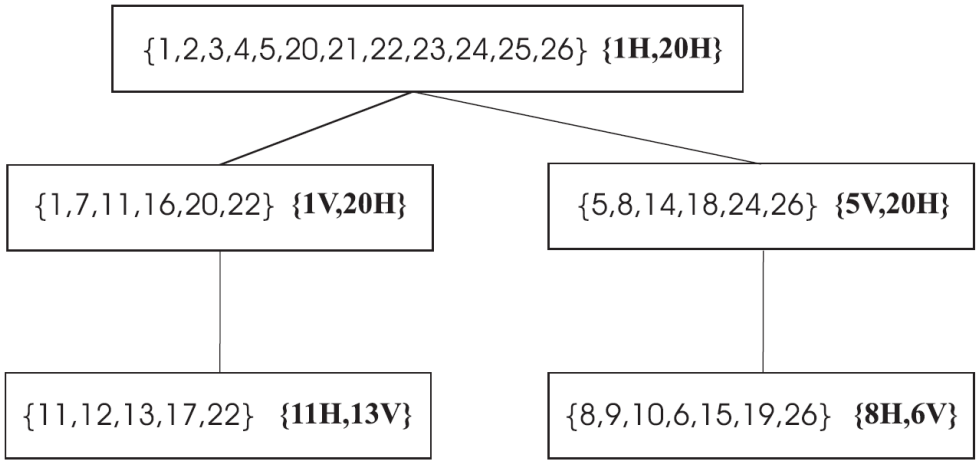
$\mathcal{H}_{\mathbb{A}}$

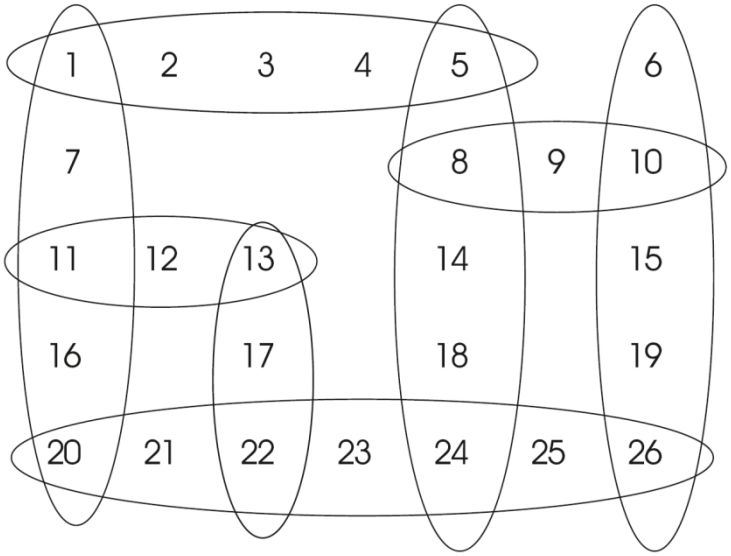
Transform the hypergraph into an acyclic one:

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



(Generalized) Hypertree Decompositions



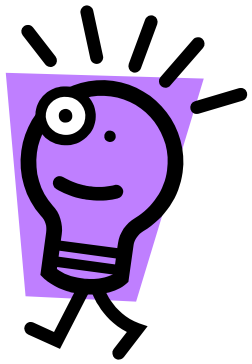


$\mathcal{H}_{\mathbb{A}}$

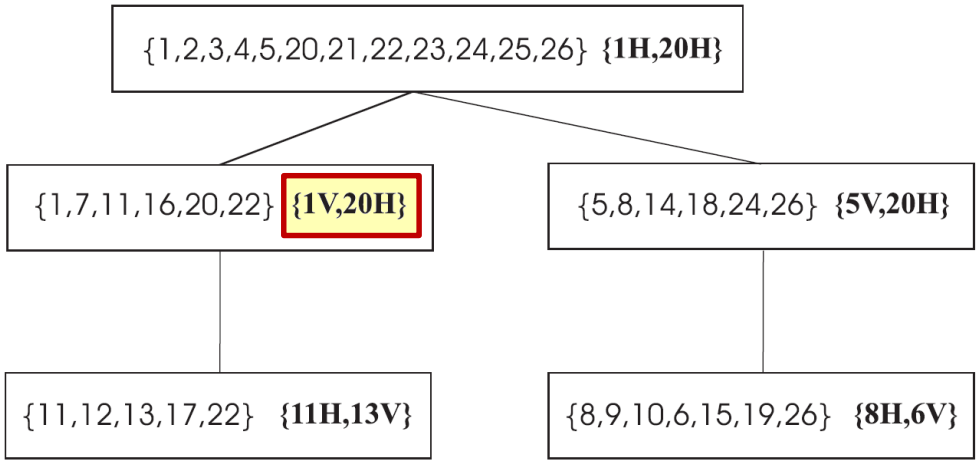
Each cluster can be seen as a **subproblem**

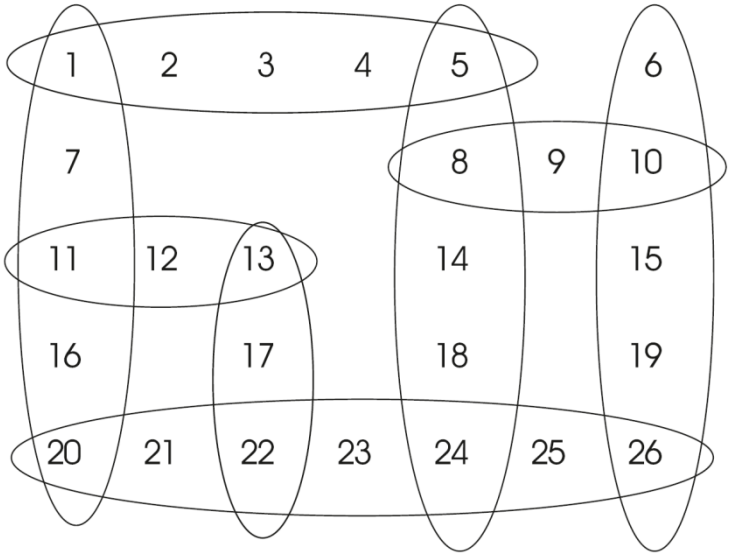
Transform the hypergraph into an acyclic one:

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



(Generalized) Hypertree Decompositions

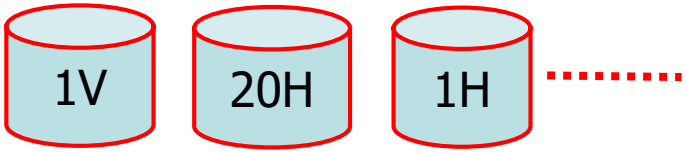




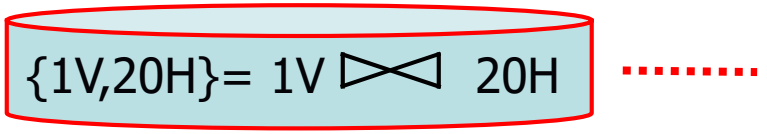
$\mathcal{H}_{\mathbb{A}}$

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

Relations:



Relations:



Outline of PART II

Beyond Tree Decompositions

Applications to Databases and CSPs

Structural and Consistency Properties

Optimization Problems

Outline of Part III

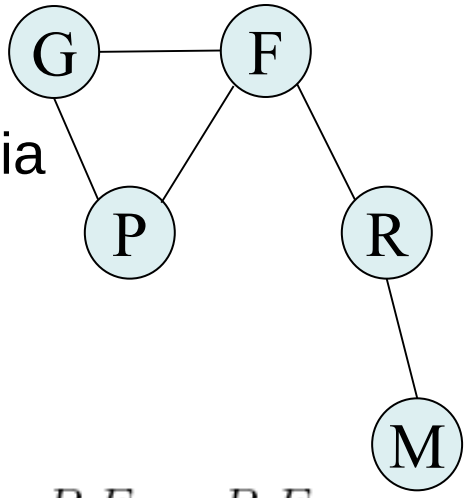
- Application: Nash Equilibria
- Application: Core
- Application: Shapley Value
- Application: Combinatorial Auctions

Outline of Part III

- **Application: Nash Equilibria**
- Application: Nucleolus
- Application: Shapley Value
- Application: Combinatorial Auctions

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

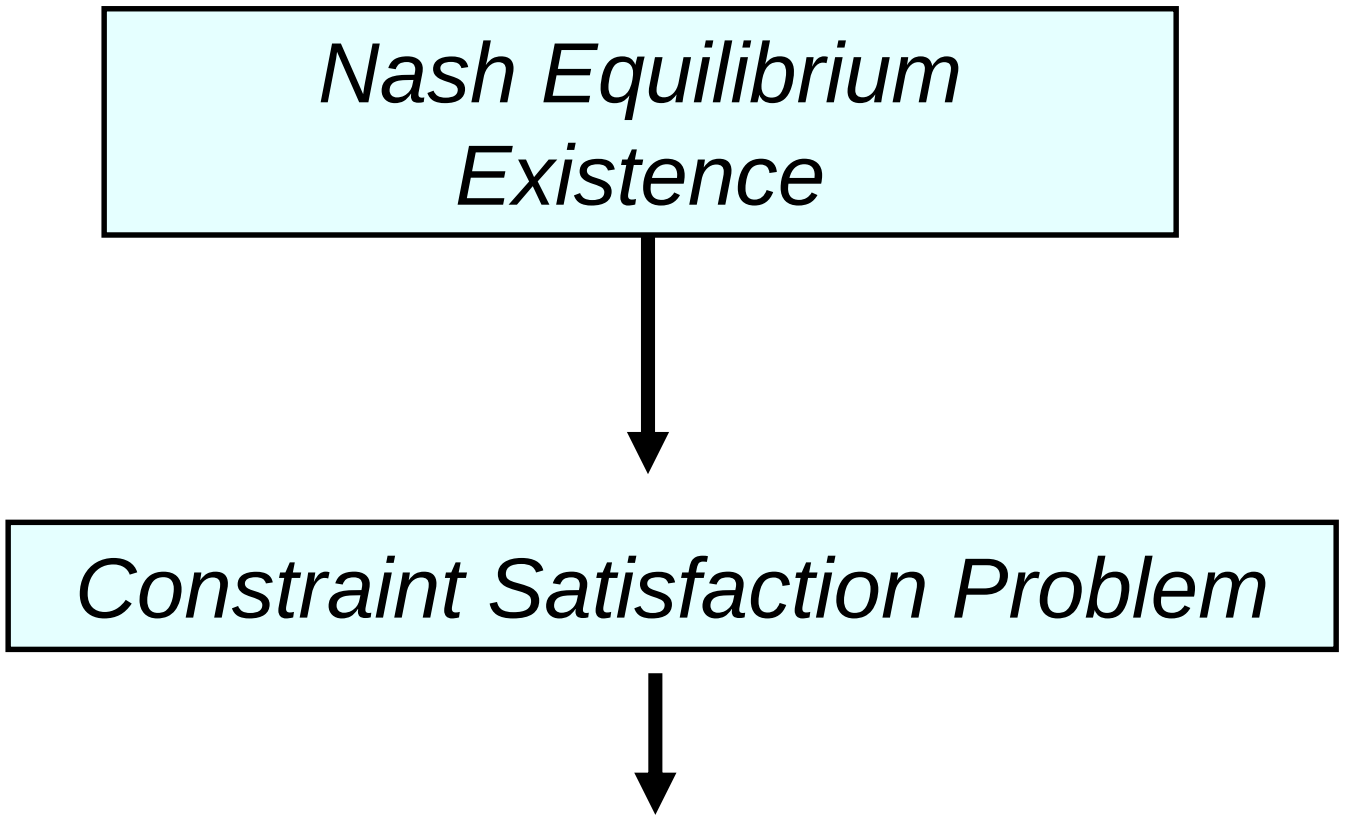
G	P_mF_m	P_mF_o	P_oF_m	P_oF_o
m	2	0	0	1
o	2	0	0	1

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



[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
o	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
o	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	o
m	o	m
o	o	m
o	o	o

r_G :

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

r_R :

R	F
o	m
m	o

r_P :

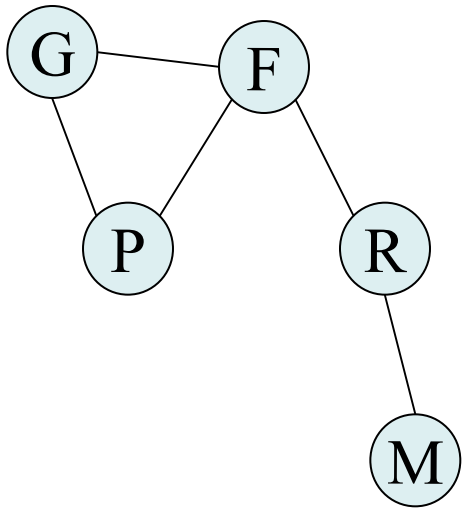
P	F
m	m
o	o

r_M :

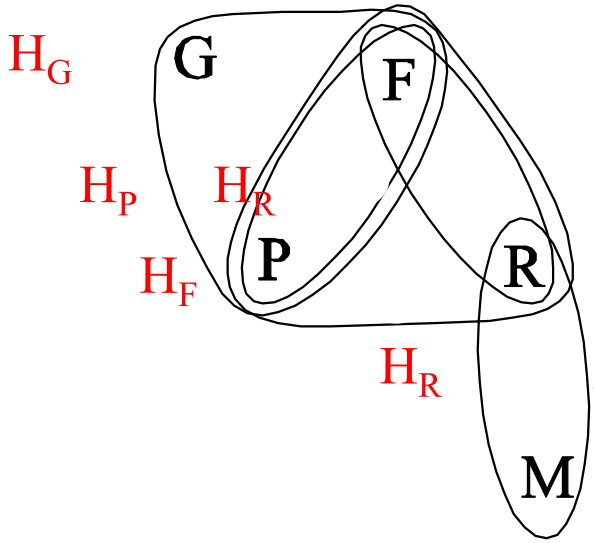
M	R
m	m
o	o

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}



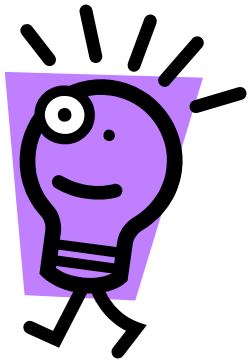
G(*FRIENDS*)

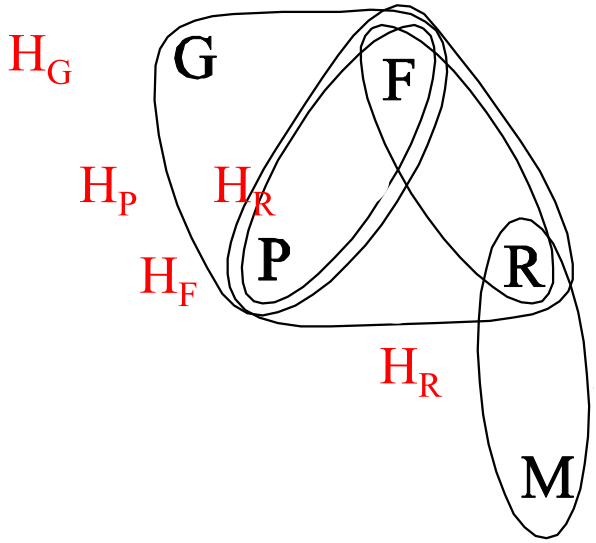


H(*FRIENDS*)

Interaction Among Players: Friends

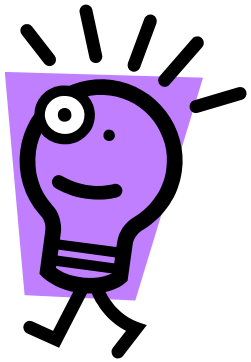
This is the same structure as the one of the associated CSP





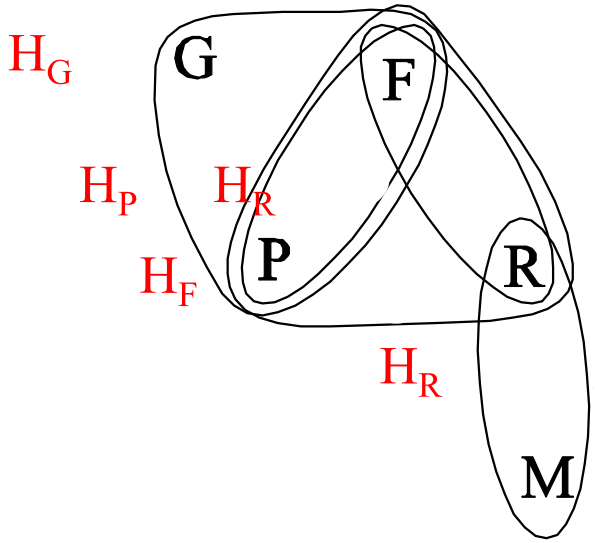
H(*FRIENDS*)

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

Application: Nash Equilibria

Application: Core

Application: Shapley Value

Application: Combinatorial Auctions

The Model

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

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

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

$$x \in X(\mathcal{G}) \begin{cases} \text{• } \textit{Efficiency} \\ \qquad x(N) = v(N) \\ \text{• } \textit{Individual Rationality} \\ \qquad x_i \geq v(\{i\}), \quad \forall i \in N \end{cases}$$

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

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

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

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

Membership in the Core on Graph Games

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

Consider the following 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 following sentence,
over the graph where *N is the set of nodes and E the set of edges* :

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

…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: $w_E(c, c') = -w(c, c');$ $w_N(c) = x_c$

Value of the edge (negated) — Value at the imputation

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

Membership in the Core on Graph Games

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

$0 \geq v(S) - \sum_{i \in S} x_i$

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)

Outline of Part III

Application: Nash Equilibria

Application: Core

Application: Shapley Value

Application: Combinatorial Auctions

Solution Concepts

- Shapley Value

$$\phi_i(\mathcal{G}) = \sum_{C \subseteq N \setminus \{i\}} \frac{|C|!(n - |C| - 1)!}{n!} \Big(v(C \cup \{i\}) - v(C) \Big)$$

- Banzhaf value

$$\beta_i(\mathcal{G}) = \frac{1}{2^{n-1}} \sum_{C \subseteq N \setminus \{i\}} \Big(v(C \cup \{i\}) - v(C) \Big)$$

Solution Concepts

- Shapley Value

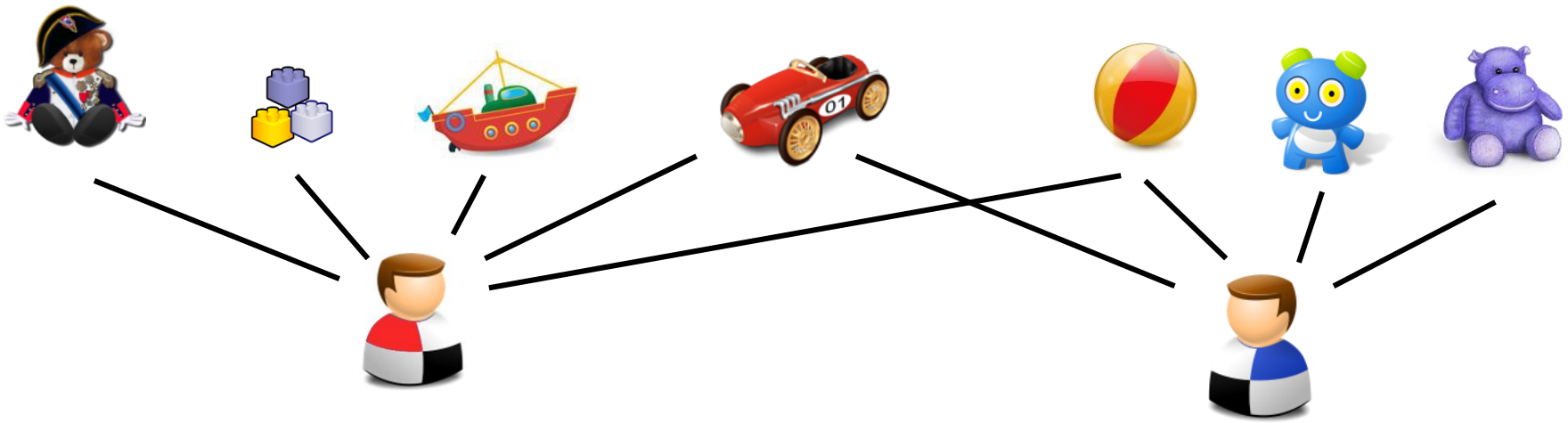
$$\phi_i(\mathcal{G}) = \sum_{C \subseteq N \setminus \{i\}} \frac{|C|!(n - |C| - 1)!}{n!} \Big(v(C \cup \{i\}) - v(C) \Big)$$

Marginal Contribution

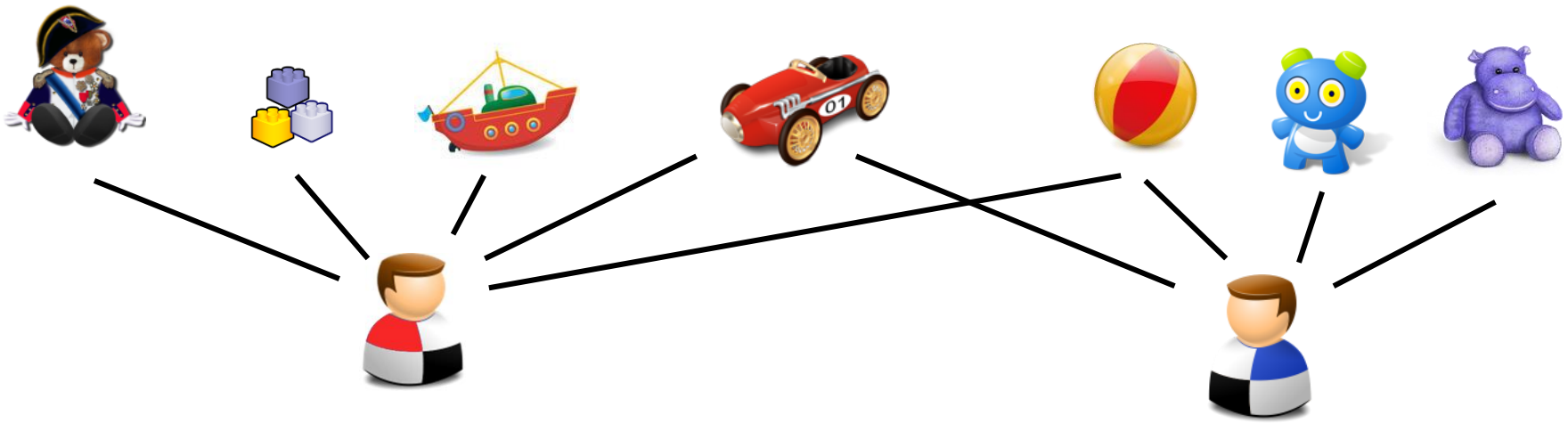
- Banzhaf value

$$\beta_i(\mathcal{G}) = \frac{1}{2^{n-1}} \sum_{C \subseteq N \setminus \{i\}} \Big(v(C \cup \{i\}) - v(C) \Big)$$

Allocation Problems

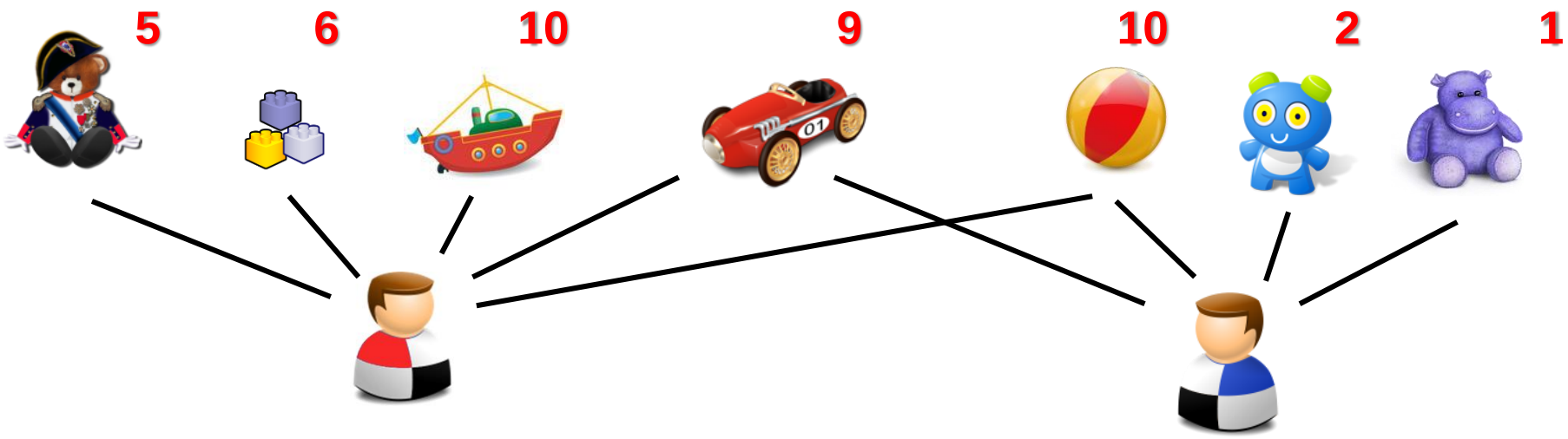


Allocation Problems



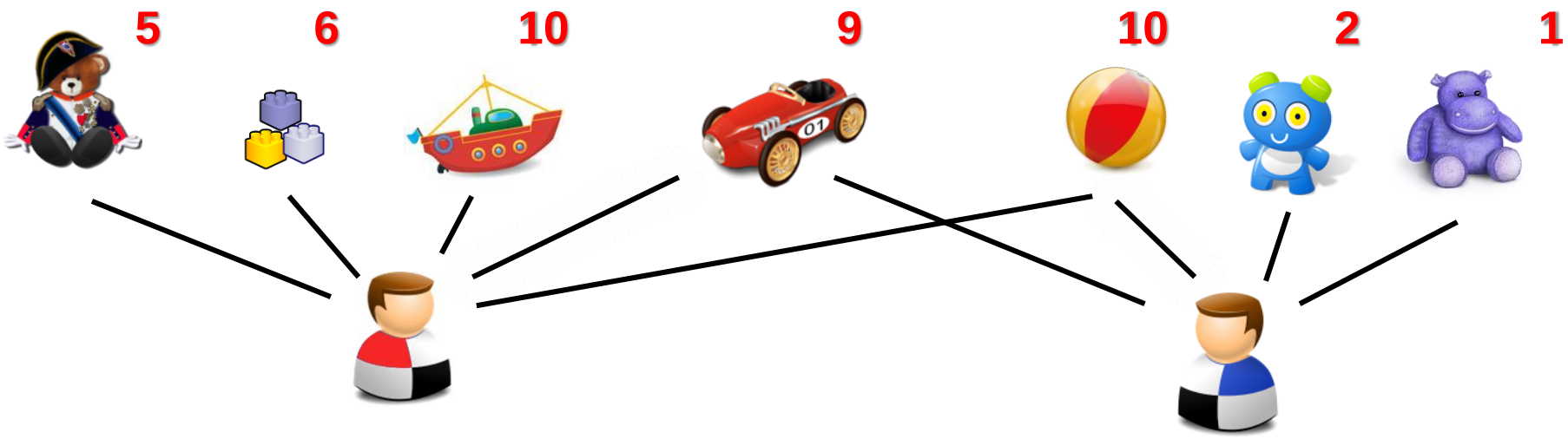
- Goods are indivisible and non-sharable

Allocation Problems



- Goods are indivisible and non-sharable
- Cardinal valuations

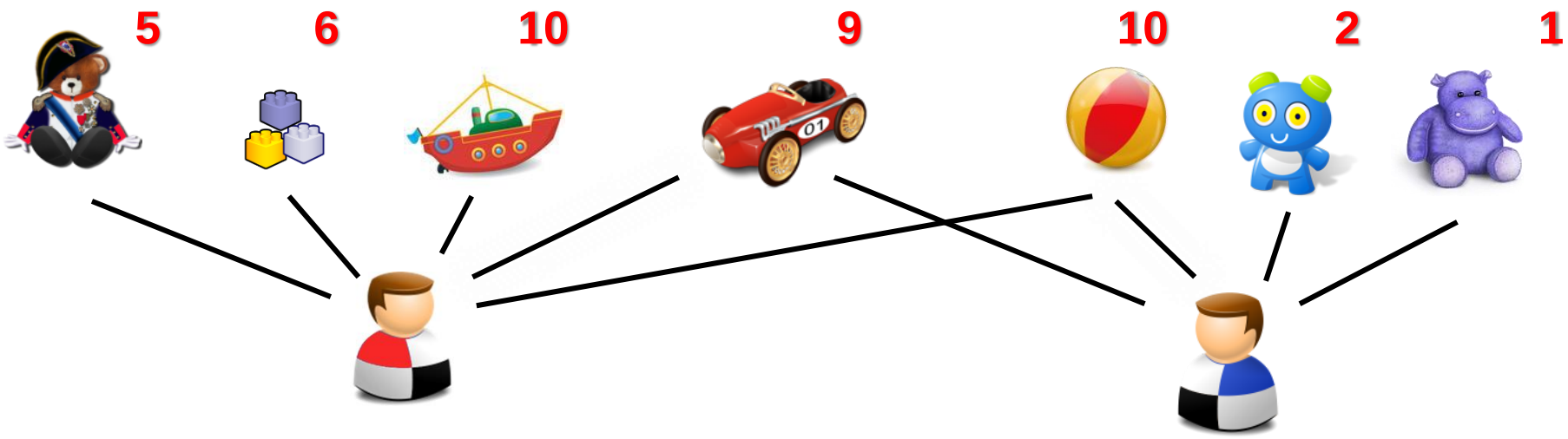
Allocation Problems



- Goods are indivisible and non-sharable
- Cardinal valuations
- Each agent/player can receive one good at most

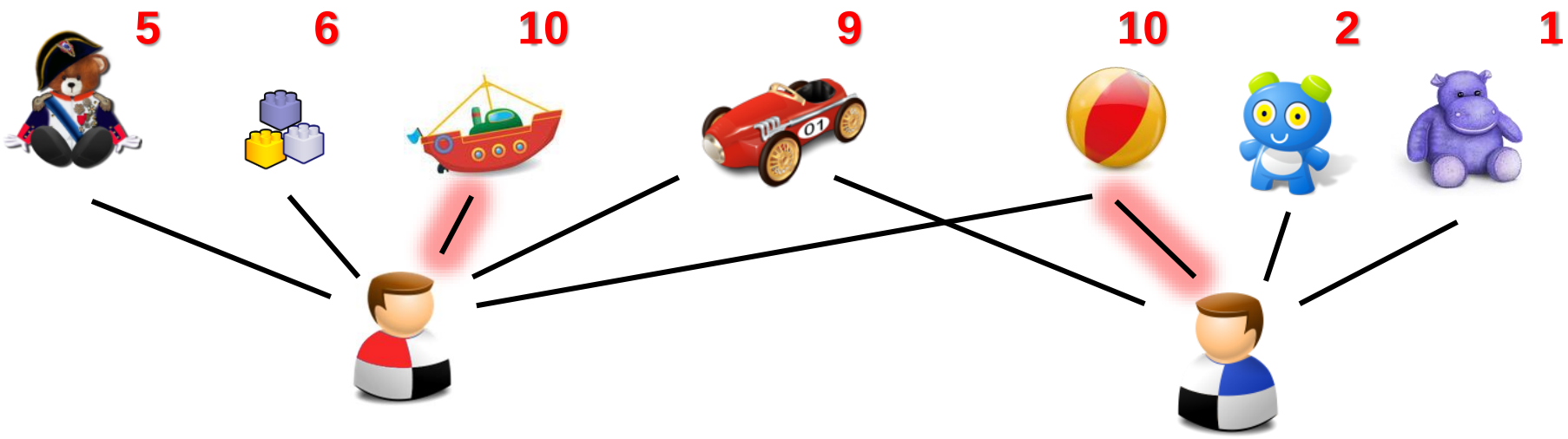
Allocation Problems

GOAL: Optimal Allocations



Allocation Problems

GOAL: Optimal Allocations



Putting It All Toghether

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

- For each coalition C, the associated worth is given by the value of the best allocation focusing on players in C only

Putting It All Toghether

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

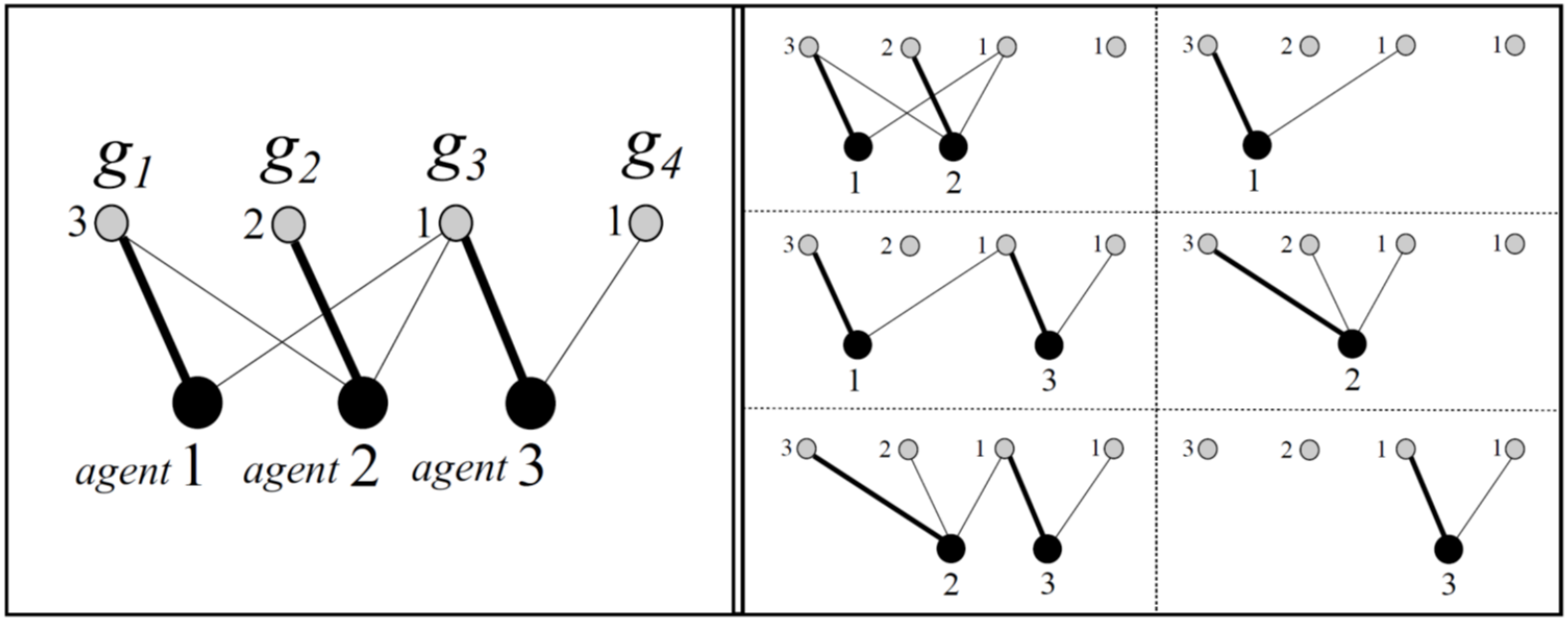
- For each coalition C, the associated worth is given by the value of the best allocation focusing on players in C only

- Framework to analyze fair division problems [Moulin, 1992]
  - Monetary compensations
  - Quasi-linear utilities

Putting It All Toghether

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

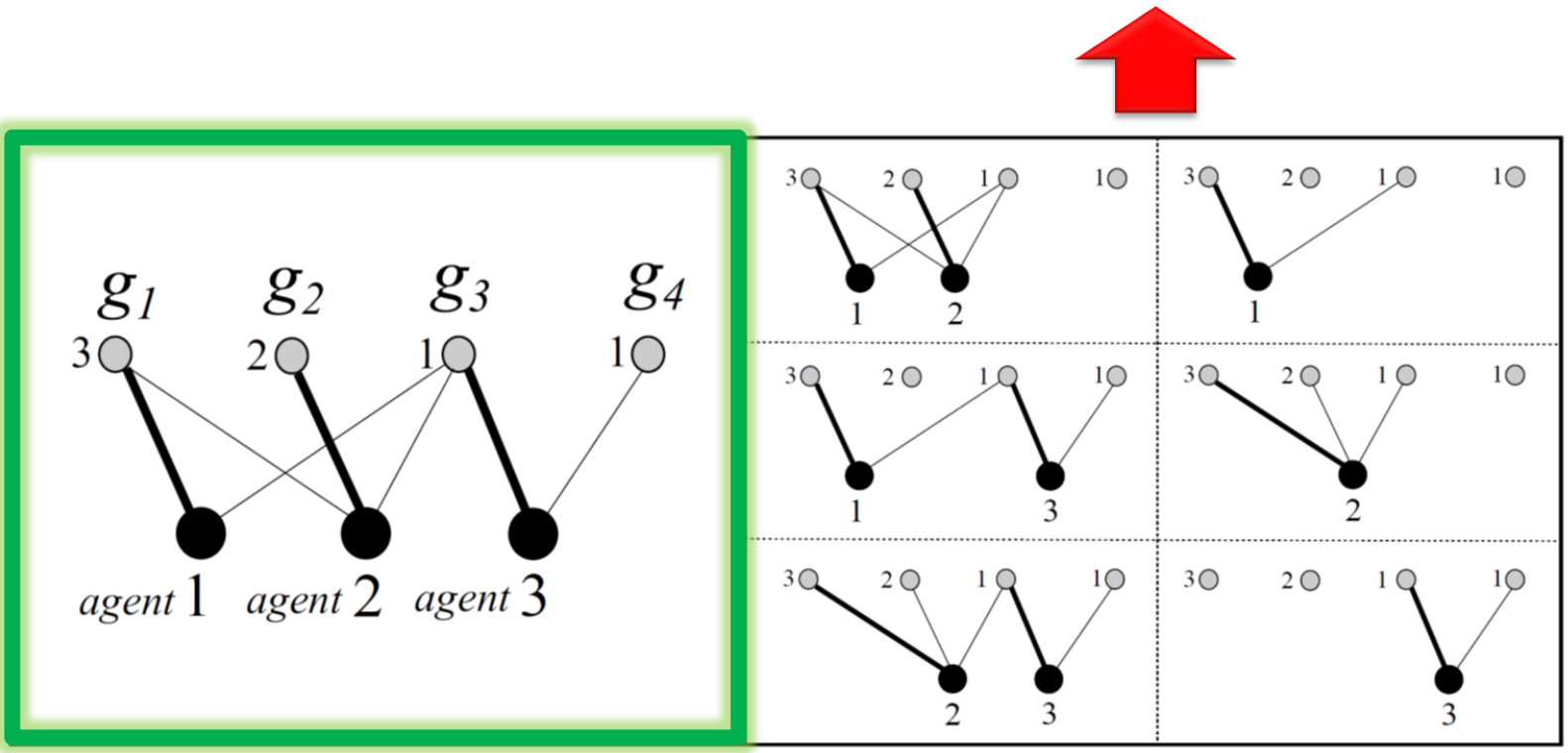
- For each coalition C, the associated worth is given by the value of the best allocation focusing on players in C only



Putting It All Toghether

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

exponentially-many coalitions

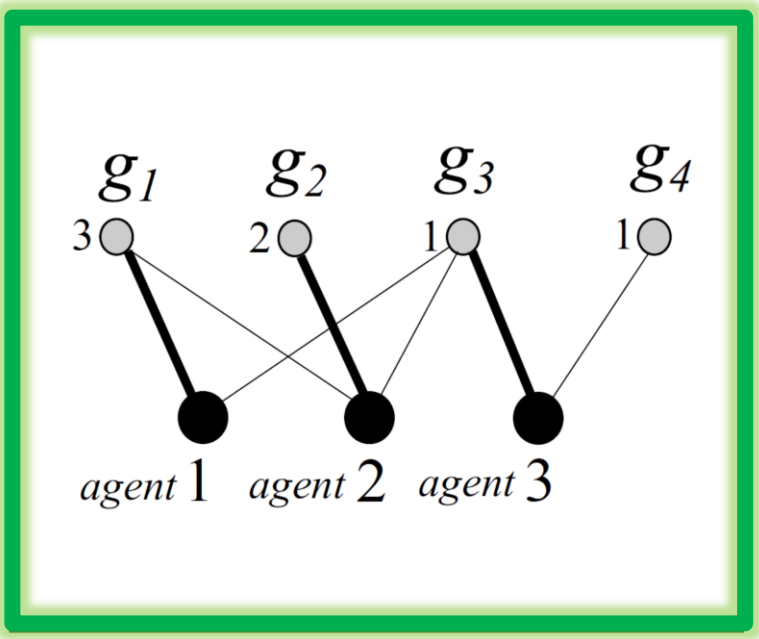


The encoding is «compact»

Putting It All Toghether

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

Given this INPUT, what about the complexity of computing the Shapley/Banzhaf value?



The encoding is «compact»

Complexity Issues

- For many classes of «compact games» (e.g., *graph games*), Shapley/Banzhaf-values can be efficiently calculated
- Here, the problem emerges to be #P-complete

Even if all goods have the same value!

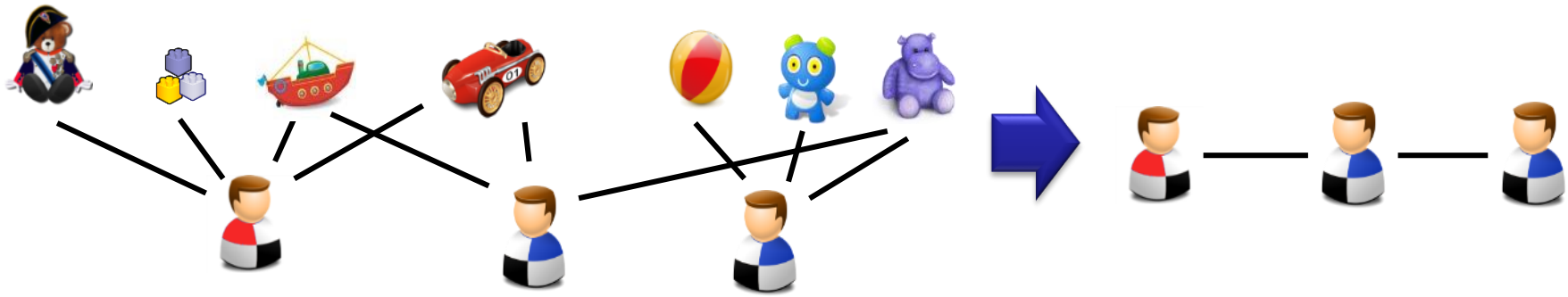
Hardness of the Banzhaf value:

- Reduction from a counting problem related to matching [Coulbourn et al, 1995]

Hardness of the Shapley value:

- The Banzhaf value can be computed in polynomial time given the Shapley value
- Adaptation of an algorithm by [Aziz et al, 2009] originally given for *simple* games

Bounded Interactions



- Interaction graph
  - There is an edge between any pair of agents competing for the same good

The Shapley value can be computed in polynomial time whenever the interaction graph is a tree.

Proof Idea: Ingredient 1

$$\phi_i(\mathcal{G}) = \sum_{C \subseteq N \setminus \{i\}} \frac{|C|!(n - |C| - 1)!}{n!} \Big(v(C \cup \{i\}) - v(C) \Big)$$

Proof Idea: Ingredient 1

$$\phi_i(\mathcal{G}) = \sum_{C \subseteq N \setminus \{i\}} \frac{|C|!(n - |C| - 1)!}{n!} \Big(v(C \cup \{i\}) - v(C) \Big)$$

- For each agent *i*, for each size *h* of coalition, and for each possible value for goods *w*

Define ***#coal(i,h,w)*** *as the number of coalitions* C *such that* $|C| = h$ *and*

$$v(C \cup \{i\}) - v(C) \geq w$$

Proof Idea: Ingredient 1

$$\phi_i(\mathcal{G}) = \sum_{C \subseteq N \setminus \{i\}} \frac{|C|!(n - |C| - 1)!}{n!} \Big(v(C \cup \{i\}) - v(C) \Big)$$

- For each agent i, for each size h of coalition, and for each possible value for goods w

Define ***#coal(i,h,w)*** *as the number of coalitions* C *such that* $|C| = h$ *and*

$$v(C \cup \{i\}) - v(C) \geq w$$

Algebraic manipulations

- The Shapley value can be written as a weigthed combination over ***#coal(i,h,w)***
- **Polynomially-many** terms are involved

Proof Idea: Ingredient 2

Define ***#coal(i,h,w)*** *as the number of coalitions* C *such that* $|C| = h$ *and*

$$v(C \cup \{i\}) - v(C) \geq w$$

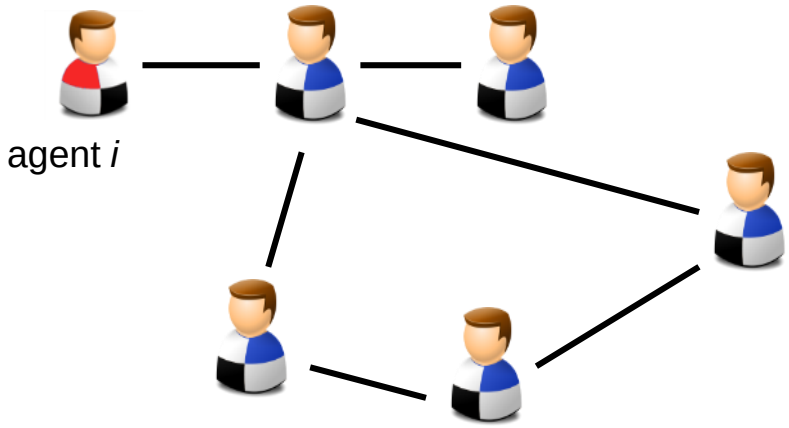
- The marginal contribution can be characterized via the existence of an allocation with certain properties

Proof Idea: Ingredient 2

Define ***#coal(i,h,w)*** *as the number of coalitions* C *such that* $|C| = h$ *and*

$$v(C \cup \{i\}) - v(C) \geq w$$

- The marginal contribution can be characterized via the existence of an allocation with certain properties



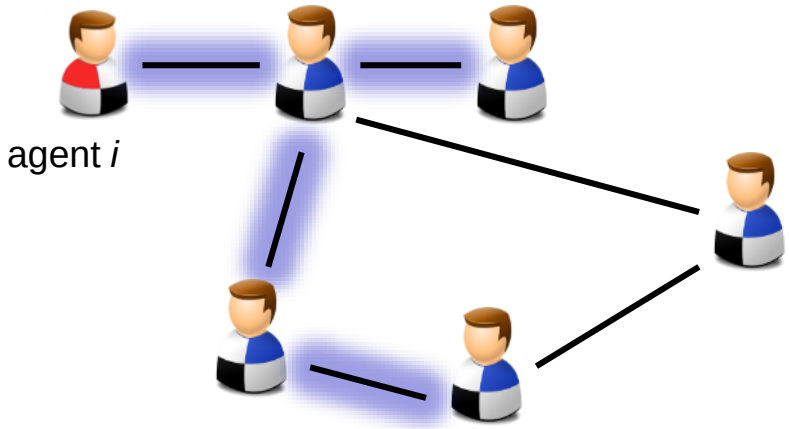
interaction graph

Proof Idea: Ingredient 2

Define ***#coal(i,h,w)*** *as the number of coalitions* C *such that* $|C| = h$ *and*

$$v(C \cup \{i\}) - v(C) \geq w$$

- The marginal contribution can be characterized via the existence of an allocation with certain properties



interaction graph

✓ *restricted w.r.t* w

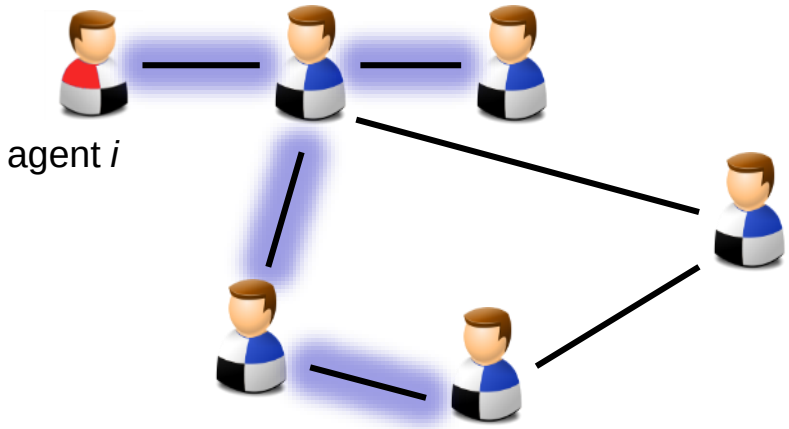
Proof Idea: Ingredient 2

Define ***#coal(i,h,w)*** *as the number of coalitions* C *such that* $|C| = h$ *and*

$$v(C \cup \{i\}) - v(C) \geq w$$

‖

there is an allocation in the induced scenario where each agent gets a good



interaction graph

✓ *restricted w.r.t* w

Proof Idea: Ingredient 2

Define ***#coal(i,h,w)*** *as the number of coalitions* C *such that* $|C| = h$ *and*

$$v(C \cup \{i\}) - v(C) \geq w$$

=

there is an allocation in the induced scenario where each agent gets a good

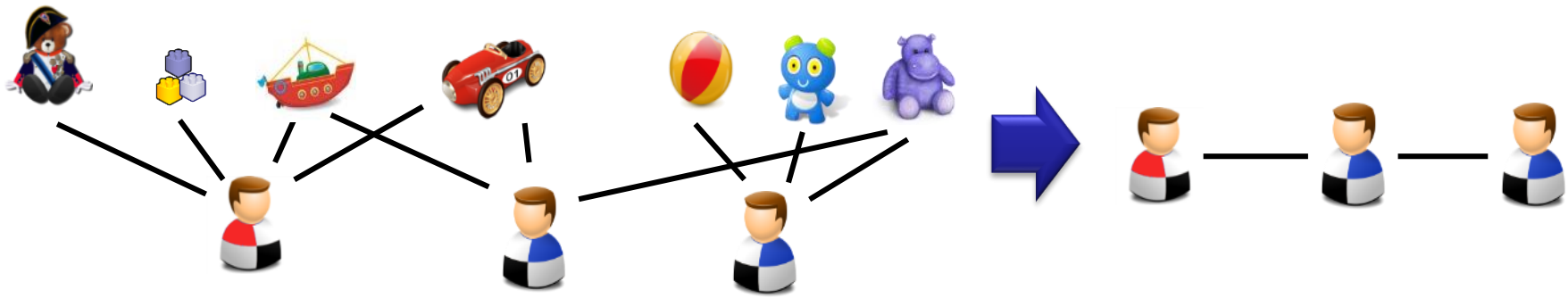
- The problem reduces to counting the number of coalitions with size *h* for which each agent can get a good

Proof Idea: Ingredient 3

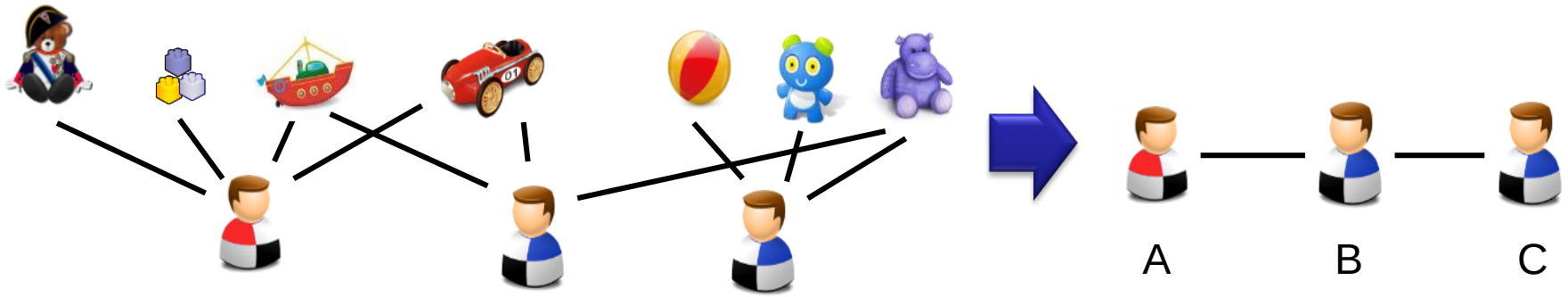
Encode as a CSP

- The problem reduces to counting the number of coalitions with size *h* for which each agent can get a good

Example Encoding



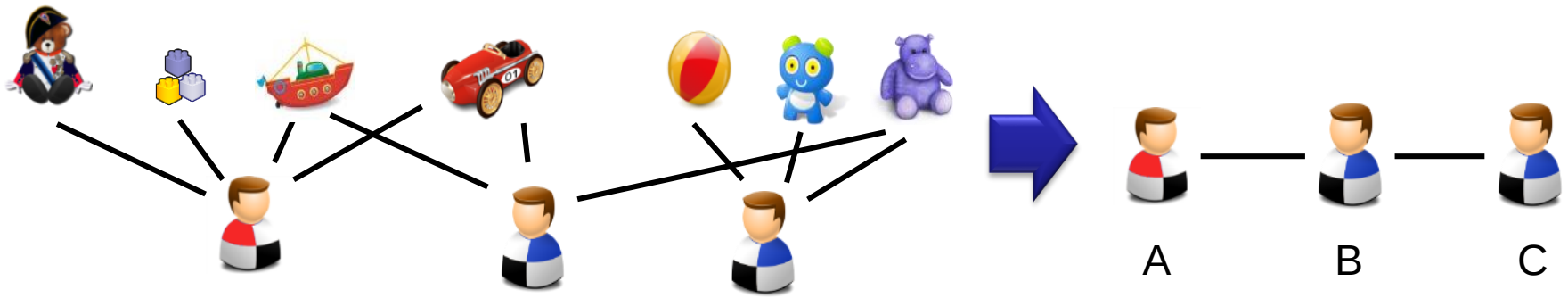
Example Encoding



- **Variables:**
  - *Agent* A, *agent* B, and *agent* C + variables IN_A, IN_B, IN_C

boolean: {true, false}

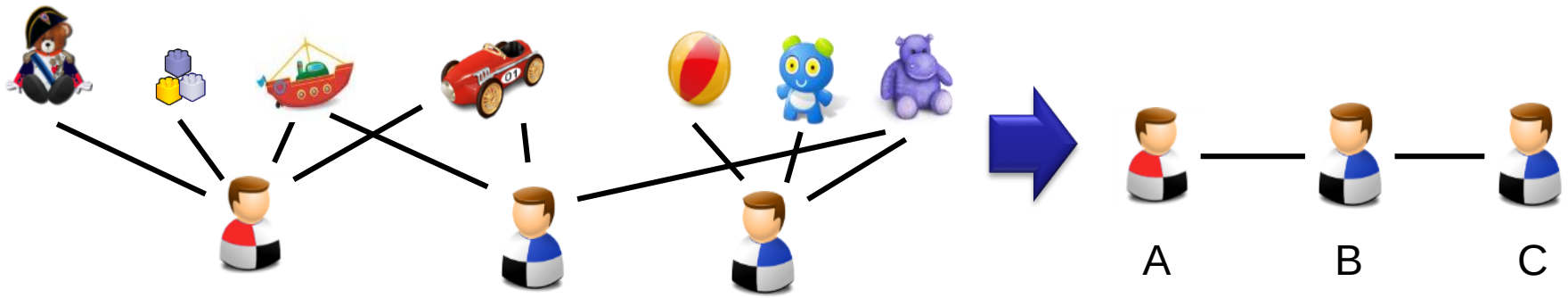
Example Encoding



- **Variables:**
  - *Agent* A, *agent* B, and *agent* C + variables IN_A, IN_B, IN_C
- **Domain:**
  - **D**(A) =
  - **D**(B) =
  - **D**(C) =

boolean: {true, false}

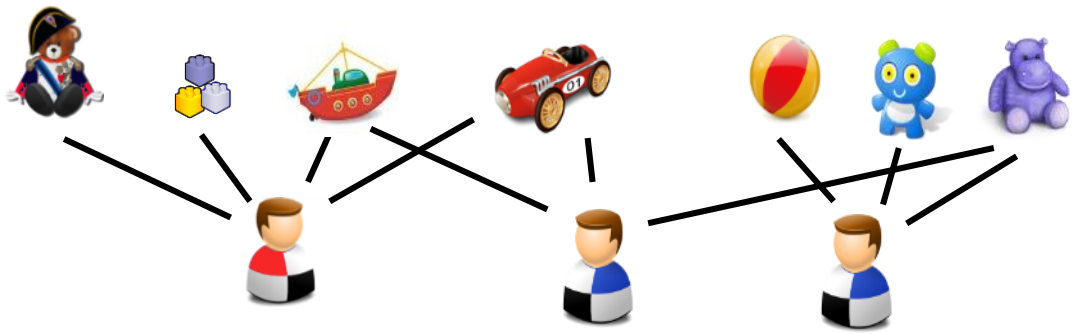
Example Encoding

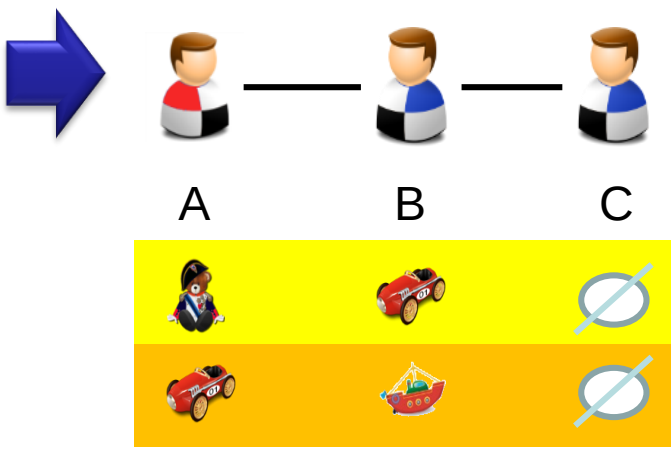


- **Variables:**
  - *Agent* A, *agent* B, and *agent* C + variables IN_A, IN_B, IN_C
- **Domain:**
  - **D**(A) =
  - **D**(B) =
  - **D**(C) =
- **Constraints:**
  - If goods are assigned, then A≠B; B≠C;
  - X= ∅ if, and only if, IN_X=false

boolean: {true, false}

Example Encoding

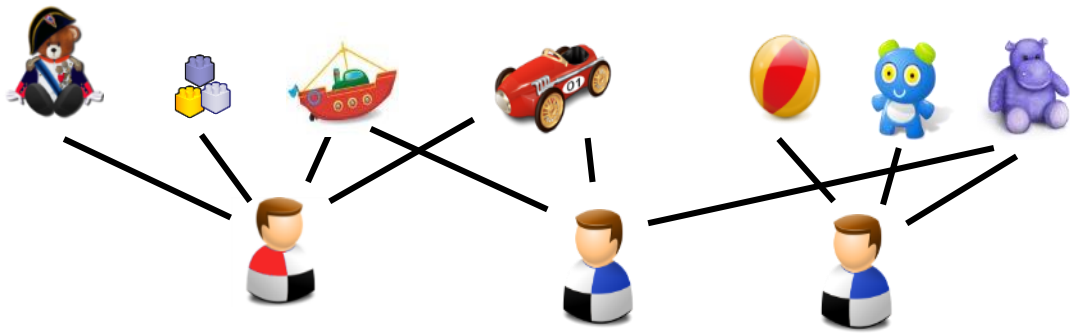


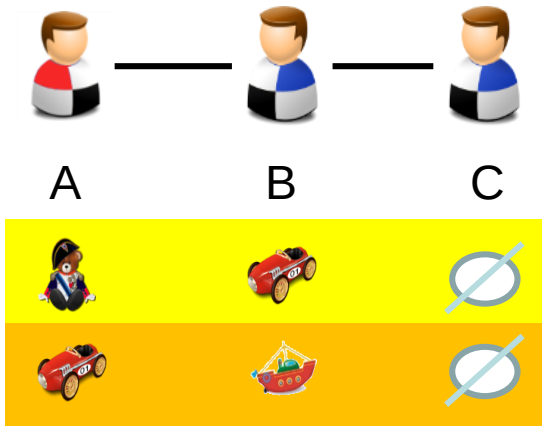


$IN_A = IN_B = true$

$IN_C = false$

Example Encoding



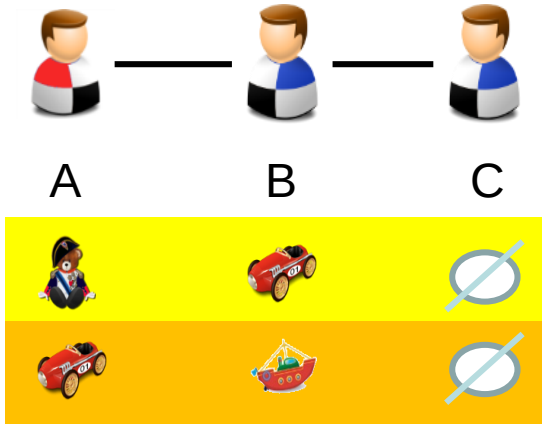


$IN_A = IN_B = true$
$IN_C = false$

- The problem reduces to counting the number of coalitions with size *h* for which each agent can get a good

Example Encoding

Count the number of solutions **projected** over the variables IN_A, IN_B, IN_C



$IN_A = IN_B = true$
$IN_C = false$

- The problem reduces to counting the number of coalitions with size *h* for which each agent can get a good

Proof Idea: Ingredient 3

in «Tractability: Practical Approaches to hard Problems»
[Gottlob, Greco and Scarcello, 2013]

Structural tractability results for CSPs

- Decision problems
- Computation Problems
- **Counting?**

Encode as a CSP

- The problem reduces to counting the number of coalitions with size *h* for which each agent can get a good

Proof Idea: Ingredient 3

Structural tractability results for CSPs

- ✓ Solutions projected over a set *W* of output variables
- ✓ Variables not in *W* are auxiliary ones

- Decision problems
- Computation Problems
- **Counting?**

Theorem (cf. [Pichler and Skritek, 2013; Greco and Scarcello, 2014]). *Counting the number of substitutions in* $\Theta(\mathcal{I}, \mathcal{W})$ *is feasible in polynomial time, on classes of CSP instances* $\mathcal{I}$ *such that the treewidth of* $G(\mathcal{I})$ *is bounded by a constant, and the size of the domain of each variable not in* $\mathcal{W}$ *is bounded by some constant, too.*

Proof Idea: Ingredient 3

Structural tractability results for CSPs

- ✓ Solutions projected over a set *W* of output variables
- ✓ Variables not in *W* are auxiliary ones

- Decision problems
- Computation Problems
- **Counting?**

Theorem (cf. [Pichler and Skritek, 2013; Greco and Scarcello, 2014]). *Counting the number of substitutions in* $\Theta(\mathcal{I}, \mathcal{W})$ *is feasible in polynomial time, on classes of CSP instances* $\mathcal{I}$ *such that the treewidth of* $G(\mathcal{I})$ *is bounded by a constant, and the size of the domain of each variable not in* $\mathcal{W}$ *is bounded by some constant, too.*

For instance, we cannot use a variable to denote the allocation for an agent, since its domain would be unbounded!

Proof Idea: Ingredient 3

Structural tractability results for CSPs

- ✓ Solutions projected over a set *W* of output variables
- ✓ Variables not in *W* are auxiliary ones

- Decision problems
- Computation Problems
- **Counting?**

Theorem (cf. [Pichler and Skritek, 2013; Greco and Scarcello, 2014]). *Counting the number of substitutions in* $\Theta(\mathcal{I}, \mathcal{W})$ *is feasible in polynomial time, on classes of CSP instances* $\mathcal{I}$ *such that the treewidth of* $G(\mathcal{I})$ *is bounded by a constant, and the size of the domain of each variable not in* $\mathcal{W}$ *is bounded by some constant, too.*

For instance, we cannot use a variable to denote the allocation for an agent, since its domain would be unbounded!

Proof Idea: Ingredient 3

- **Variables:**
  - *Agent* A, *agent* B, and *agent* C
  - Boolean variables IN_A, IN_B, IN_C
  - Products are associated with Boolean variables

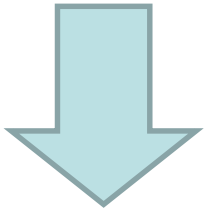
  - Further auxiliary variables

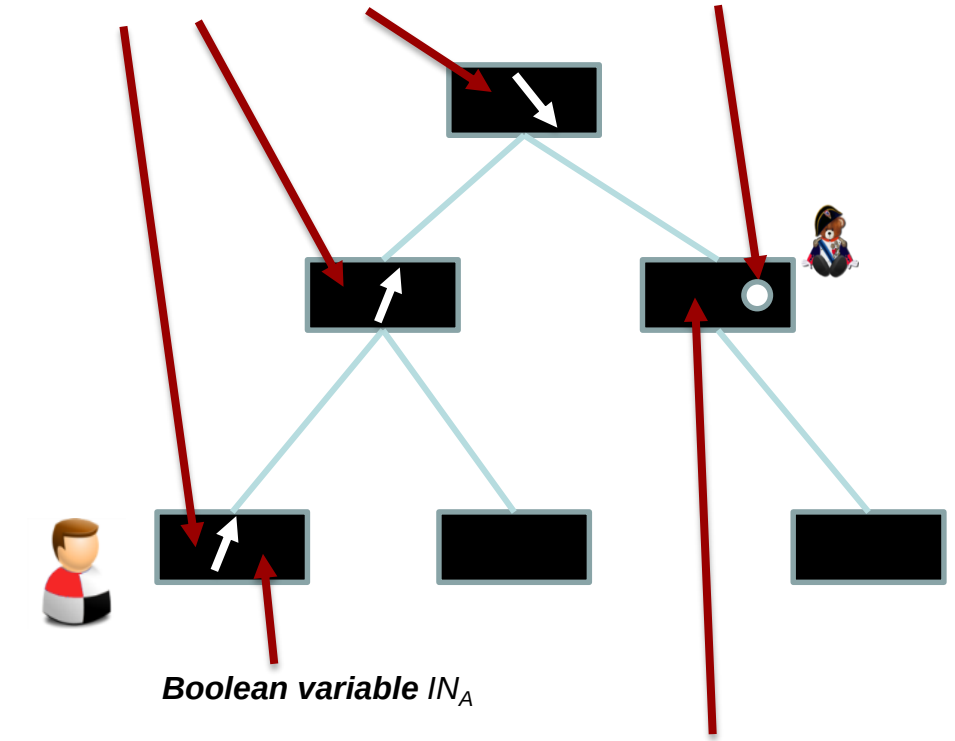
Proof Idea: Ingredient 3

- **Variables:**
  - *Agent* A, *agent* B, and *agent* C
  - Boolean variables IN_A, IN_B, IN_C
  - Products are associated with Boolean variables

  - Further auxiliary variables

From the original graph, we build a tree decomposition



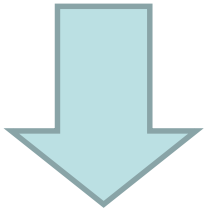


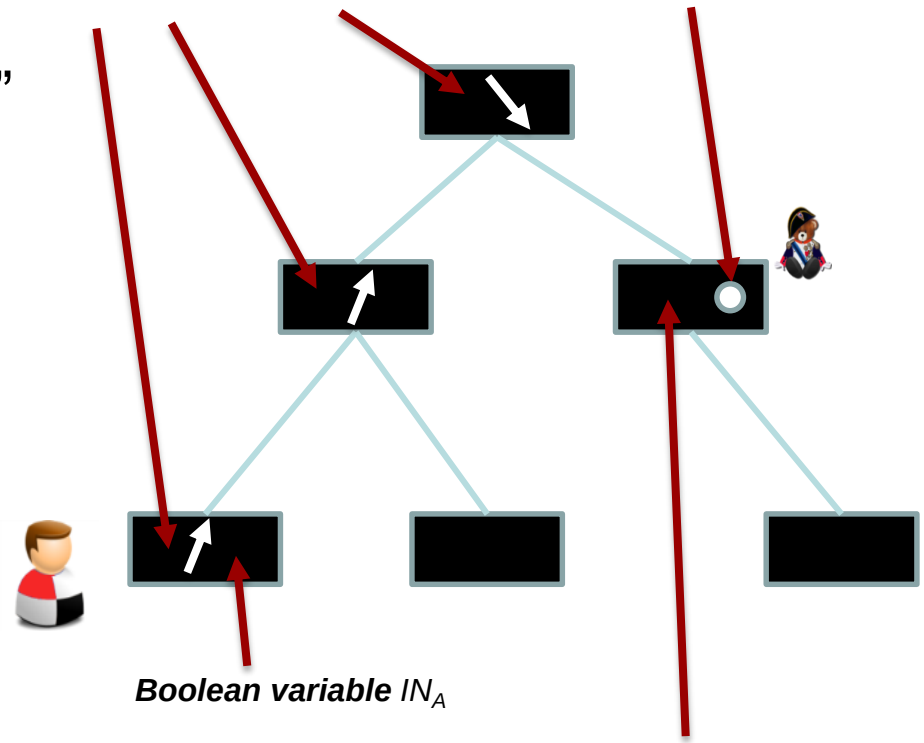
Proof Idea: Ingredient 3

- **Variables:**
  - *Agent* A, *agent* B, and *agent* C
  - Boolean variables IN_A, IN_B, IN_C
  - Products are associated with Boolean variables

  - Further auxiliary variables “R”
- **Domain:**
  - **D**(R) = {  , … }

From the original graph, we build a tree decomposition



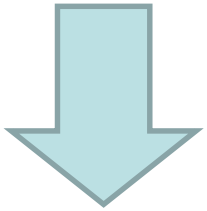


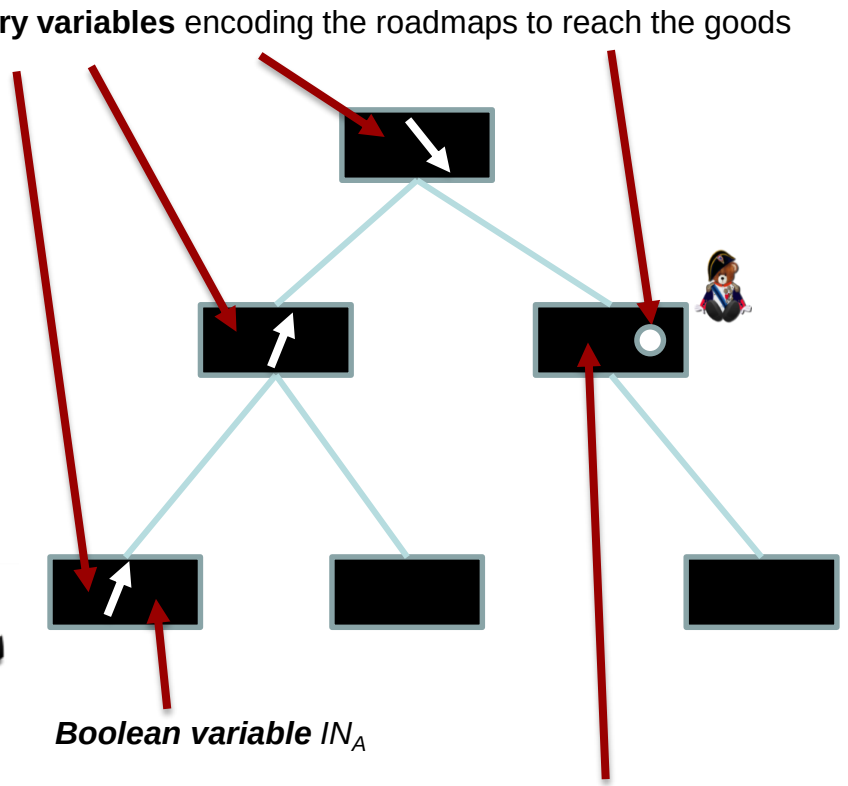
Proof Idea: Ingredient 3

- **Variables:**
  - *Agent* A, *agent* B, and *agent* C
  - Boolean variables IN_A, IN_B, IN_C
  - Products are associated with Boolean variables

  - Further auxiliary variables “R”
- **Domain:**
  - **D**(R) = { , , … }
- **Constraints:**
  - Compatibility of the road signs
  - Built from the decomposition!

From the original graph, we build a tree decomposition



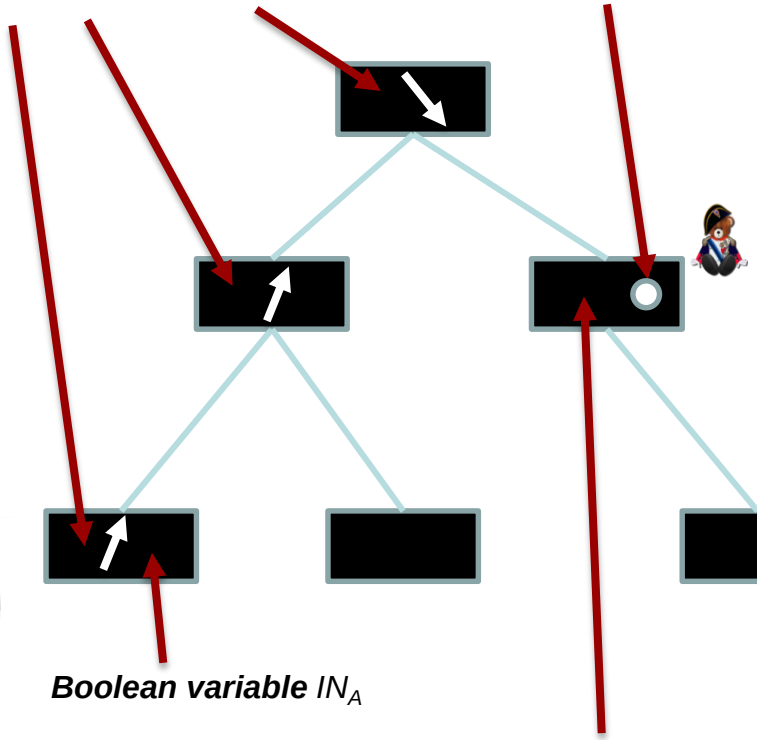


Proof Idea: Ingredient 3

W.l.o.g. the tree is binary.
Hence, a few «road signs» suffices

From the original graph,
we build a tree decomposition

auxiliary variables encoding the roadmaps to reach the goods



Boolean variable IN_A

Booelan variable associated with good

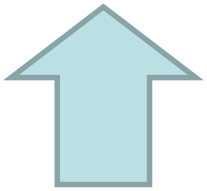
- **Domain:**
  - **D**(R) = { , , … }
- **Constraints:**
  - Compatibility of the road signs
  - Built from the decomposition!

Proof Idea: Ingredient 3

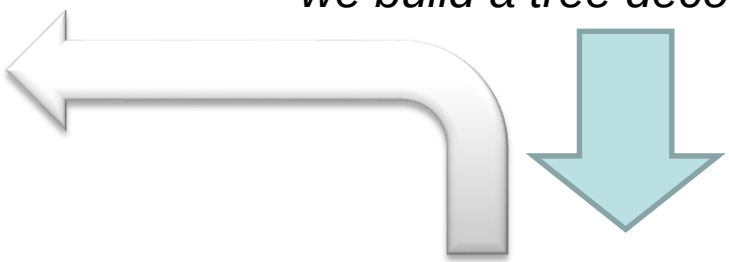
From the original graph, we build a tree decomposition

W.l.o.g. the tree is binary.
Hence, a few «road signs» suffices

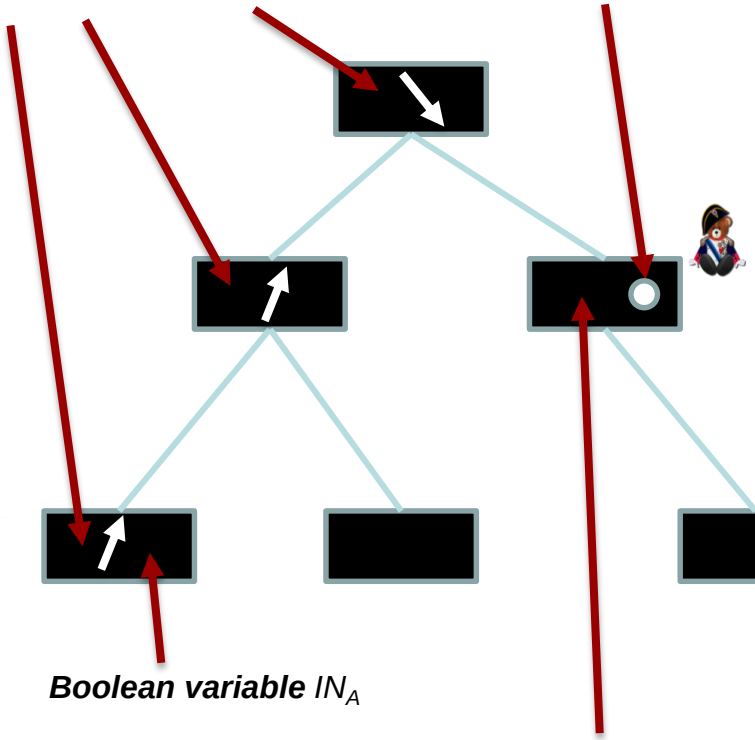
*The CSP has **still** bounded treewidth, and the domain of the auxiliary variables is bounded*



- **Constraints:**
  - Compatibility of the road signs
  - Built from the decomposition!



auxiliary variables encoding the roadmaps to reach the goods



Boolean variable IN_A

Booelan variable associated with good

Outline of Part III

Application: Nash Equilibria

Application: Nucleolus

Application: Shapley Value

Application: Combinatorial Auctions

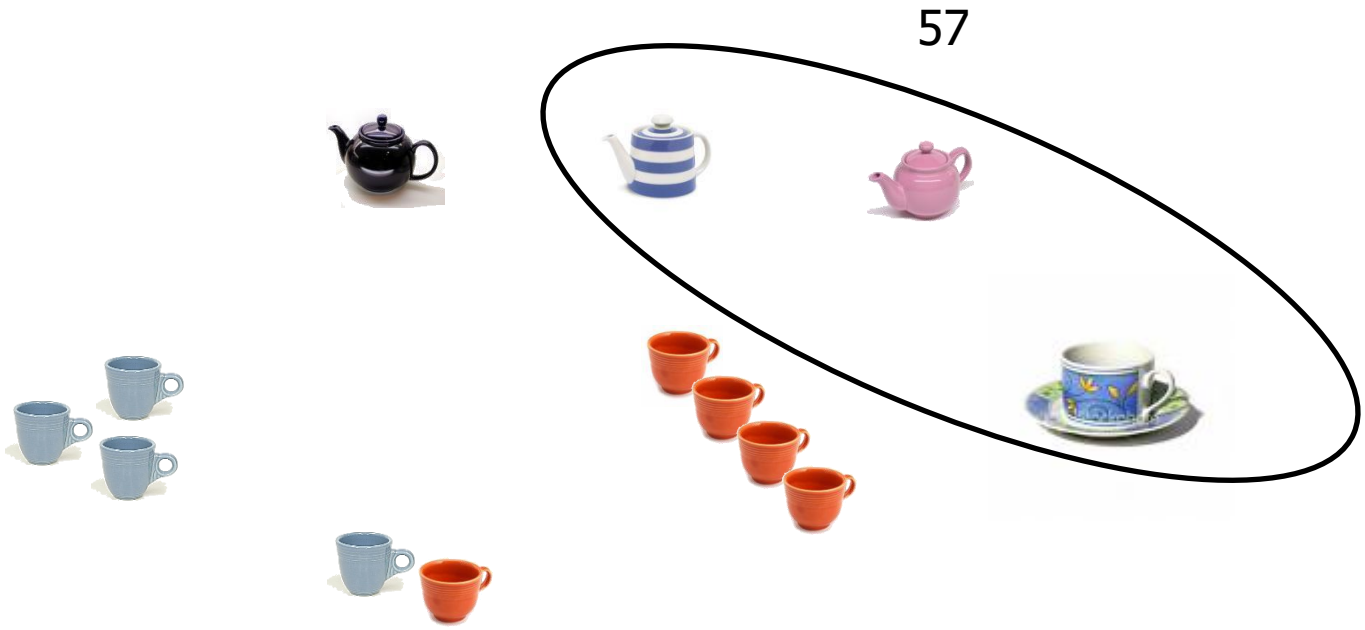
Example: Combinatorial Auctions



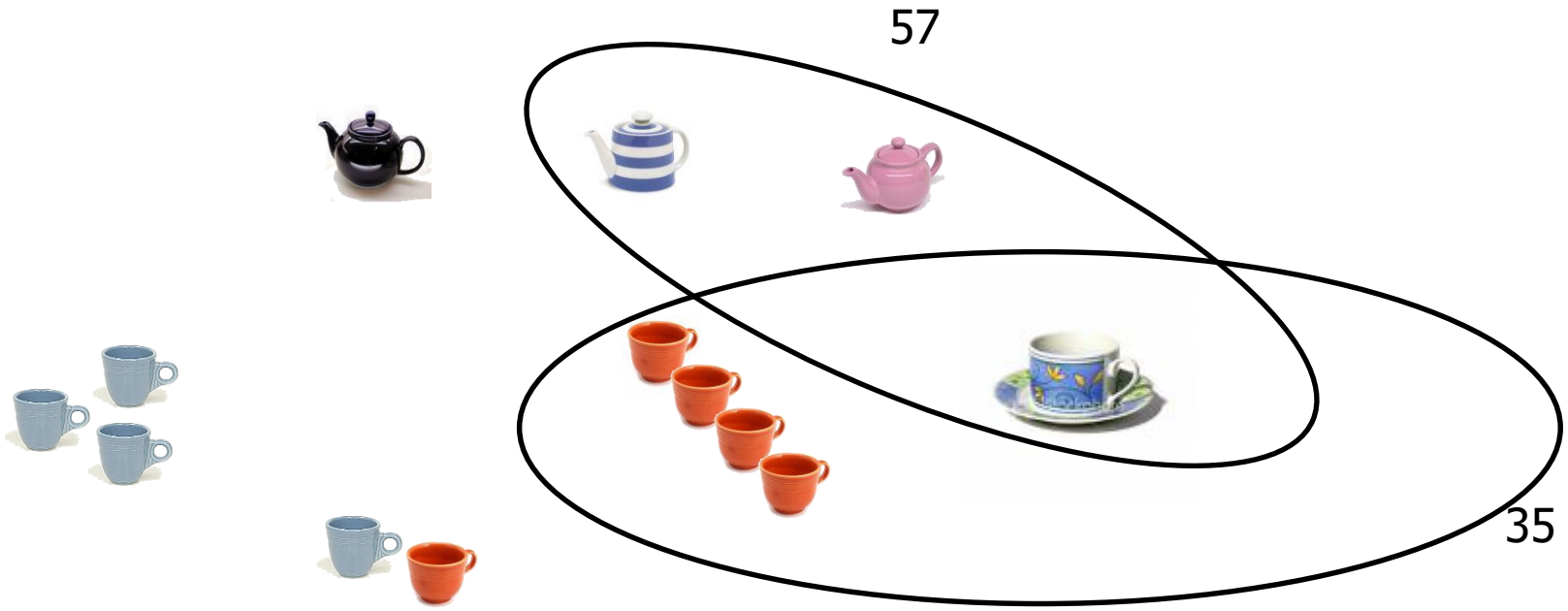




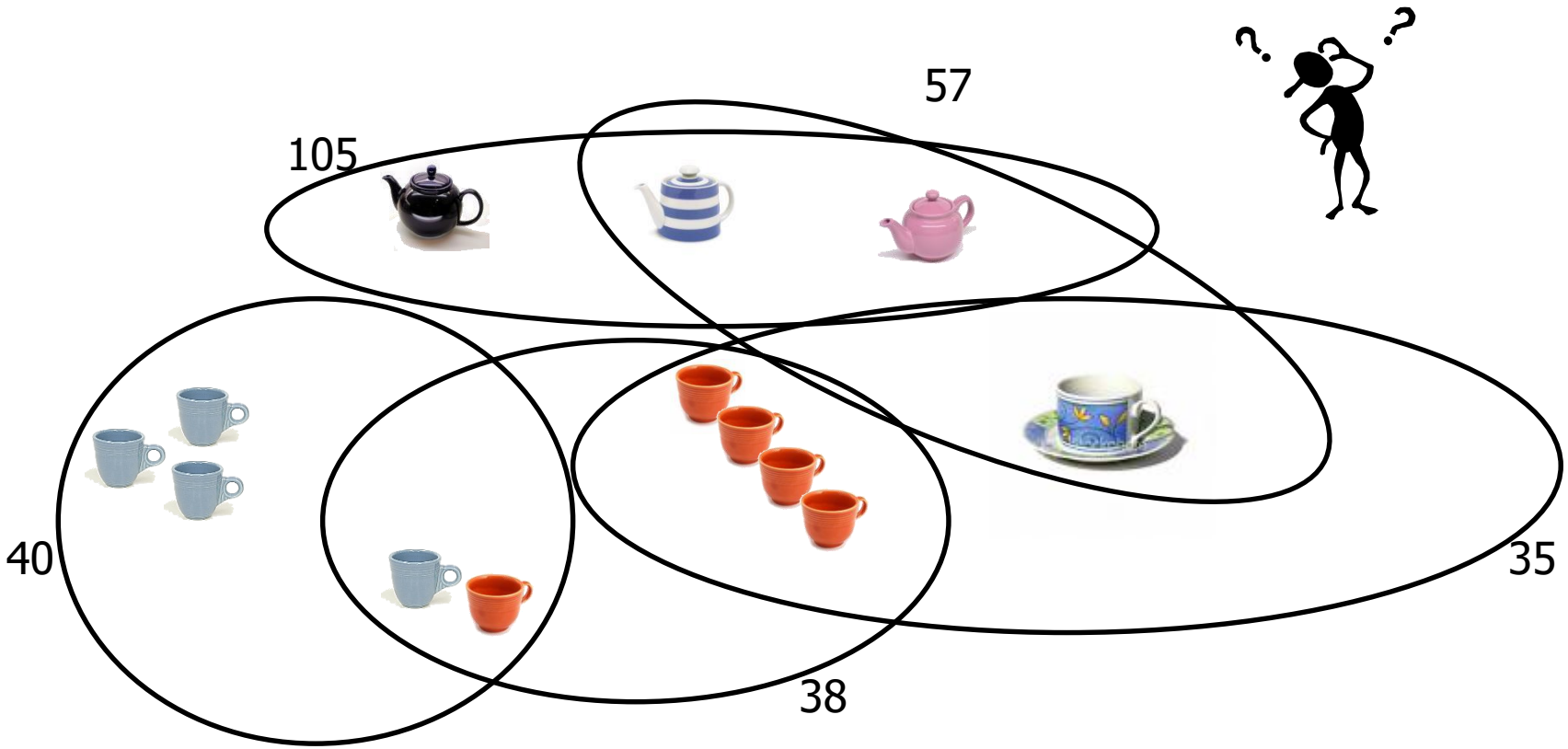
Example: Combinatorial Auctions



Example: Combinatorial Auctions



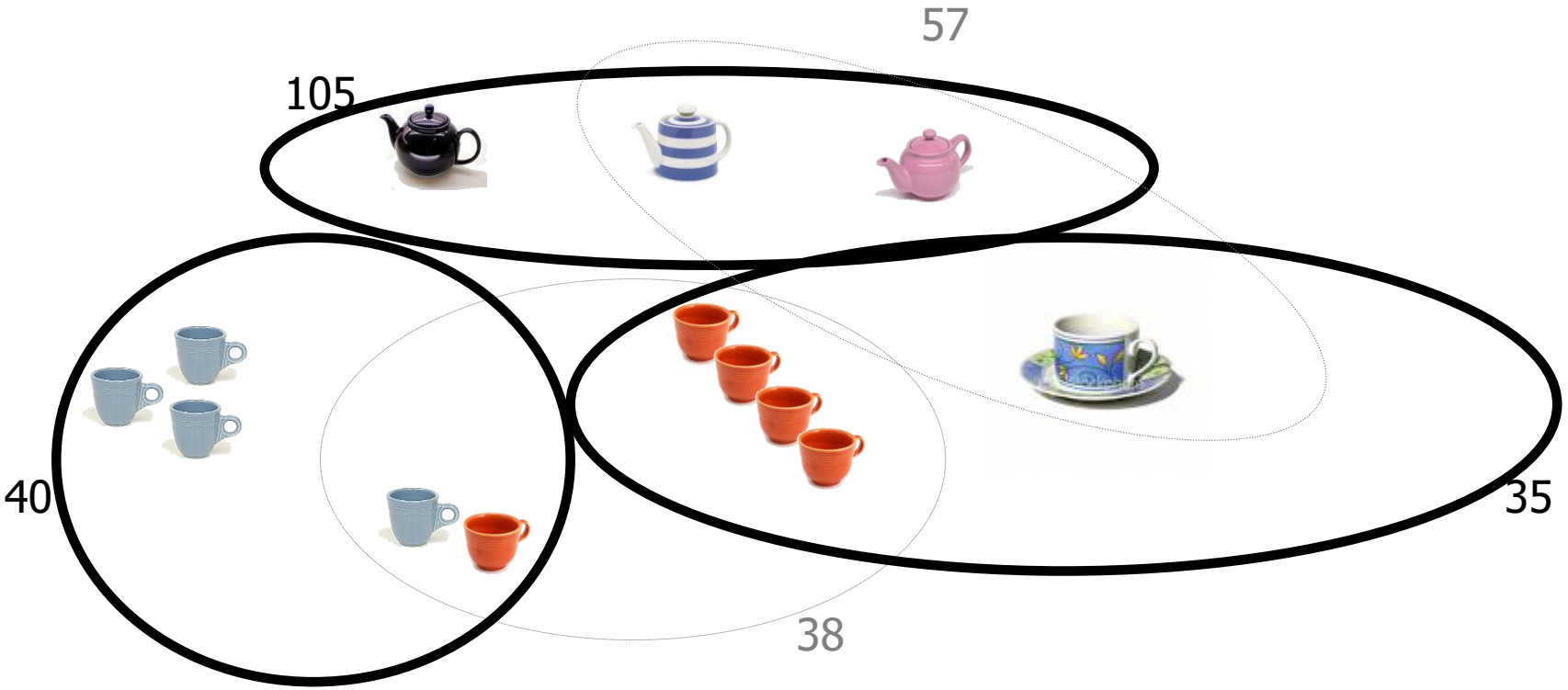
Example: Combinatorial Auctions



Winner Determination Problem

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

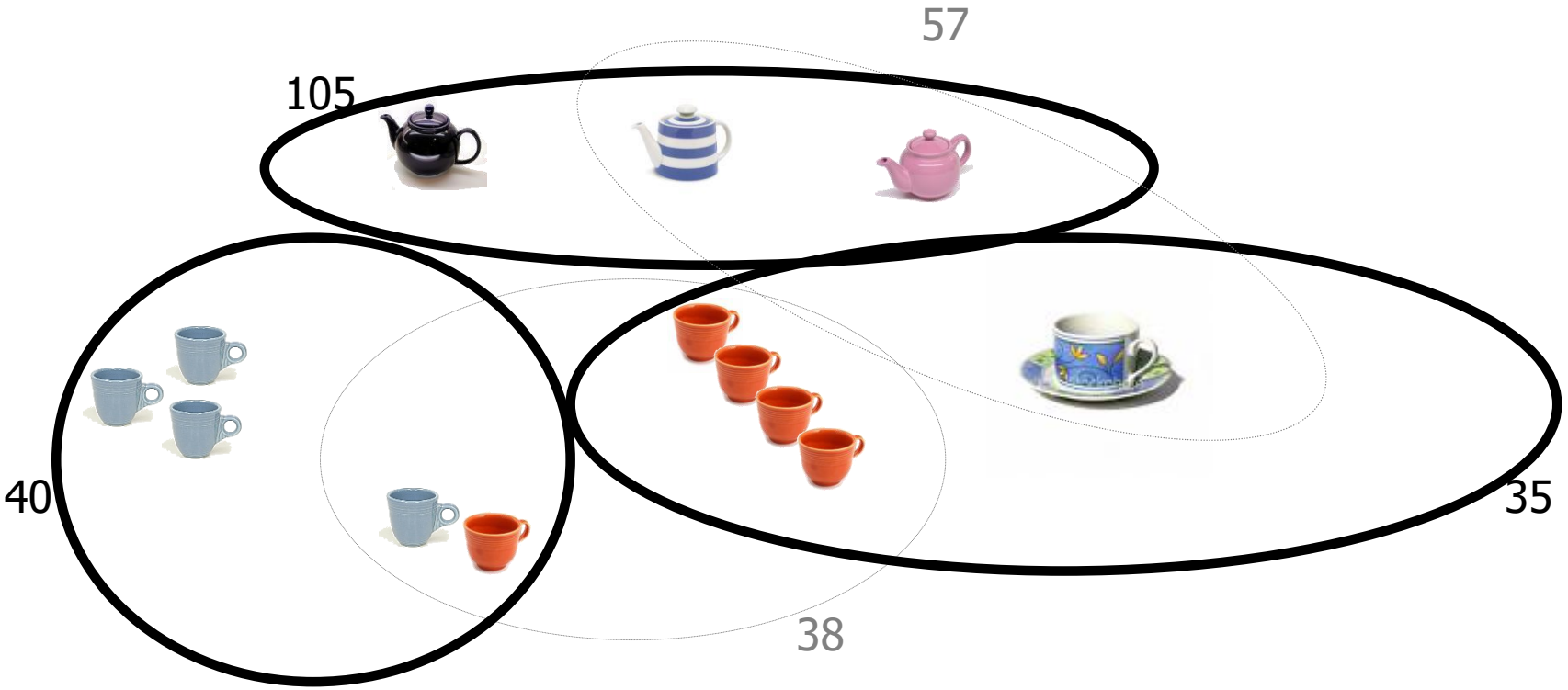
Example: Combinatorial Auctions



Winner Determination Problem **180**

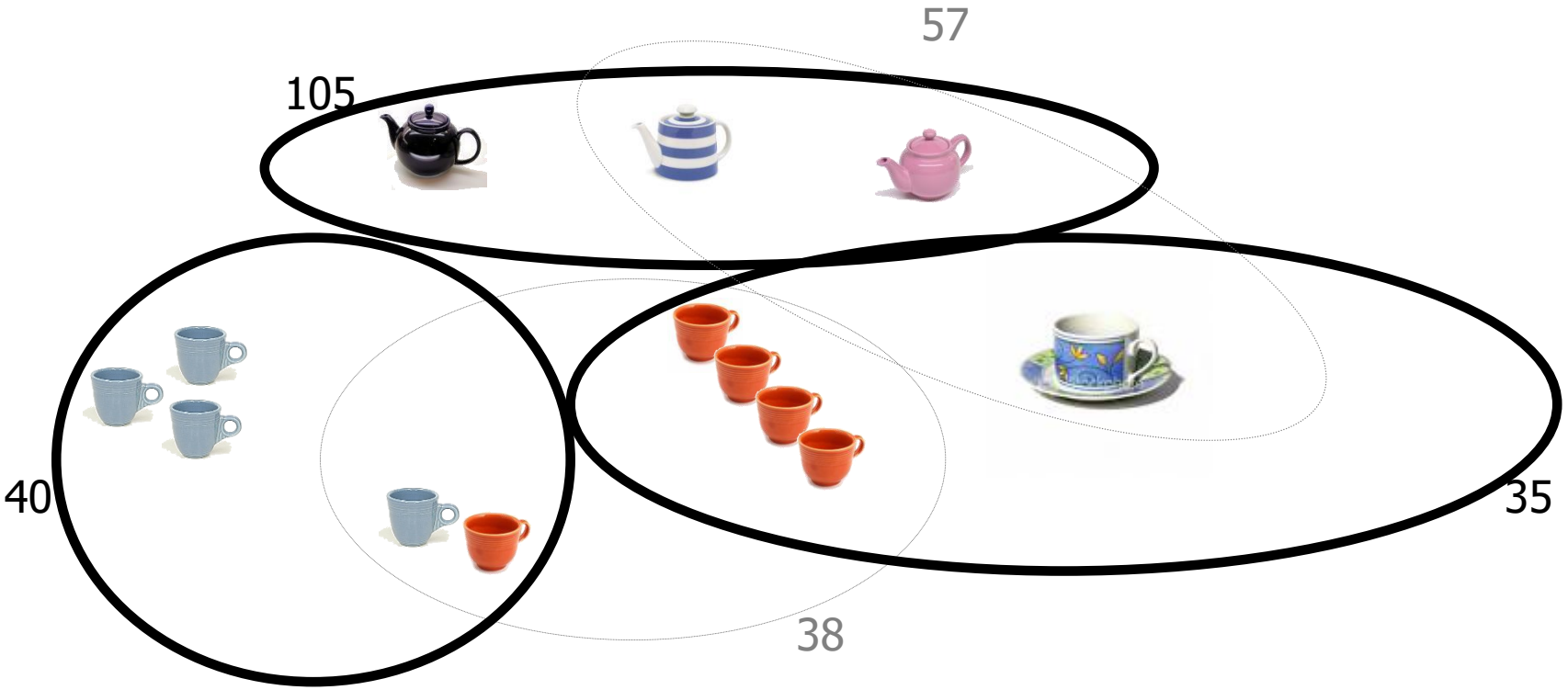
- Determine the outcome that maximizes the sum of accepted bid prices

Example: Combinatorial Auctions



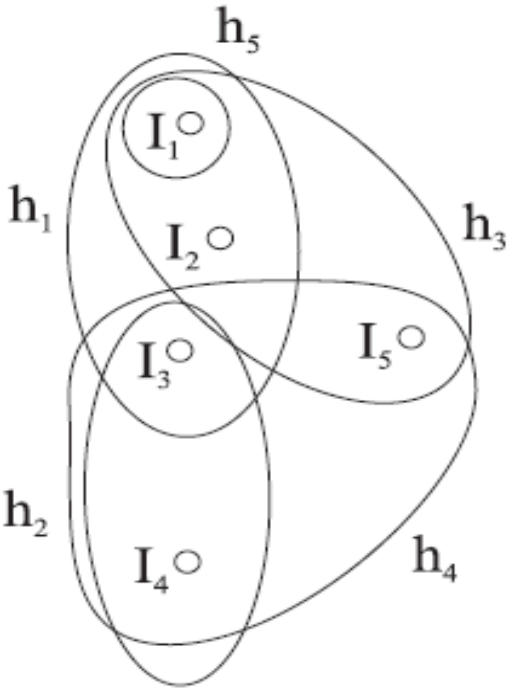
- Other applications [Cramton, Shoham, and Steinberg, '06]
  - airport runway access
  - trucking
  - bus routes
  - industrial procurement

Example: Combinatorial Auctions



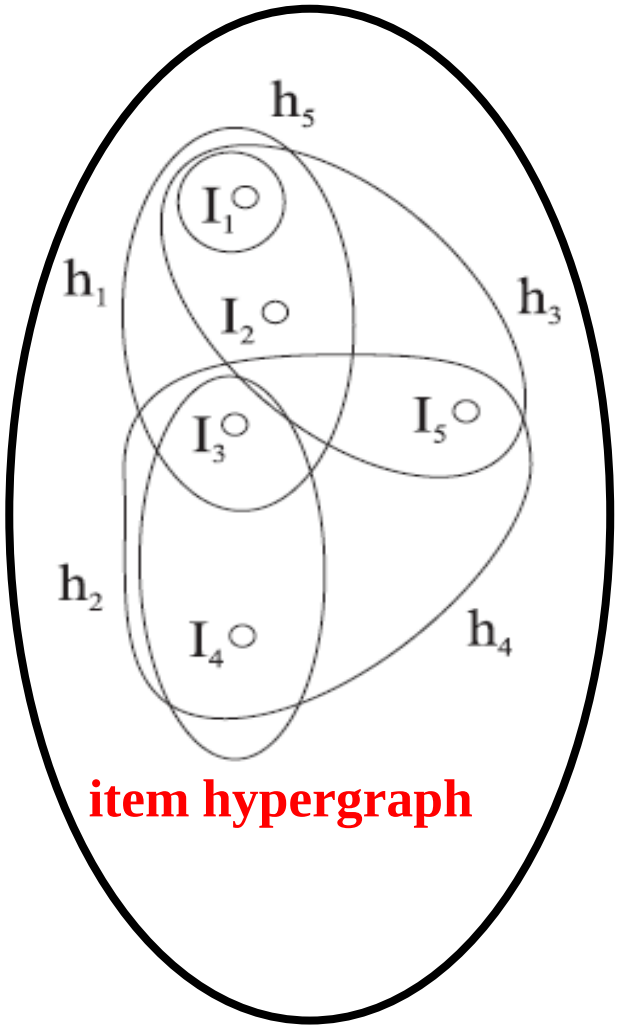
Winner Determination is NP-hard

Structural Properties



item hypergraph

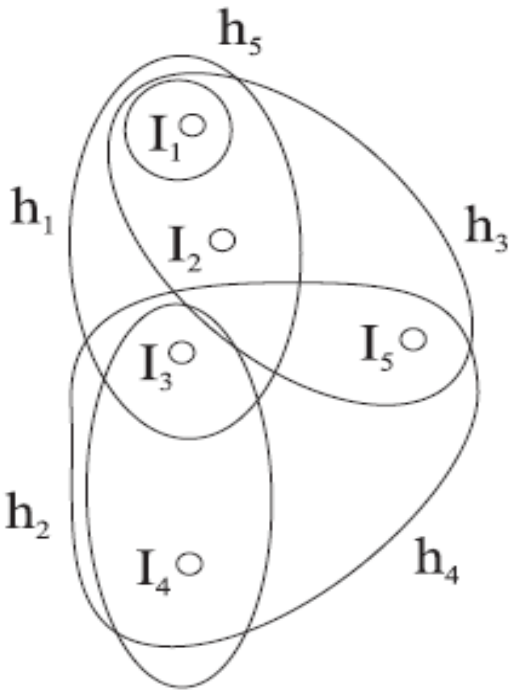
Structural Properties



item hypergraph

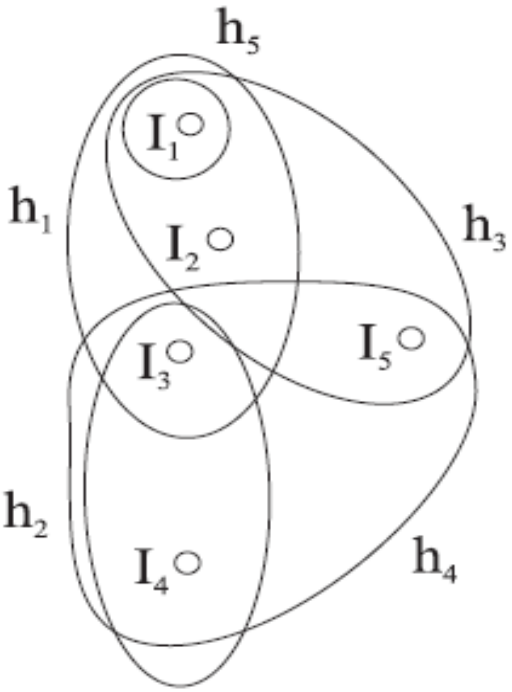
The Winner Determination Problem remains NP-hard even in case of acyclic hypergraphs

Dual Hypergraph



item hypergraph

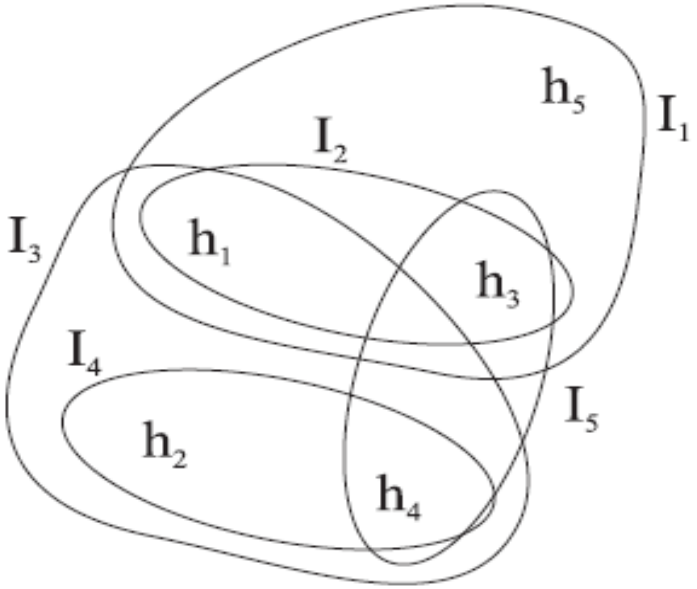
Dual Hypergraph



item hypergraph

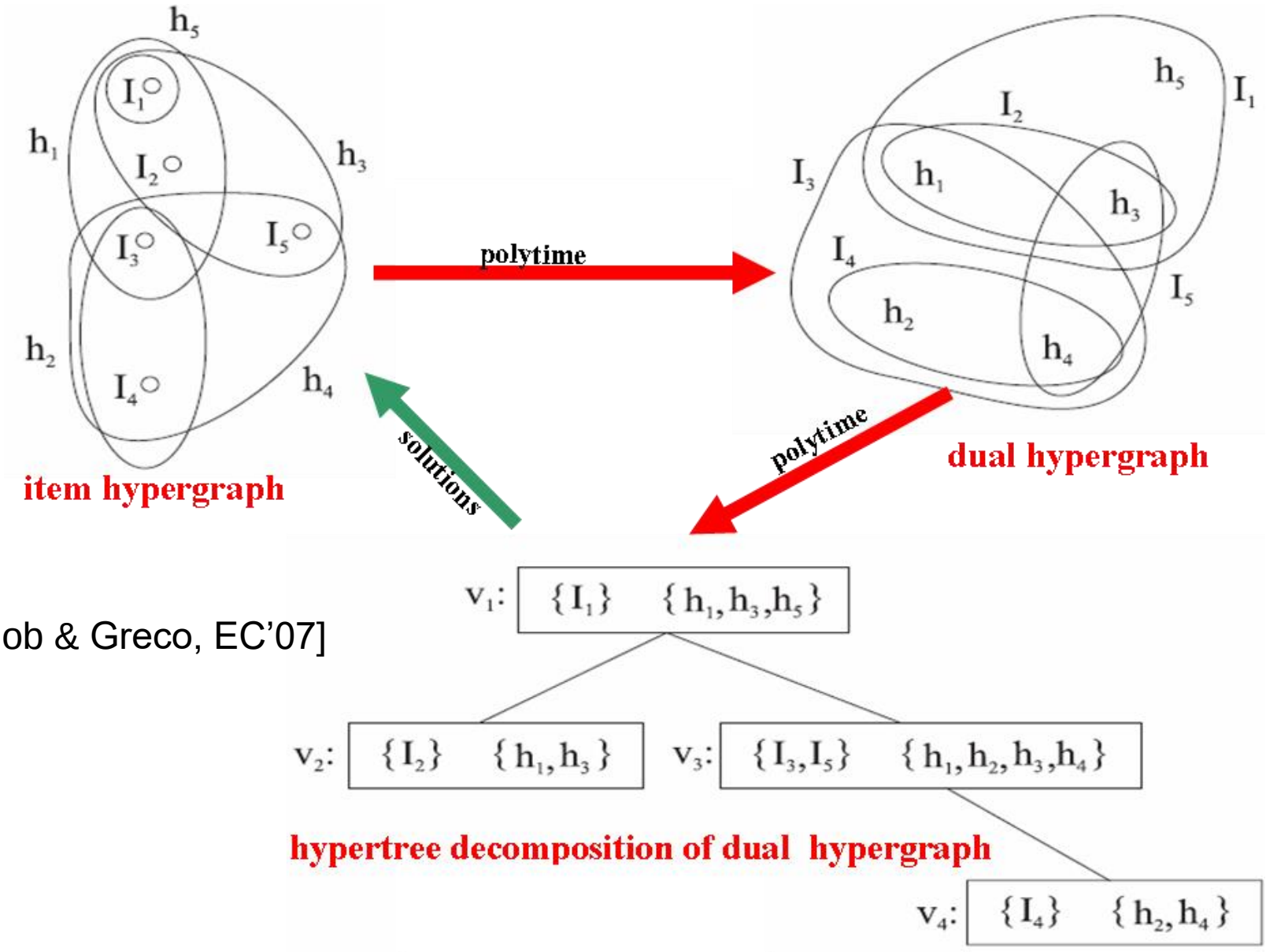
polytime

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



dual hypergraph

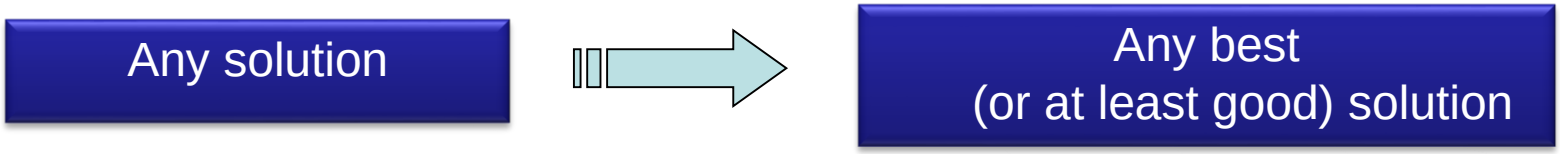
The Approach



[Gottlob & Greco, EC'07]

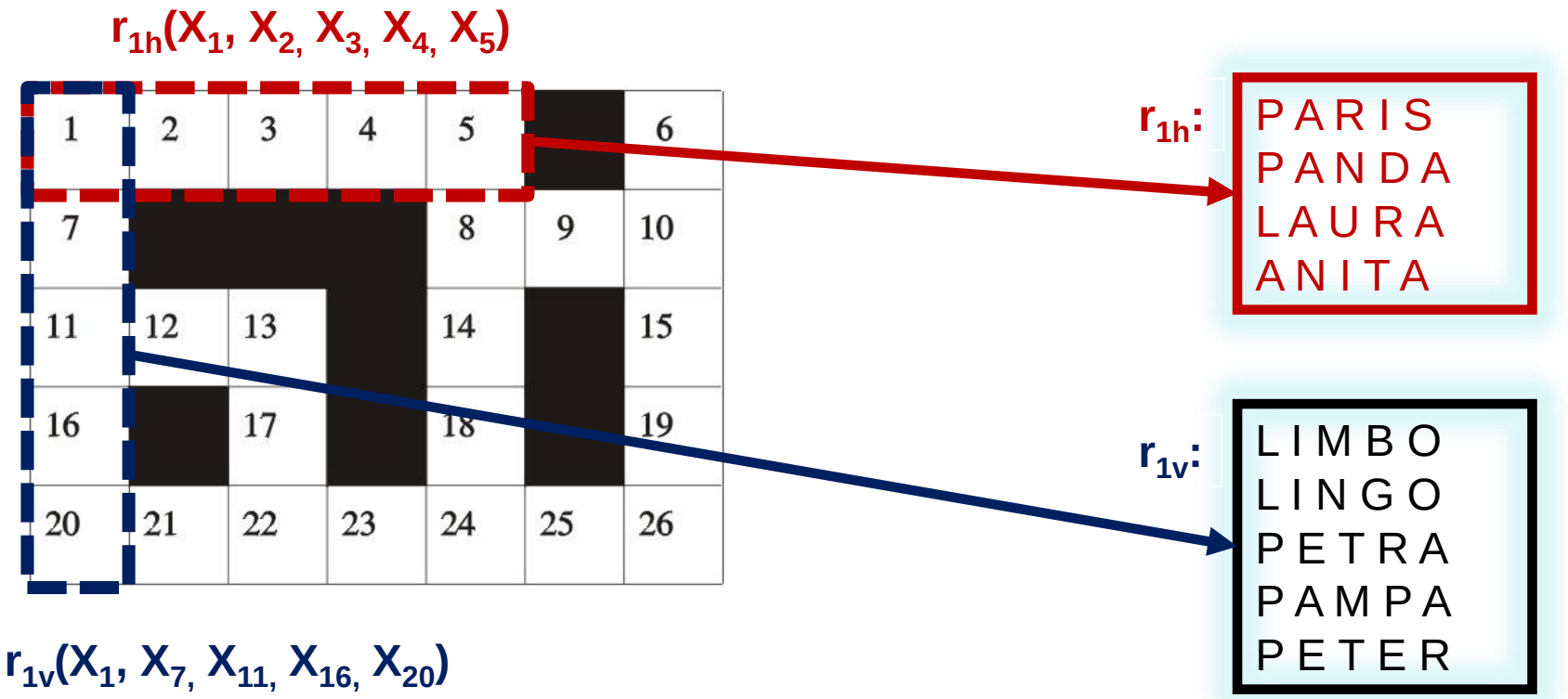
Constraint Optimization Problems

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



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

Classical CSPs



- 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

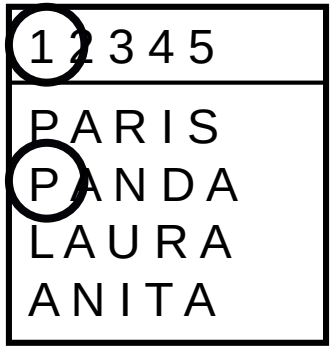
A Classification for Optimization Problems

CSOP

Each mapping variable-value has a cost.

Then, find an assignment:

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



A Classification for Optimization Problems

CSOP

Each mapping variable-value has a cost.

Then, find an assignment:

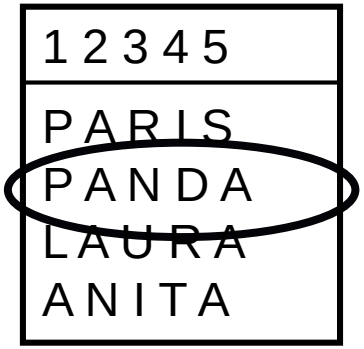
- Satisfying all the constraints, and
- Having the minimum total cost.

WCSP

Each tuple has a cost.

Then, find an assignment:

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



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:

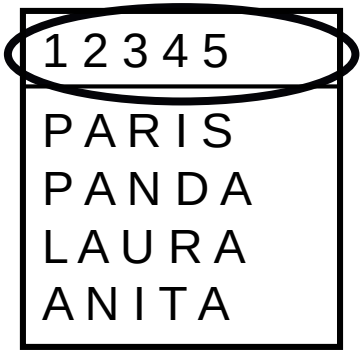
- Satisfying all the constraints, and
- Having the minimum total cost.

MAX-CSP

Each constraint relation has a cost.

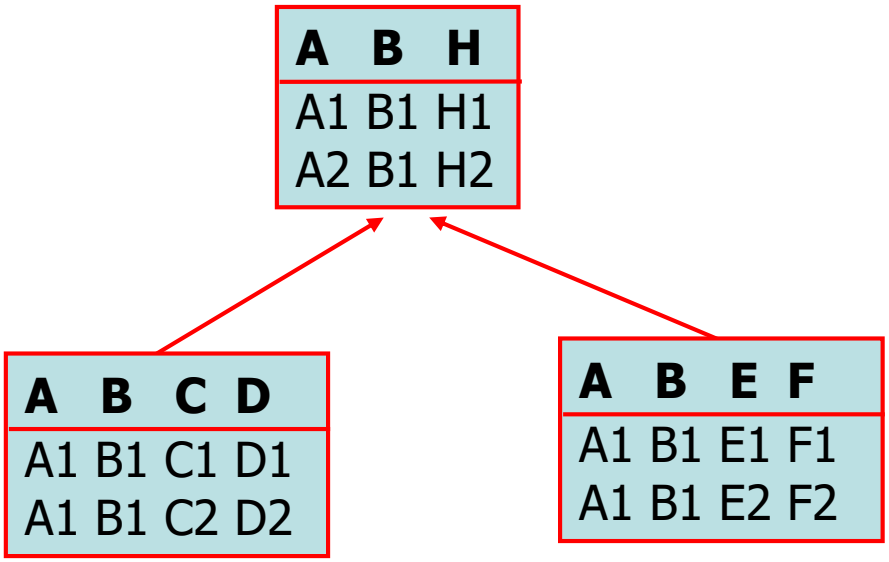
Then, find an assignment:

- Minimizing the cost of violated relations.



CSOP: Tractability of Acyclic Instances

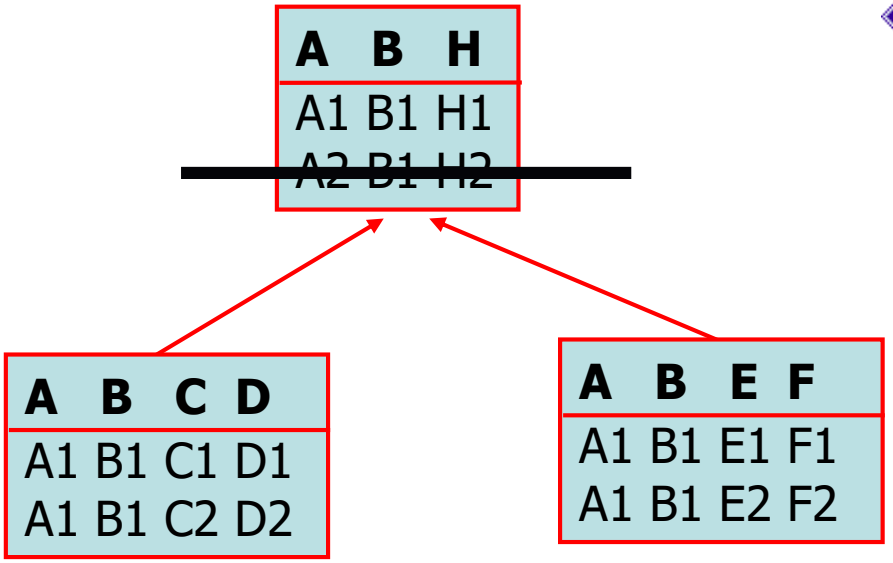
- Adapt the dynamic programming approach in **(Yannakakis'81)**



[Gottlob & Greco, EC'07]

CSOP: Tractability of Acyclic Instances

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

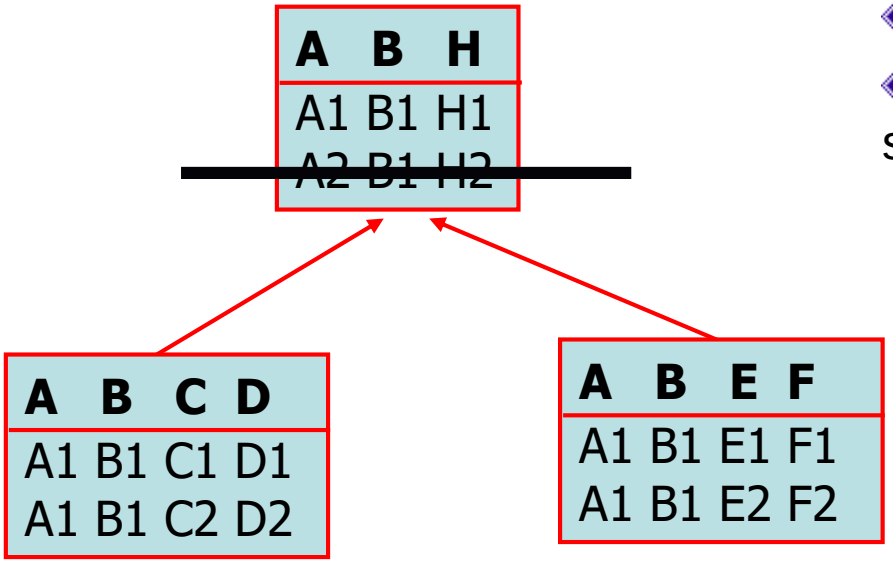


With a bottom-up computation:

- Filter the tuples that do not match

CSOP: Tractability of Acyclic Instances

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



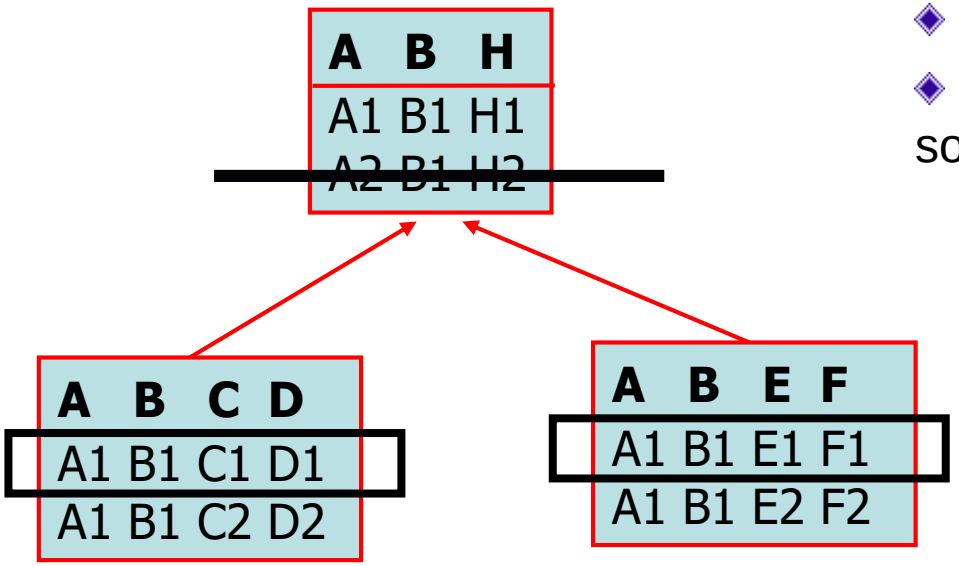
With a bottom-up computation:

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

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)**



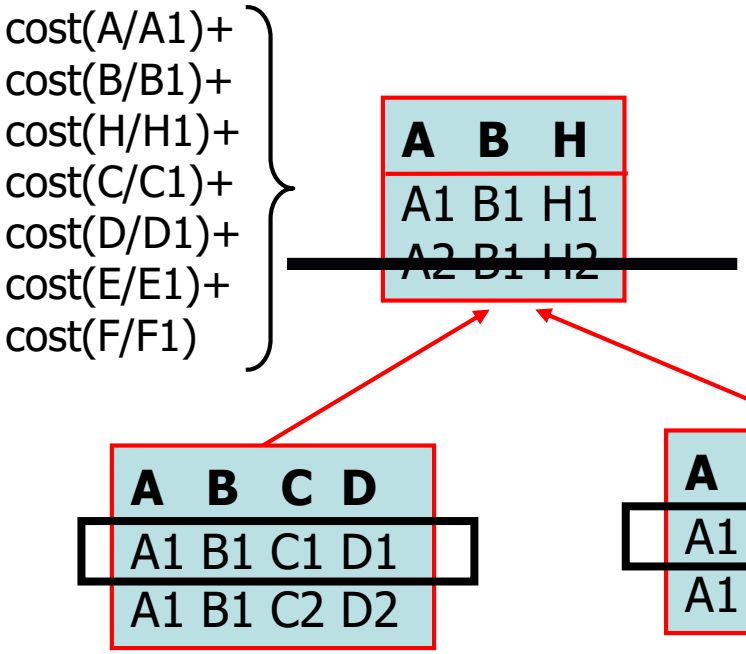
With a bottom-up computation:

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

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)**



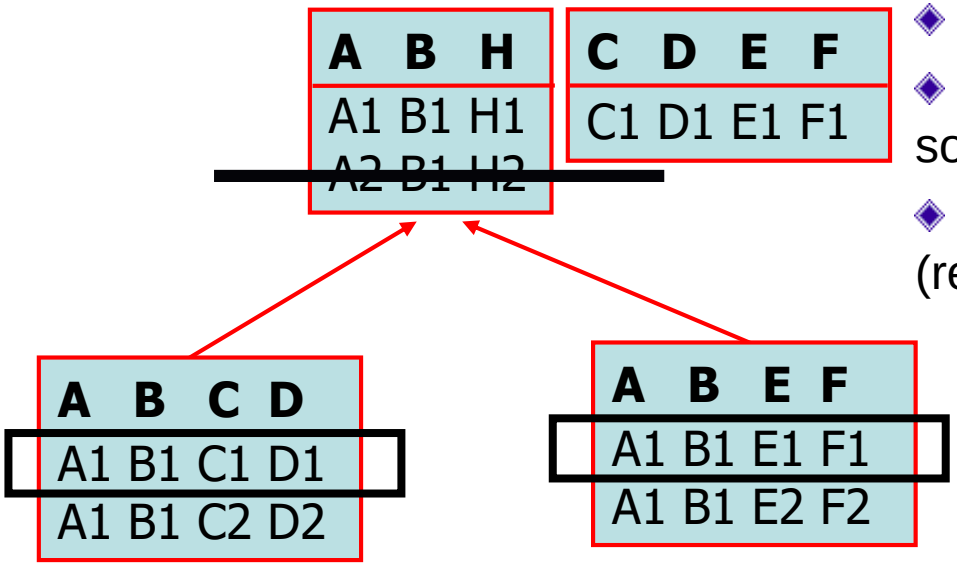
With a bottom-up computation:

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

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

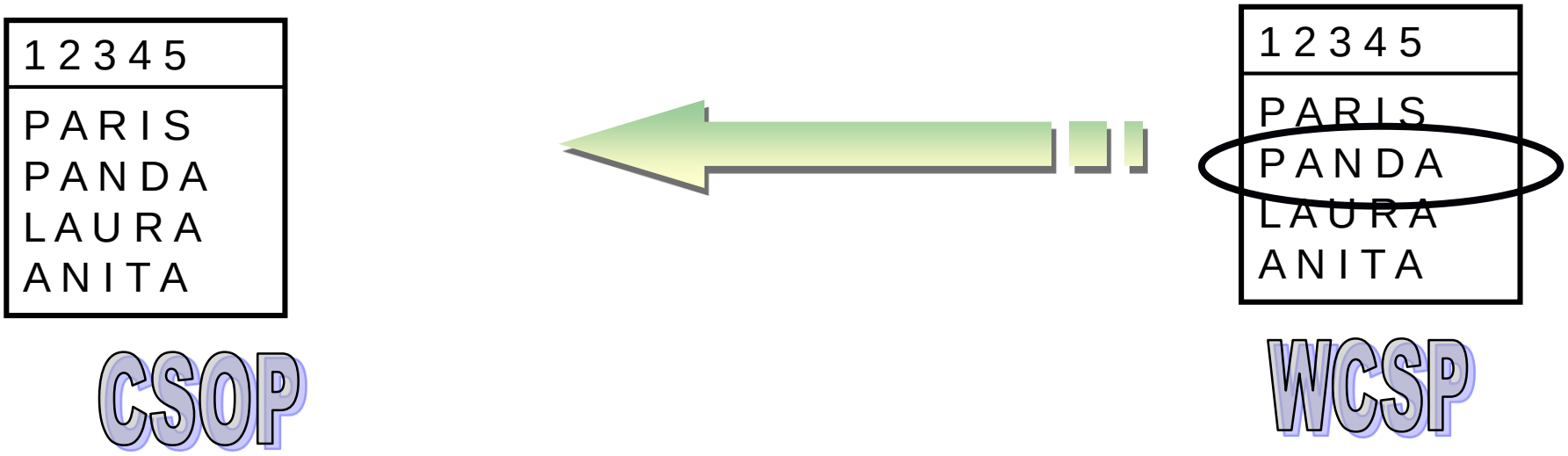
- Adapt the dynamic programming approach in **(Yannakakis'81)**



With a bottom-up computation:

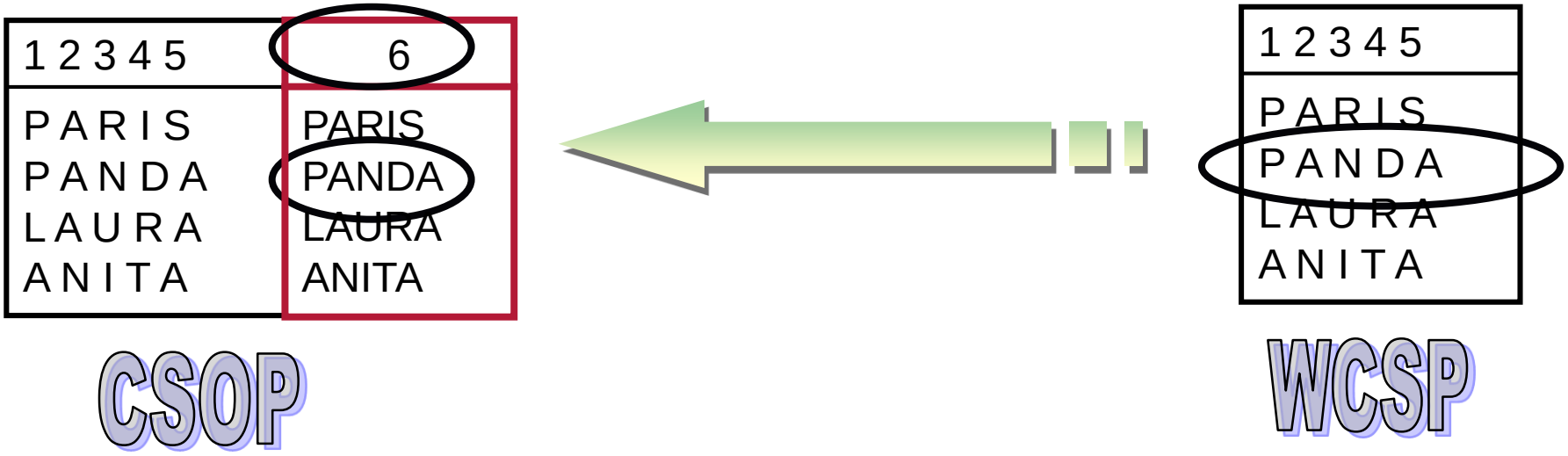
- Filter the tuples that do not match
- Compute the cost of the best partial solution, by looking at the children
- Propagate the best partial solution (resolving ties arbitrarily)

WCSP: Tractability of Acyclic Instances



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

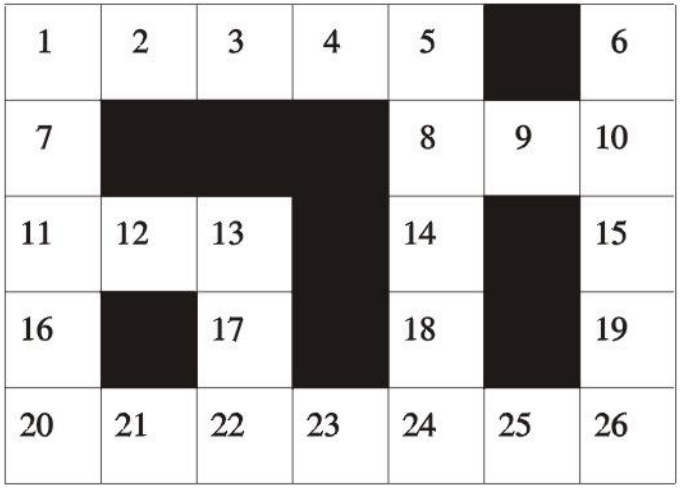
WCSP: Tractability of Acyclic Instances



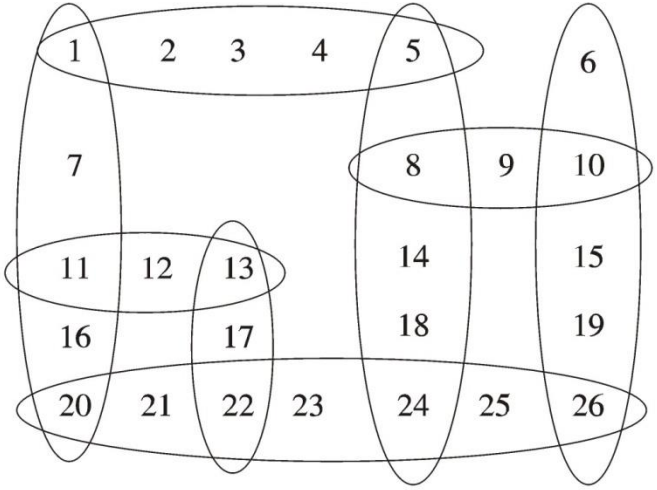
The mapping:

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

In-Tractability of MAX-CSP Instances

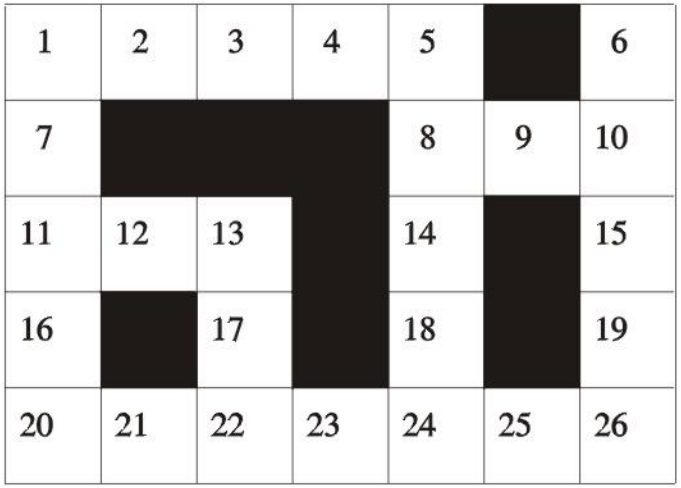


- Maximize the number of words placed in the puzzle

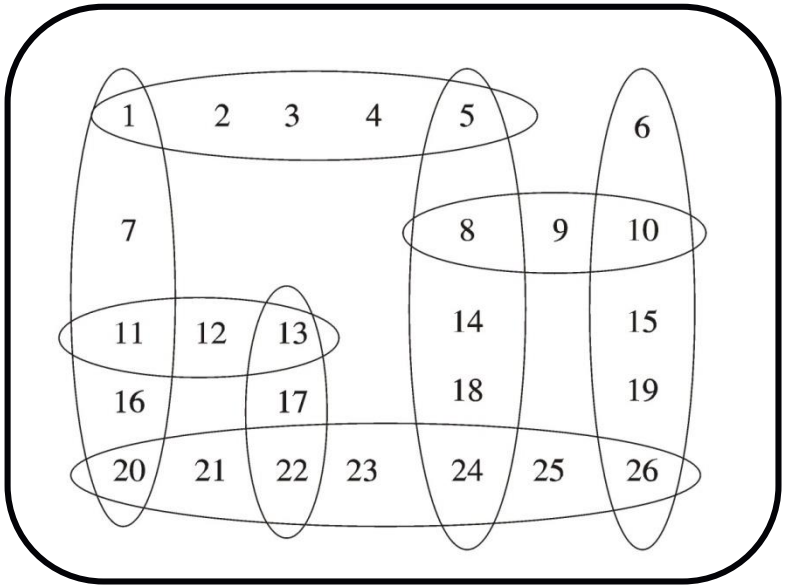


[Gottlob, Greco, and Scarcello, ICALP‘09]

In-Tractability of MAX-CSP Instances



- Maximize the number of words placed in the puzzle

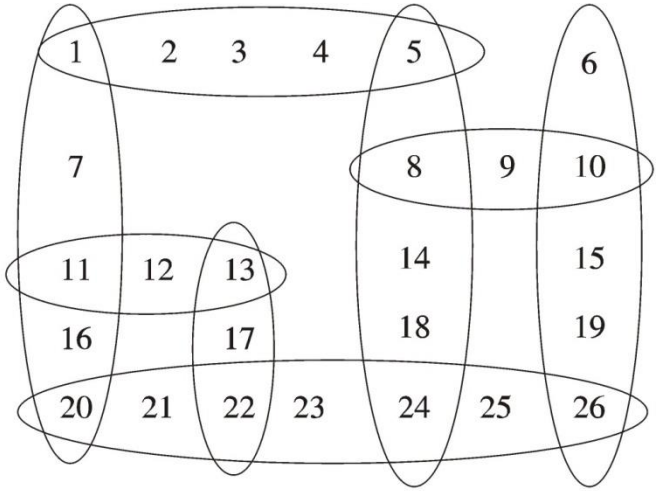


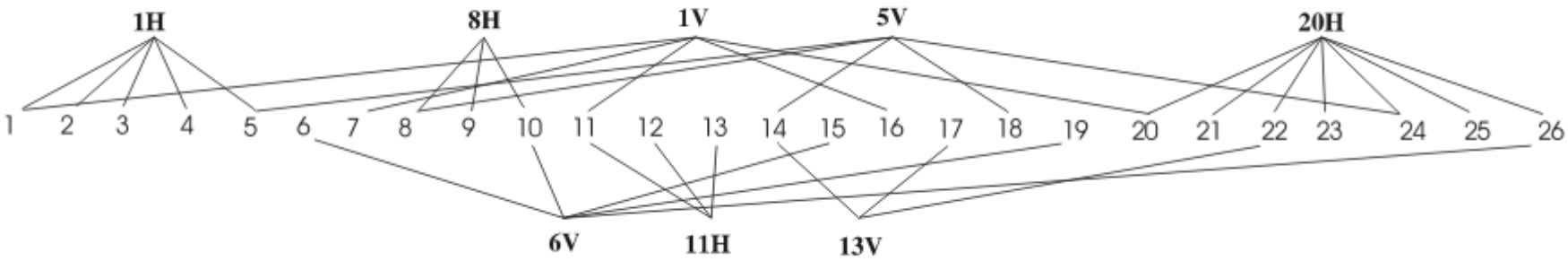
- Add a “big” constraint with no tuple

The puzzle is satisfiable $\leftrightarrow$ exactly one constraint is violated in the **acyclic** MAX-CSP

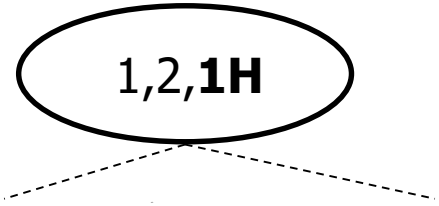
Tractability of MAX-CSP Instances

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

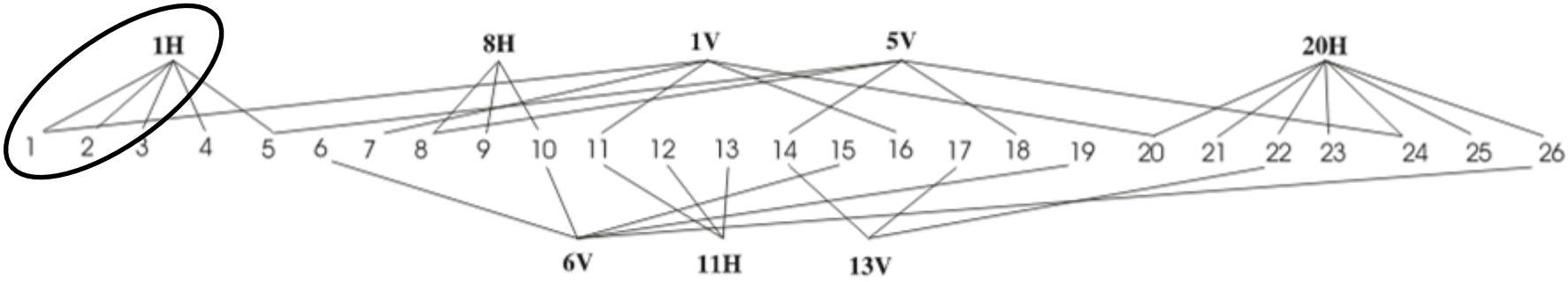




Tractability of MAX-CSP Instances



MAX-CSP



Tractability of MAX-CSP Instances

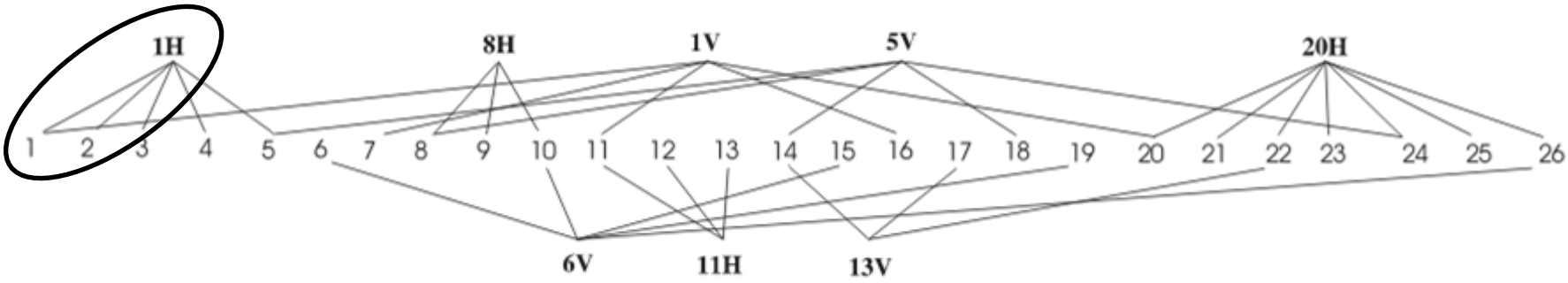
1 2	**1H**
P A	PARIS
P A	PANDA
L A	LAURA
A N	ANITA
A A	*unsat*
A B	*unsat*
...	*unsat*

CSOP

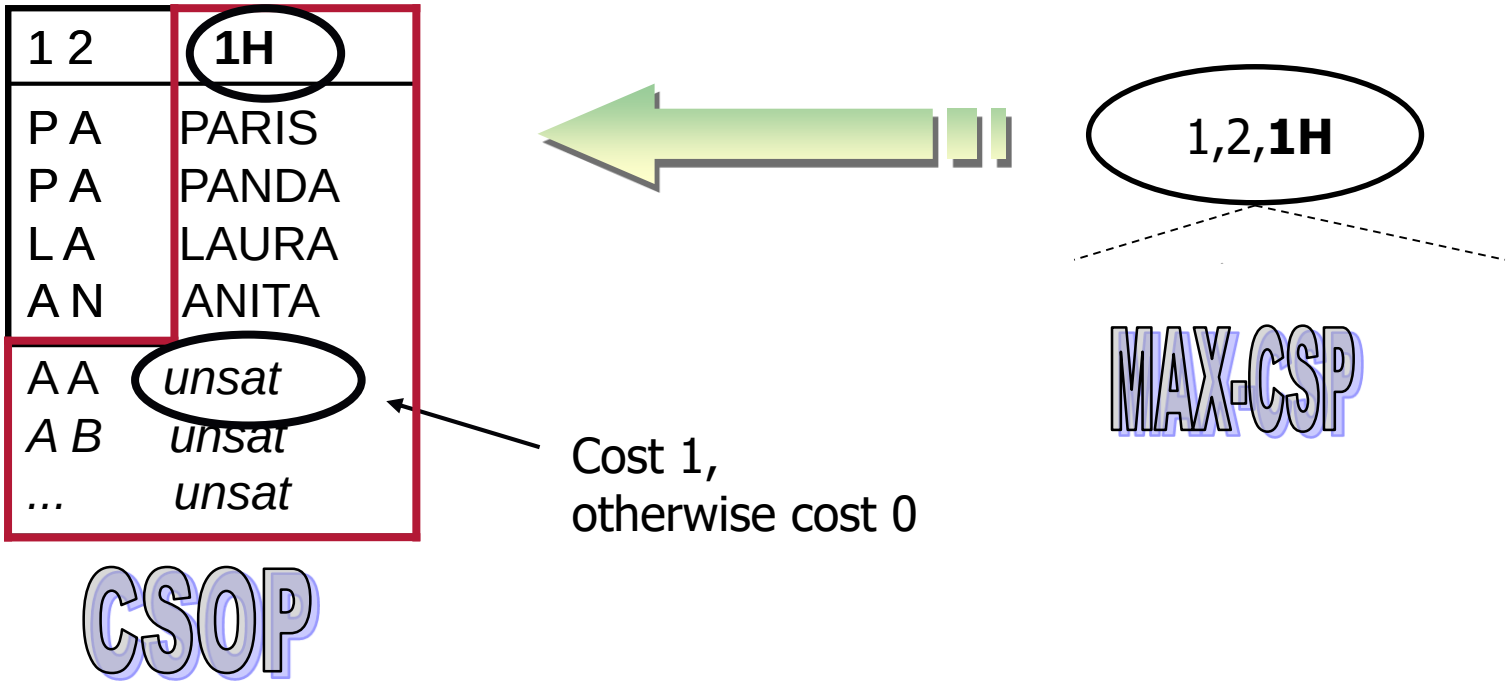
Cost 1,
otherwise cost 0

1,2,**1H**

MAX-CSP



In-Tractability of MAX-CSP Instances



The mapping:

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

Thank you!