Winter School and Workshop on Algorithmic Game Theory
Singapore, 14-18 Jan 2013

Structural Decomposition Methods:

Basic Concepts and Applications in Algorithmic Game Theory

Gianluigi Greco
University of Calabria

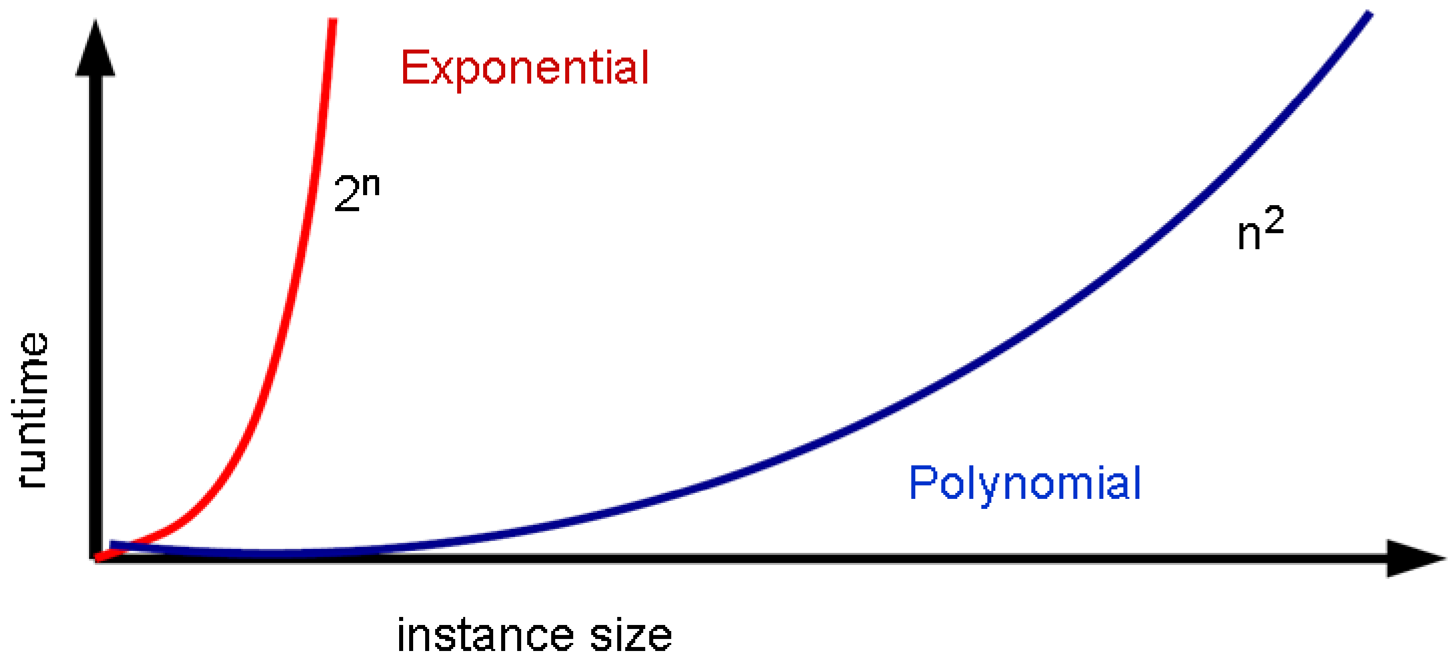
Inherent Problem Complexity

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

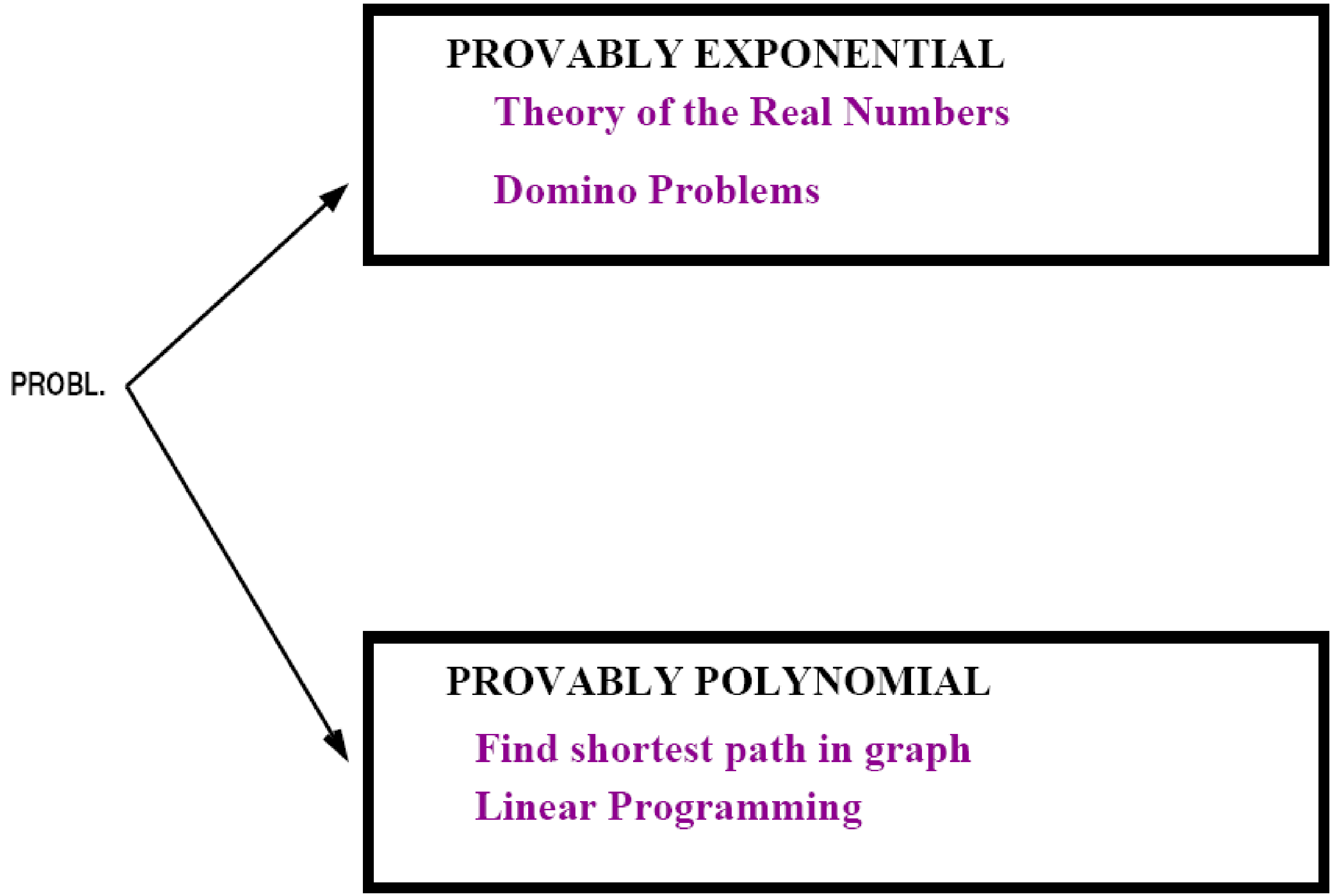
Inherent Problem Complexity

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

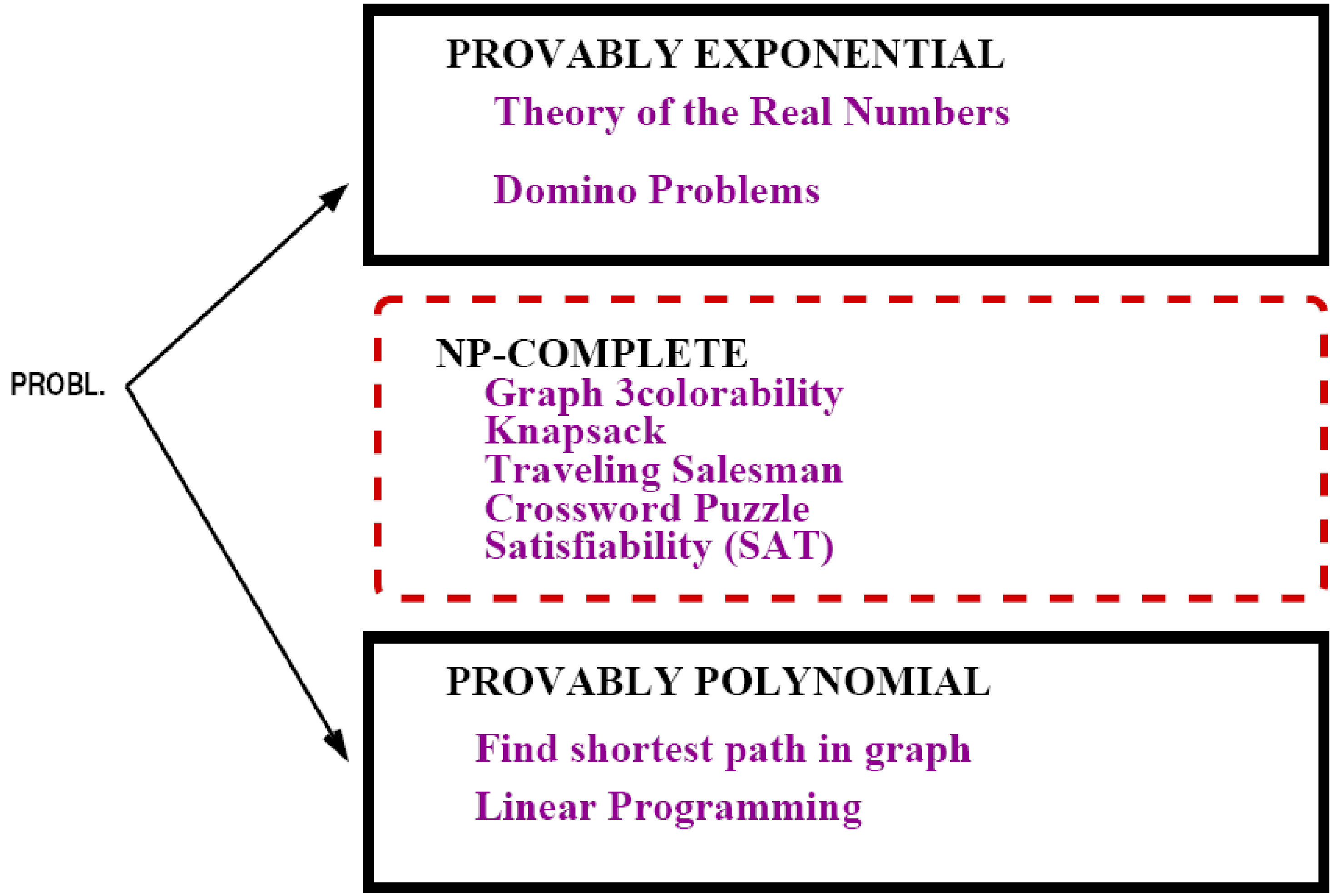
Number of steps it takes for input of size n



Time Complexity



Time Complexity

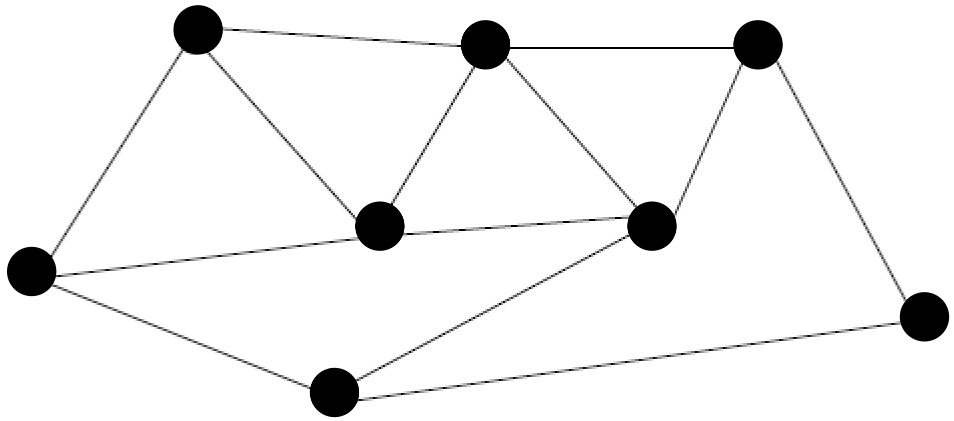


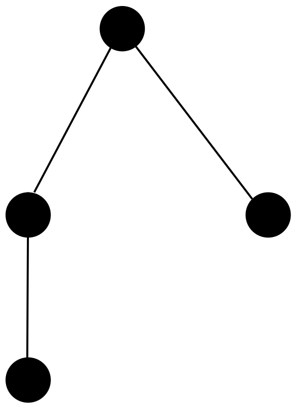
Graph Three-colorability

Instance: A graph G.

Question: Is G 3-colorable?

Examples of instances:



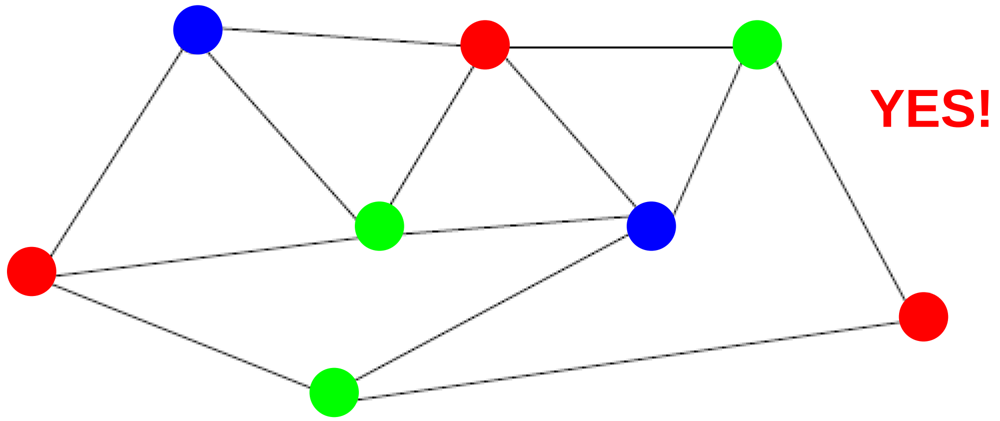


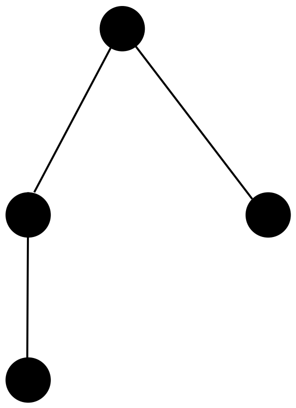
Graph Three-colorability

Instance: A graph G.

Question: Is G 3-colorable?

Examples of instances:





Outline

Identification of “Easy” Classes

Applications of Tree Decompositions

Beyond Tree Decompositions

Decision/Computation Problems

Optimization Problems

Enumeration Problems

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

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

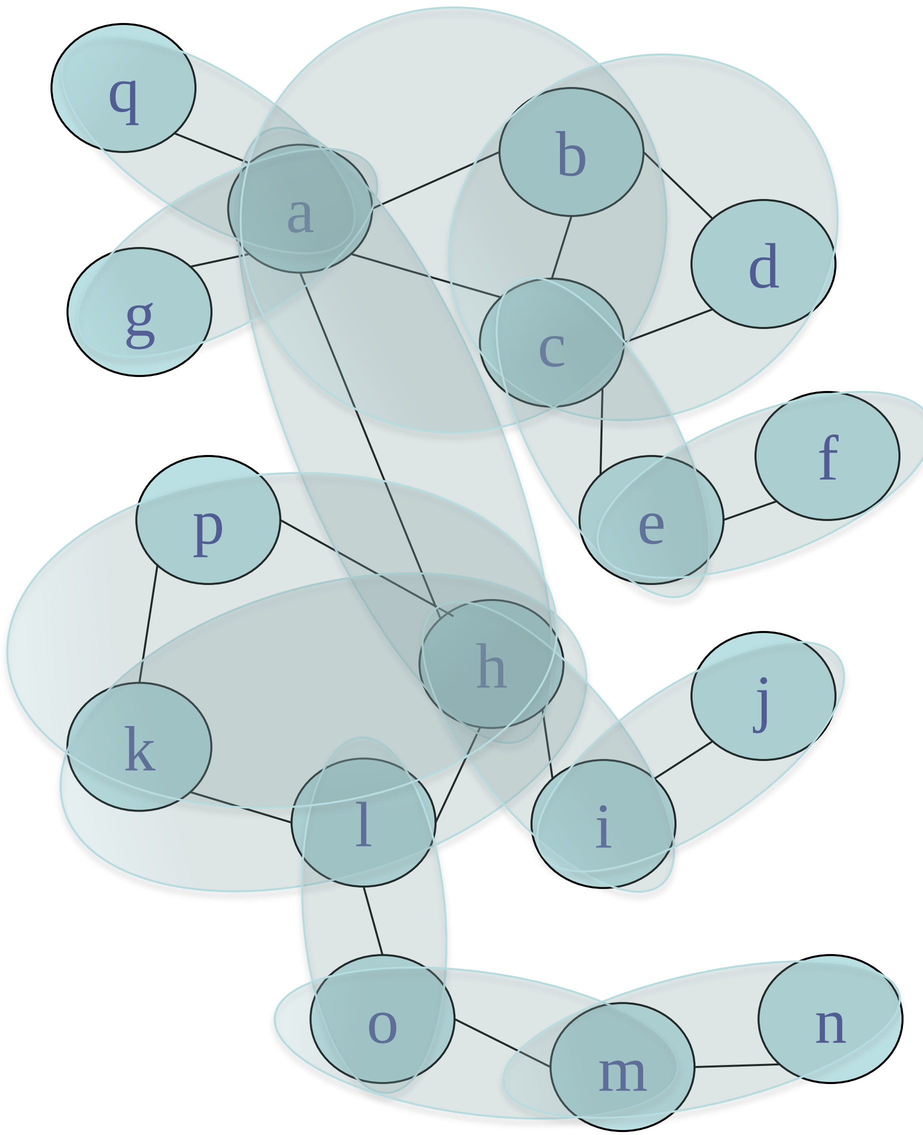
- With graph-based problems, high complexity is mostly due to *cyclicity*.

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

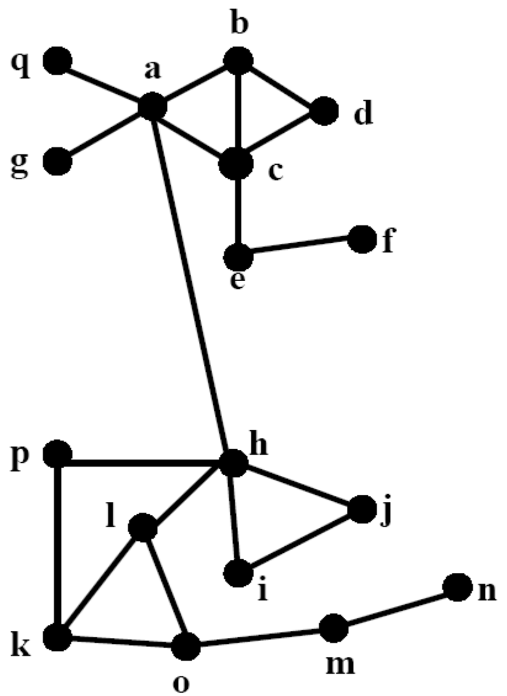
- Moreover, many graph problems are polynomially solvable if restricted to instances of *low cyclicity*.

How can we measure the degree of cyclicity?

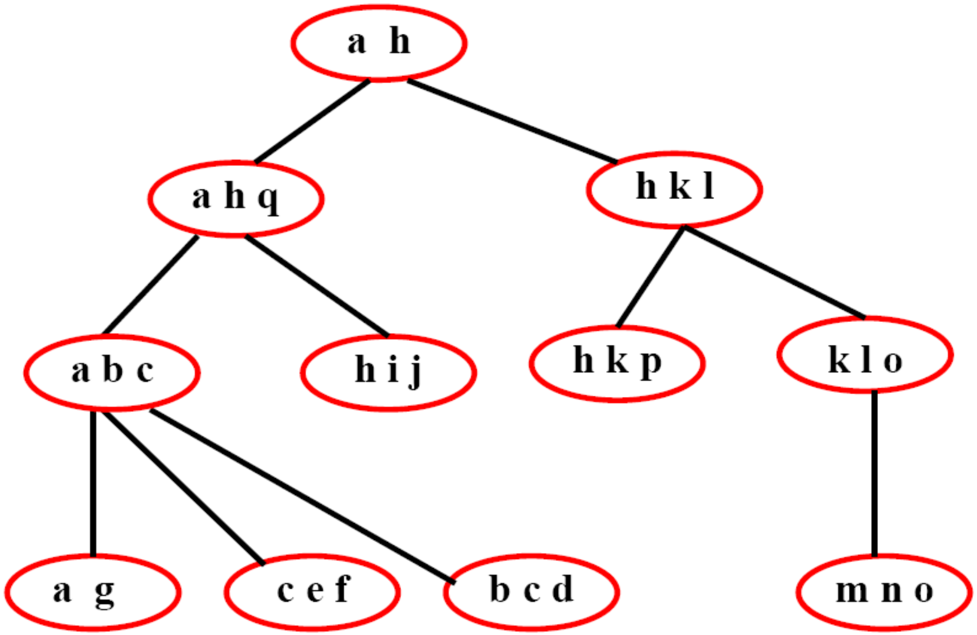
Tree Decompositions [Robertson & Seymour '86]



Tree Decompositions [Robertson & Seymour '86]

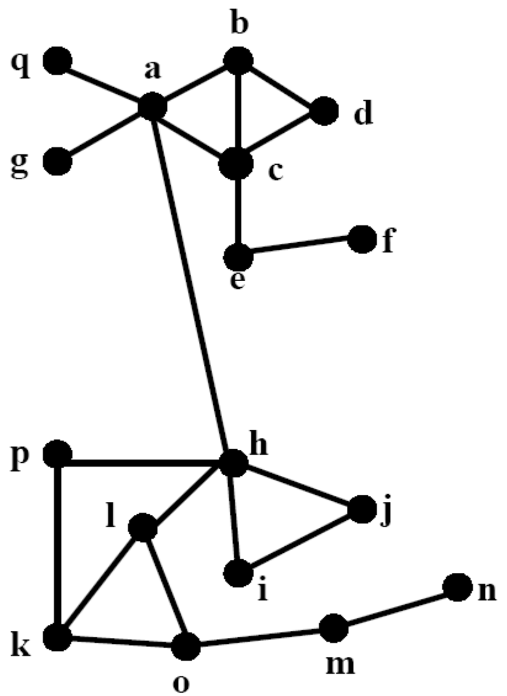


Graph G

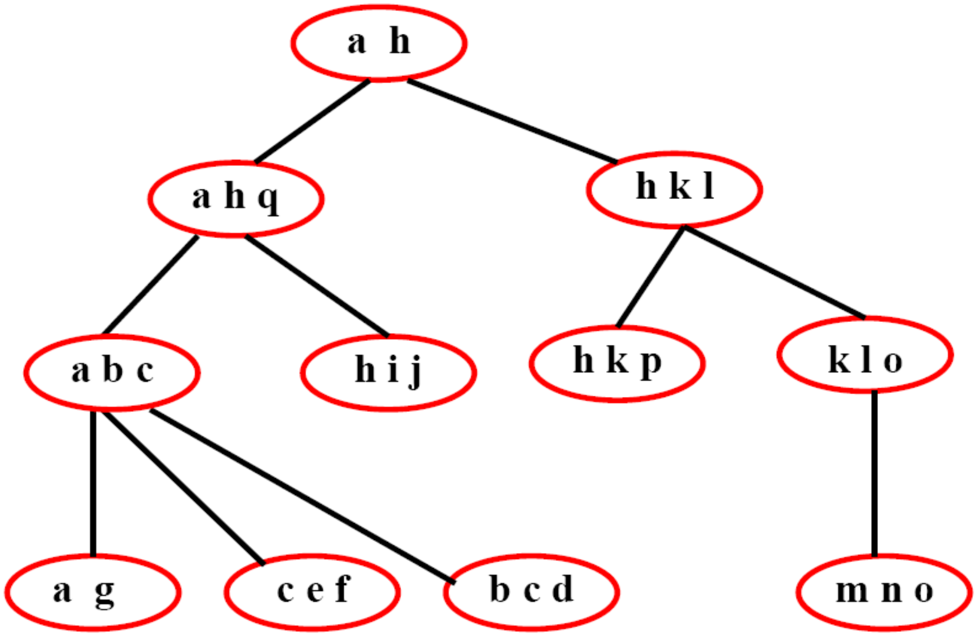


Tree decomposition of width 2 of G

Tree Decompositions [Robertson & Seymour '86]



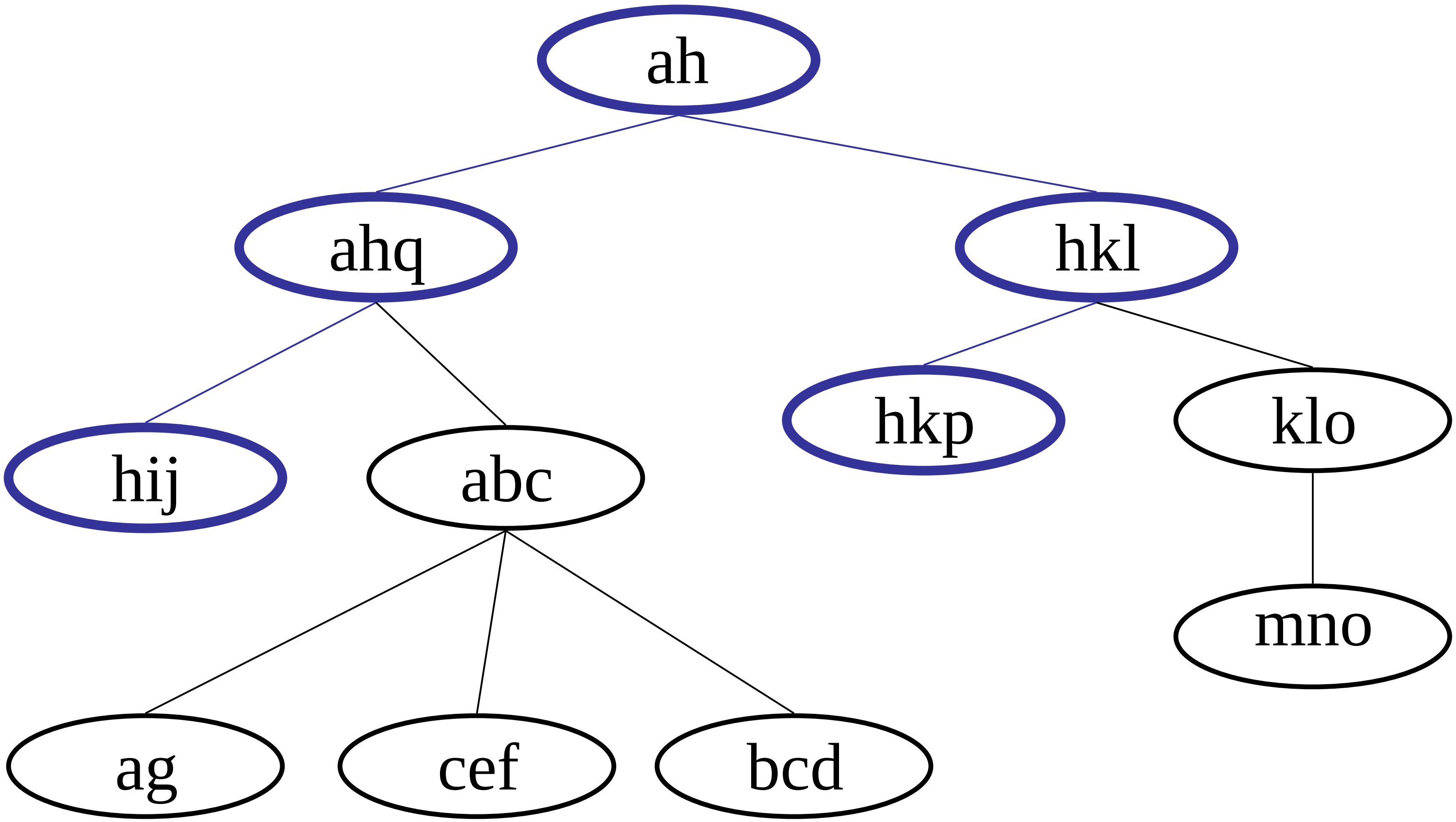
Graph G



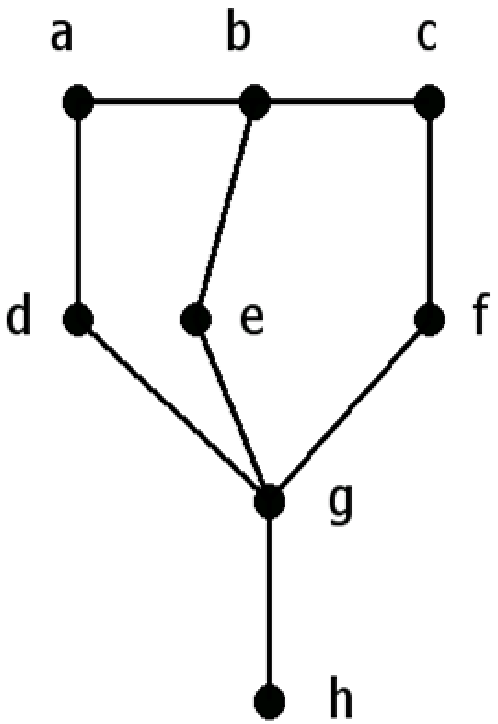
Tree decomposition of width 2 of G

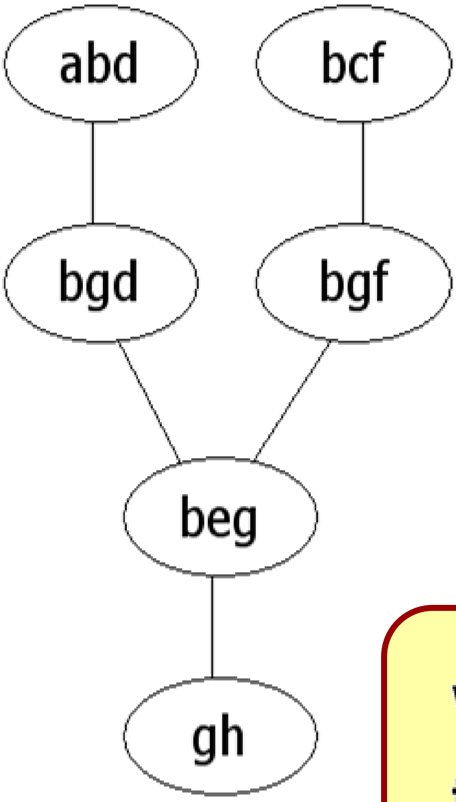
- Every edge realized in some bag
- Connectedness condition

Connectedness condition for *h*



Tree Decompositions and Treewidth





$width(T,X_i)$ = max $|X_i|$ -1

$tw(G)$ = min $width(T,X_i)$

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

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

---

[1] tw is fixed-parameter tractable (parameter: treewidth)

[2] tw is a key for tractability ⇐

Classical Computational Complexity

n = input size

good → poly(n) — P

bad → $2^{O(n)}$ — NP-h

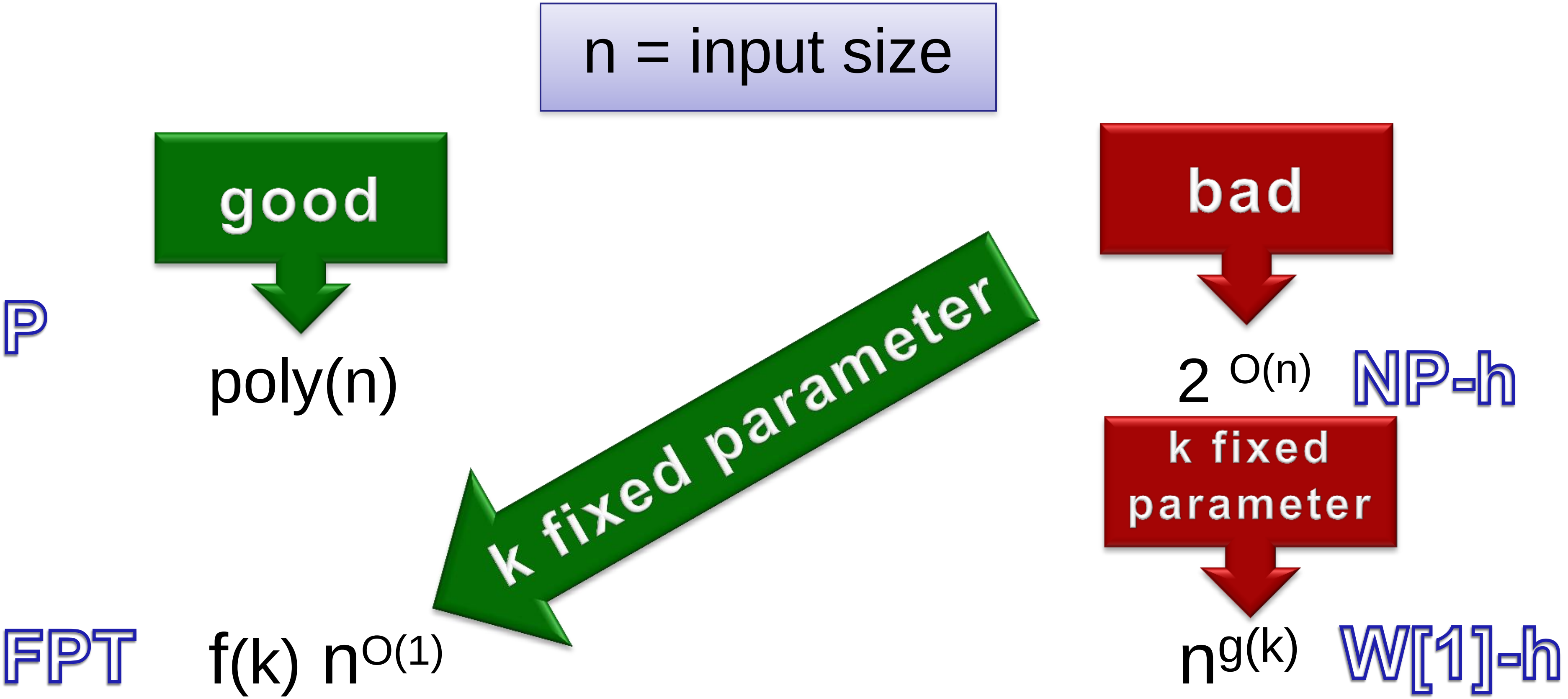
Assuming P≠NP

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



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)
Vertex Cover	1.2738^k
Connected Vertex Cover	2^k
Multiway Cut	2^k
Directed Multiway Cut	$2^{O(k^3)}$
Almost-2-SAT (VC-PM)	4^k
Multicut	$2^{O(k^3)}$
Pathwidth One Deletion Set	4.65^k
Undirected Feedback Vertex Set	3.83^k
Undirected Feedback Vertex Set	3^k
Subset Feedback Vertex Set	$2^{O(k \log k)}$
Directed Feedback Vertex Set	$4^k k!$
Odd Cycle Transversal	3^k
Edge Bipartization	2^k
Planar DS	$2^{11.98\sqrt{k}}$
1-Sided Crossing Min	$2^{O(\sqrt{k} \log k)}$
Max Leaf	3.72^k
Directed Max Leaf	3.72^k
Set Splitting	1.8213^k
Nonblocker	2.5154^k
Edge Dominating Set	2.3147^k
k-Path	4^k
k-Path	1.66^k
Convex Recolouring	4^k
VC-max degree 3	1.1616^k
Clique Cover	2^{2^k}
Clique Partition	2^{k^2}
Cluster Editing	1.62^k
Steiner Tree	2^k
3-Hitting Set	2.076^k

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

---

[1] tw is fixed-parameter tractable (parameter: treewidth)

[2] tw is a key for tractability

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

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

Master Theorems for Treewidth

Arnborg, Lagergren, Seese '91:

Optimization version of Courcelle's Theorem.

Finding an optimal set P such that G |= Φ(P) is FP-linear over inputs G of bounded treewidth.

Example:

Given a graph G=(V,E)

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

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

Optimality (More General)

- $G = \langle (N, E), w_N, w_E \rangle$ is a graph weighted on vertices and edges, and ϕ an **MSO$_2$** formula
- A solution to ϕ is an interpretation (z_N, z_E) (a pair (set of vertices, set of edges)) such that $(N, E) \models \phi[(z_n, z_E)]$ and its cost is $\sum_{x \in z_N} w_N(x) + \sum_{y \in z_E} w_E(y)$.
- A solution of minimum cost is said optimal

Theorem (simplified from Arnborg et al., 1991)

Let ϕ be a fixed **MSO$_2$** *sentence and let $G = \langle (N, E), f_N, f_E \rangle$ be a weighted graph such that $(N, E) \in \mathcal{C}_k$. Then, computing an optimal solution to ϕ over G is feasible in polynomial time (w.r.t. $||G||$).*

Outline

- Identification of “Easy” Classes
- Applications of Tree Decompositions
- Beyond Tree Decompositions
- Decision/Computation Problems
- Optimization Problems
- Enumeration Problems

The Model

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

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

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 $X(\mathcal{G})$

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

- *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

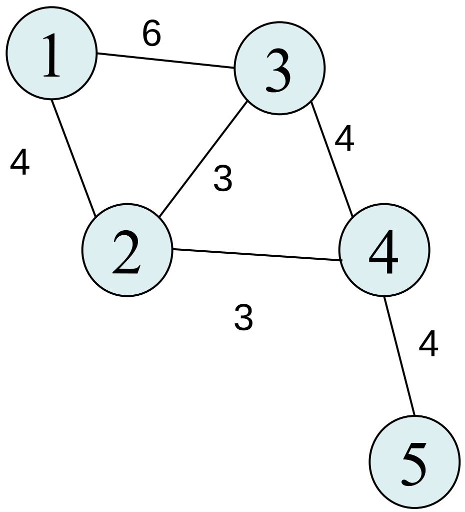
$$\mathcal{G} = \langle N, v \rangle,\ v : 2^N \mapsto \mathbb{R}$$

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

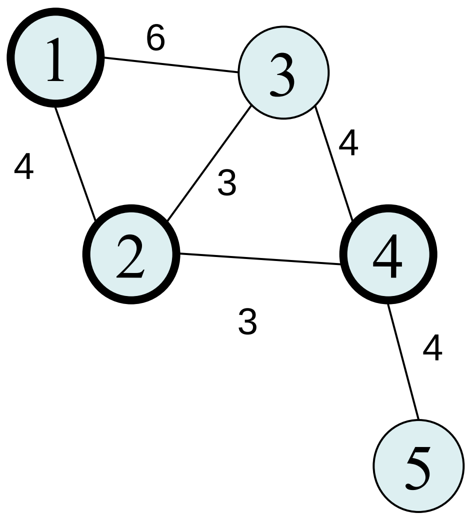
The Core: $\forall S \subseteq N, x(S) \geq v(S);$

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

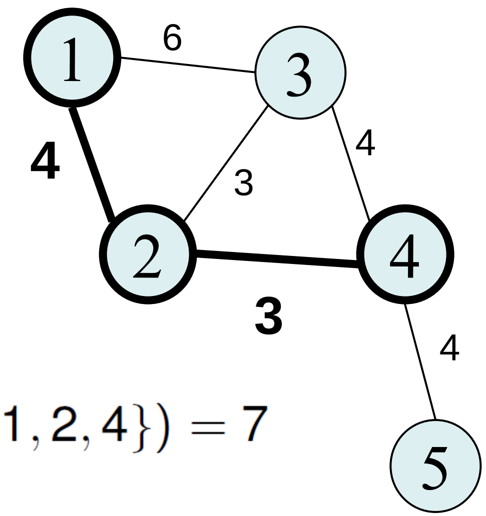
Compact Games



Compact Games

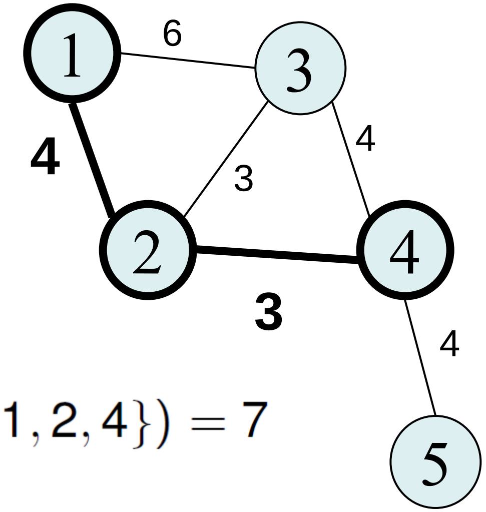


Compact Games



$v(\{1,2,4\}) = 7$

Compact Games



$v(\{1, 2, 4\}) = 7$

- *Graph Games* [Deng and Papadimitriou, 1994]
  - Computational issues of several solution concepts

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* :

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

…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

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

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

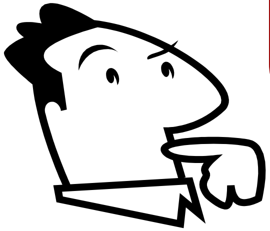
- Identification of “Easy” Classes
- Applications of Tree Decompositions
- Beyond Tree Decompositions
- Decision/Computation Problems
- Optimization Problems
- Enumeration Problems

Beyond Treewidth

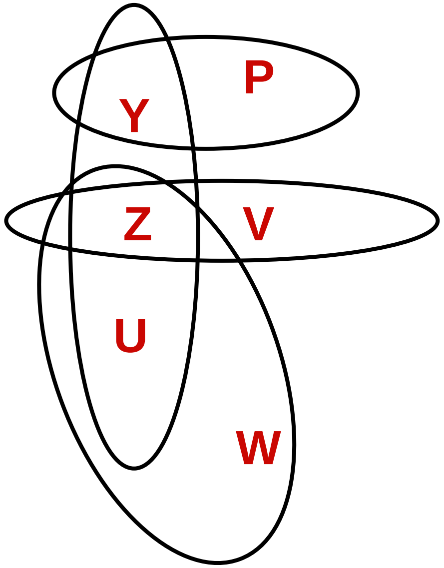
- Treewidth is currently the most successful measure of graph cyclicity. It subsumes most other methods.
- However, there are “simple” graphs that are heavily cyclic. For example, a clique.

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…

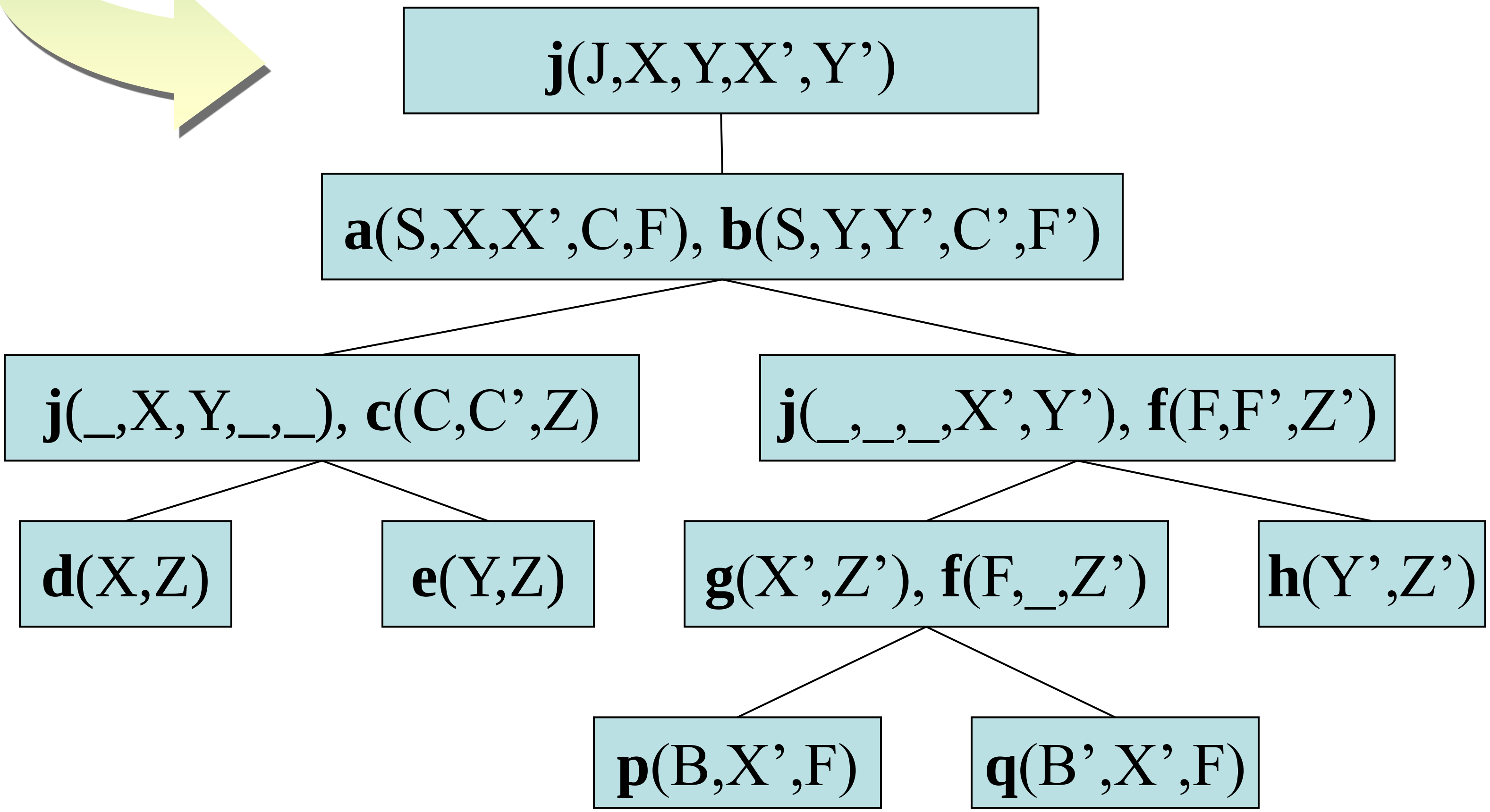


Generalized Hypertree Decompositions

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

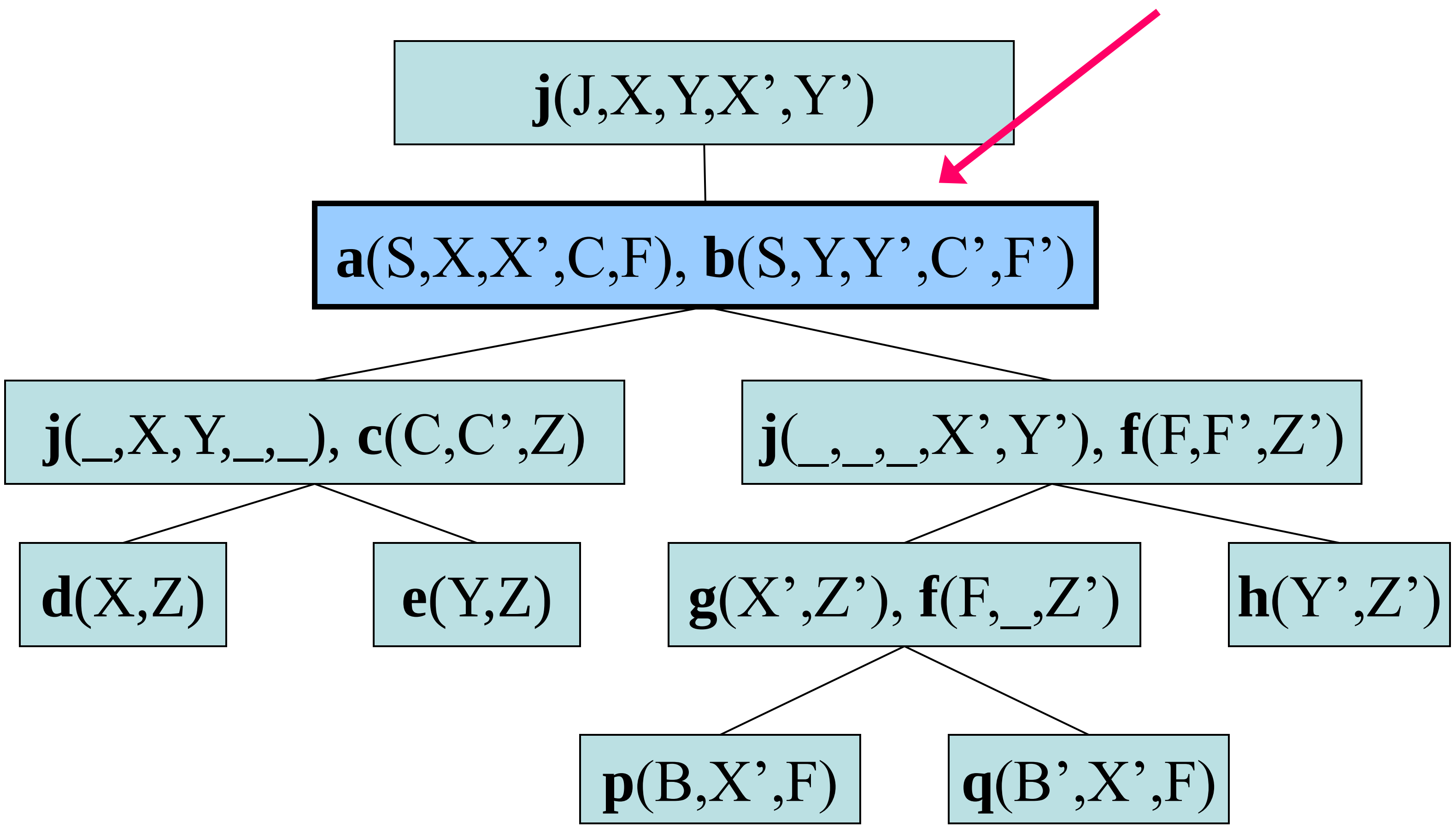
$e(Y,Z)$ $f(F,F',Z')$ $g(X',Z')$ $h(Y',Z')$

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

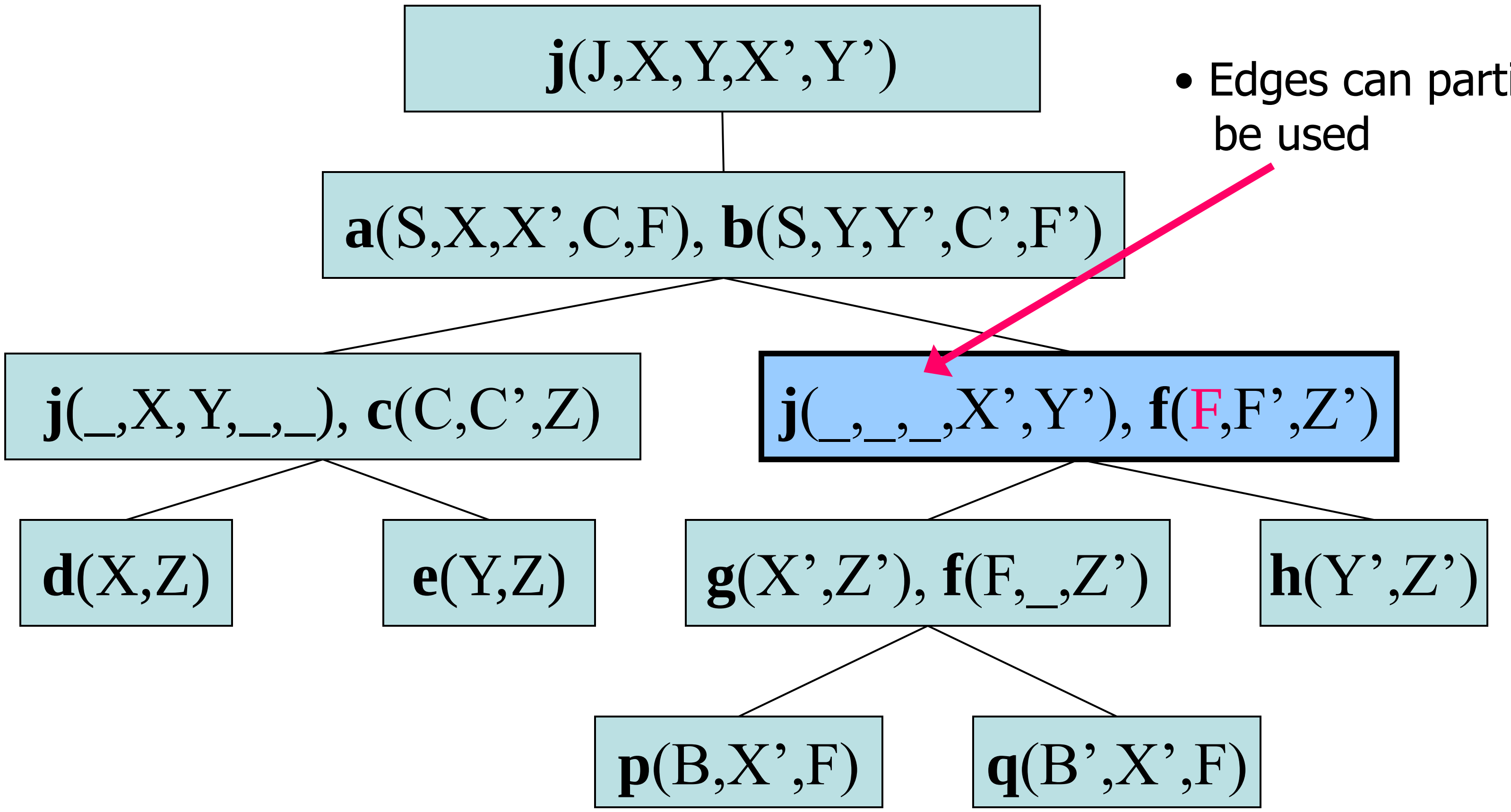


Basic Conditions(1/2)

- We group edges

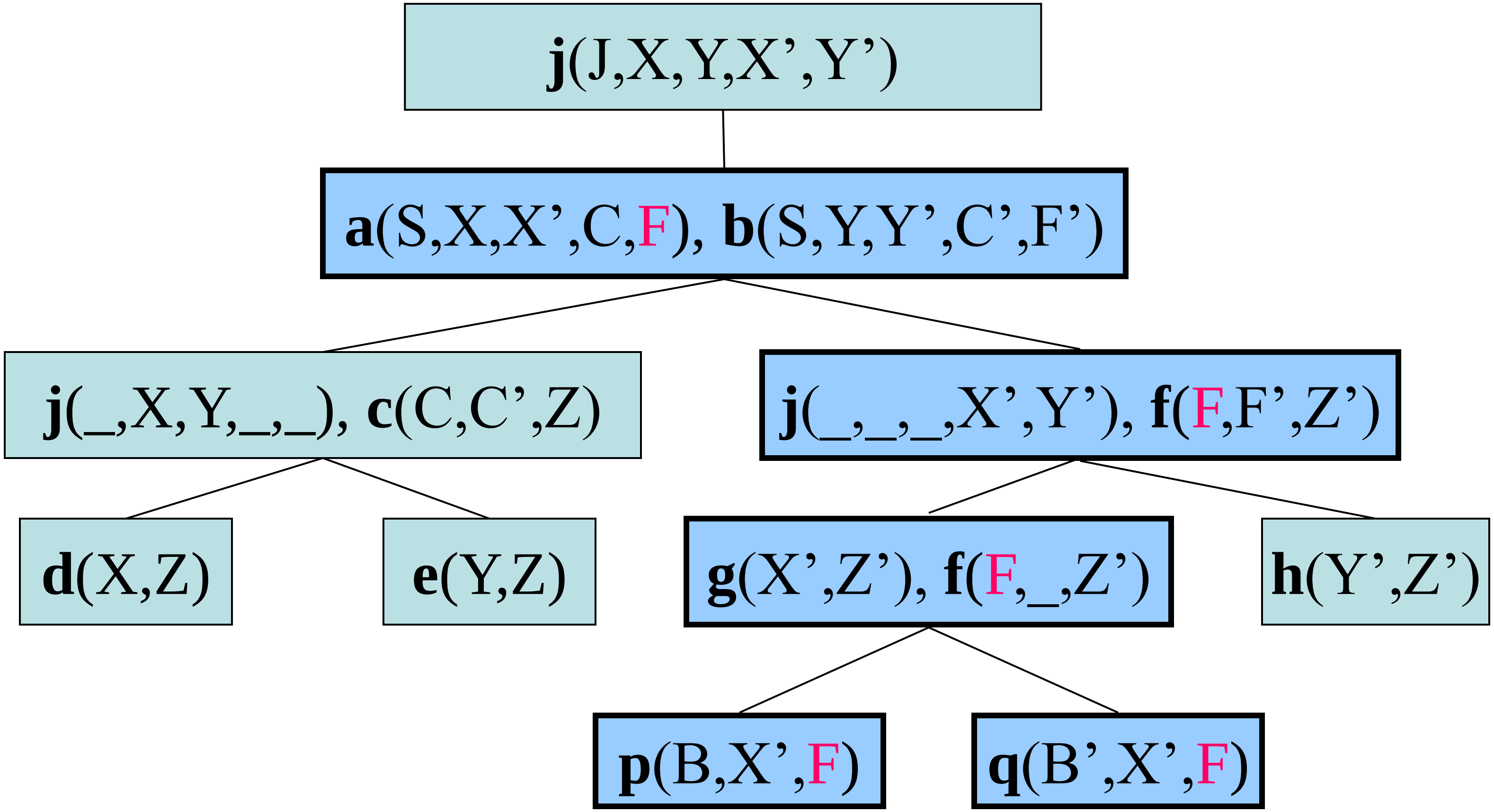


Basic Conditions(2/2)



- Edges can partially be used

Connectedness Condition

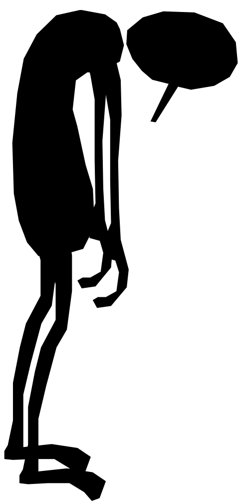


Computational Question

- 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 ?



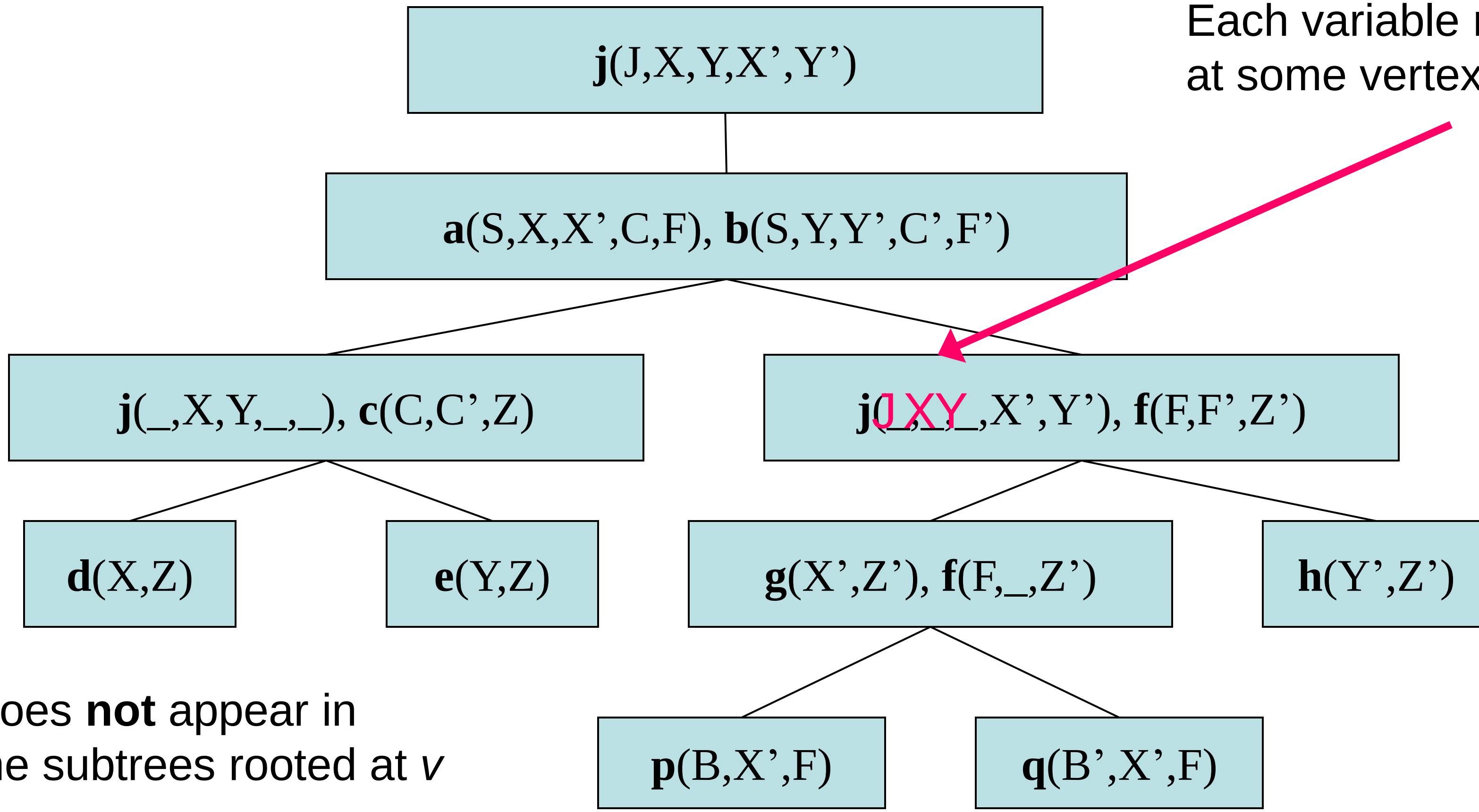
Bad news: ghw(H) < 4? NP-complete

[Schwentick et. al. 06]

Hypertree Decomposition (HTD)

HTD = Generalized HTD +Special Condition

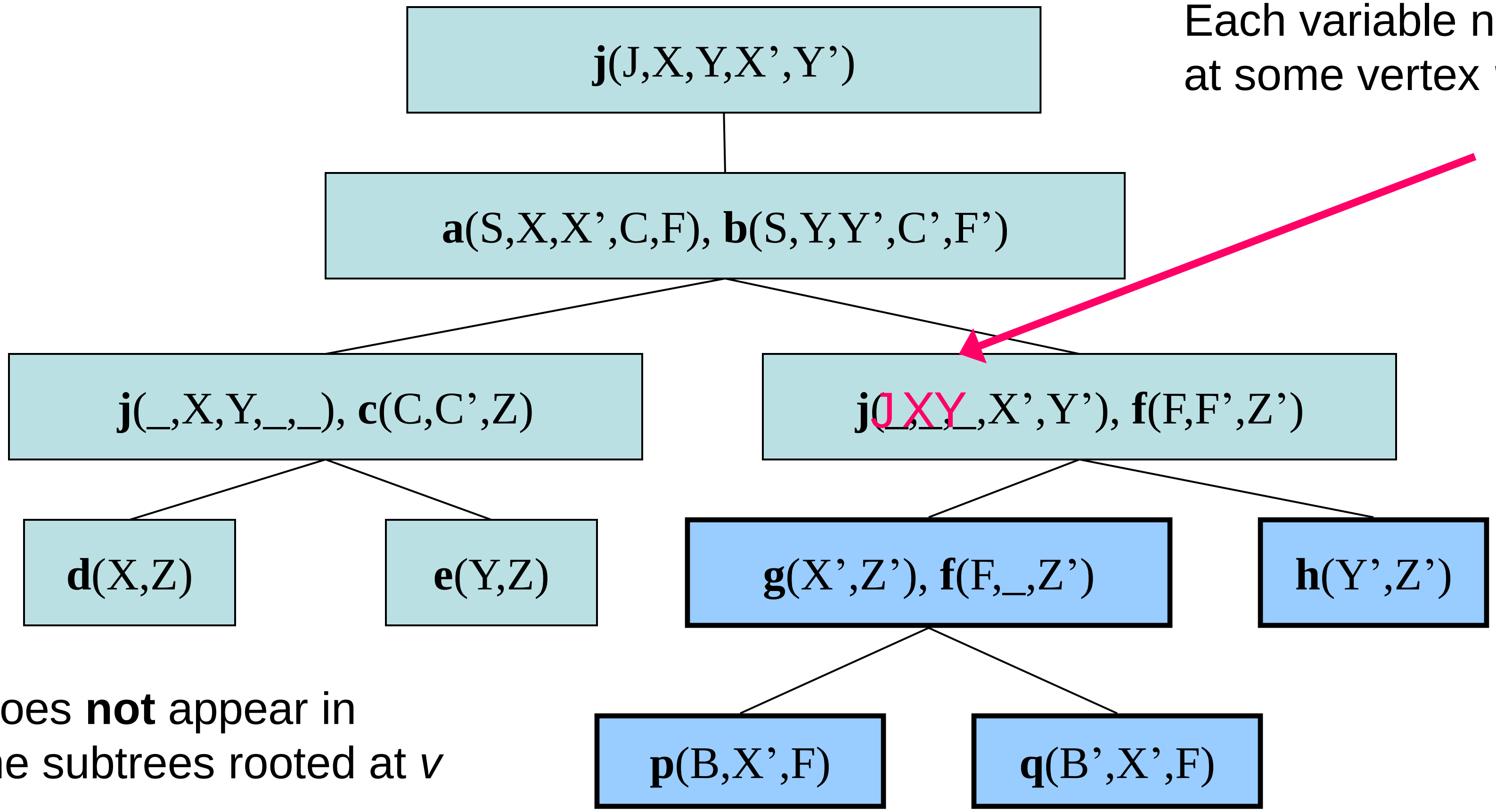
Each variable not used at some vertex *v*



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

Special Condition

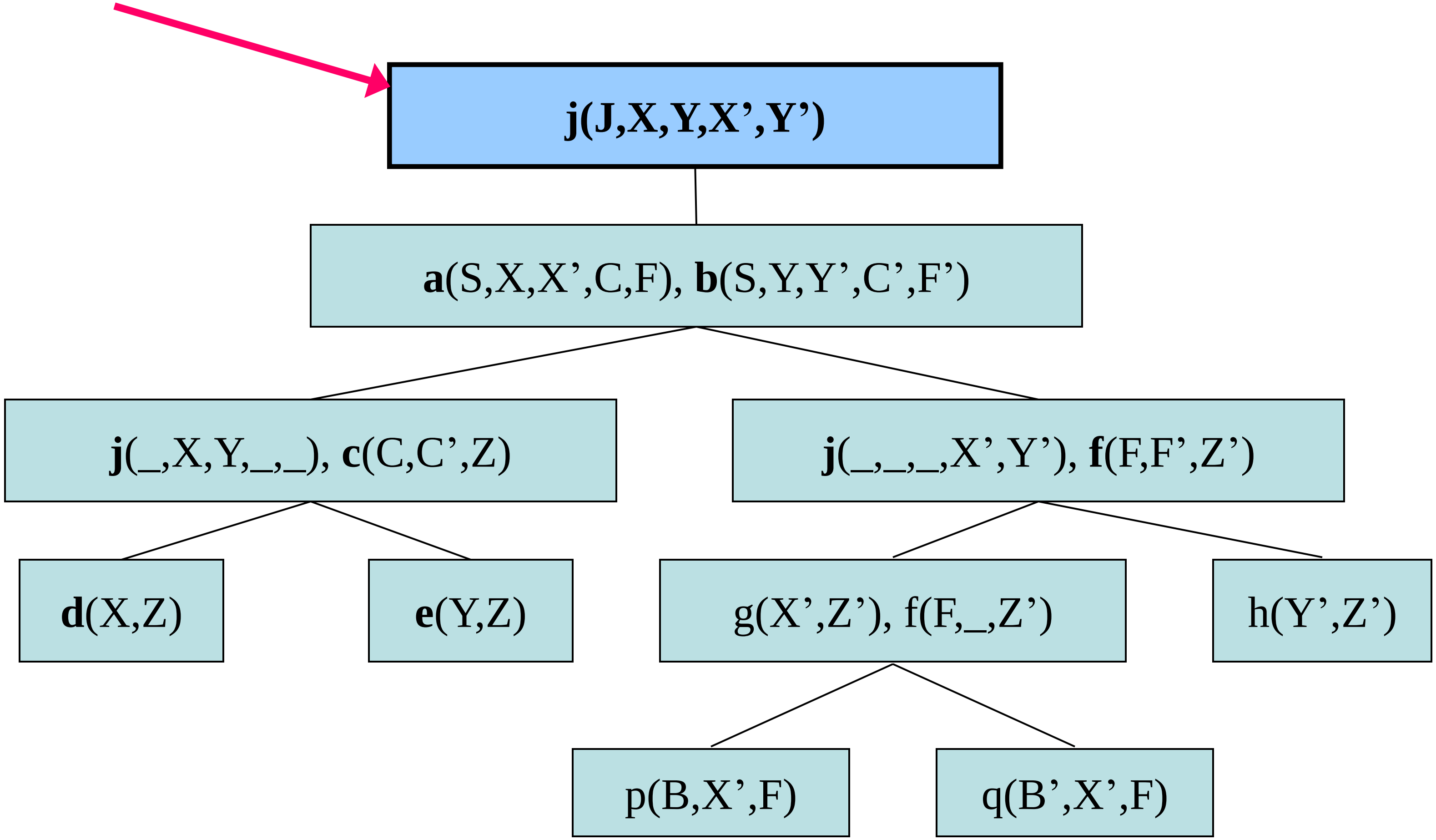
Each variable not used at some vertex *v*



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 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] :

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

Outline

- Identification of “Easy” Classes
- Applications of Tree Decompositions
- Beyond Tree Decompositions
- Decision/Computation Problems
- Optimization Problems
- Enumeration Problems

Three 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 $\mathbb{B}$

$$h : U \longrightarrow V$$

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

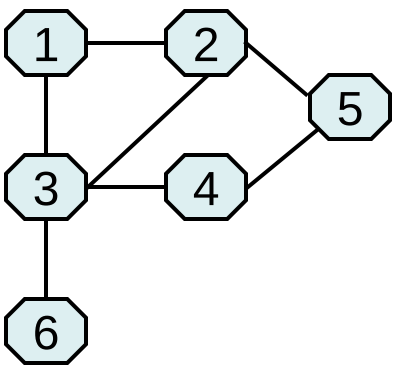
$$\mathbf{x} \in R_i \implies h(\mathbf{x}) \in S_i$$

HOM is NP-complete

(well-known, independently proved in various contexts)

Membership: Obvious, guess *h.*

Hardness: Transformation from 3COL.



$\mathbb{A}$

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

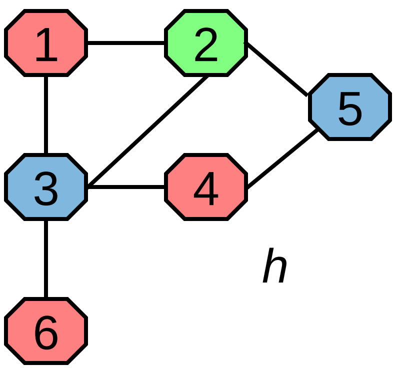
Graph 3-colourable *iff* HOM(*A*,*B*) yes-instance.

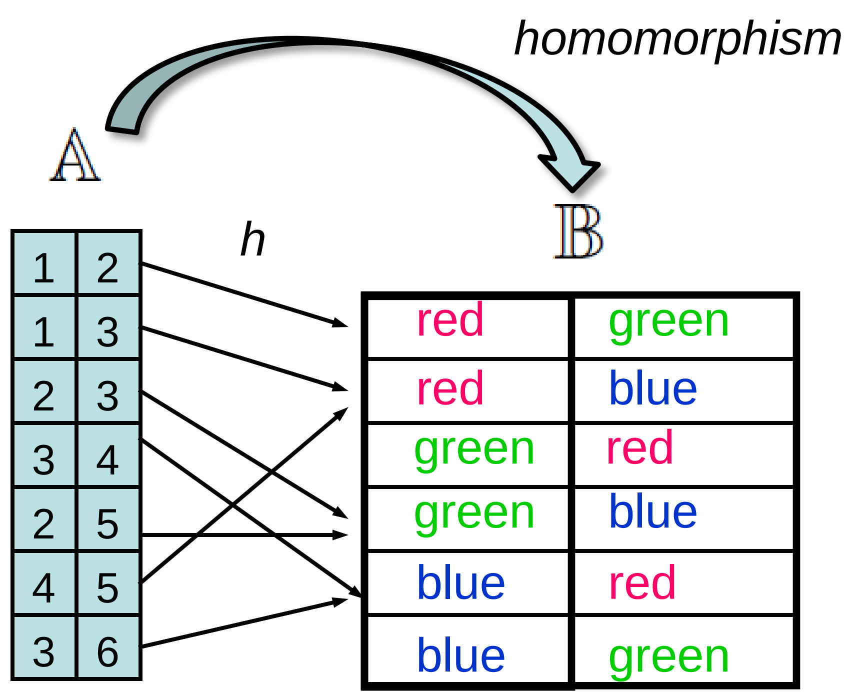
HOM is NP-complete

(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:

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
………	…..

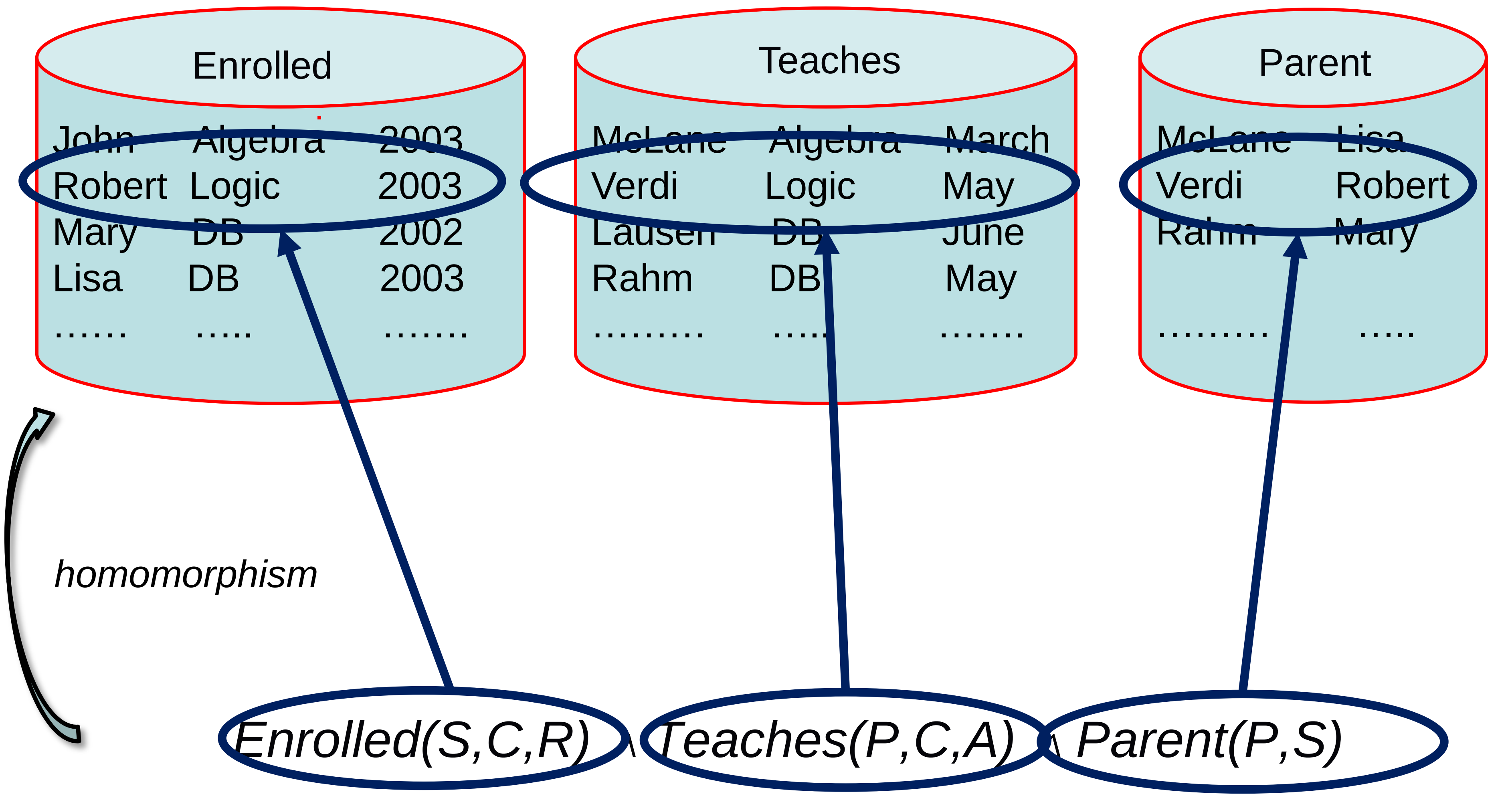
QUERY:

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

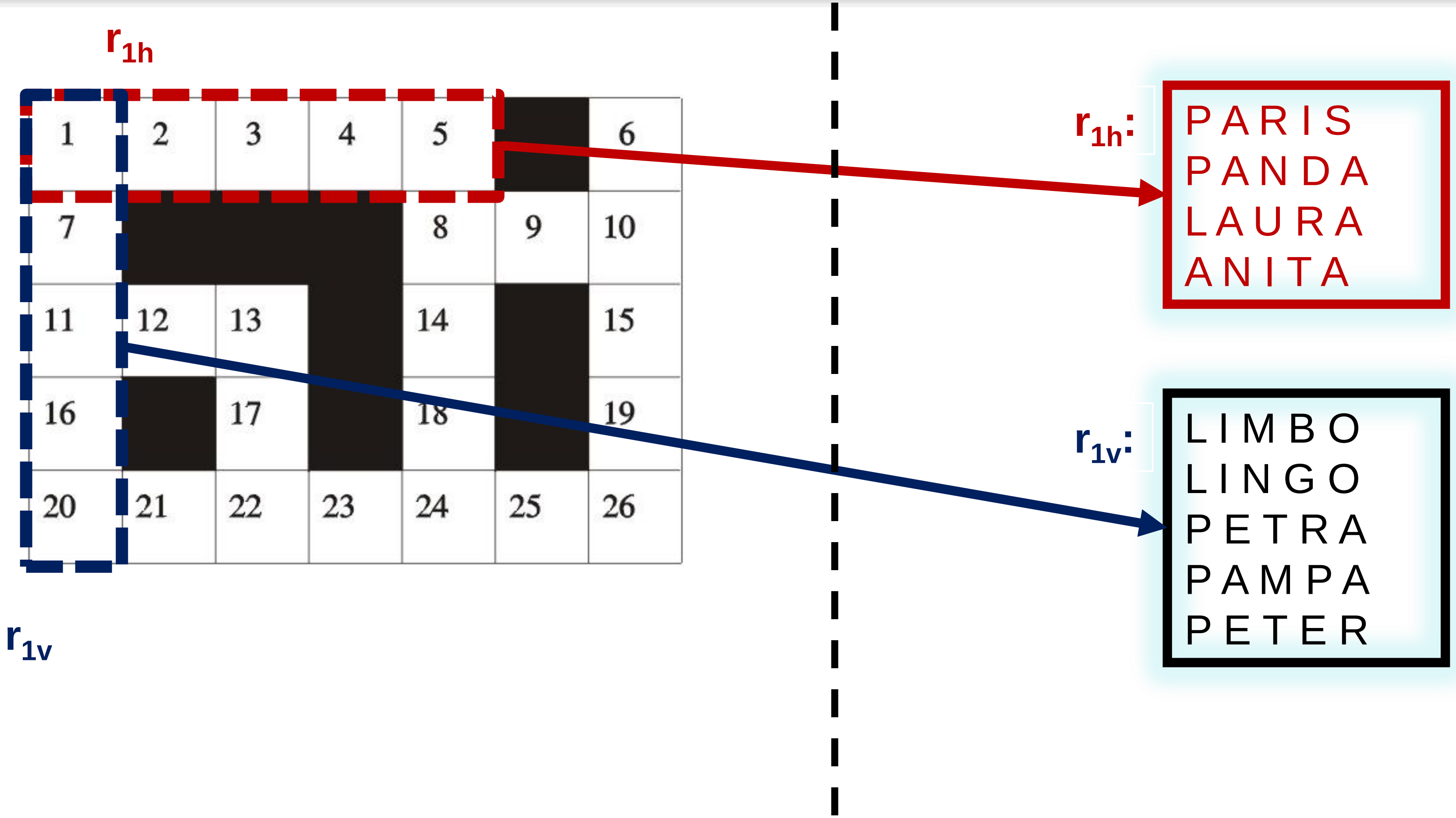
$$ans \leftarrow Enrolled(S,C,R) \wedge Teaches(P,C,A) \wedge Parent(P,S)$$

Conjunctive Database Queries

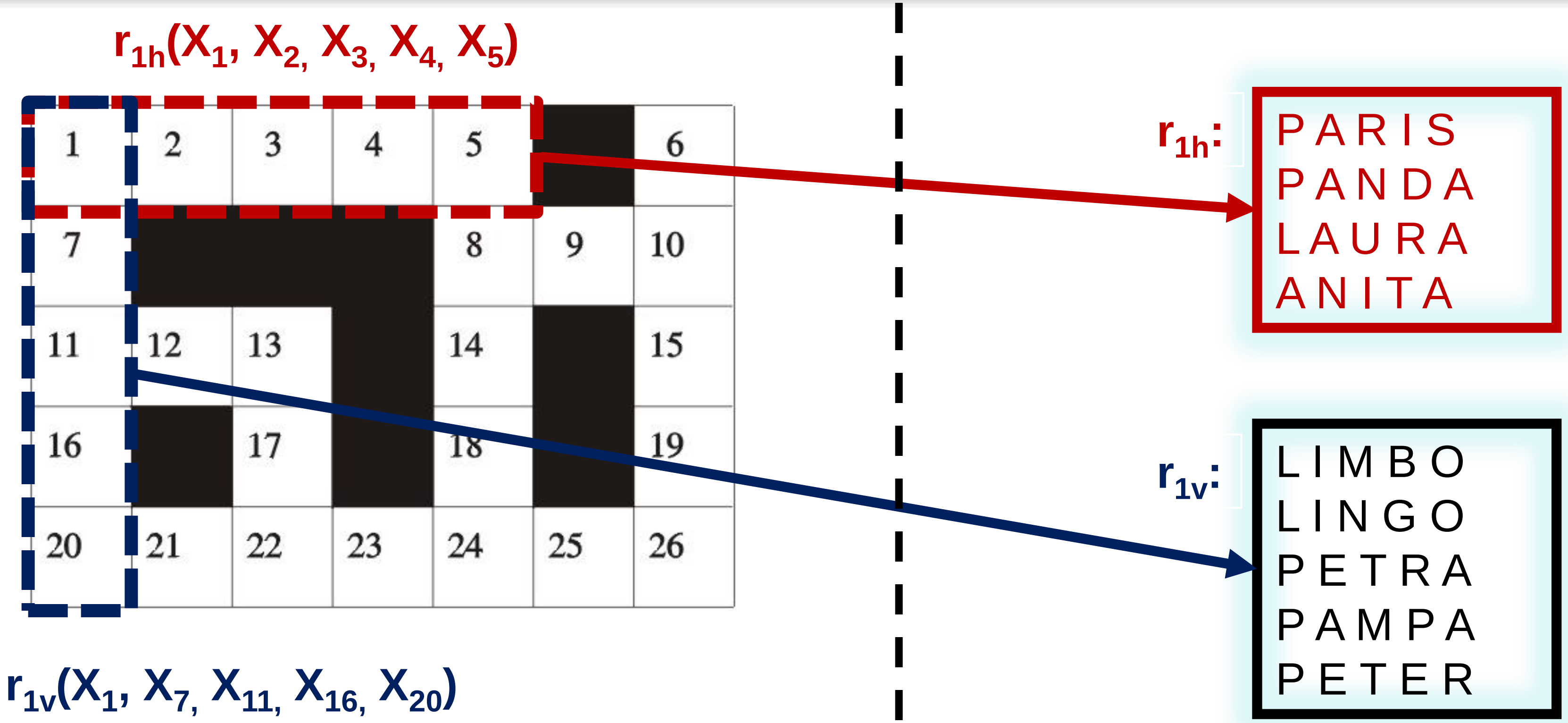
DATABASE:



CSPs as Homomorphism Problems



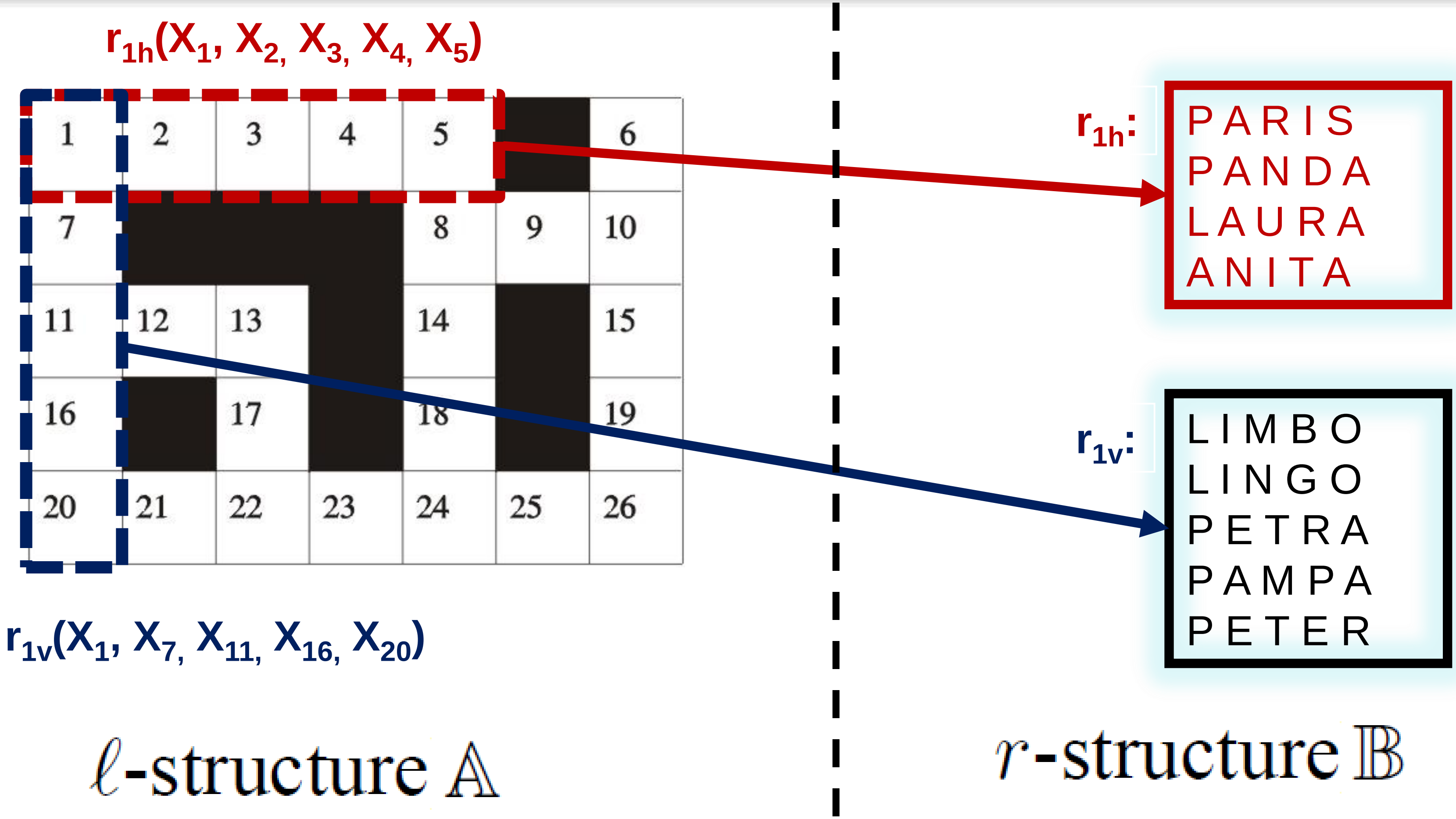
CSPs as Homomorphism Problems



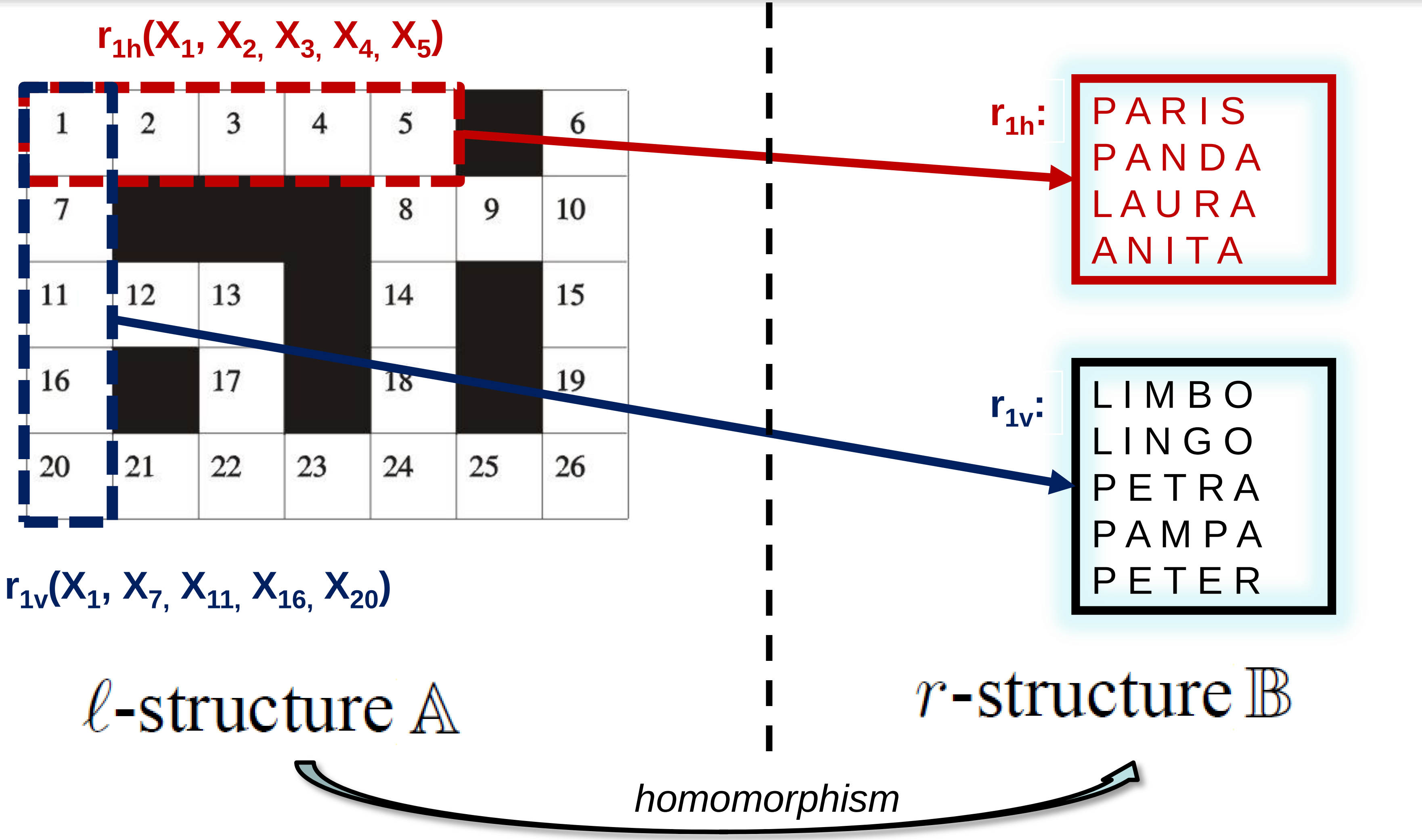
- Set of variables $\{X_1,\ldots,X_{26}\}$
- Set of constraint scopes

- Set of constraint relations

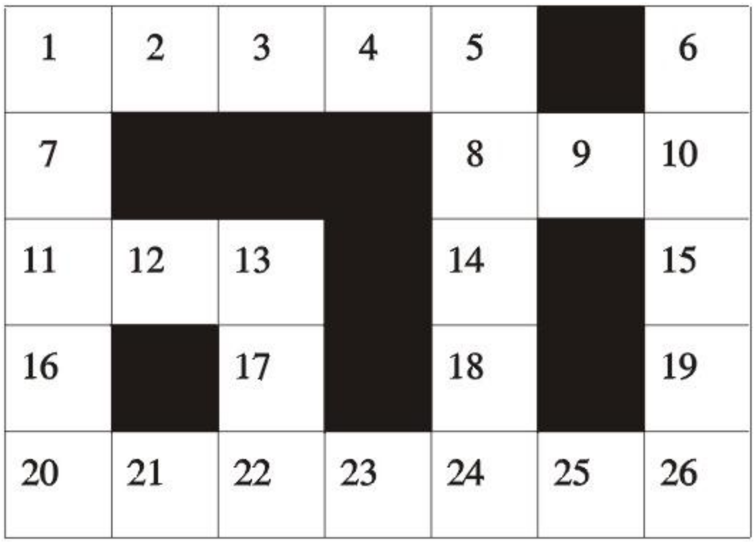
CSPs as Homomorphism Problems

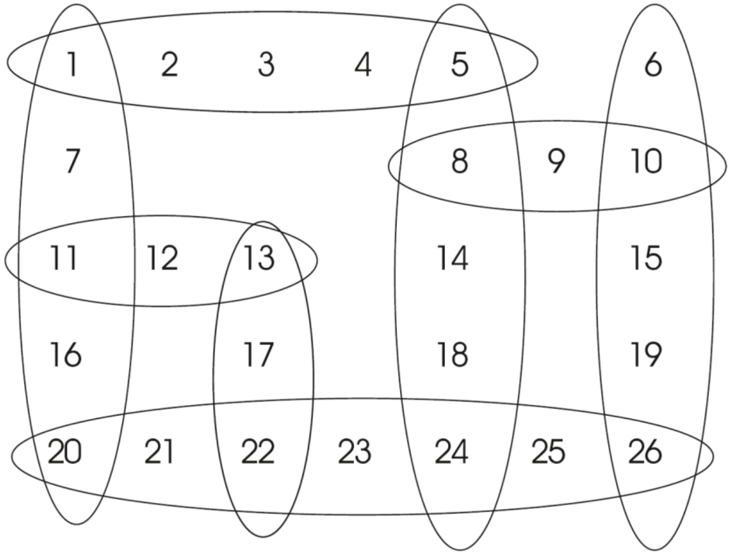


CSPs as Homomorphism Problems



CSPs and Hypergraphs



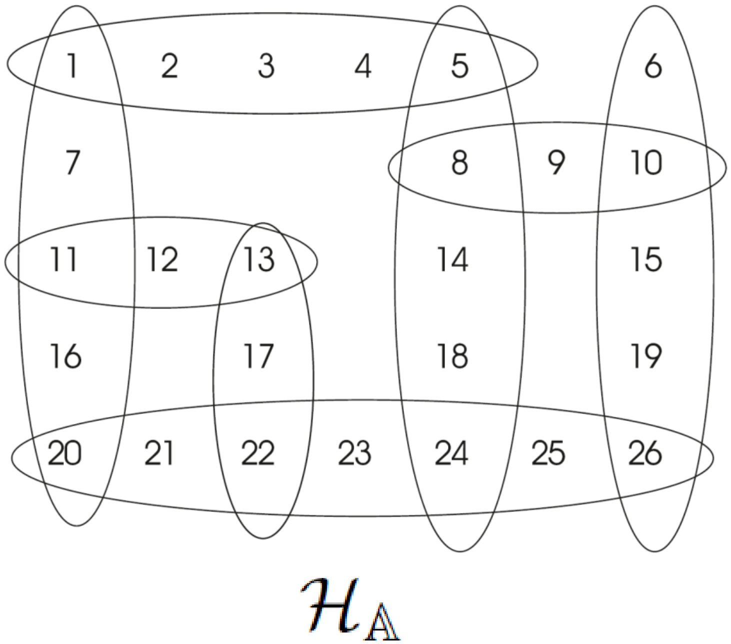


$\mathcal{H}_{\mathbb{A}}$

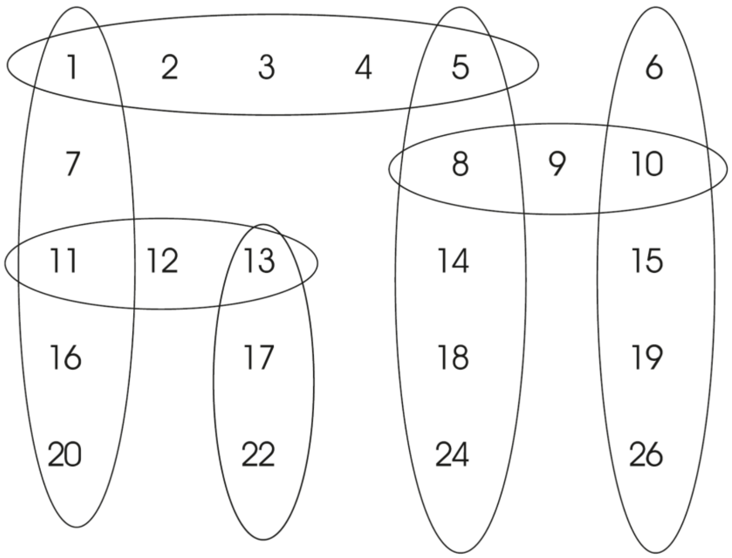
ℓ-structure $\mathbb{A}$

- Variables map to nodes
- Scopes map to hyperedges

Structurally Restricted CSPs



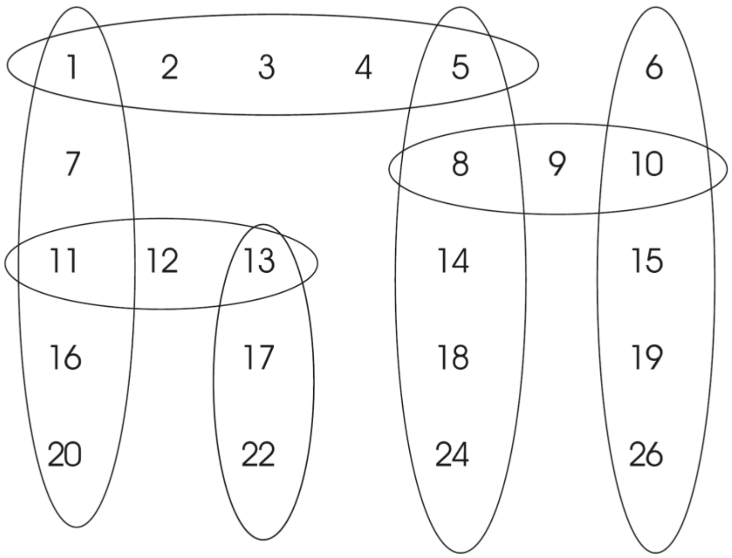
Structurally Restricted CSPs



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

Structurally Restricted CSPs

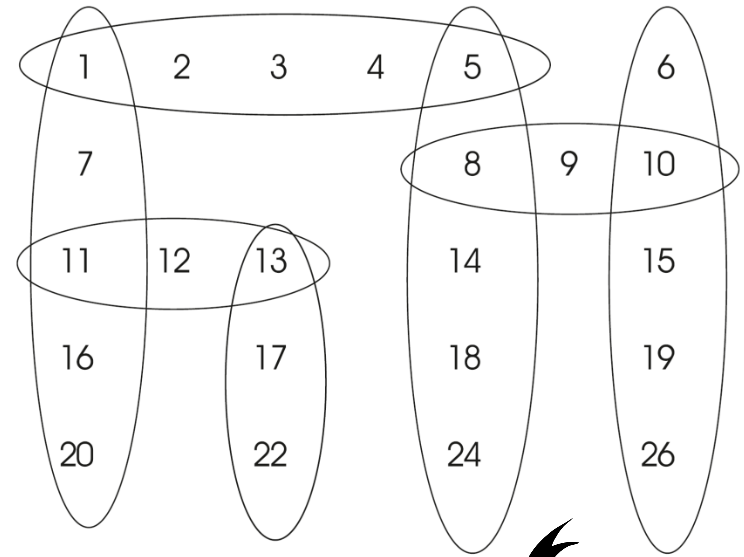
The hypergraph is acyclic



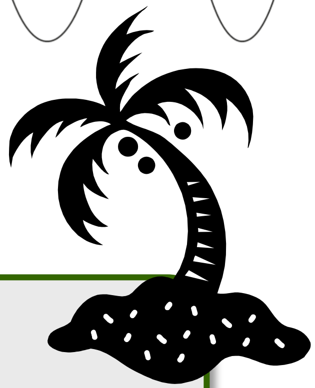
$\mathcal{H}_{\mathbb{A}}$

Structurally Restricted CSPs

The hypergraph is acyclic

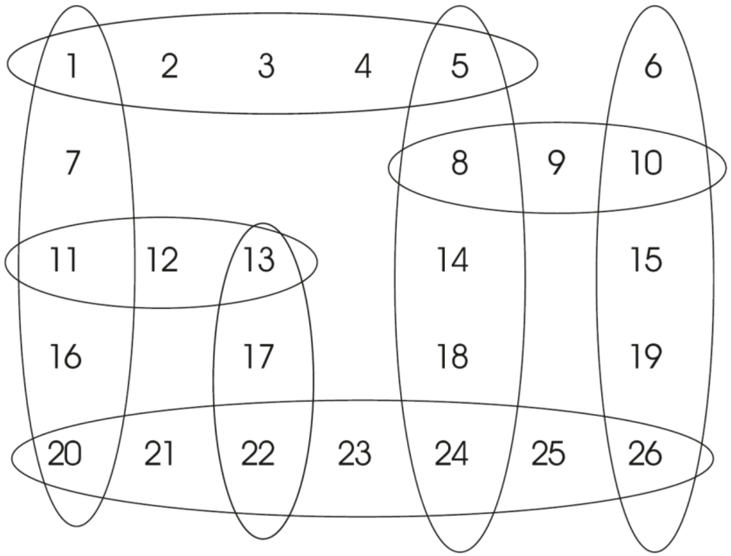


$\mathcal{H}_{\mathbb{A}}$



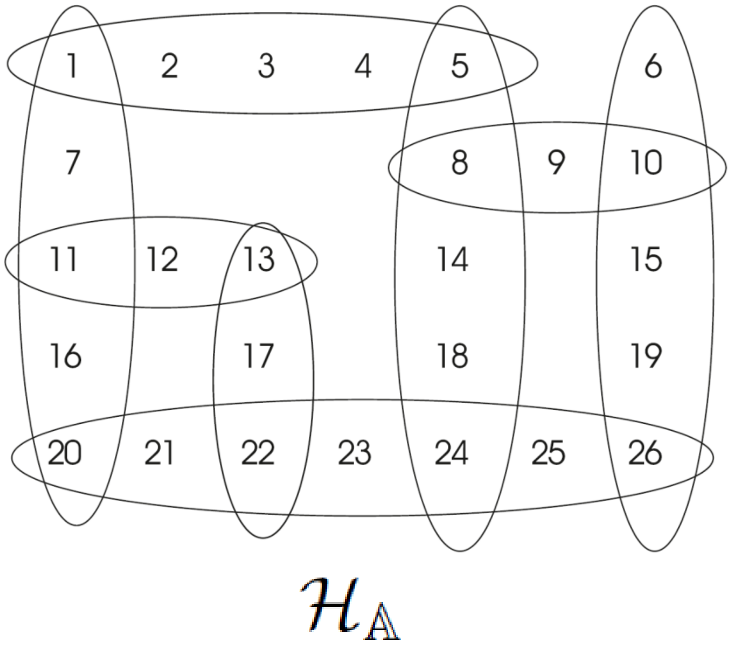
- We have seen that *Acyclicity is efficiently recognizable*
- We shall see that *Acyclic CSPs can be efficiently solved*

Decomposition Methods



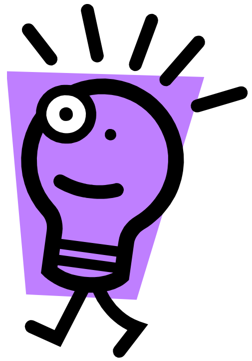
$\mathcal{H}_{\mathbb{A}}$

Decomposition Methods

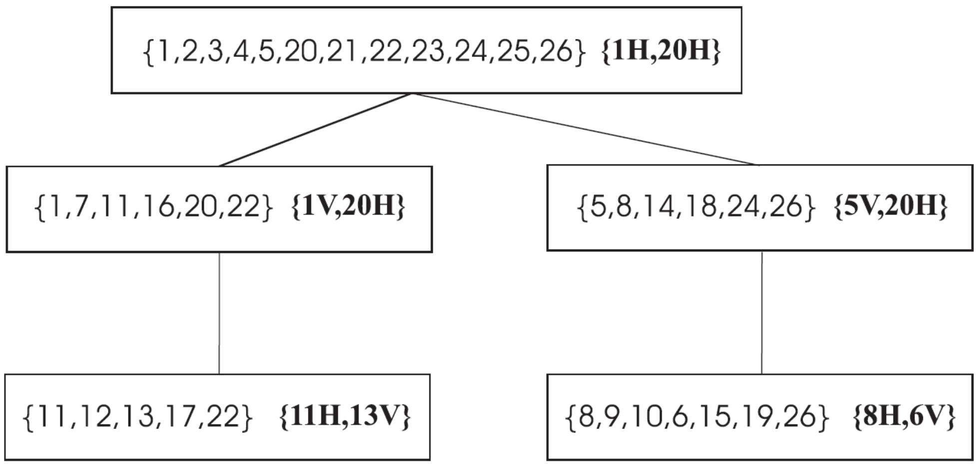


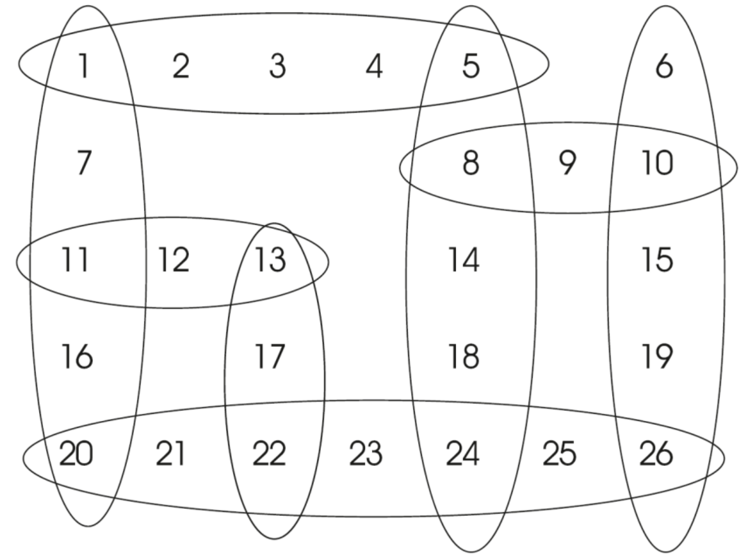
Transform the hypergraph into an acyclic one:

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



Generalized Hypertree Decompositions

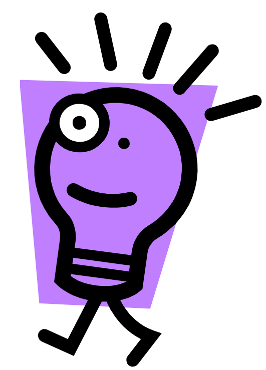




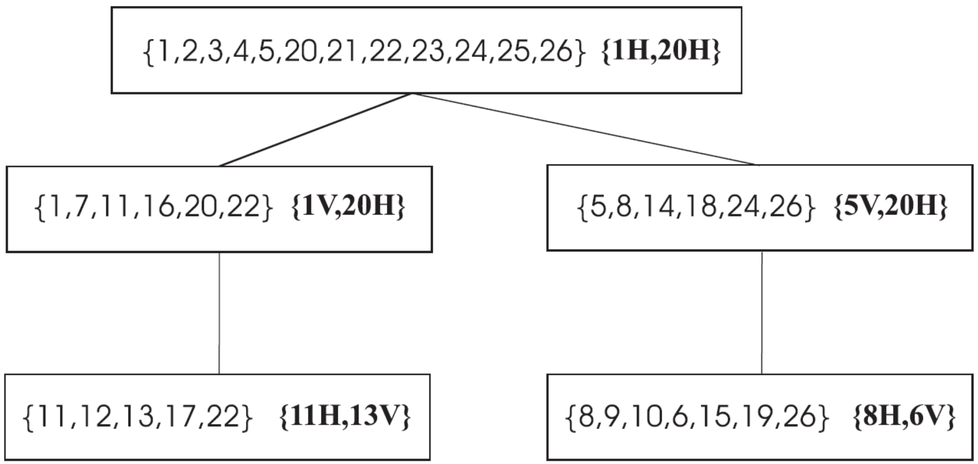
$\mathcal{H}_{\mathbb{A}}$

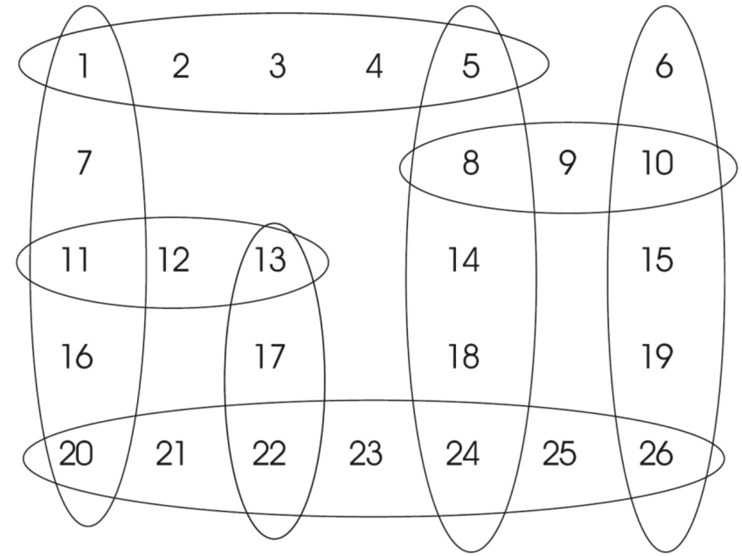
Transform the hypergraph into an acyclic one:

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



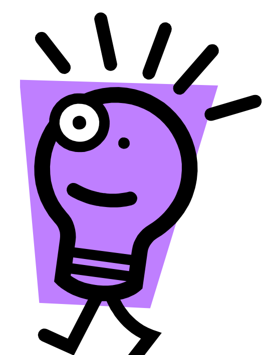
Generalized Hypertree Decompositions





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

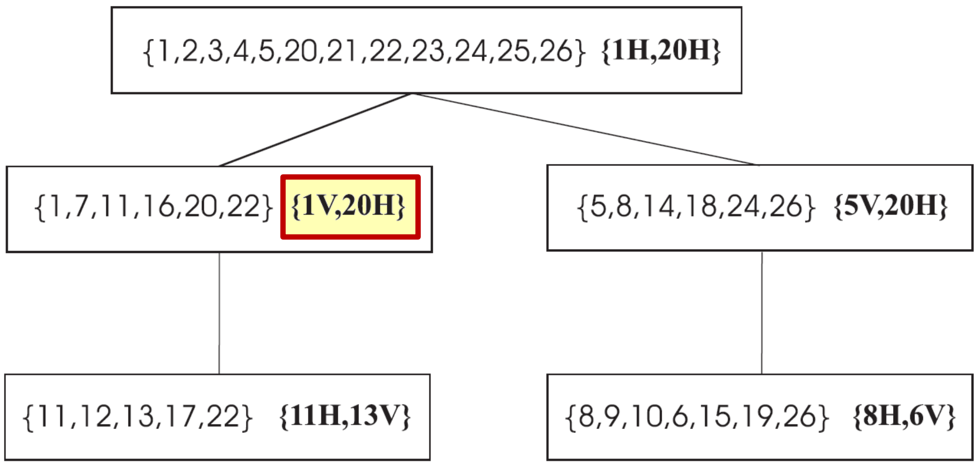
Each cluster can be seen as a **subproblem**

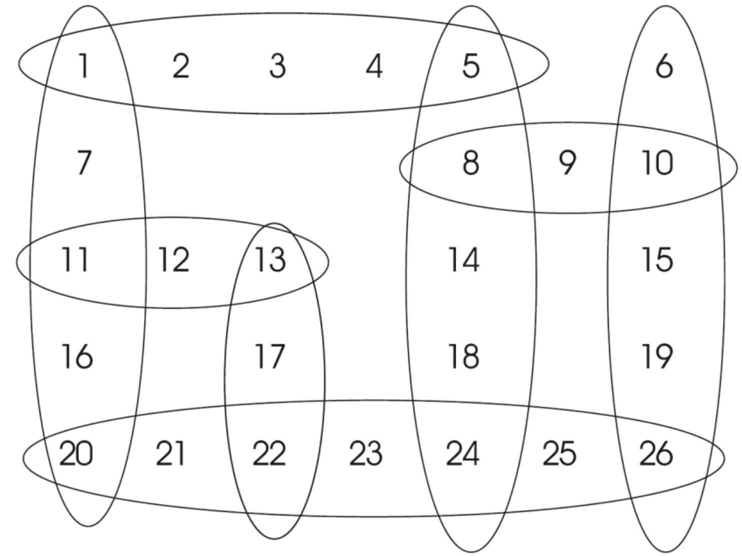


Transform the hypergraph into an acyclic one:

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

Generalized Hypertree Decompositions



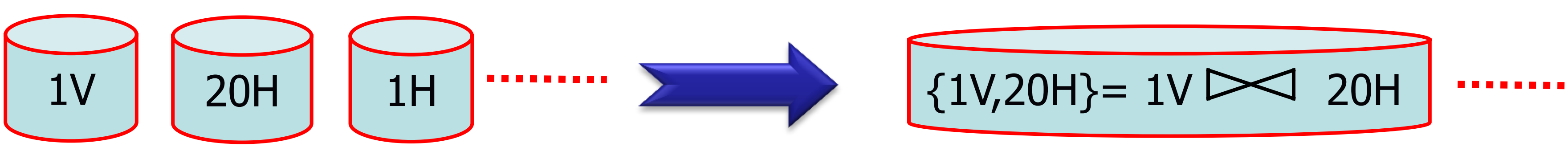


Each cluster can be seen as a **subproblem**

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

Relations:

Relations:



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]

A Polynomial-time Algorithm

HOM: The homomorphism problem

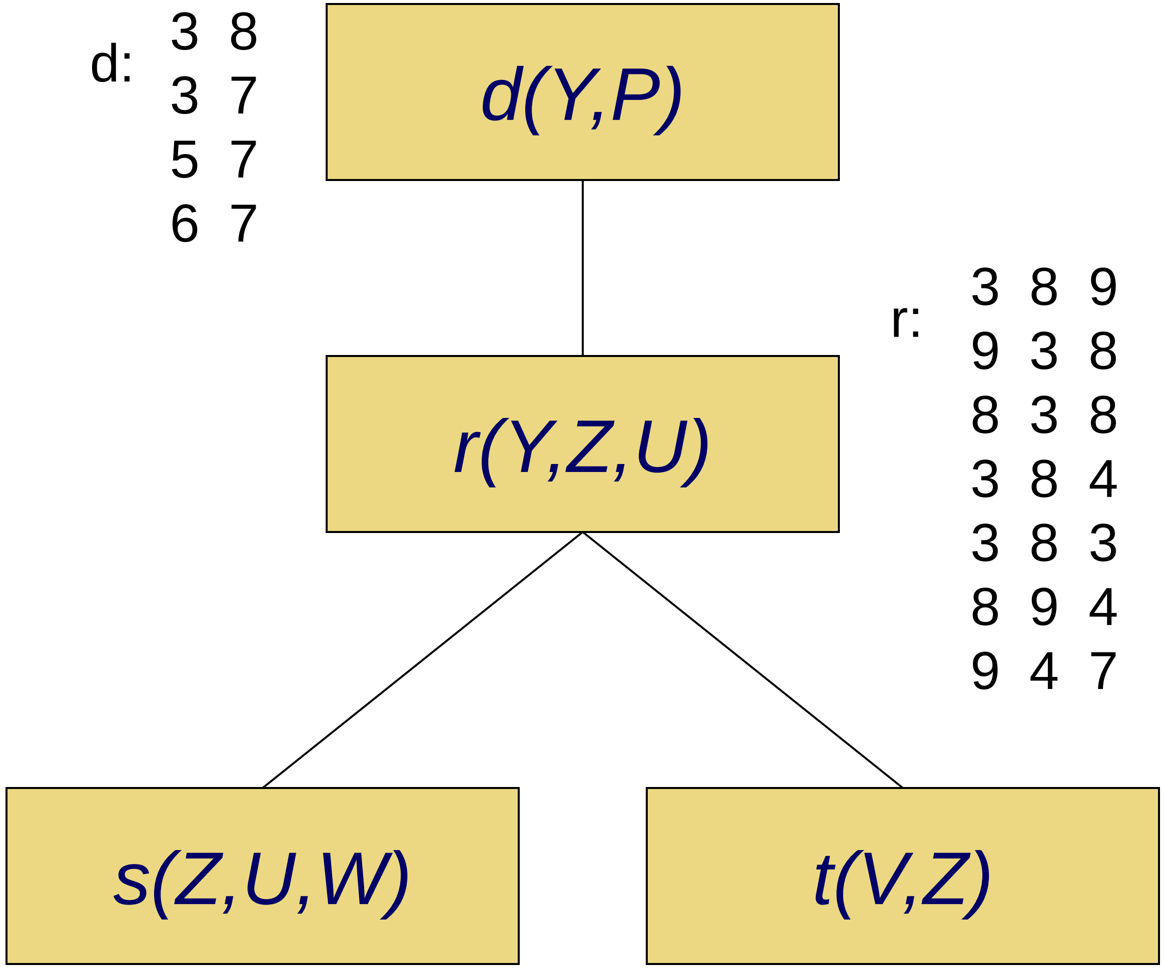
BCQ: Boolean conjunctive query evaluation

CSP: Constraint satisfaction problem

Yannakakis's Algorithm (ABCQs):
Dynamic Programming over a Join Tree

d:

3	8
3	7
5	7
6	7



r:

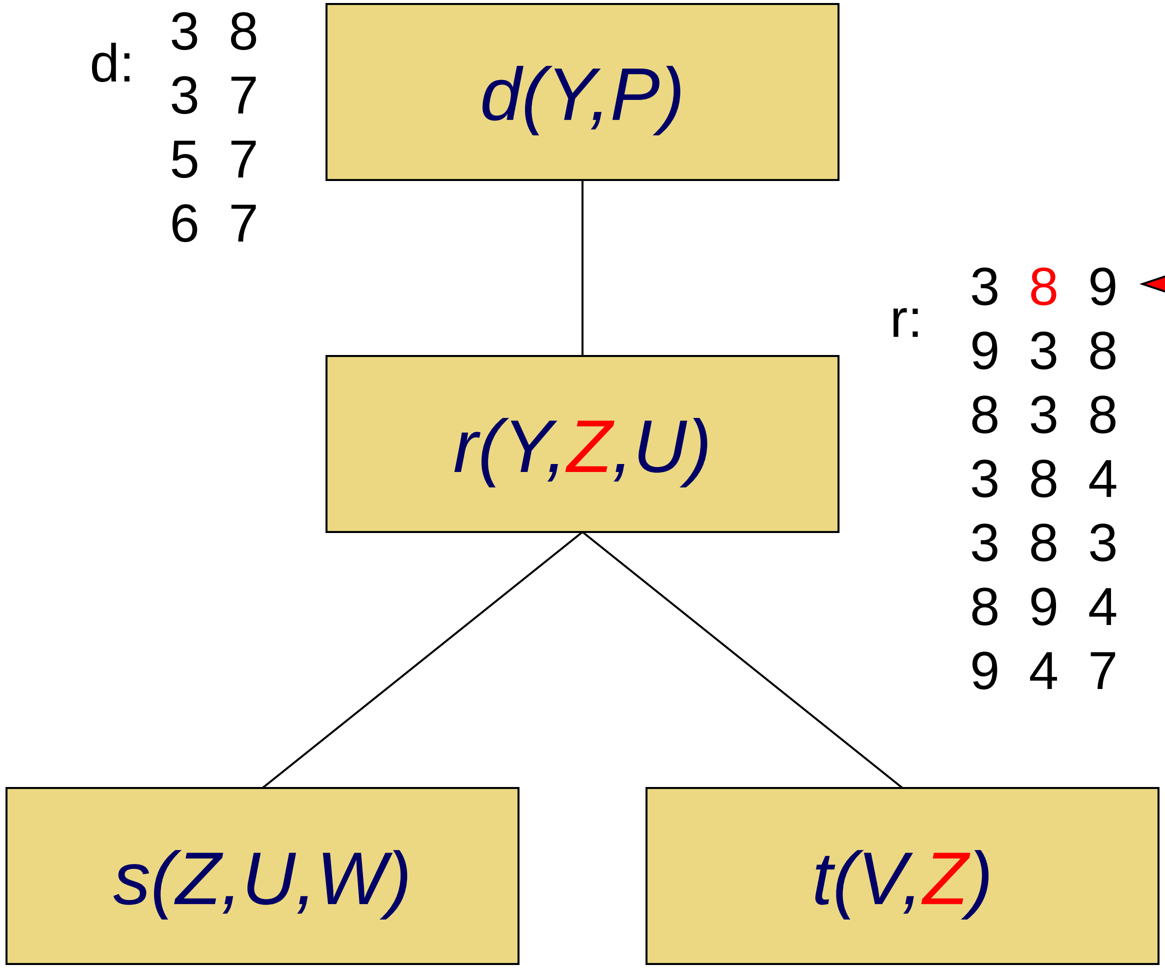
3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

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



d:

3	8
3	7
5	7
6	7

r:

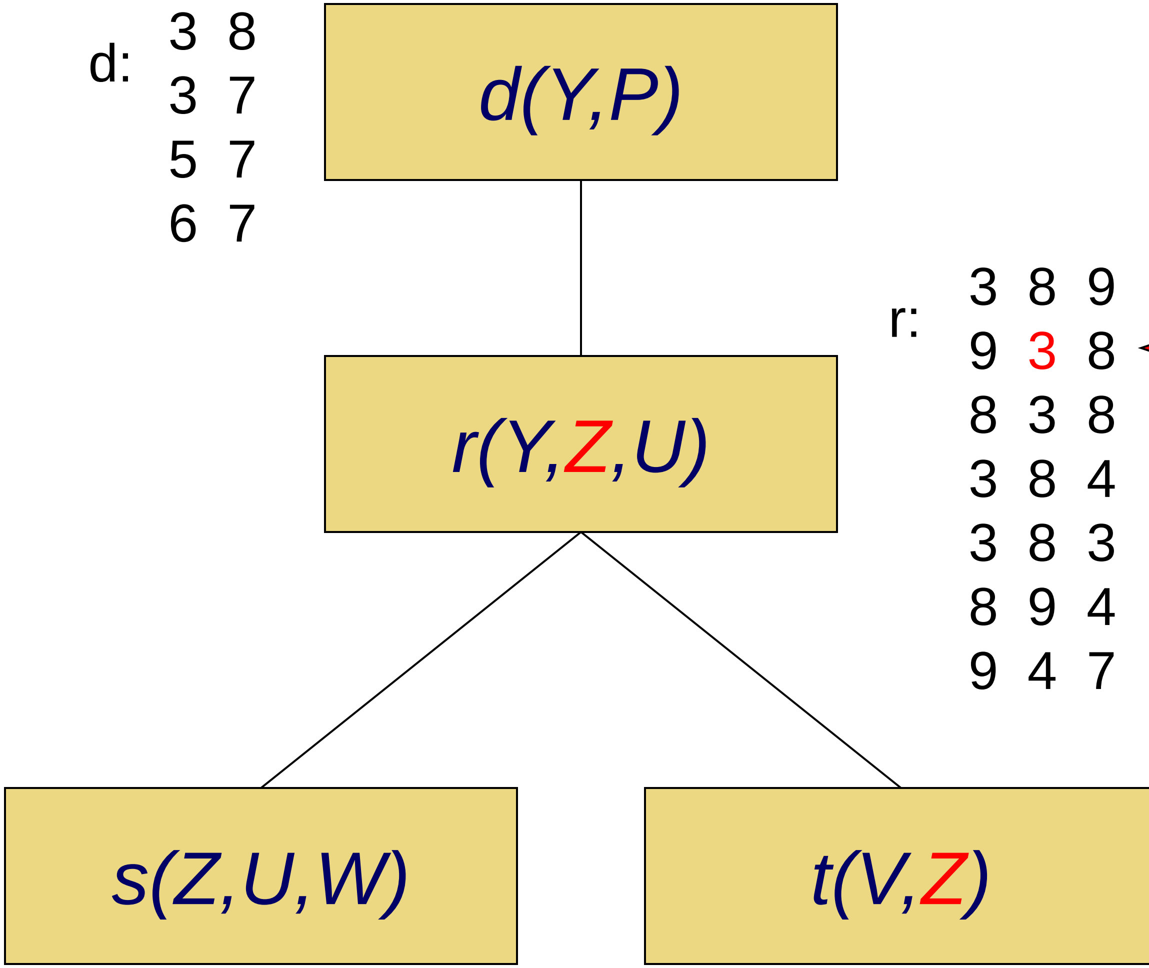
3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

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:

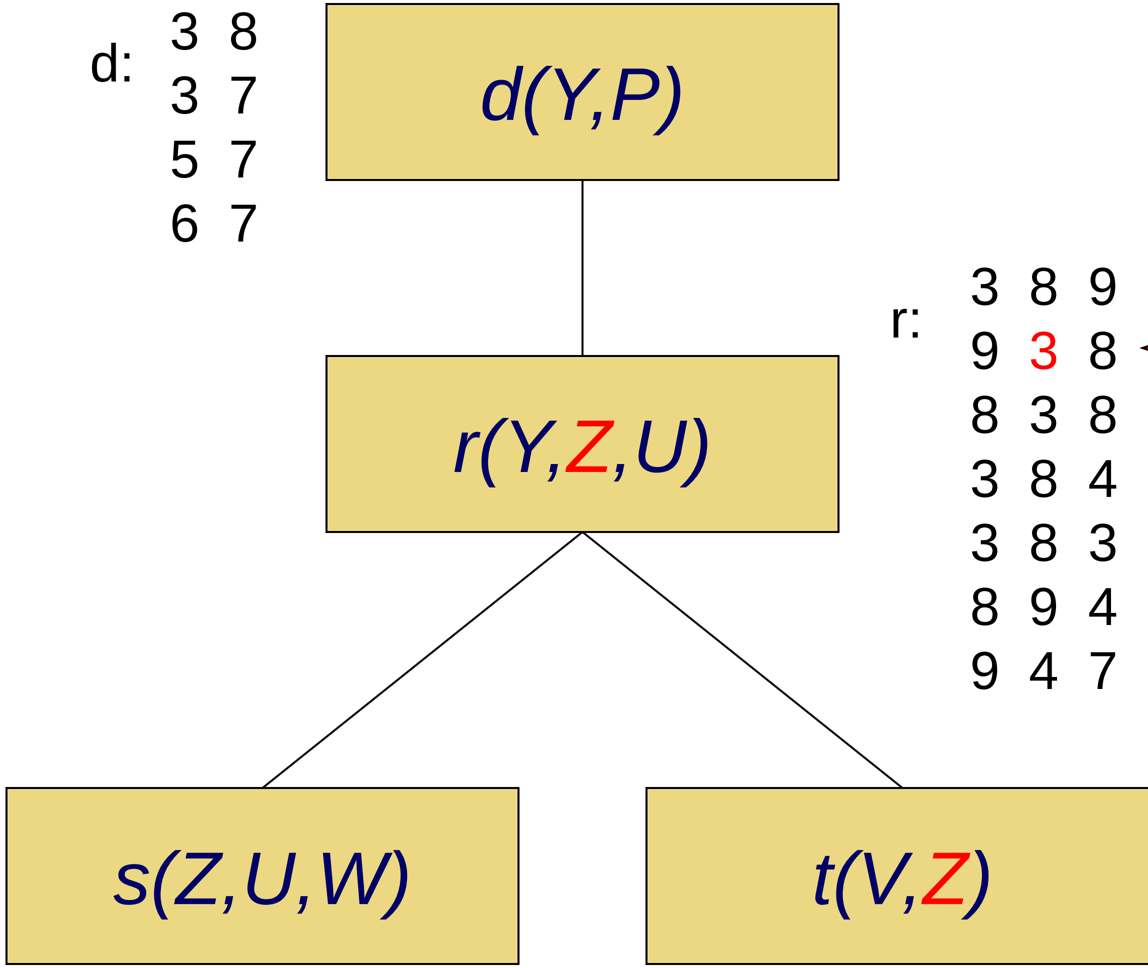
3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

t:

9	8
9	3
9	5

d:

3	8
3	7
5	7
6	7



r:

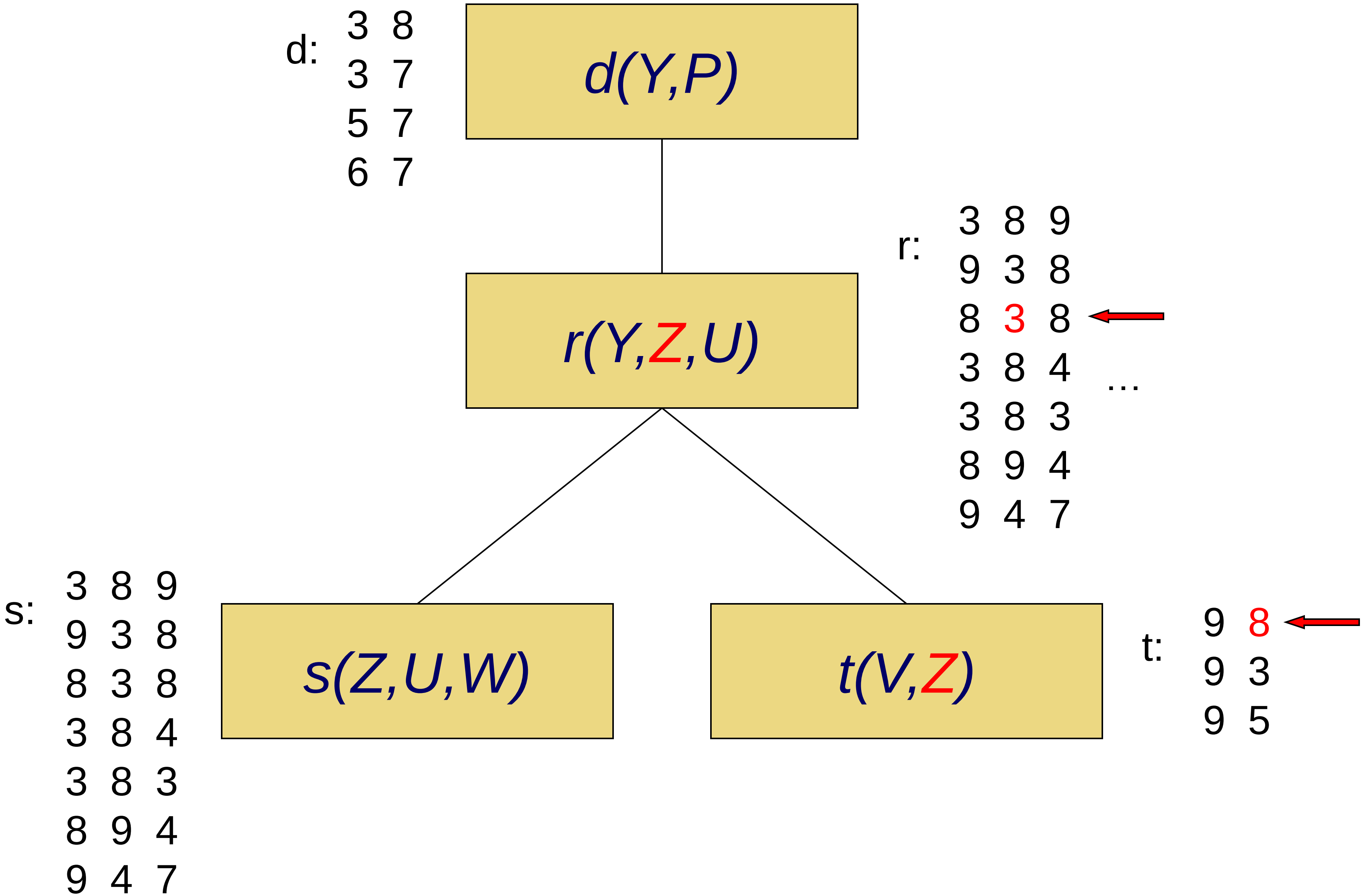
3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

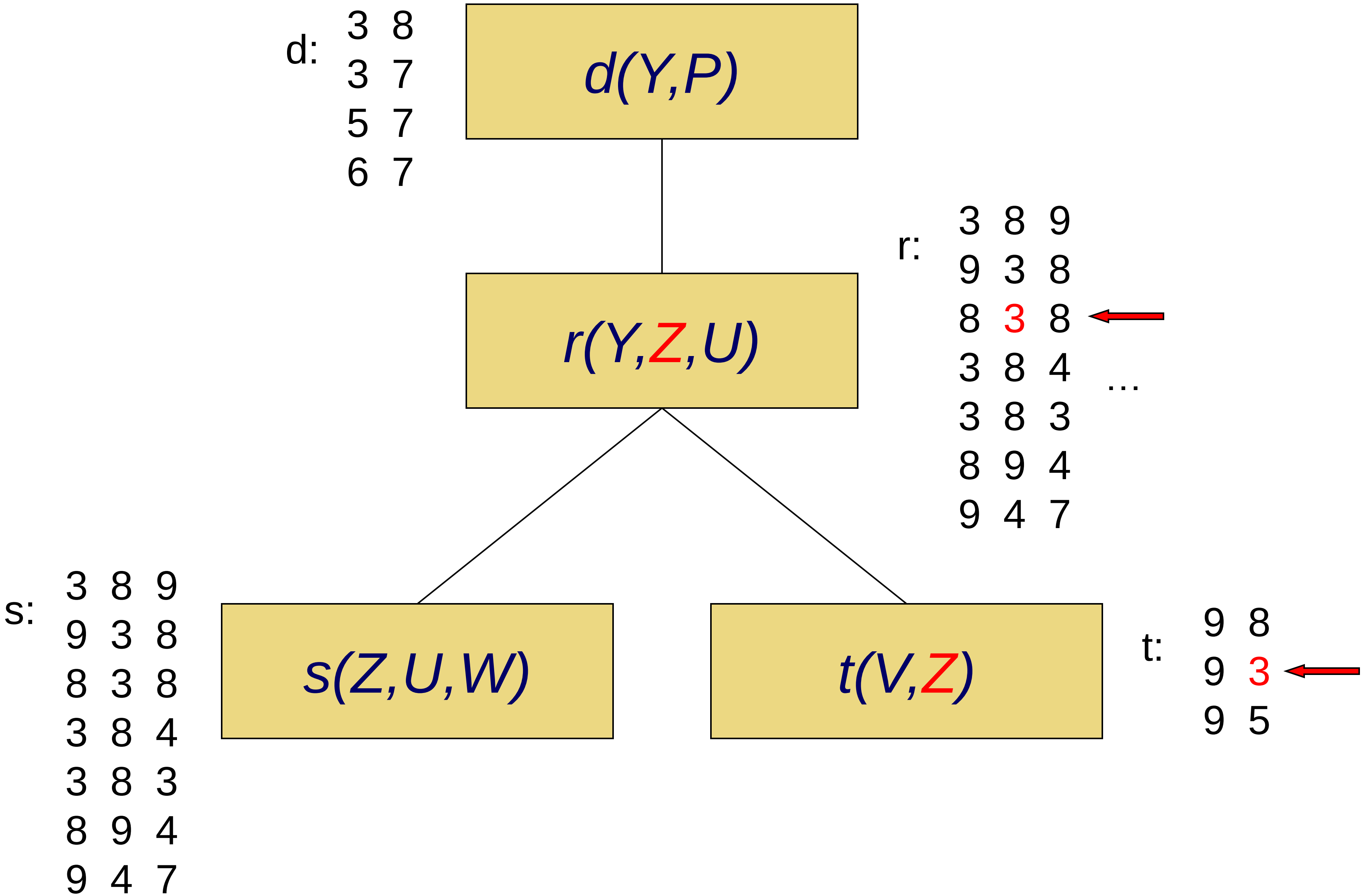
s:

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

t:

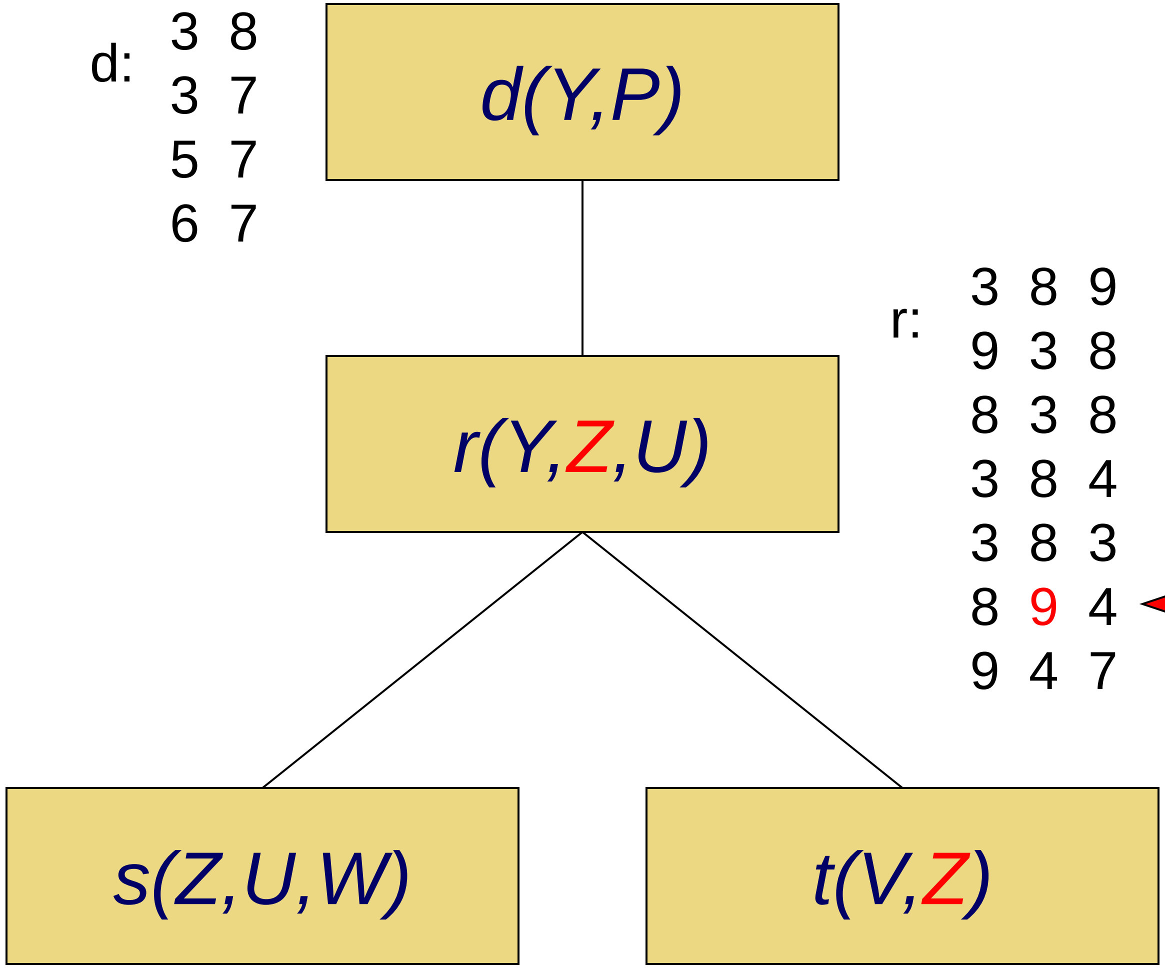
9	8
9	3
9	5


d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
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



r:

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

s:

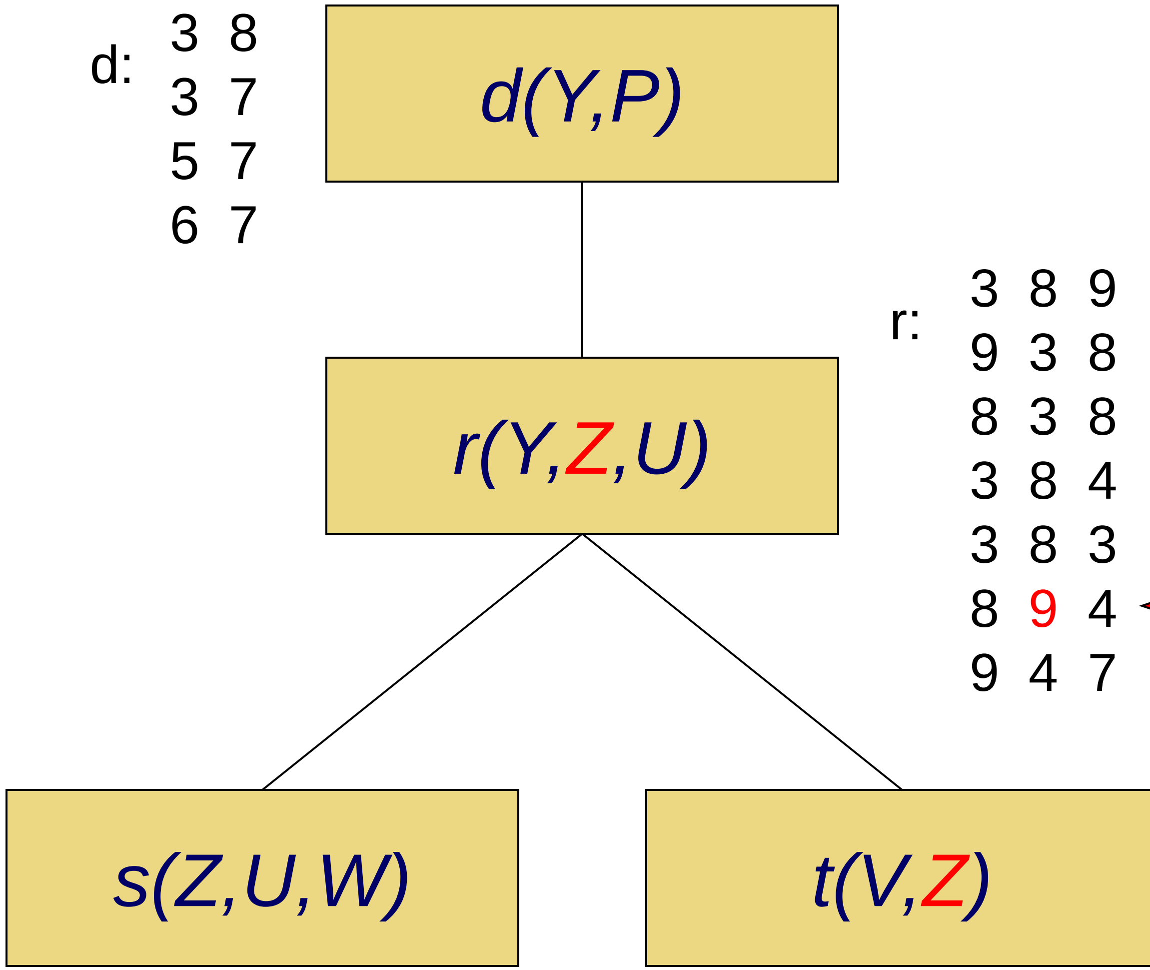
3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

t:

9	8
9	3
9	5

d:

3	8
3	7
5	7
6	7



r:

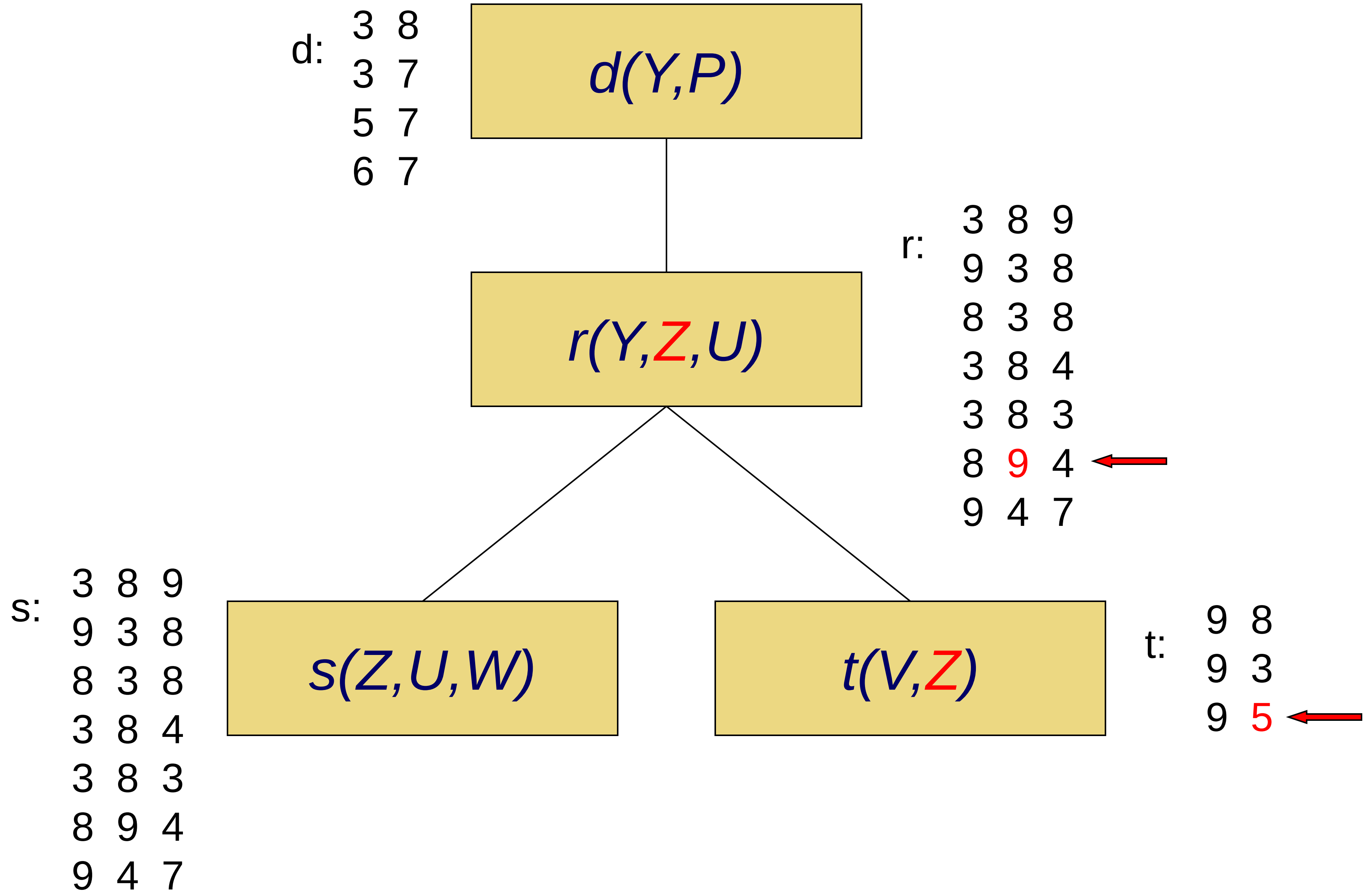
3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

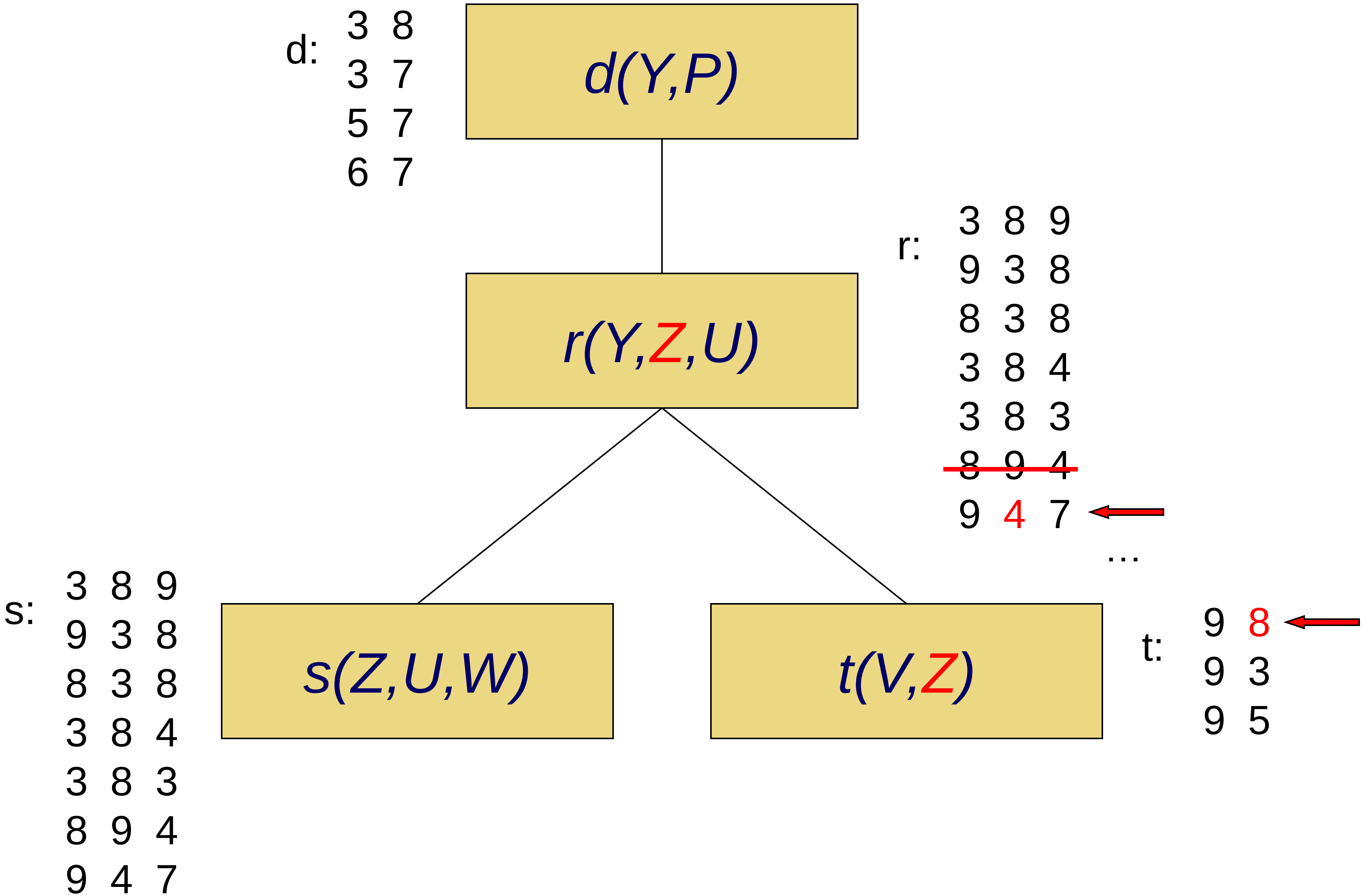
s:

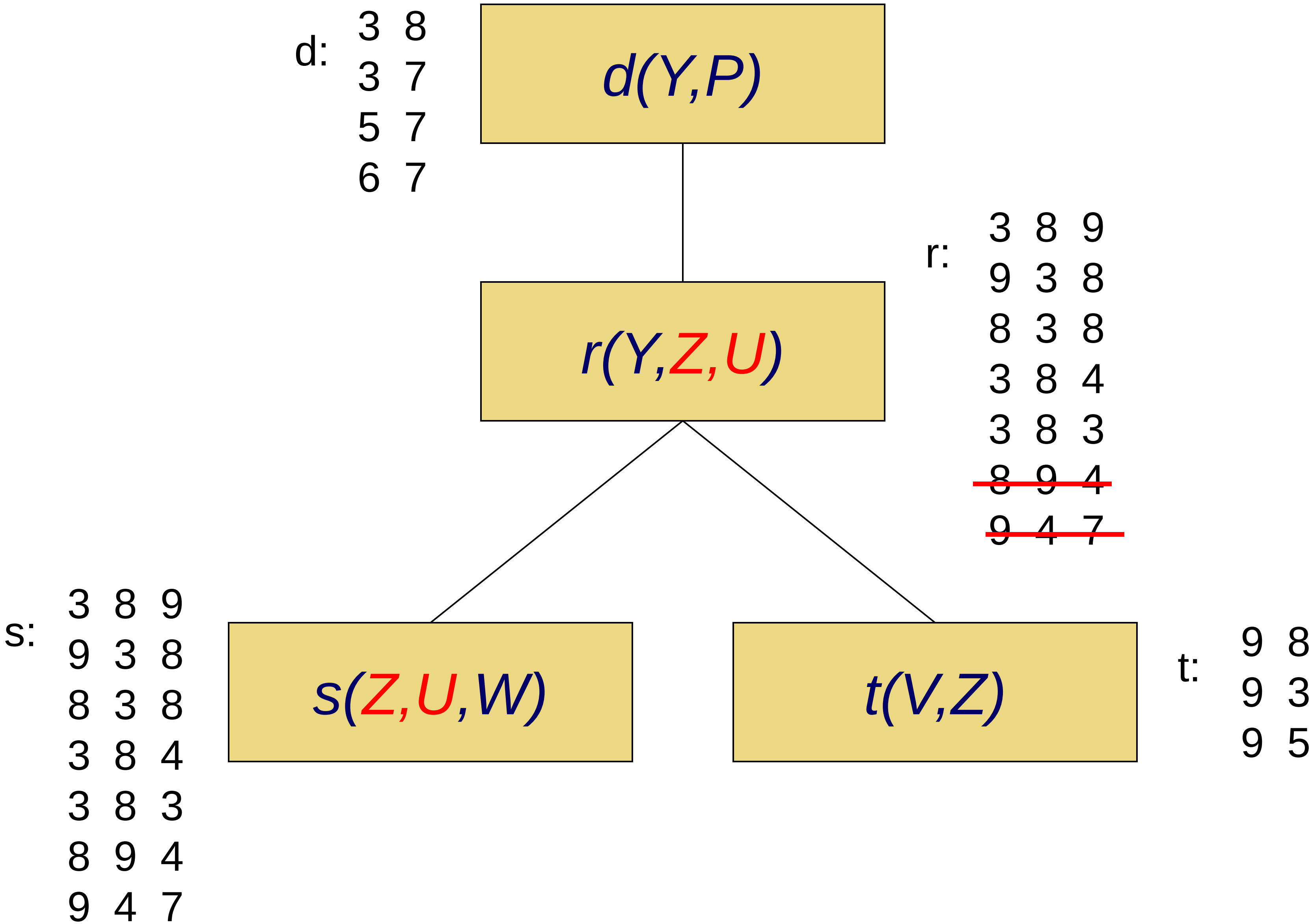
3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

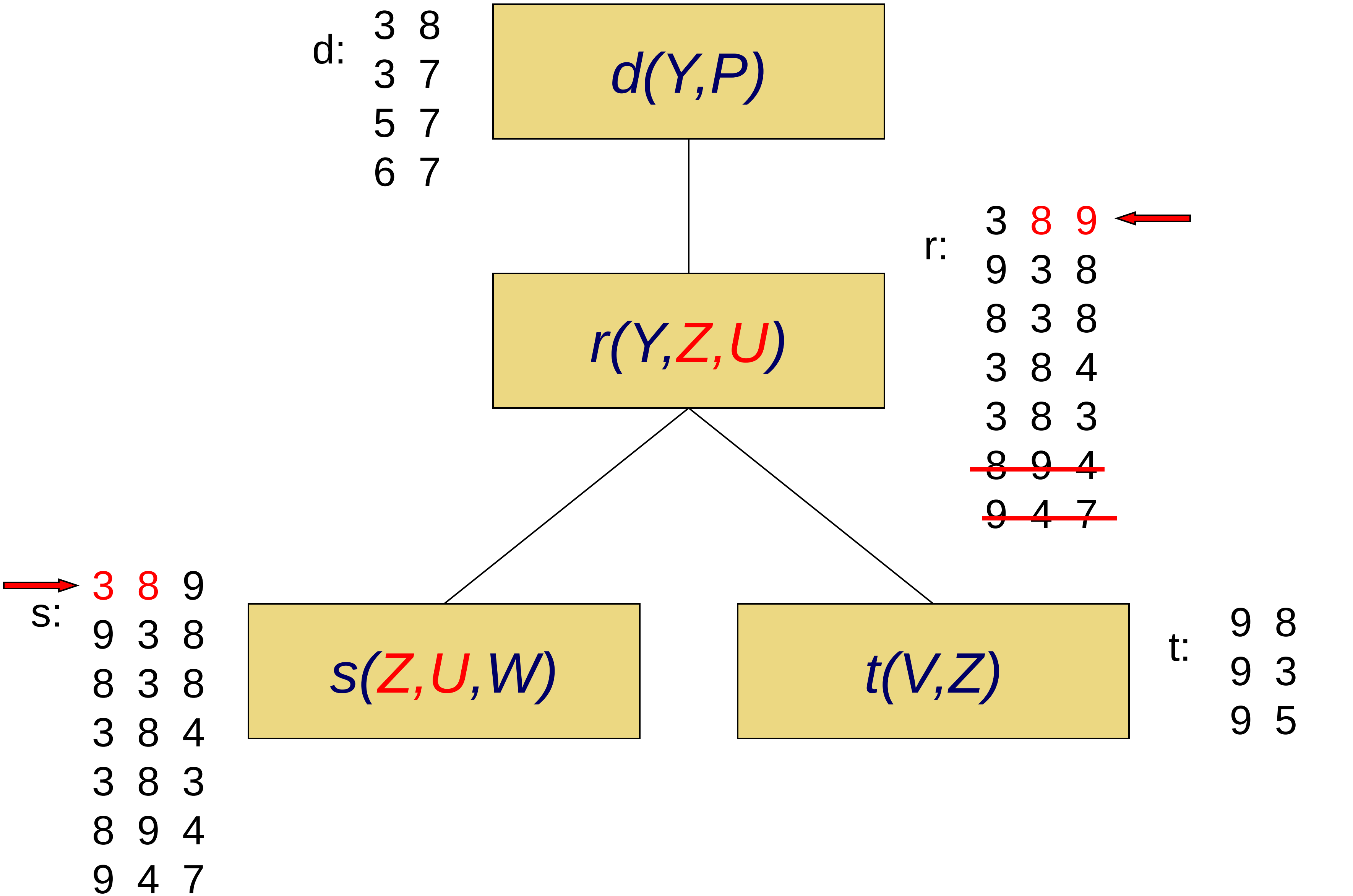
t:

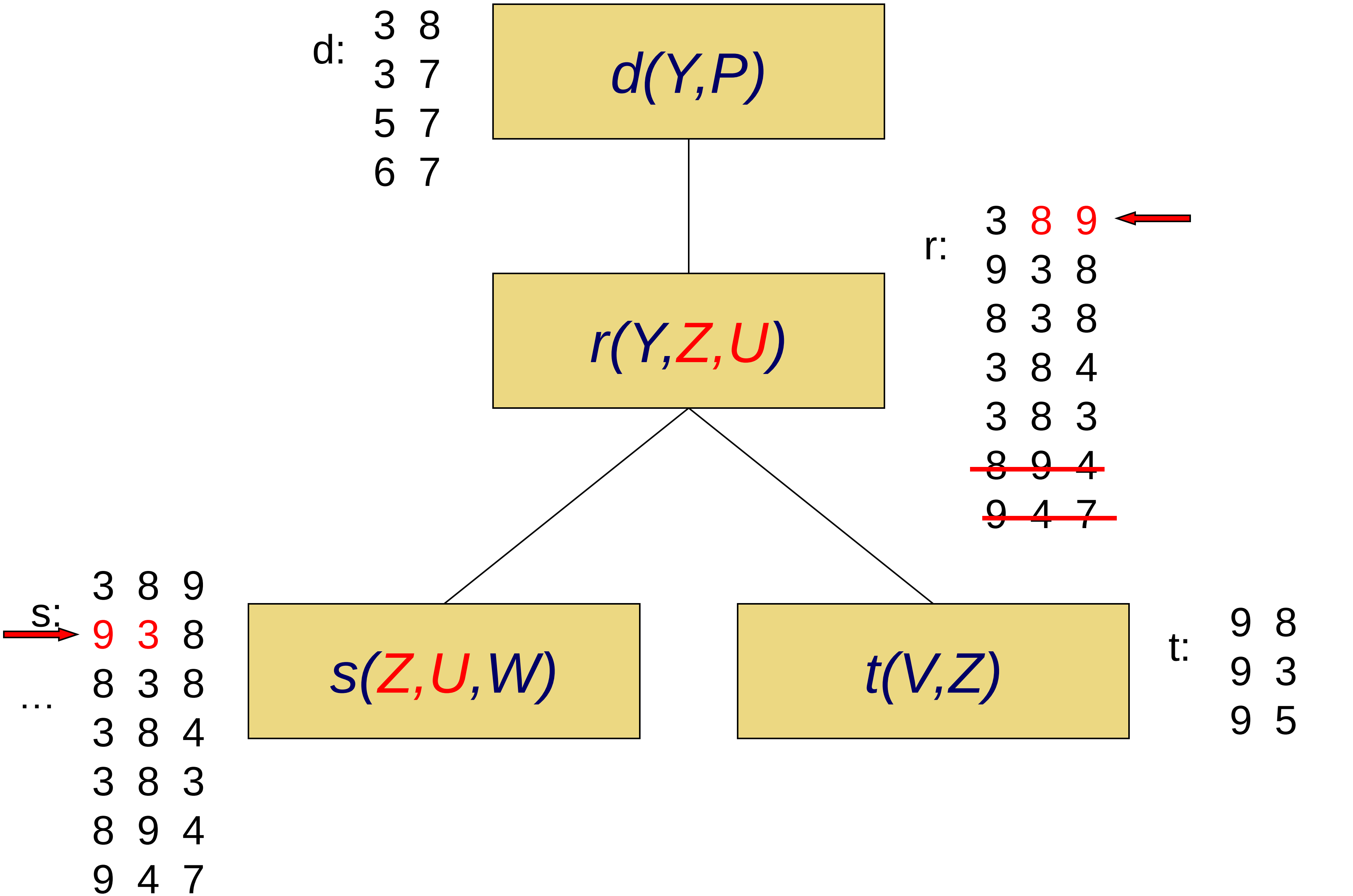
9	8
9	3
9	5

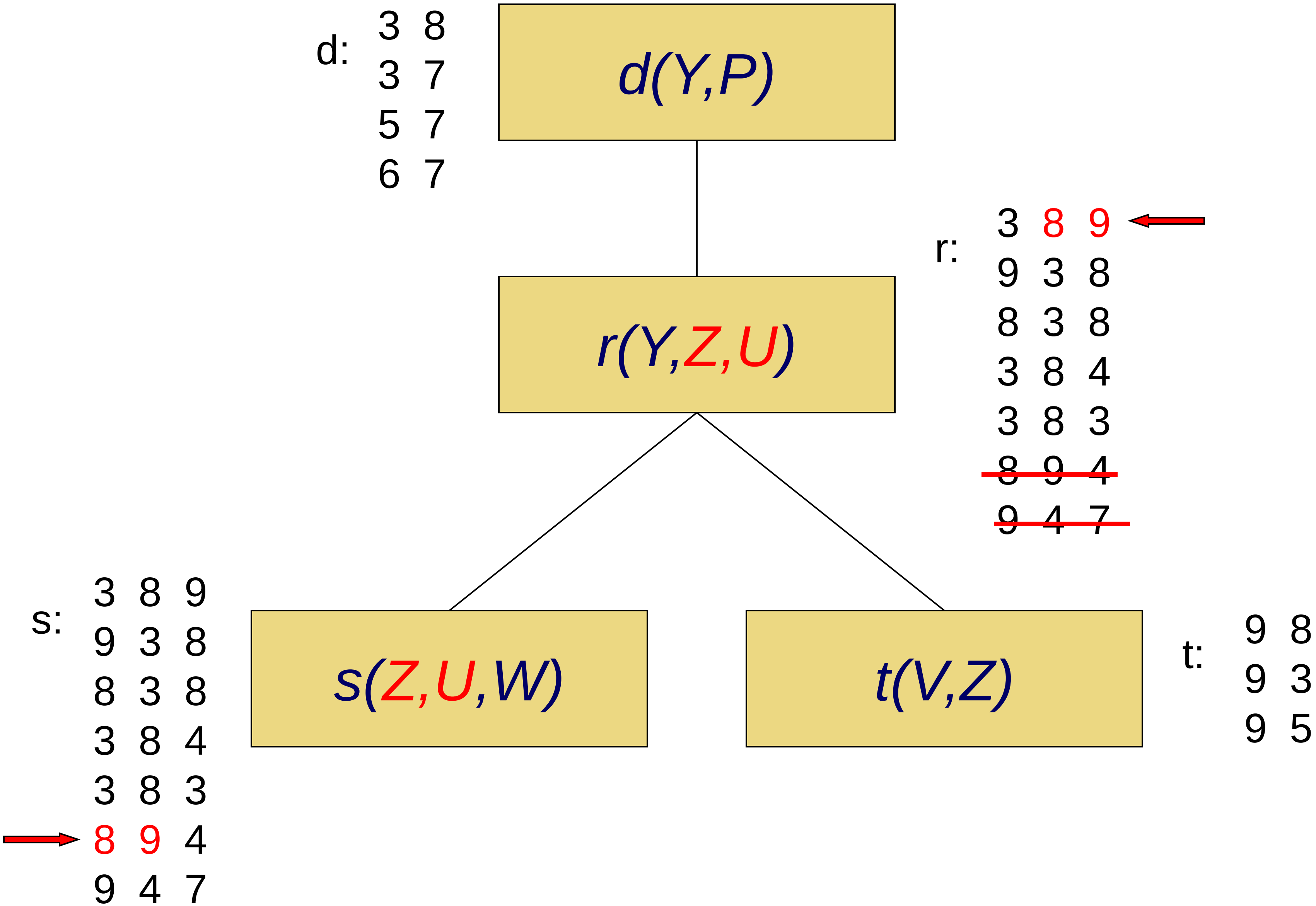

d:
3 8
3 7
5 7
6 7
d(Y,P)
r(Y,Z,U)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

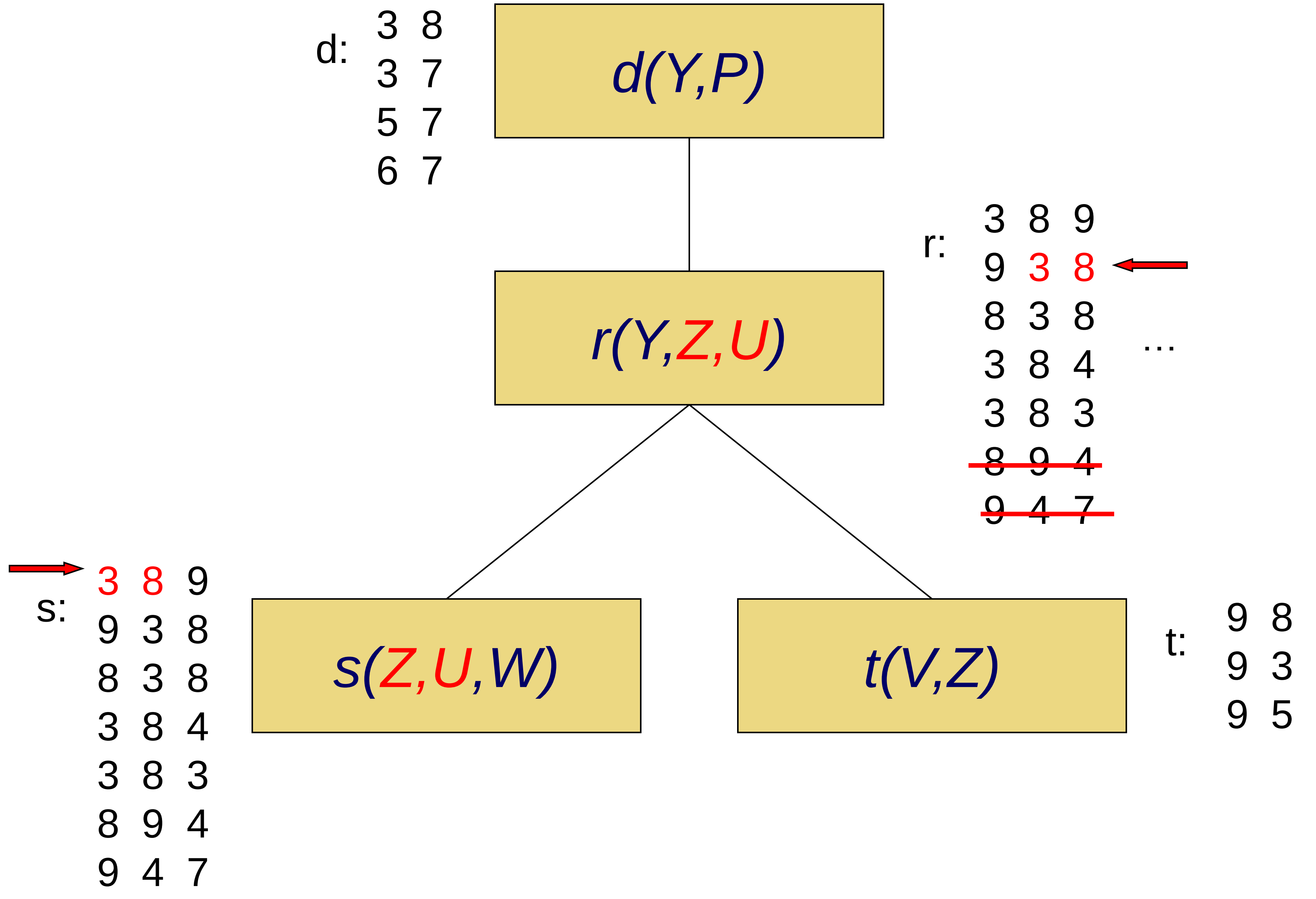

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
...
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5


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

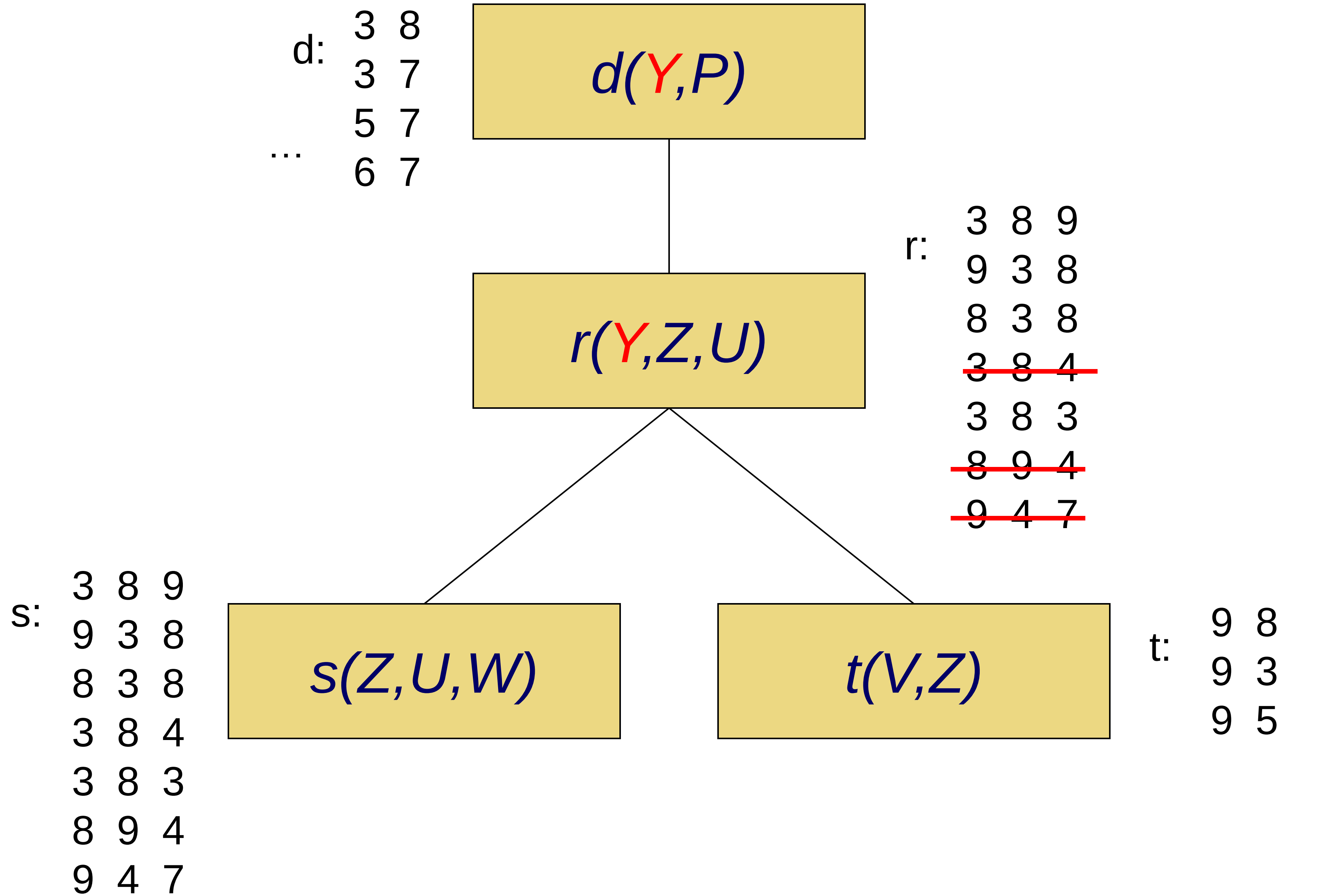

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
~~8 9 4~~
~~9 4 7~~
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

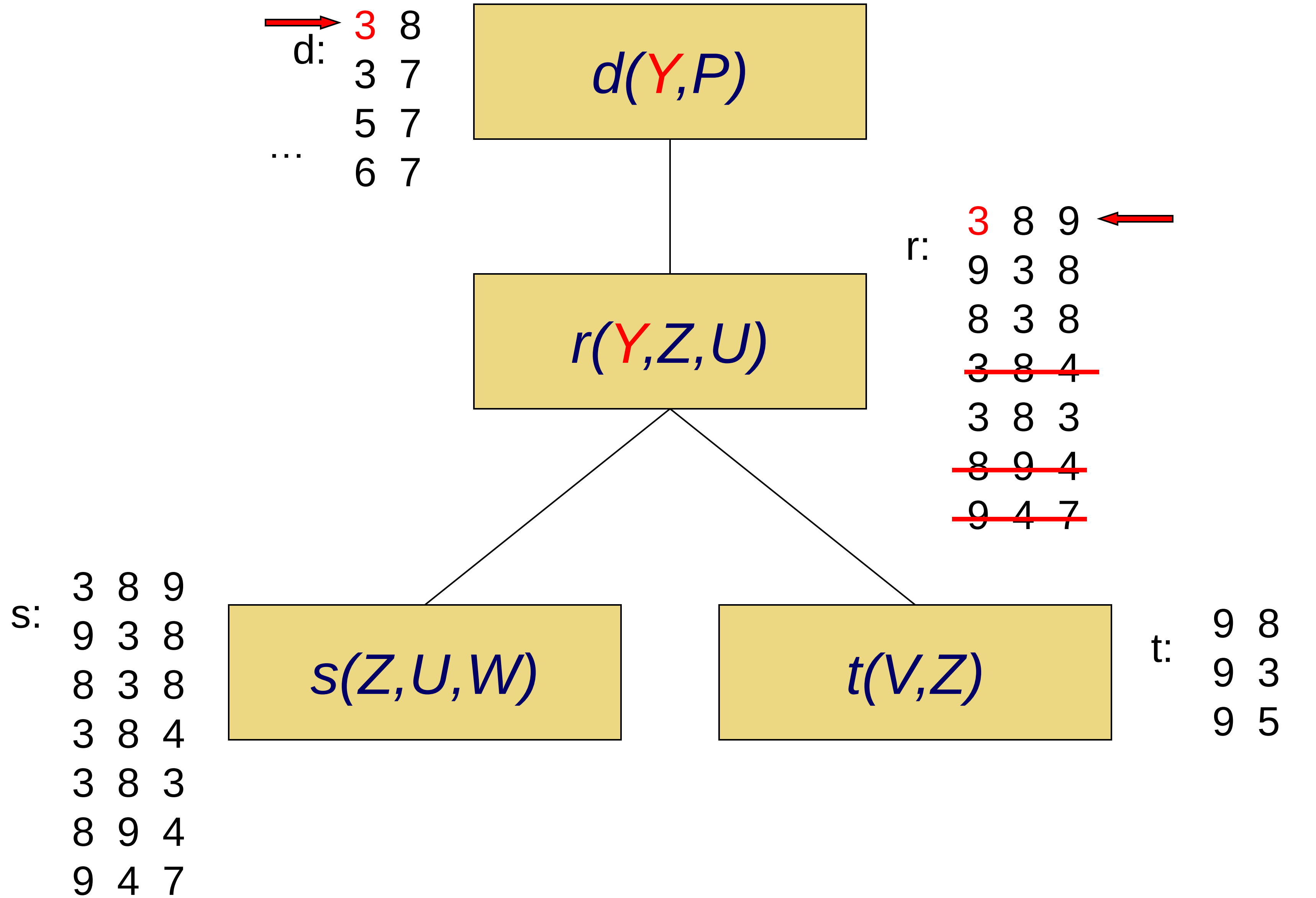

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
…
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5


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



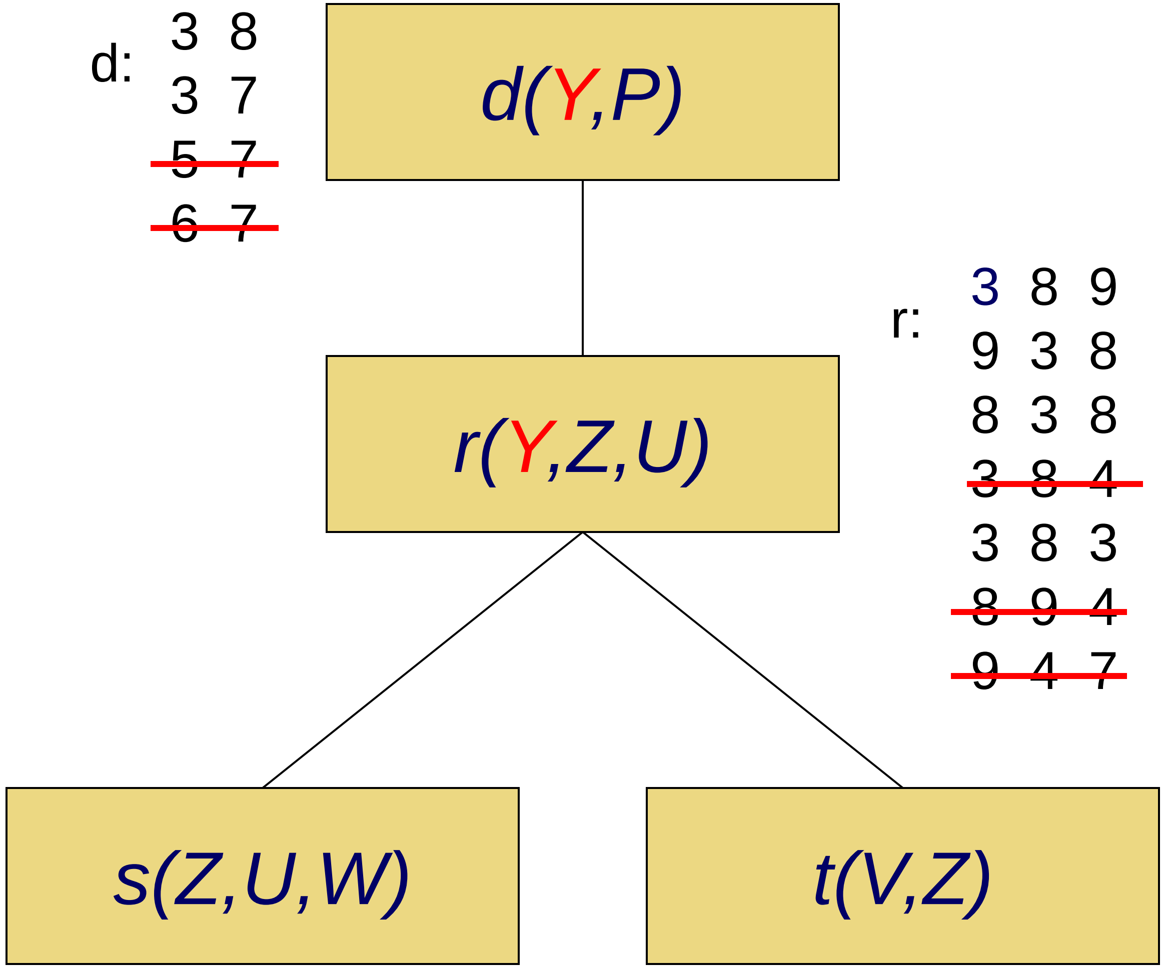
d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
…
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5


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

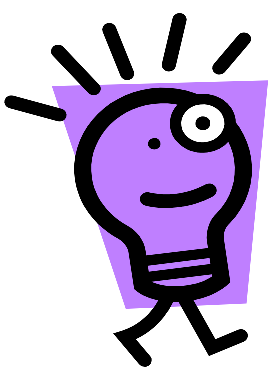
«Answering» Acyclic Instances

HOM: The homomorphism problem

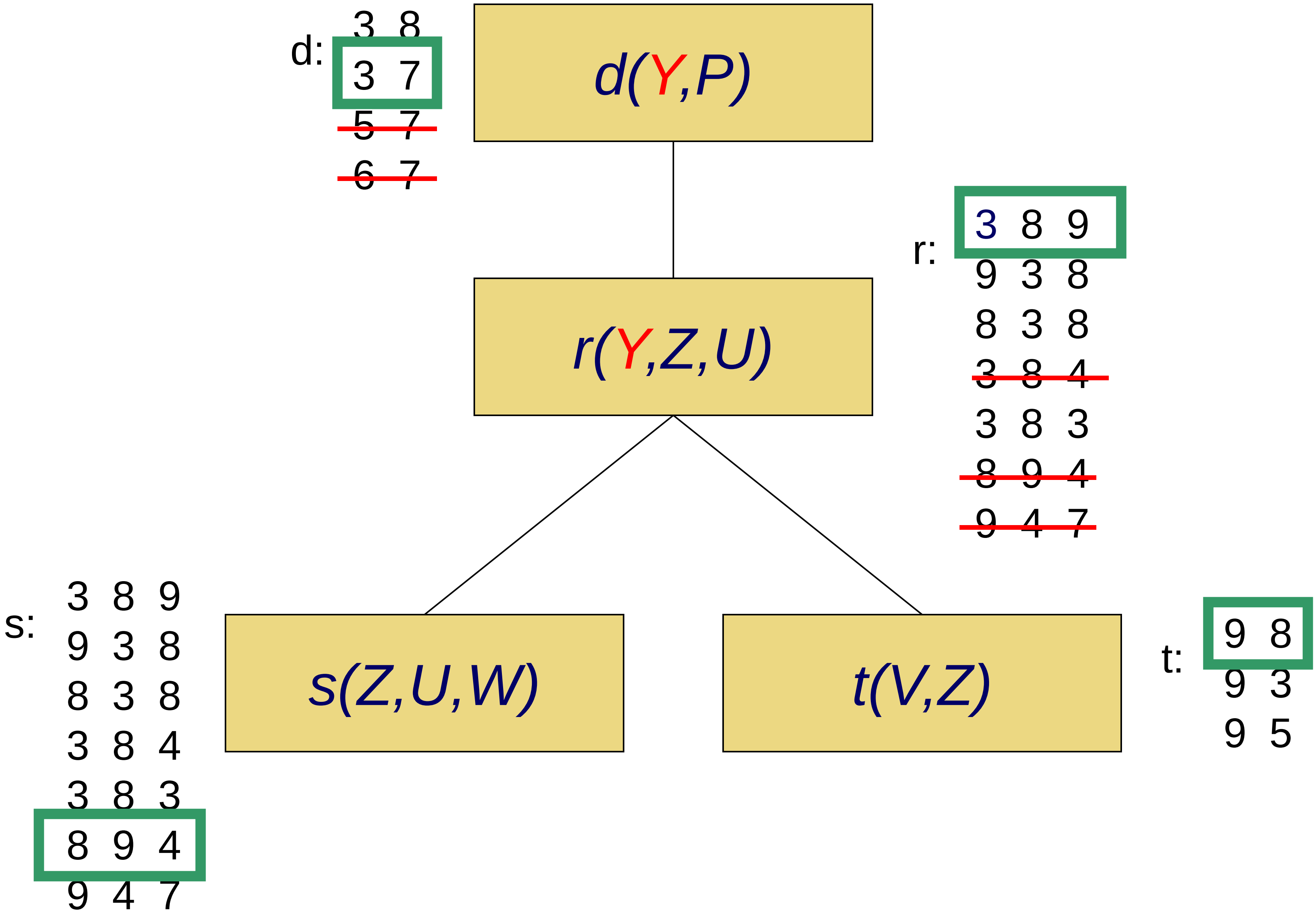
BCQ: Boolean conjunctive query evaluation

CSP: Constraint satisfaction problem

Yannakakis's Algorithm (ABCQs):
Dynamic Programming over a Join Tree



- Answering ACQs can be done adding a top-down phase to Yannakakis' algorithm for ABCQs



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

Example Application: Strategic Games

- Game *G=(P,Neigh,Act,U)* where
  - *P*: set of players
  - *Neigh(p)*: neighbors of player *p*
  - *Act(p):* actions (strategies) of player *p*
  - *U:* utility function *u(p),* for each player *p*

Example Application: Strategic Games

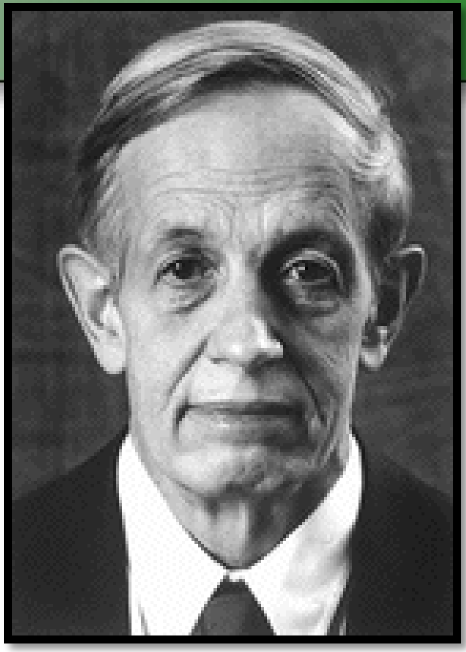
- Game *G=(P,Neigh,Act,U)* where
  - *P*: set of players
  - *Neigh(p)*: neighbors of player *p*
  - *Act(p):* actions (strategies) of player *p*
  - *U:* utility function *u(p)*, for each player *p*

Ex.: Prisoners' Dilemma

- $P = \{P_1, P_2\}$
- $Neigh(P_1) = \{P_2\}$; $Neigh(P_2) = \{P_1\}$;
- $Act(P_1) = Act(P_2) =$ {collaborate, defeat}
- Utility functions: listed in the matrix

P_1 \ P_2	Collaborate	Defeat
Collaborate	(-5,-5)	(-1,-10)
Defeat	(-10,-1)	(-2,-2)

Nash's theorem



Nash equilibrium: a global strategy, from which no player has an incentive to unilaterally deviate.

Nash's theorem

Nash equilibrium: a global strategy, from which no player has an incentive to unilaterally deviate.

P_2

P_1

	Collaborate	Defeat
Collaborate	(-5,-5)	(-1,-10)
Defeat	(-10,-1)	(-2,-2)

Nash's theorem

Nash equilibrium: a global strategy, from which no player has an incentive to unilaterally deviate.

P_2

P_1		Collaborate	Defeat
	Collaborate	(-5,-5)	(-1,-10)
	Defeat	(-10,-1)	(-2,-2)

Nash's theorem

Nash equilibrium: a global strategy, from which no player has an incentive to unilaterally deviate.

Theorem: Every game admits a *mixed* Nash equilibrium, where players chose their strategies according to probability distributions

.

P_2

P_1	Collaborate	Defeat
Collaborate	(-5,-5)	(-1,-10)
Defeat	(-10,-1)	(-2,-2)

Nash's theorem

Nash equilibrium: a global strategy, from which no player has an incentive to unilaterally deviate.

Theorem: Every game admits a *mixed* Nash equilibrium, where players chose their strategies according to probability distributions

	Collaborate	Defeat
Collaborate	(0,1)	(1,0)
Defeat	(1,0)	(0,1)

P_2

P_1	Collaborate	Defeat
Collaborate	(-5,-5)	(-1,-10)
Defeat	(-10,-1)	(-2,-2)

Pure Equilibria

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

F	$P_m R_m$	$P_m R_o$	$P_o R_m$	$P_o R_o$
m	2	2	1	0
o	0	2	1	2

G	$P_m F_m$	$P_m F_o$	$P_o F_m$	$P_o F_o$
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	$P_m R_m$	$P_m R_o$	$P_o R_m$	$P_o R_o$
m	2	2	1	0
o	0	2	1	2

G	$P_m F_m$	$P_m F_o$	$P_o F_m$	$P_o F_o$
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
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

τ_F :

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

τ_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

τ_R :

R	F
o	m
m	o

τ_P :

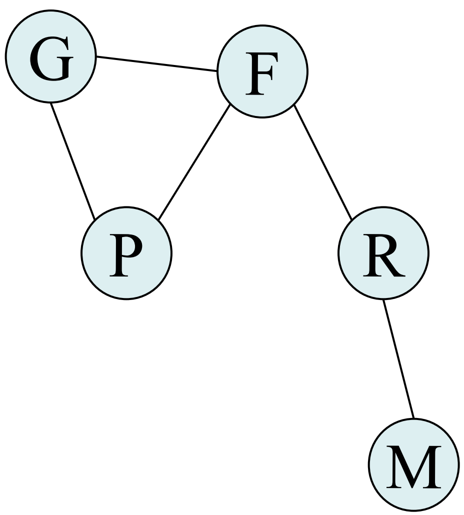
P	F
m	m
o	o

τ_M :

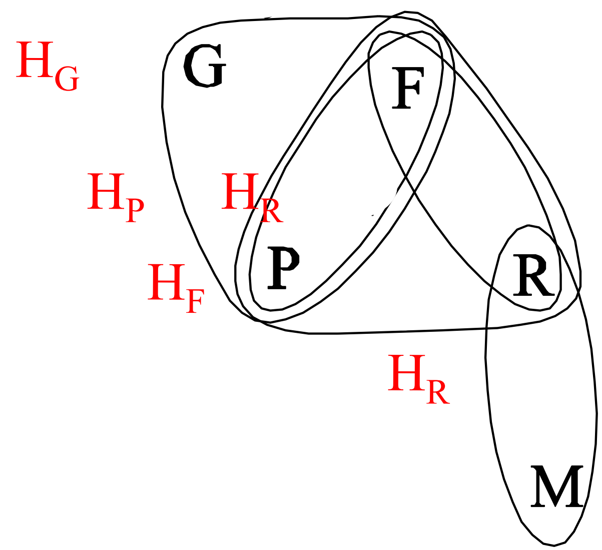
M	R
m	m
o	o

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}

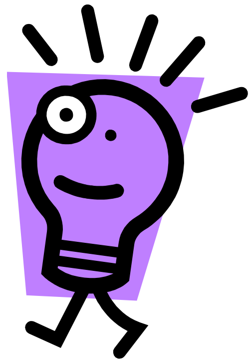


G(*FRIENDS*)

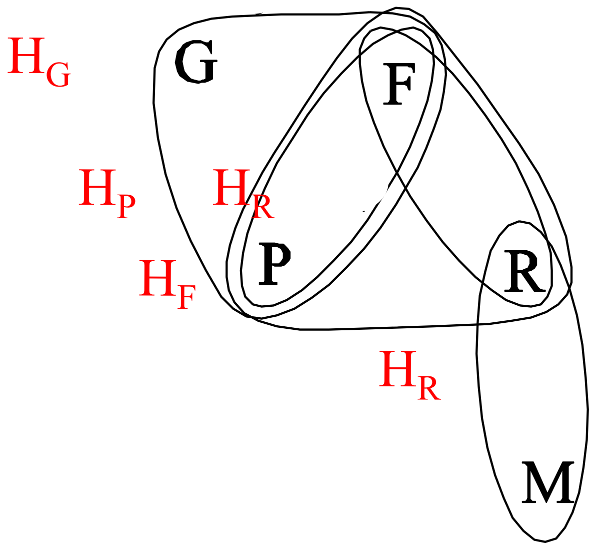


H(*FRIENDS*)

Interaction Among Players: Friends

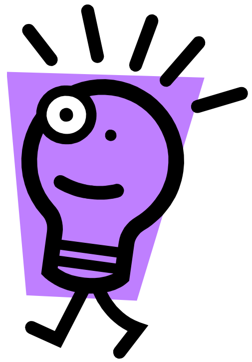


This is the same structure as the one of the associated CSP



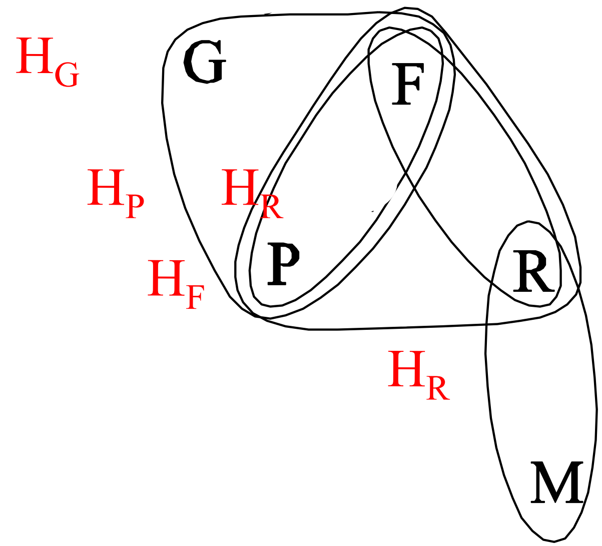
H(*FRIENDS*)

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

Identification of "Easy" Classes

Applications of Tree Decompositions

Beyond Tree Decompositions

Decision/Computation Problems

Optimization Problems

Enumeration Problems

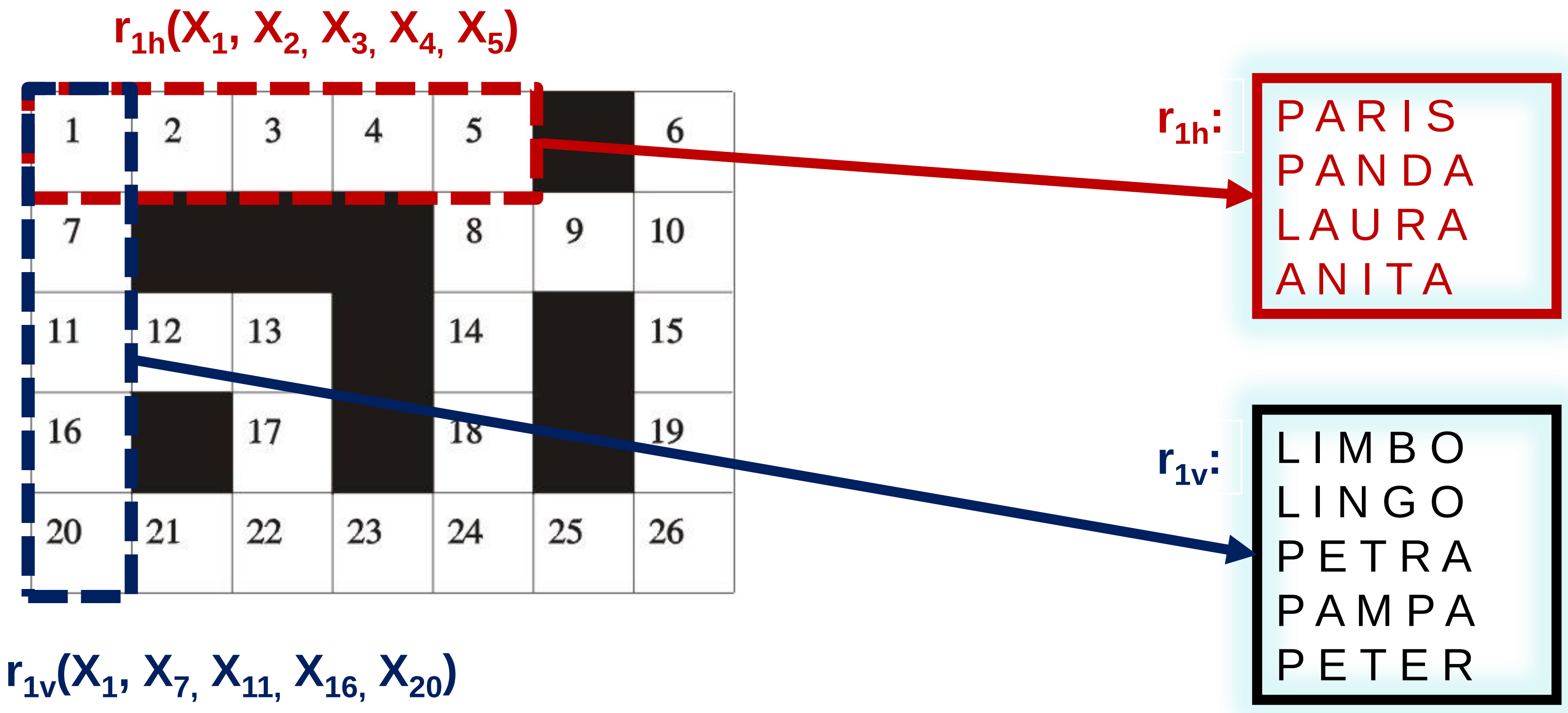
Constraint Optimization Problems

- 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, …

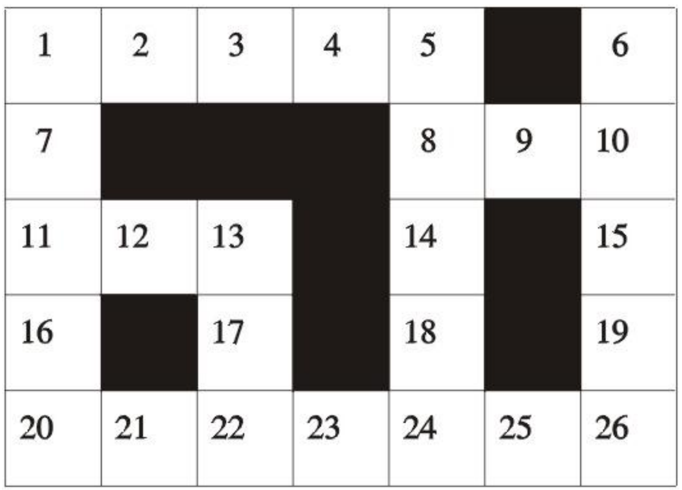
Classical CSPs



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

- Set of constraint relations

Puzzles for Experts…



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

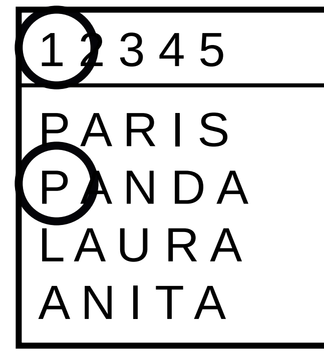
A Classification for Optimization Problems

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:

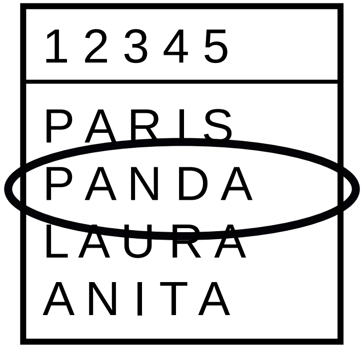
- Satisfying all the constraints, and
- Having the minimum total cost.

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.

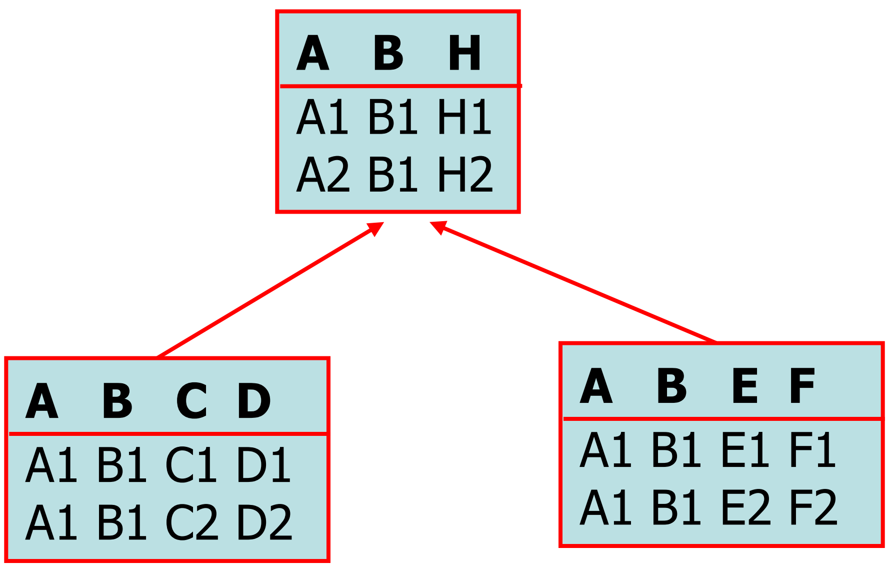
Then, find an assignment:

- Minimizing the cost of violated relations.

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)**



[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

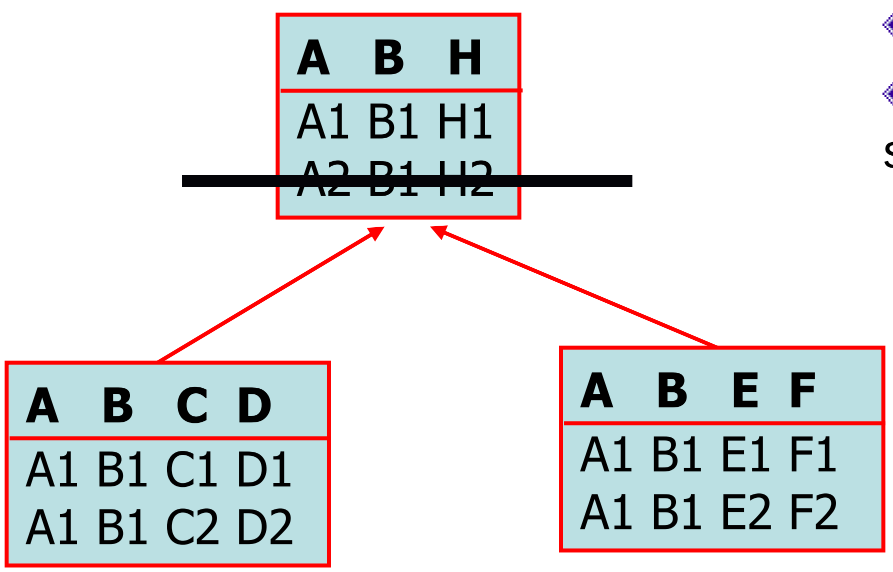
A	**B**	**E**	**F**
A1	B1	E1	F1
A1	B1	E2	F2

CSOP: Tractability of Acyclic Instances

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

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



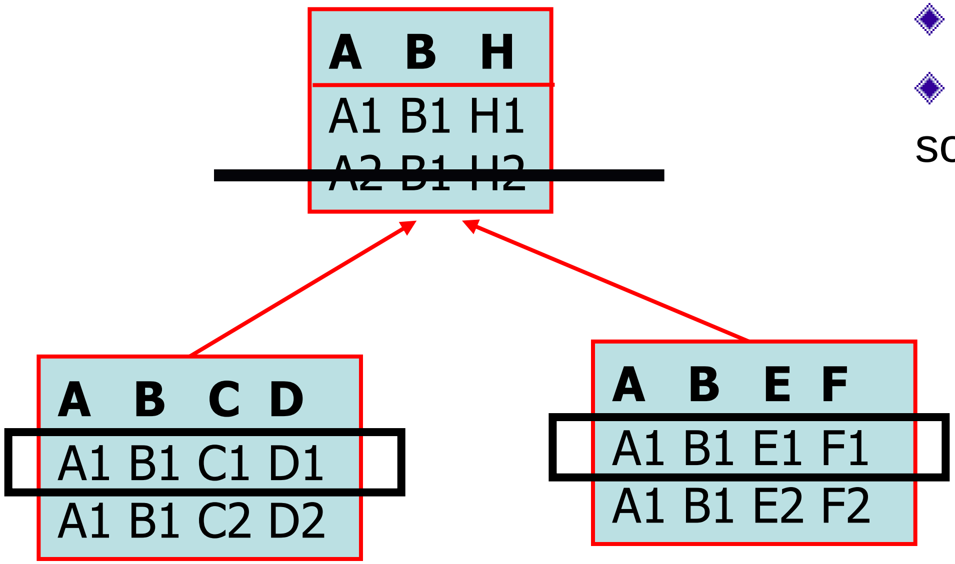
cost(C/C1)=cost(D/D1)=0
cost(C/C2)=cost(D/D2)=1
cost(E/E1)=cost(F/F1)=0
cost(E/E2)=cost(F/F2)=1

CSOP: Tractability of Acyclic Instances

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

With a bottom-up computation:

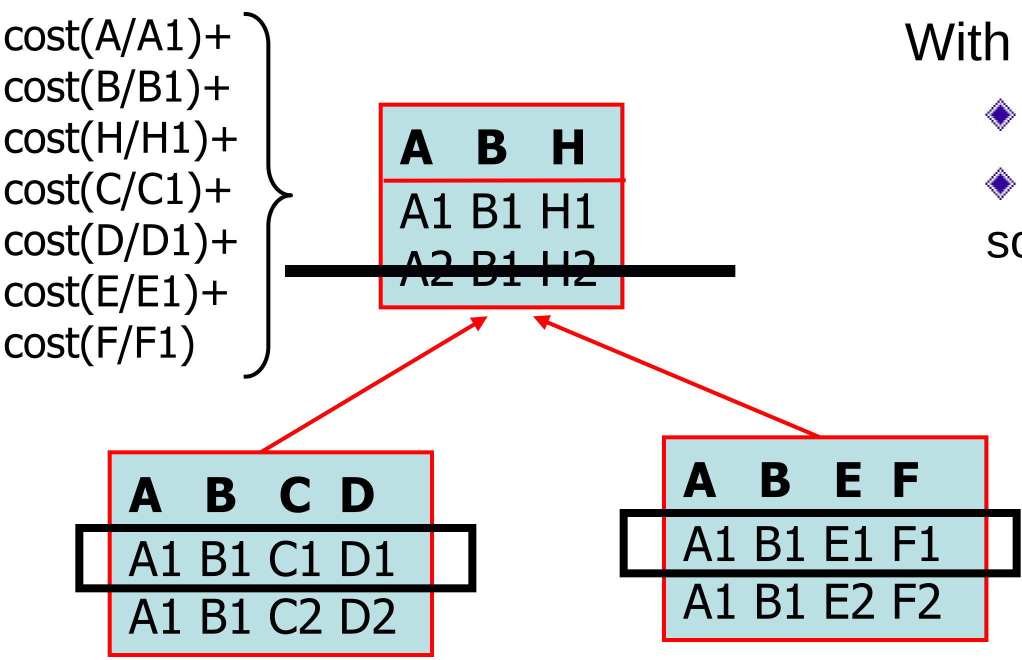
- Filter the tuples that do not match
- Compute the cost of the best partial solution, by looking at the children



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)**



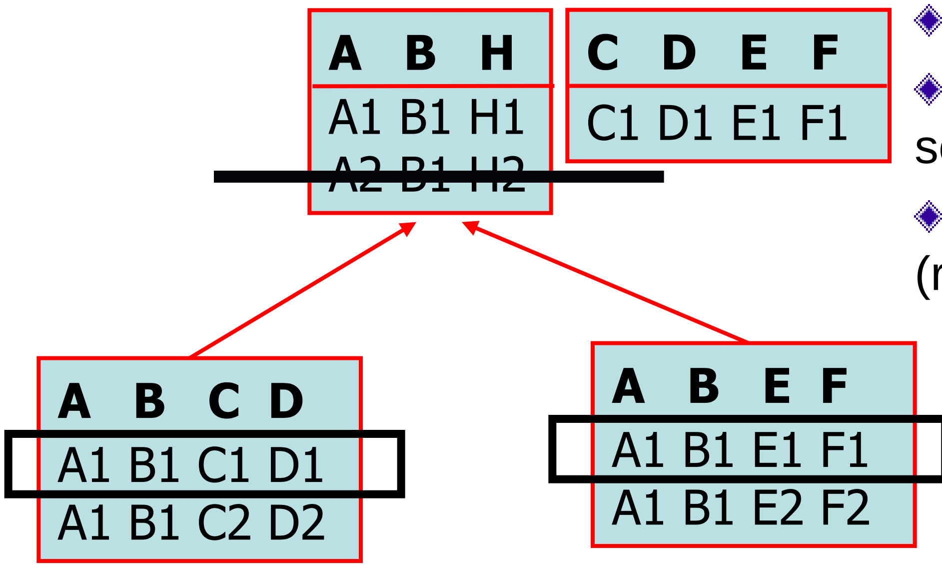
With a bottom-up computation:

- 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

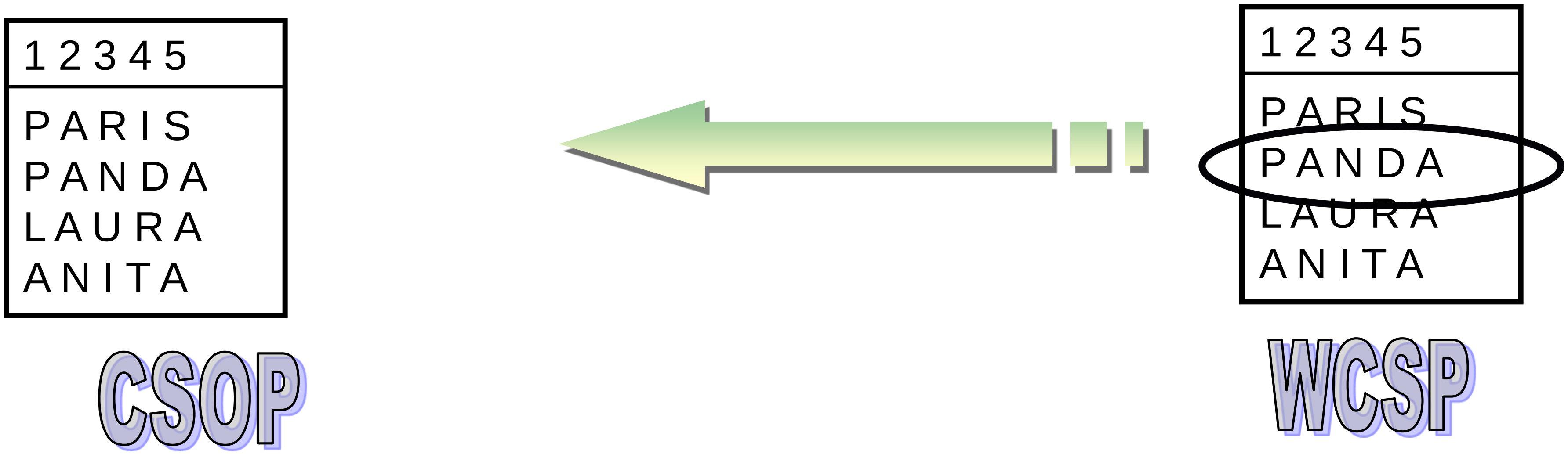
- Adapt the dynamic programming approach in **(Yannakakis'81)**



With a bottom-up computation:

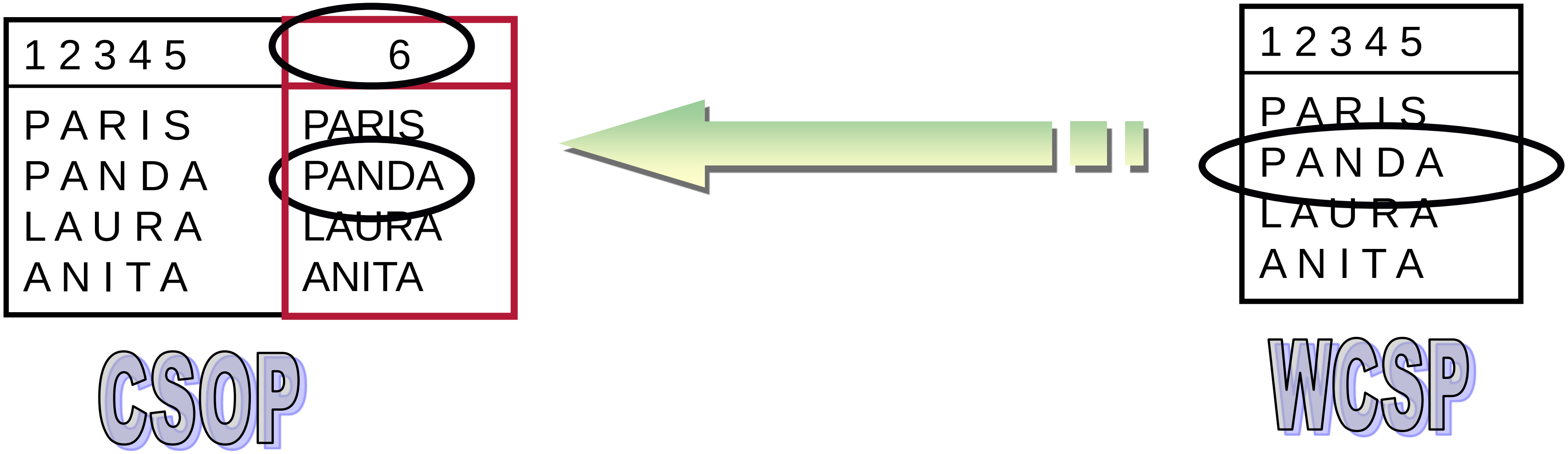
- Filter the tuples that do not match
- Compute the cost of the best partial solution, by looking at the children
- Propagate the best partial solution (resolving ties arbitrarily)

WCSP: Tractability of Acyclic Instances



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

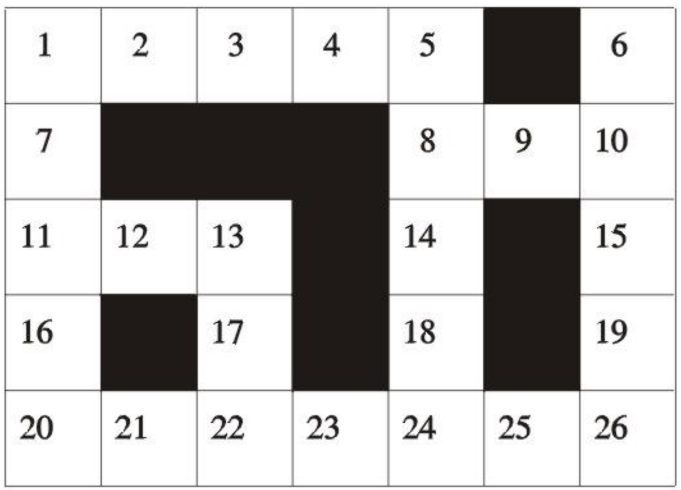
WCSP: Tractability of Acyclic Instances



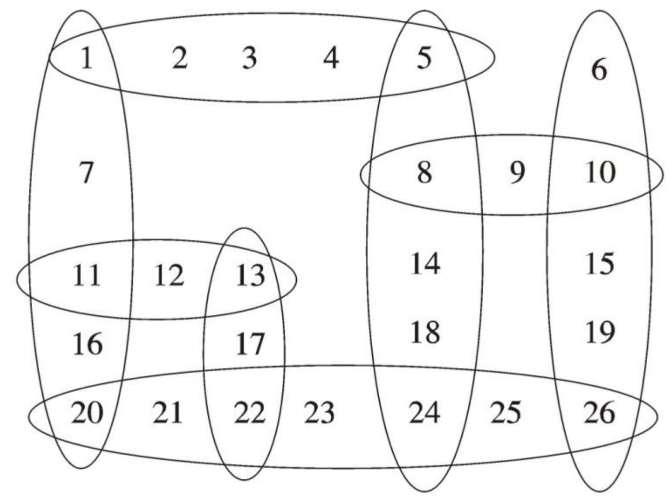
The mapping:

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

In-Tractability of MAX-CSP Instances

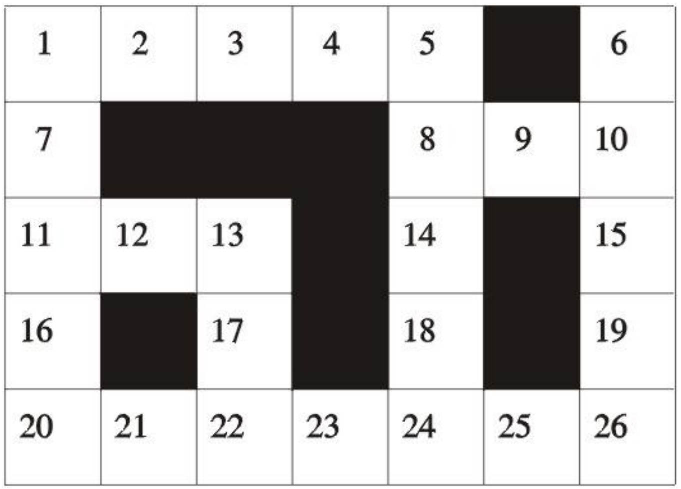


- Maximize the number of words placed in the puzzle

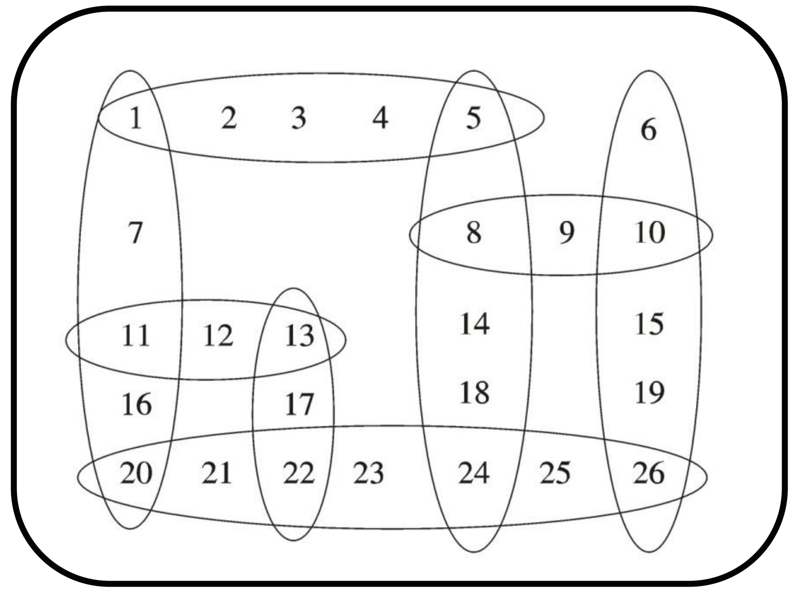


[Gottlob, Greco, and Scarcello, ICALP'09]

In-Tractability of MAX-CSP Instances



- Maximize the number of words placed in the puzzle

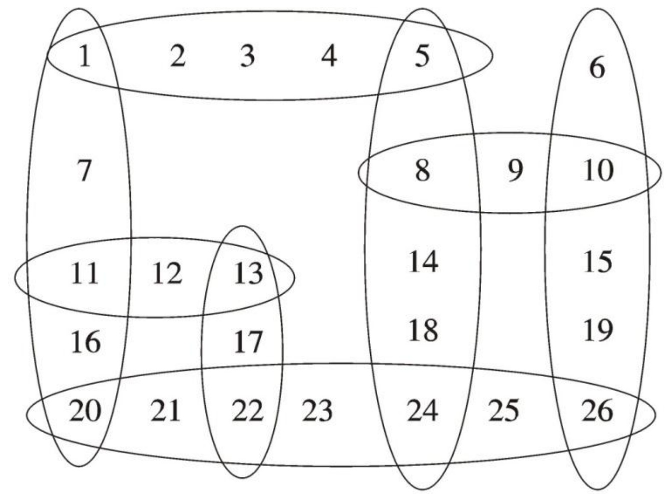


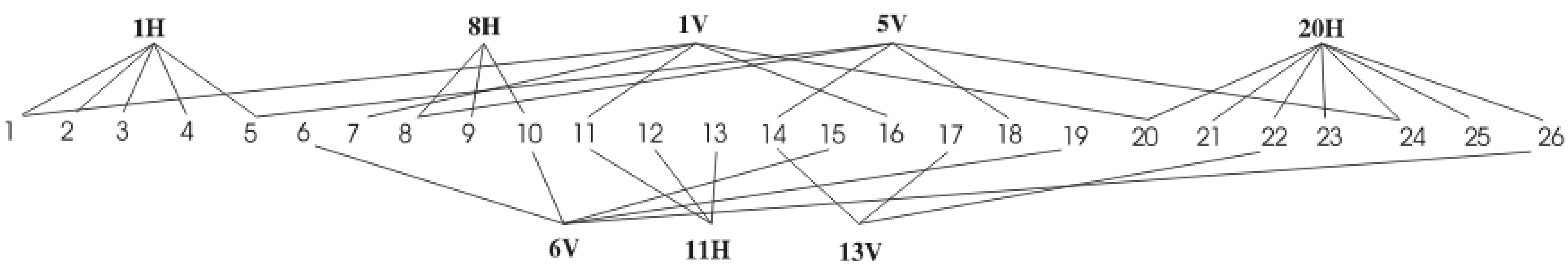
- Add a "big" constraint with no tuple

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

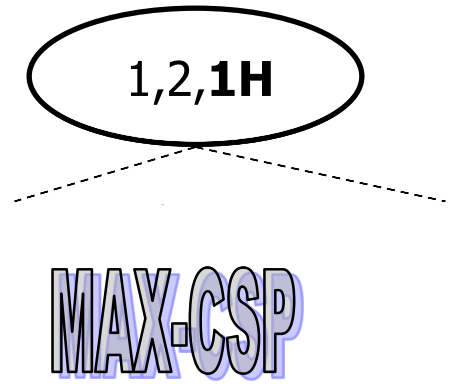
Tractability of MAX-CSP Instances

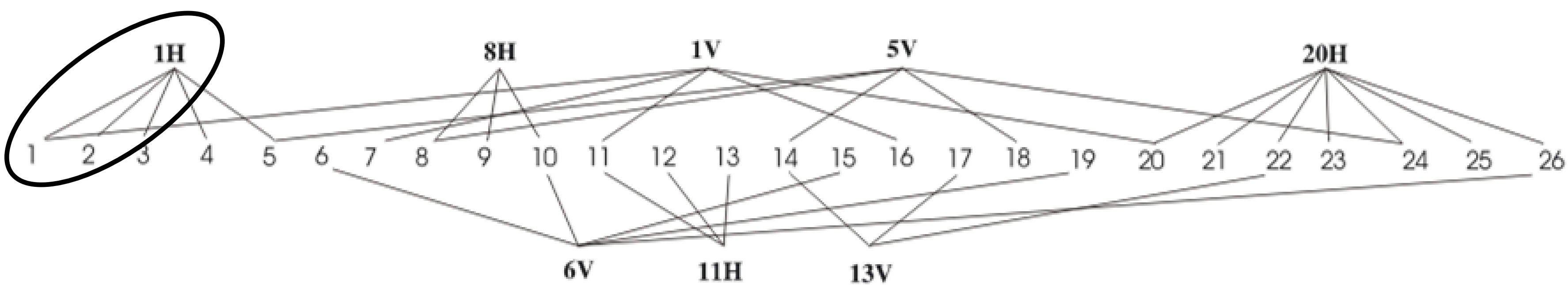
1. Consider the incidence graph
2. Compute a Tree Decomposition





Tractability of MAX-CSP Instances



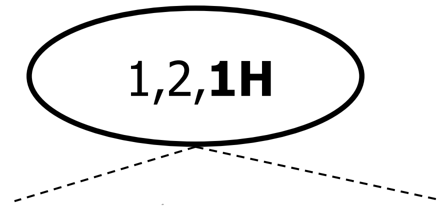


Tractability of MAX-CSP Instances

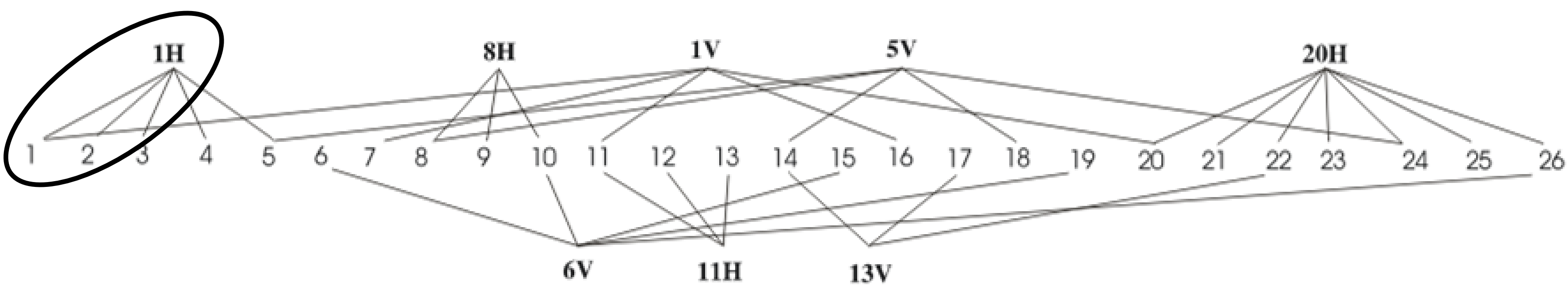
1 2	**1H**
P A	PARIS
P A	PANDA
L A	LAURA
A N	ANITA
A A	*unsat*
A B	*unsat*
...	*unsat*

Cost 1,
otherwise cost 0

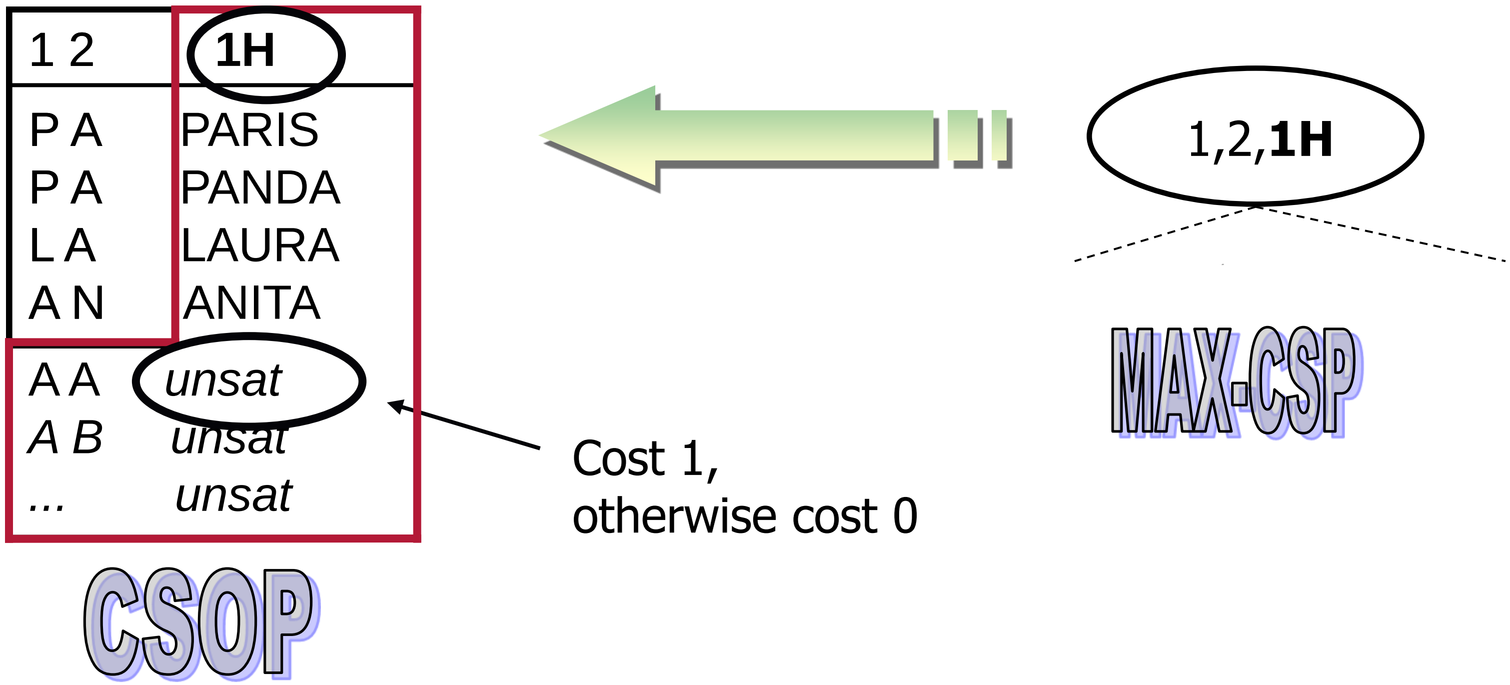
CSOP



MAX-CSP



In-Tractability of MAX-CSP Instances

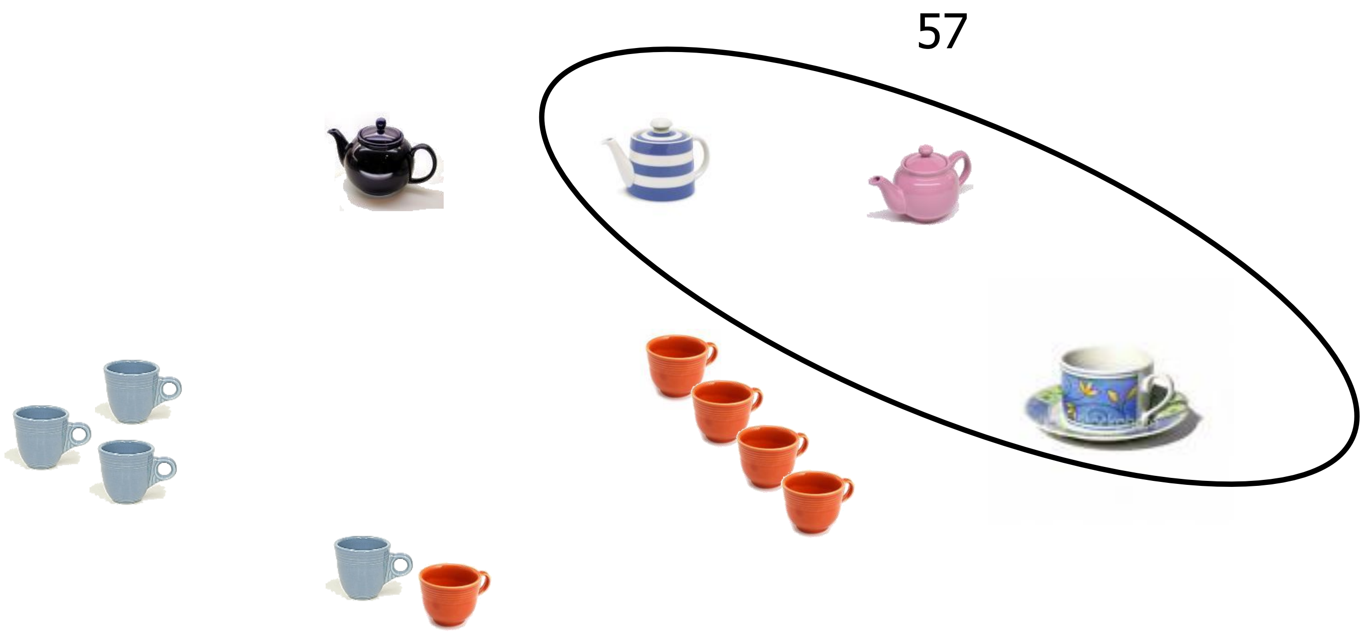


The mapping:

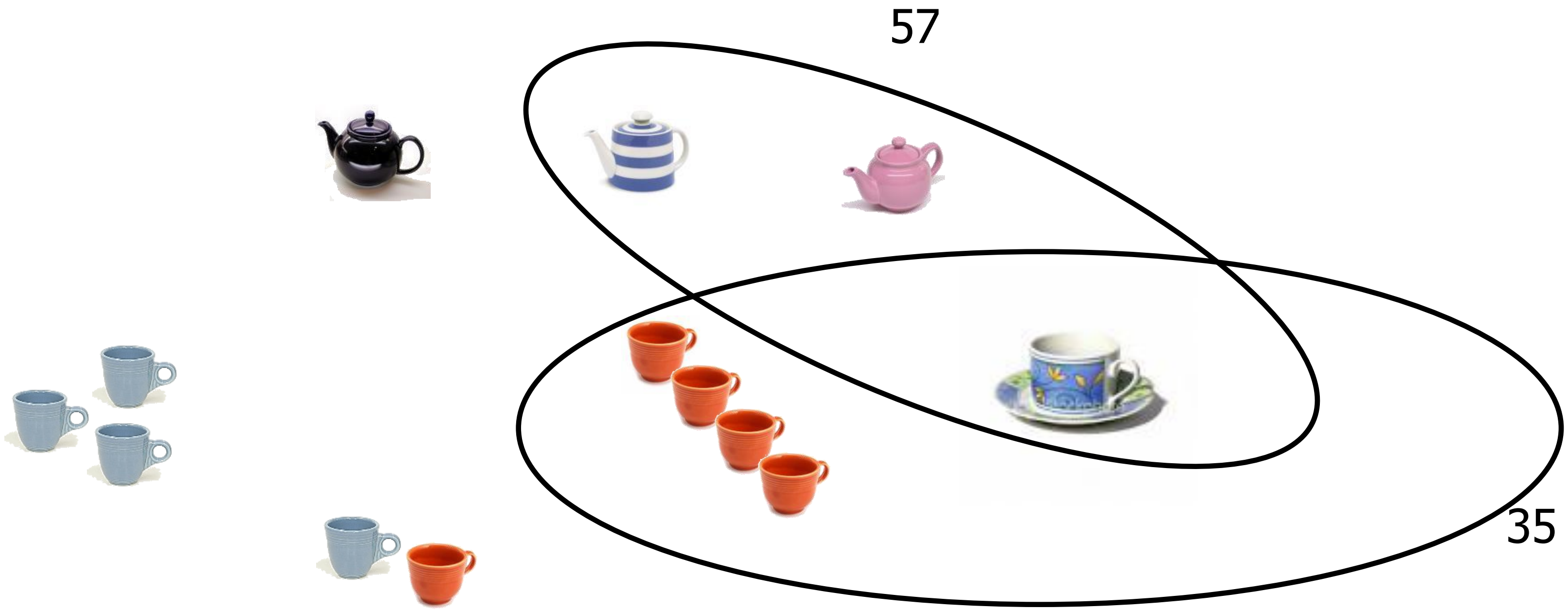
- Is feasible in time exponential in the width
- Preserves the solutions
- Leads to an Acyclic CSOP Instance

Example Application: Combinatorial Auctions

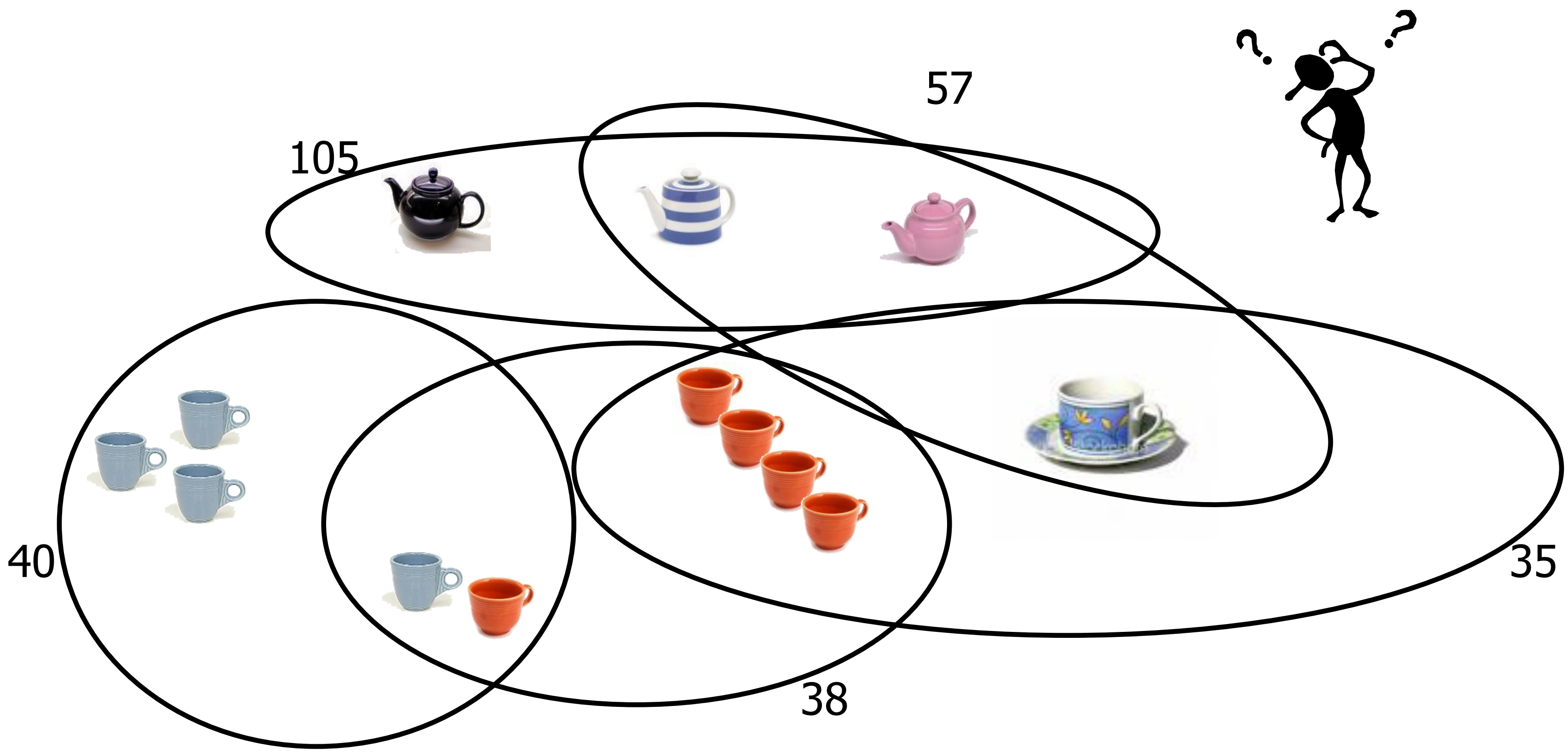
Example Application: Combinatorial Auctions



Example Application: Combinatorial Auctions



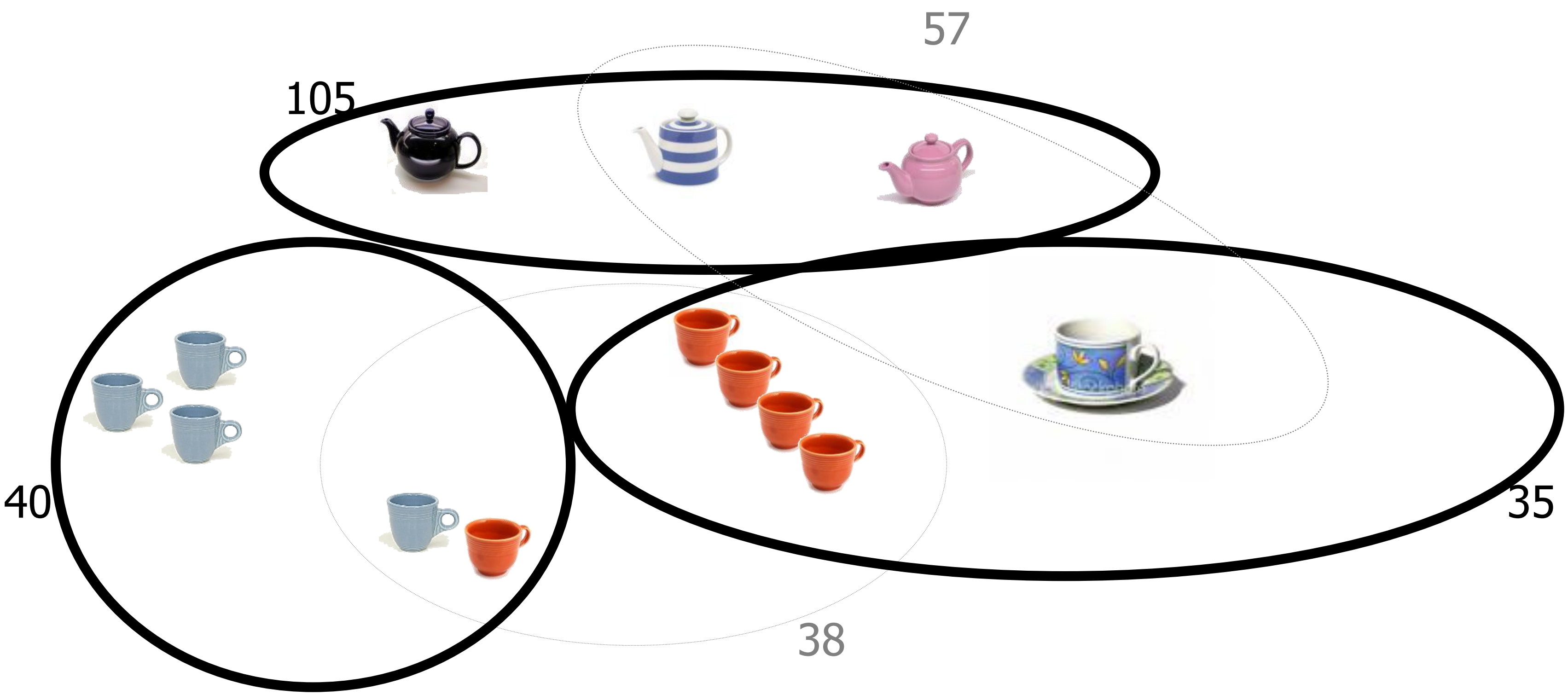
Example Application: Combinatorial Auctions



Winner Determination Problem

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

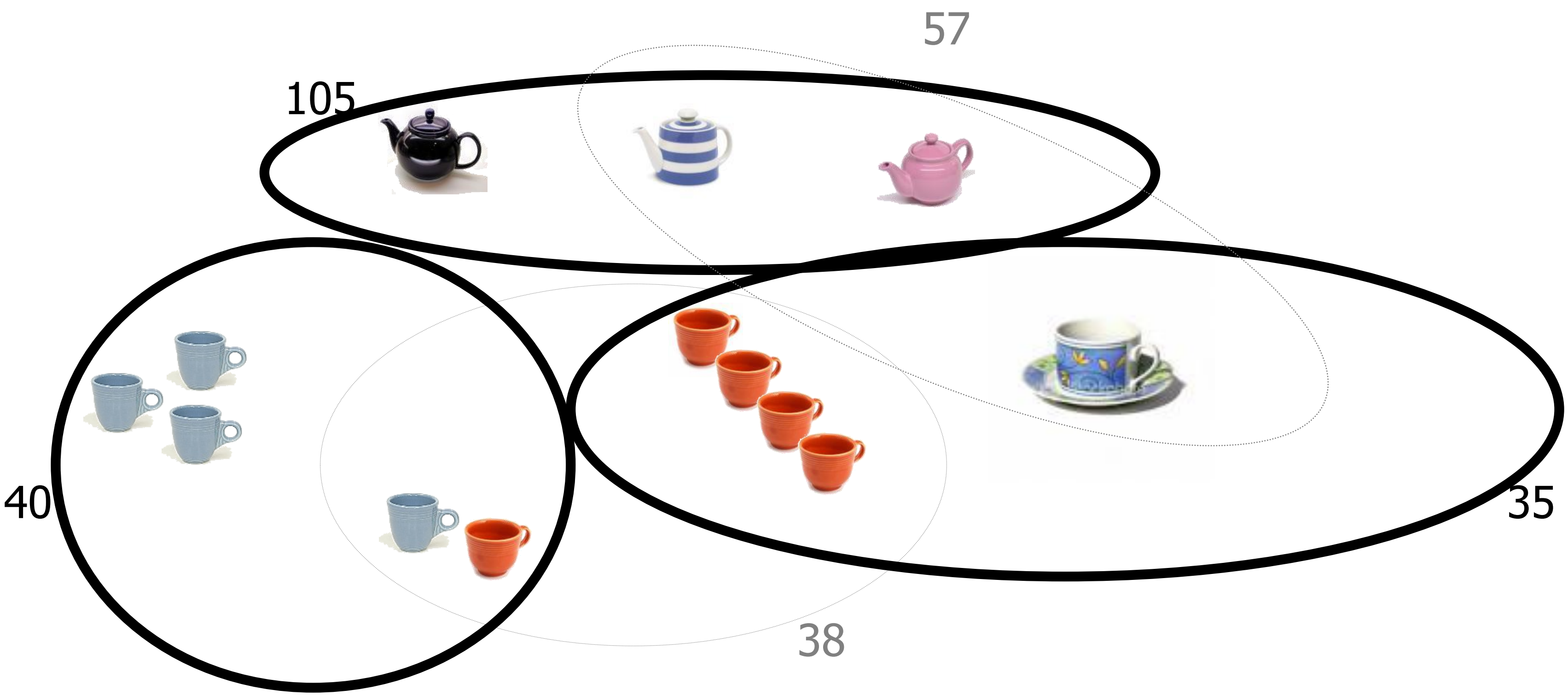
Example Application: Combinatorial Auctions



Winner Determination Problem → **180**

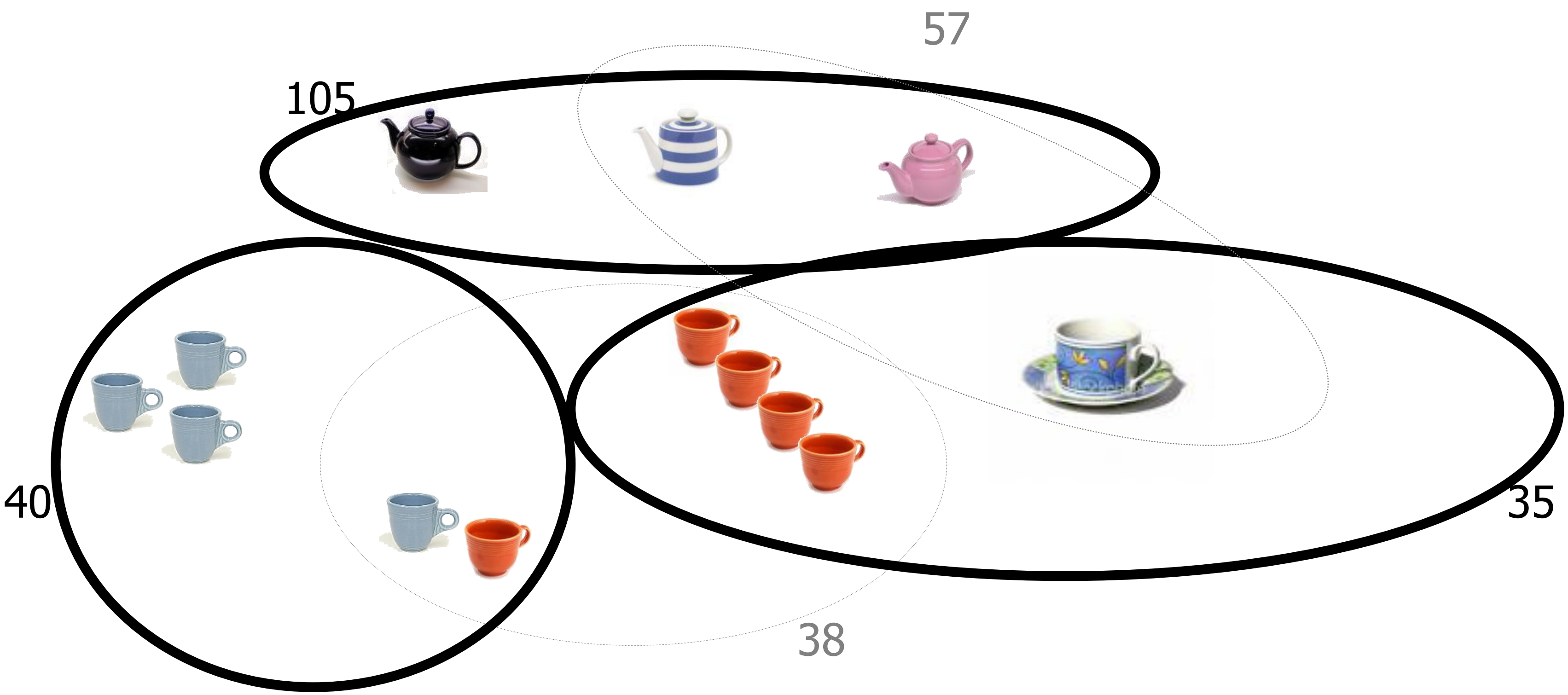
- Determine the outcome that maximizes the sum of accepted bid prices

Example Application: Combinatorial Auctions



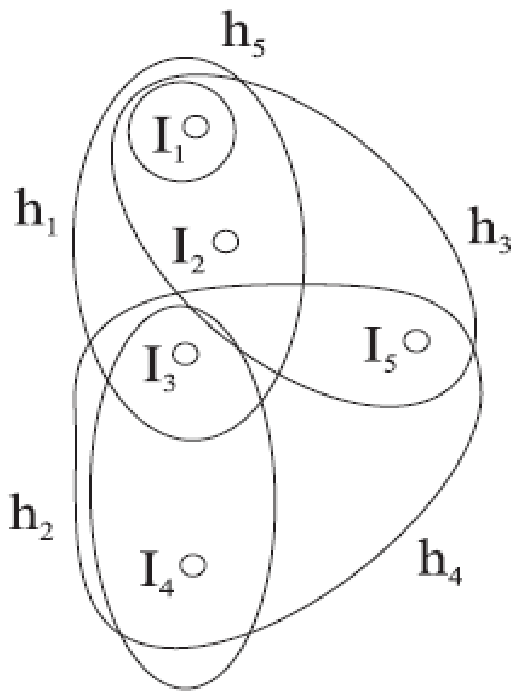
- Other applications [Cramton, Shoham, and Steinberg, '06]
  - airport runway access
  - trucking
  - bus routes
  - industrial procurement

Example Application: Combinatorial Auctions



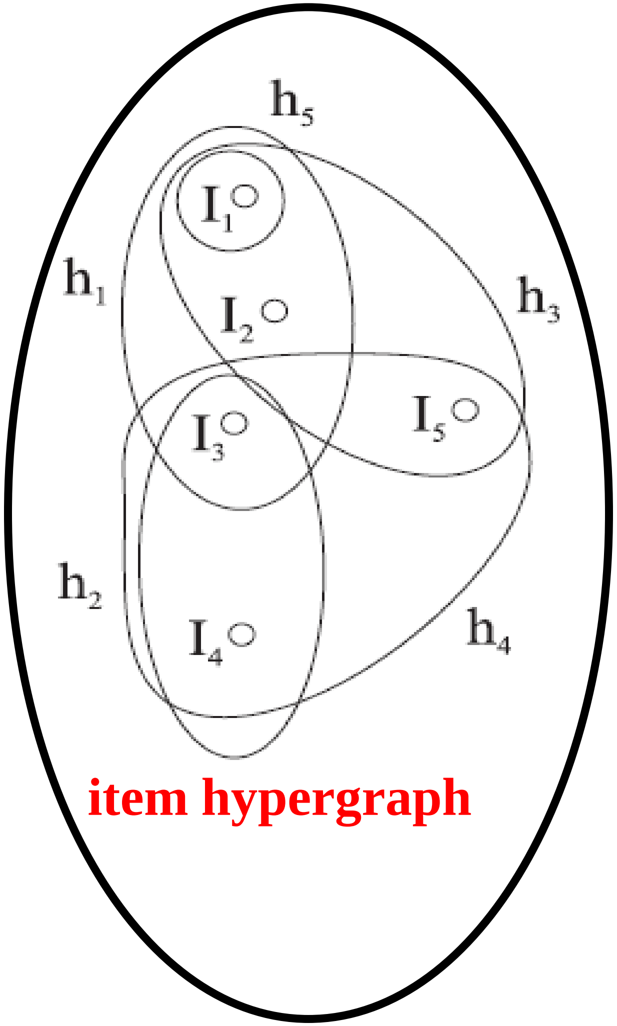
Winner Determination is NP-hard

Structural Properties



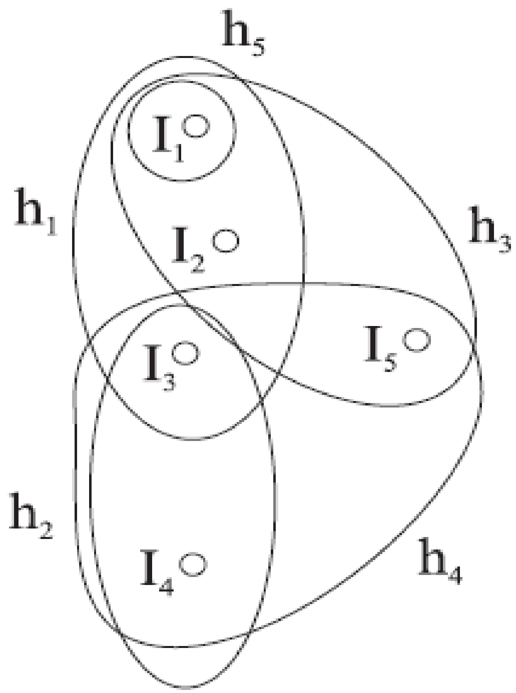
item hypergraph

Structural Properties



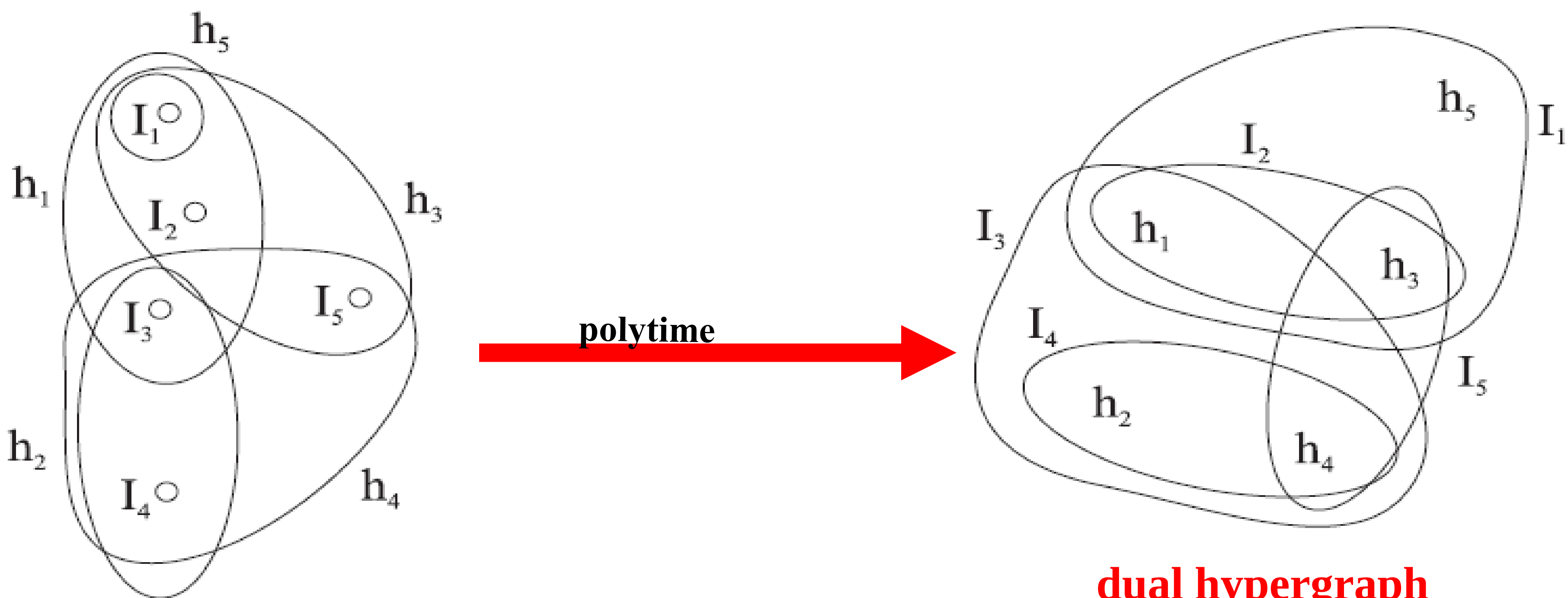
The Winner Determination Problem remains NP-hard even in case of acyclic hypergraphs

Dual Hypergraph



item hypergraph

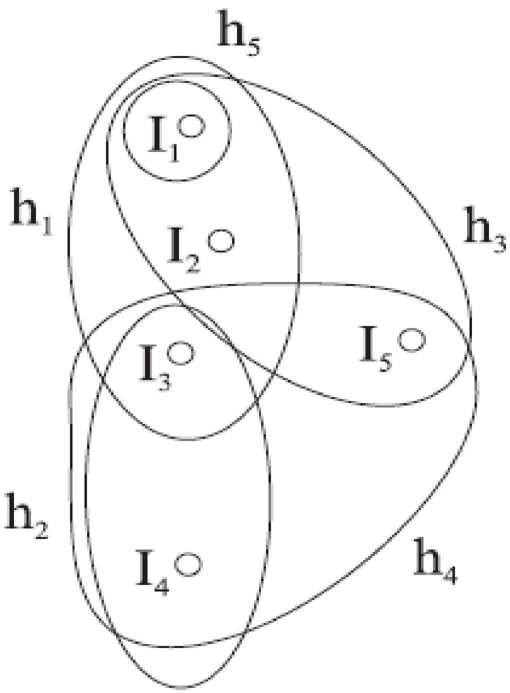
Dual Hypergraph



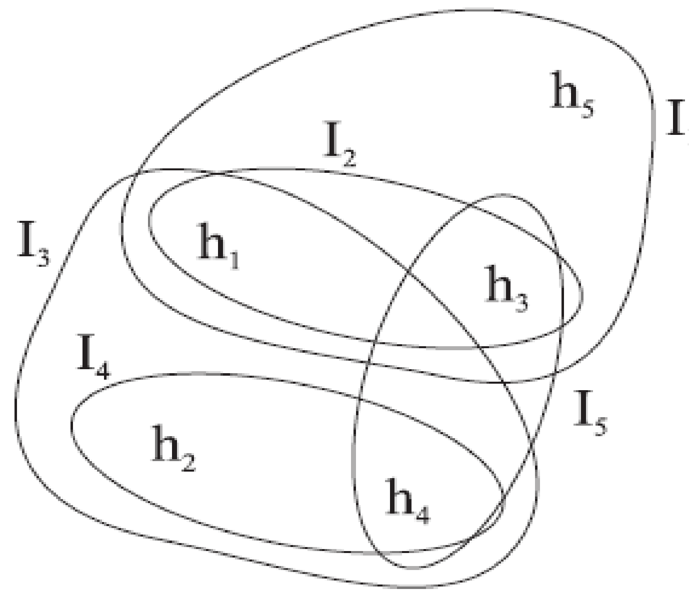
item hypergraph

dual hypergraph

Dual Hypergraph



item hypergraph

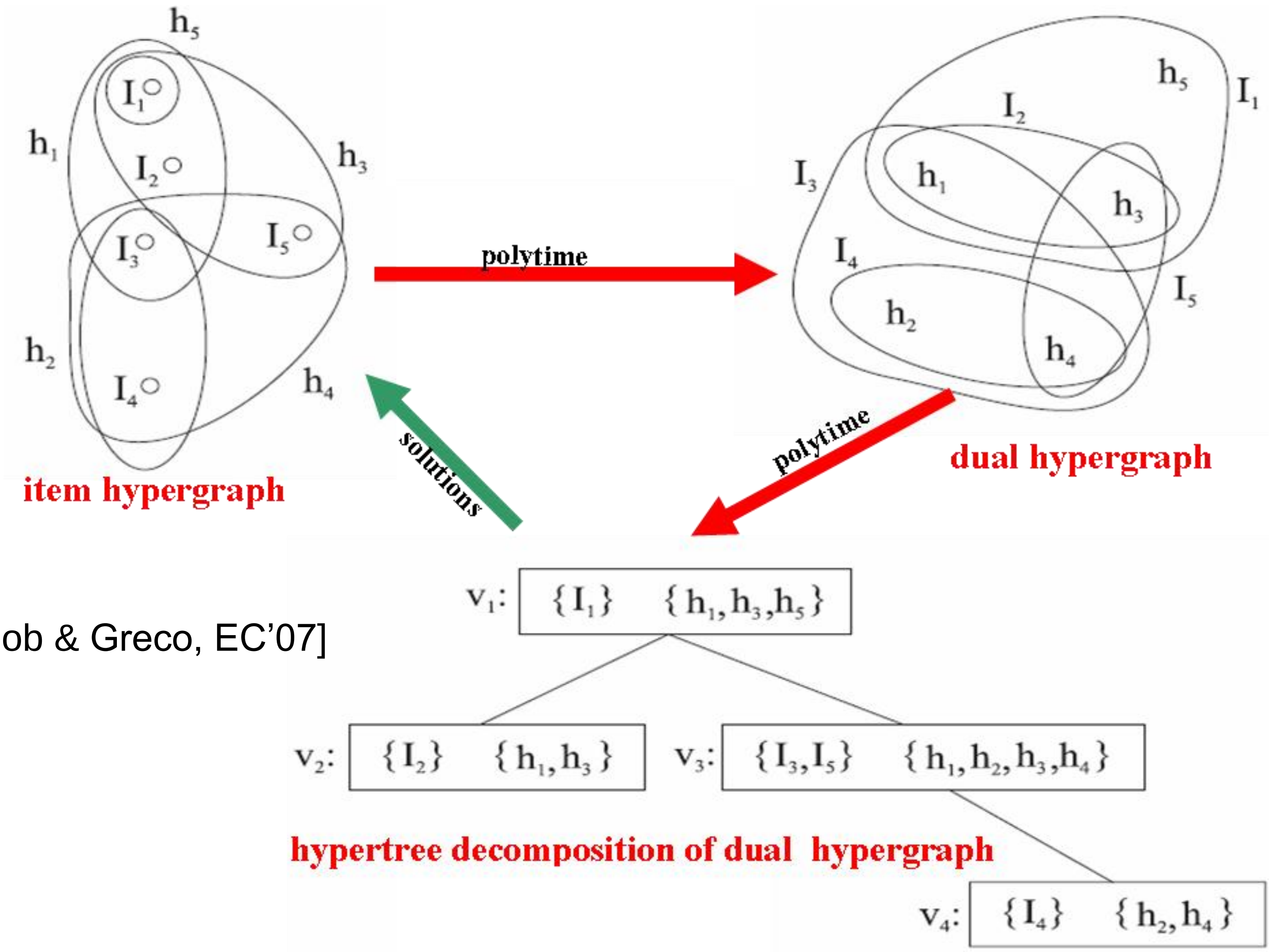


dual hypergraph

I_1

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

The Approach



[Gottlob & Greco, EC'07]

Outline

Identification of “Easy” Classes

Applications of Tree Decompositions

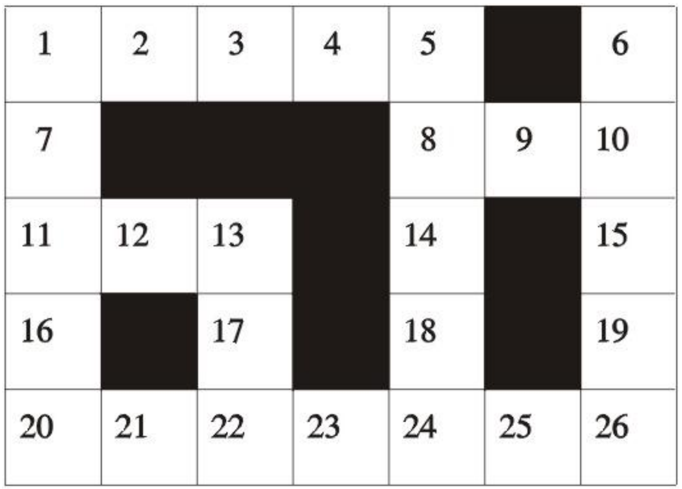
Beyond Tree Decompositions

Decision/Computation Problems

Optimization Problems

Enumeration Problems

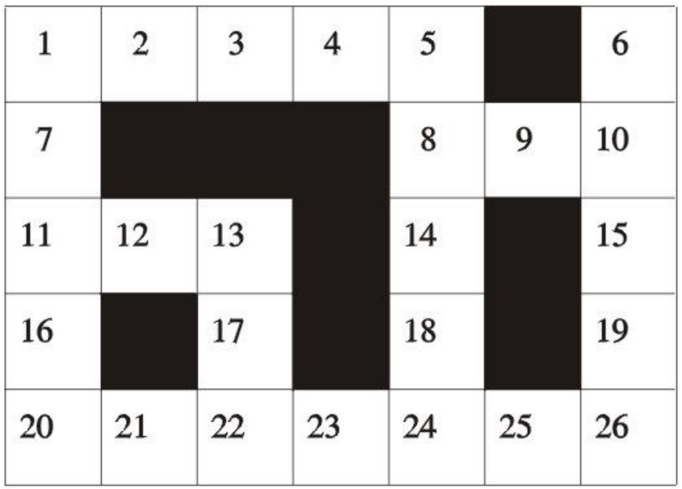
Puzzles for «Very» Experts…



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

- Generate all solutions

Puzzles for «Very» Experts…



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

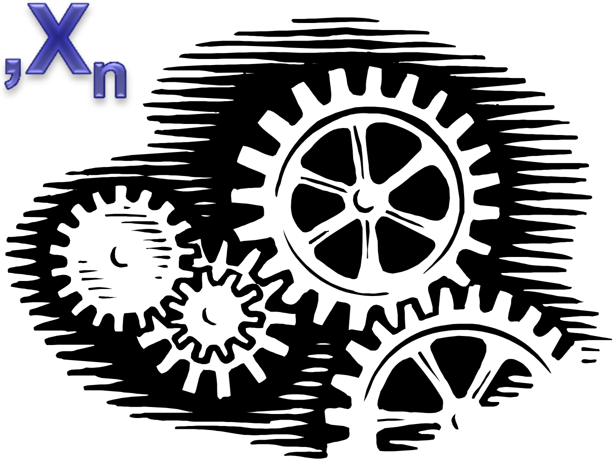
- Generate all solutions

Plain Enumeration

Projection

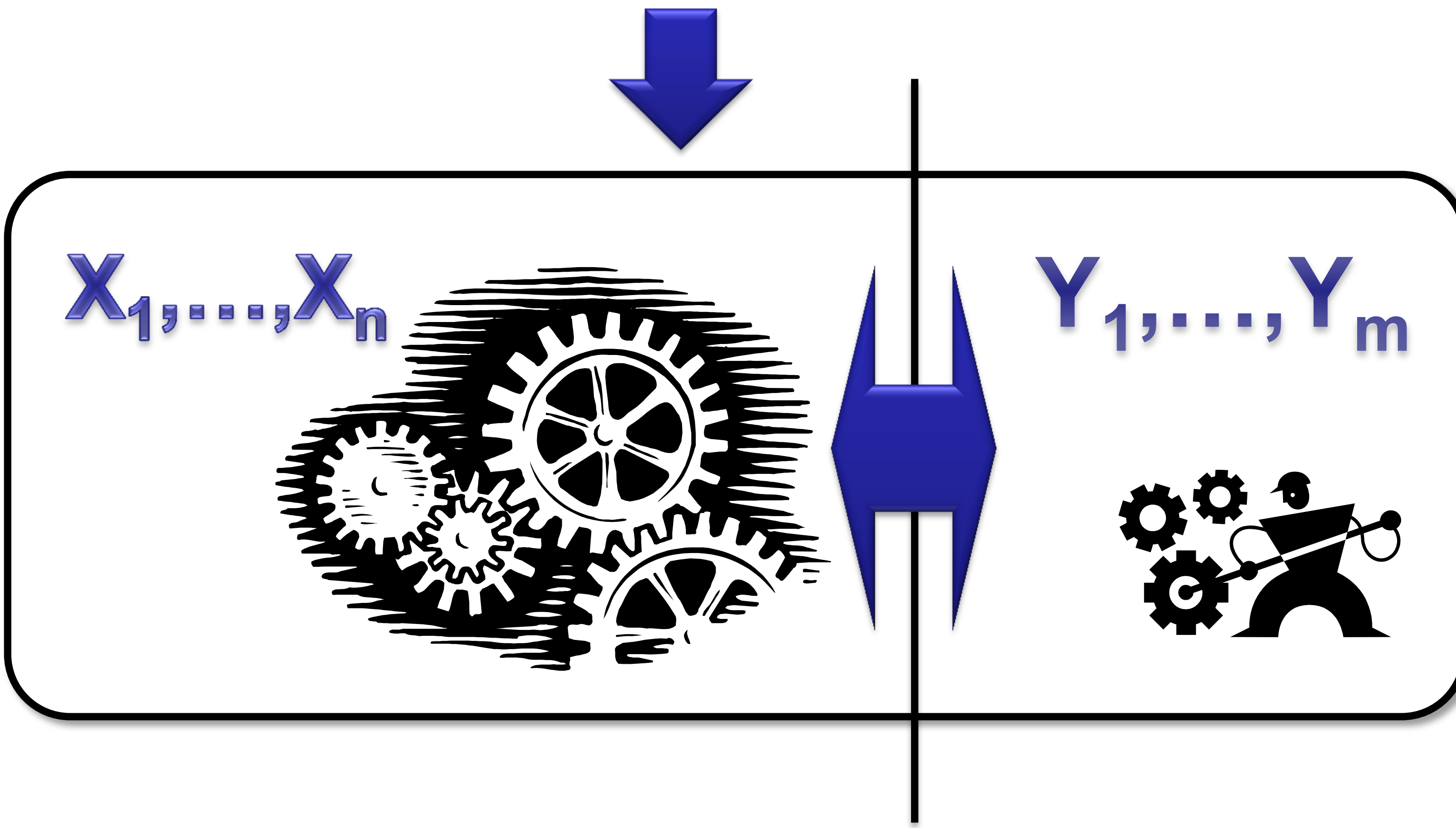
Problem over Variables $X_1,...,X_n$

$X_1,...,X_n$



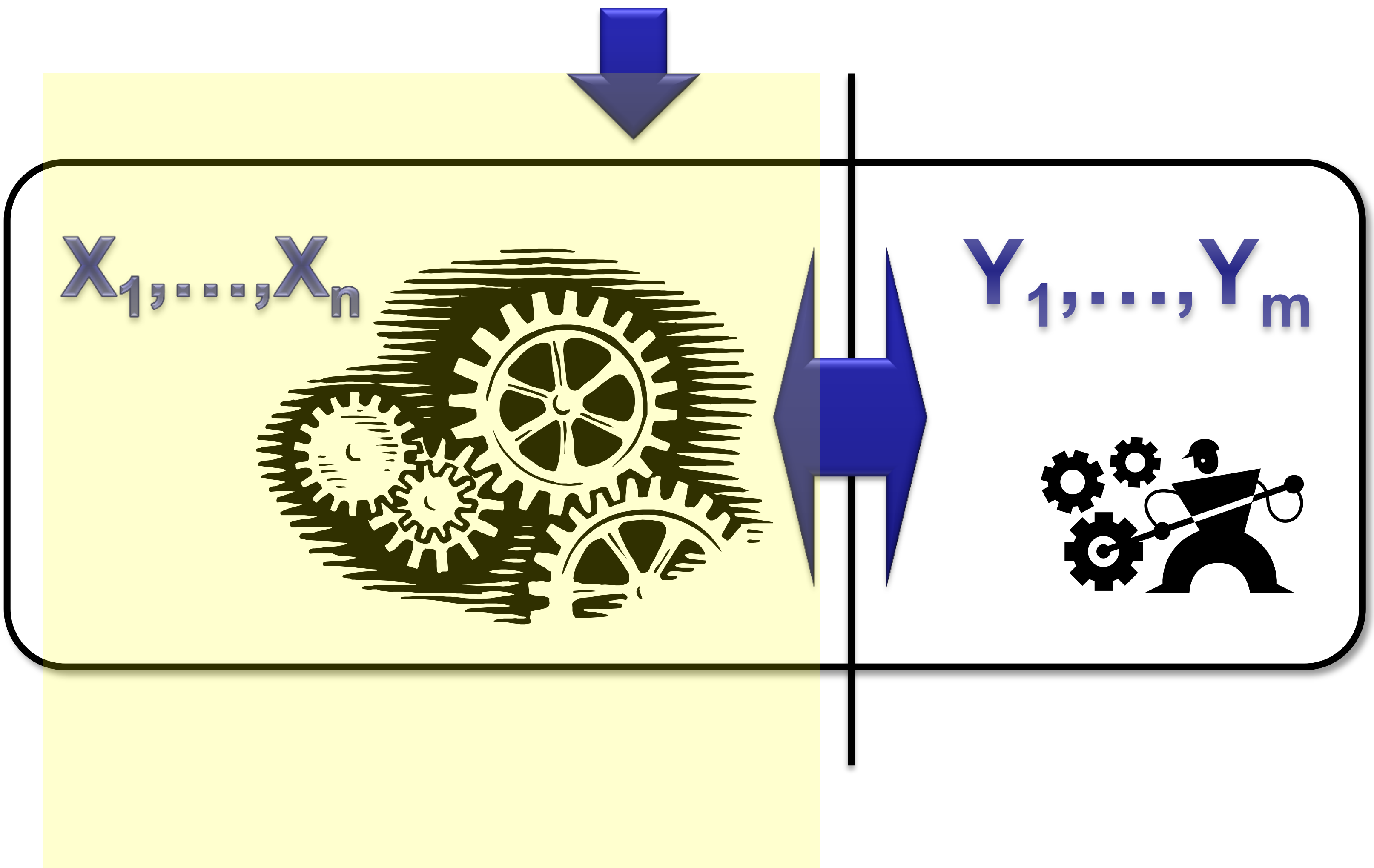
Projection

Problem over Variables $X_1,...,X_n$



Projection

Problem over Variables $X_1,...,X_n$



Questions

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

- *Enumerate* all the homomorphisms in $\mathbb{A}^{\mathbb{B}}$
- For a set of variables X, enumerate the *projection* $\mathbb{A}^{\mathbb{B}}[X]$

Questions

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

- *Enumerate* all the homomorphisms in $\mathbb{A}^{\mathbb{B}}$
- For a set of variables X, enumerate the *projection* $\mathbb{A}^{\mathbb{B}}[X]$

What about efficiency here ?

Enumeration With Polynomial Delay

- All the following tasks are in **POLYNOMIAL TIME**
  - Decide whetere there is no solution
  - Find the *first* solution
  - Given the current solution, find the *next* one
  - After the last solution, check that there are no further ones

Enumeration With Polynomial Delay

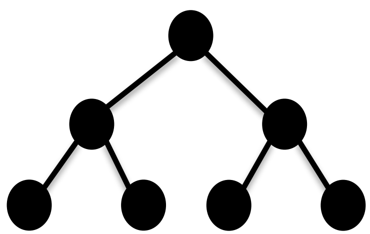
- All the following tasks are in **POLYNOMIAL TIME**
  - Decide whetere there is no solution
  - Find the *first* solution
  - Given the current solution, find the *next* one
  - After the last solution, check that there are no further ones

What about the SPACE

Enumeration With Polynomial Delay

- All the following tasks are in **POLYNOMIAL TIME**
  - Decide whetere there is no solution
  - Find the *first* solution
  - Given the current solution, find the *next* one
  - After the last solution, check that there are no further ones

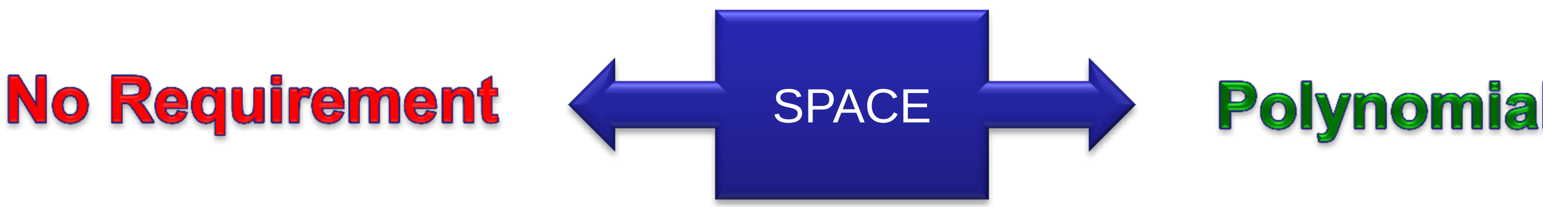
---



exponential space, but
operations in *polynomial time*

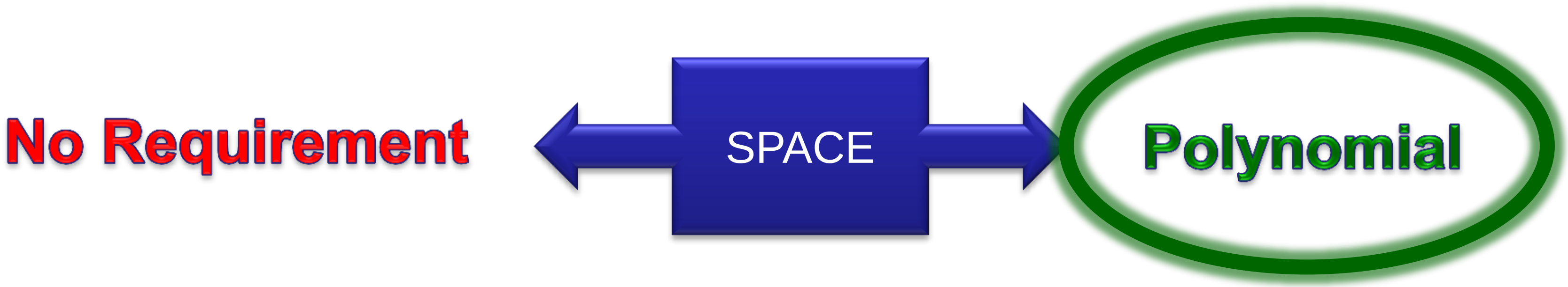
Enumeration With Polynomial Delay

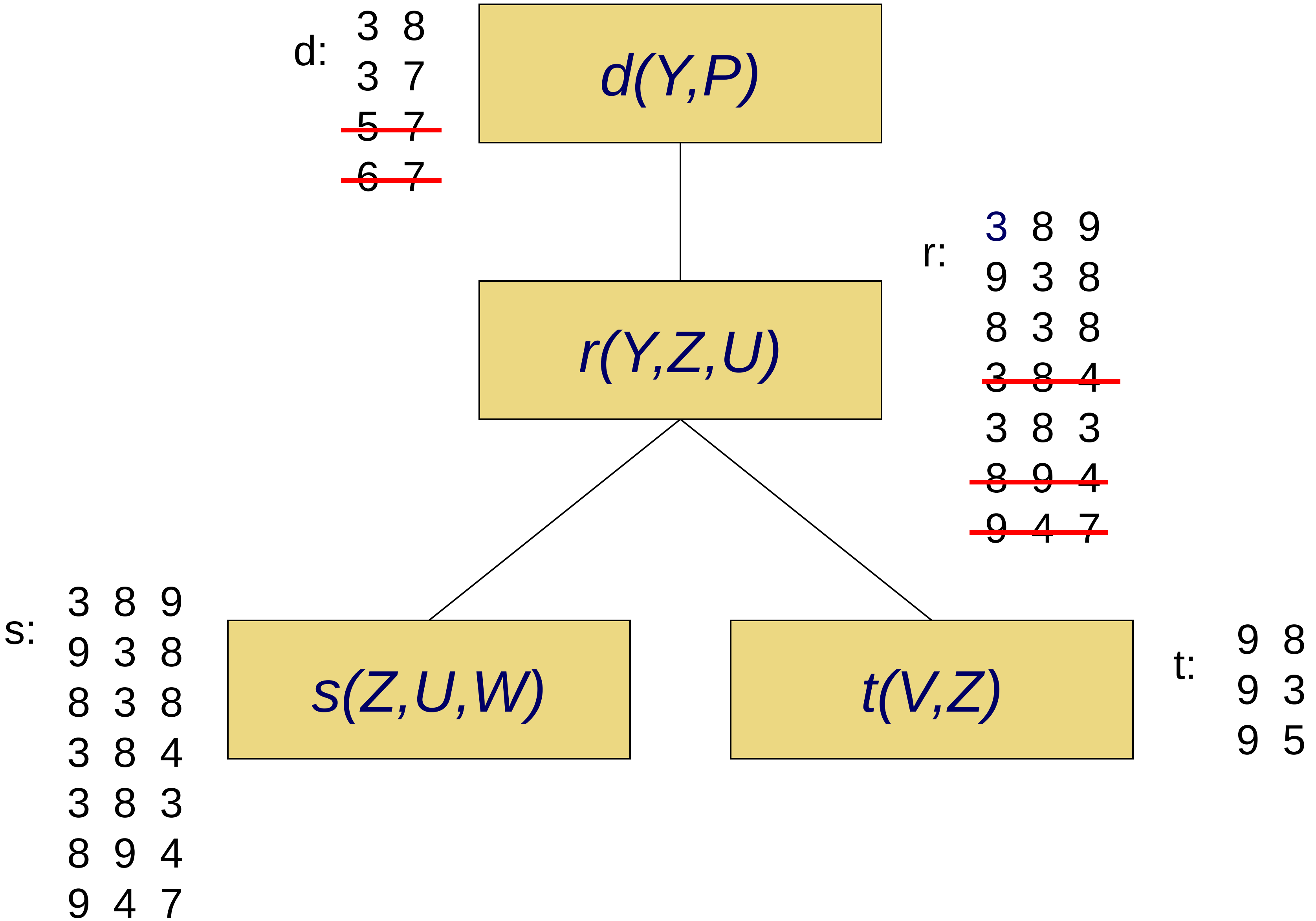
- All the following tasks are in **POLYNOMIAL TIME**
  - Decide whetere there is no solution
  - Find the *first* solution
  - Given the current solution, find the *next* one
  - After the last solution, check that there are no further ones

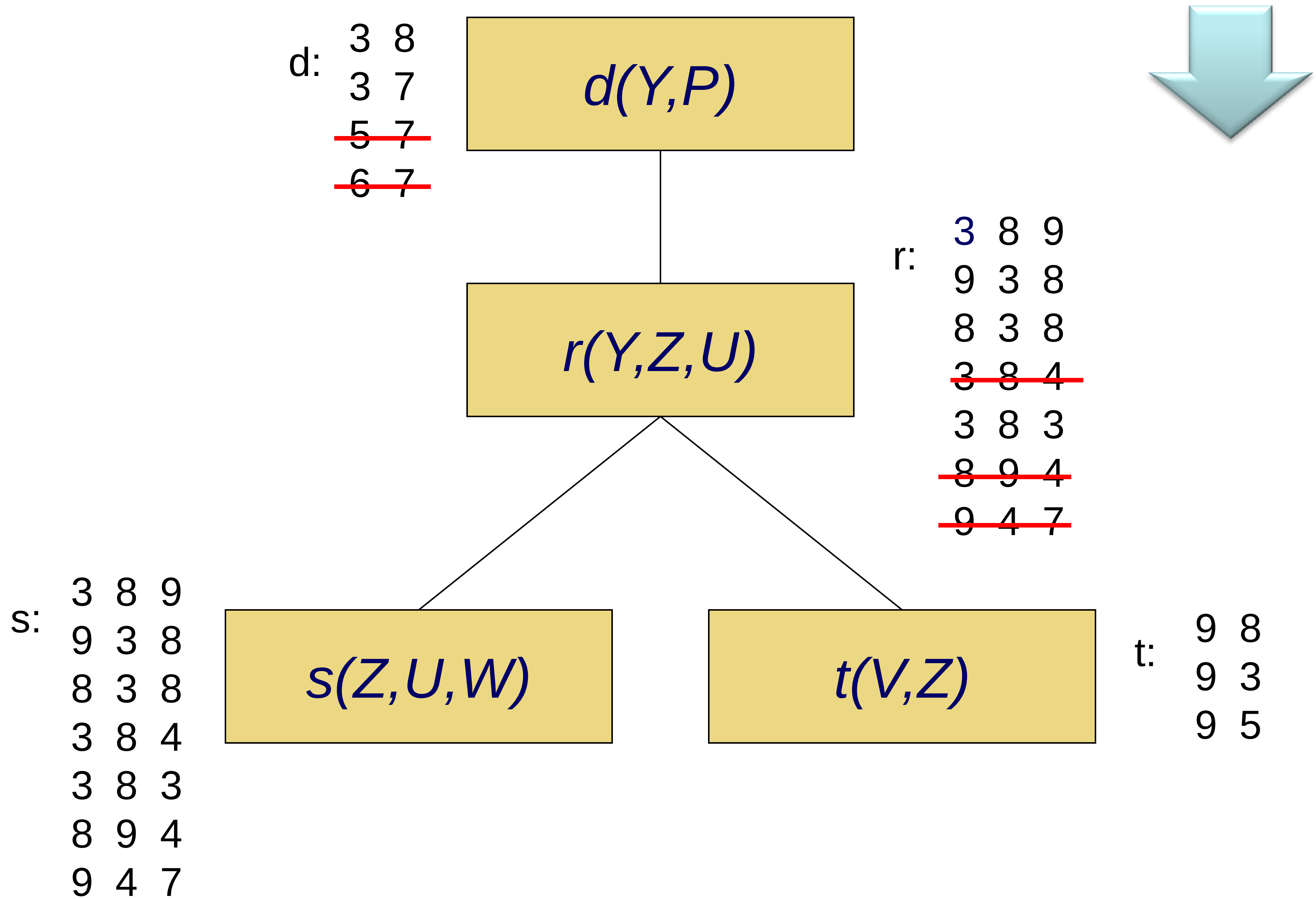


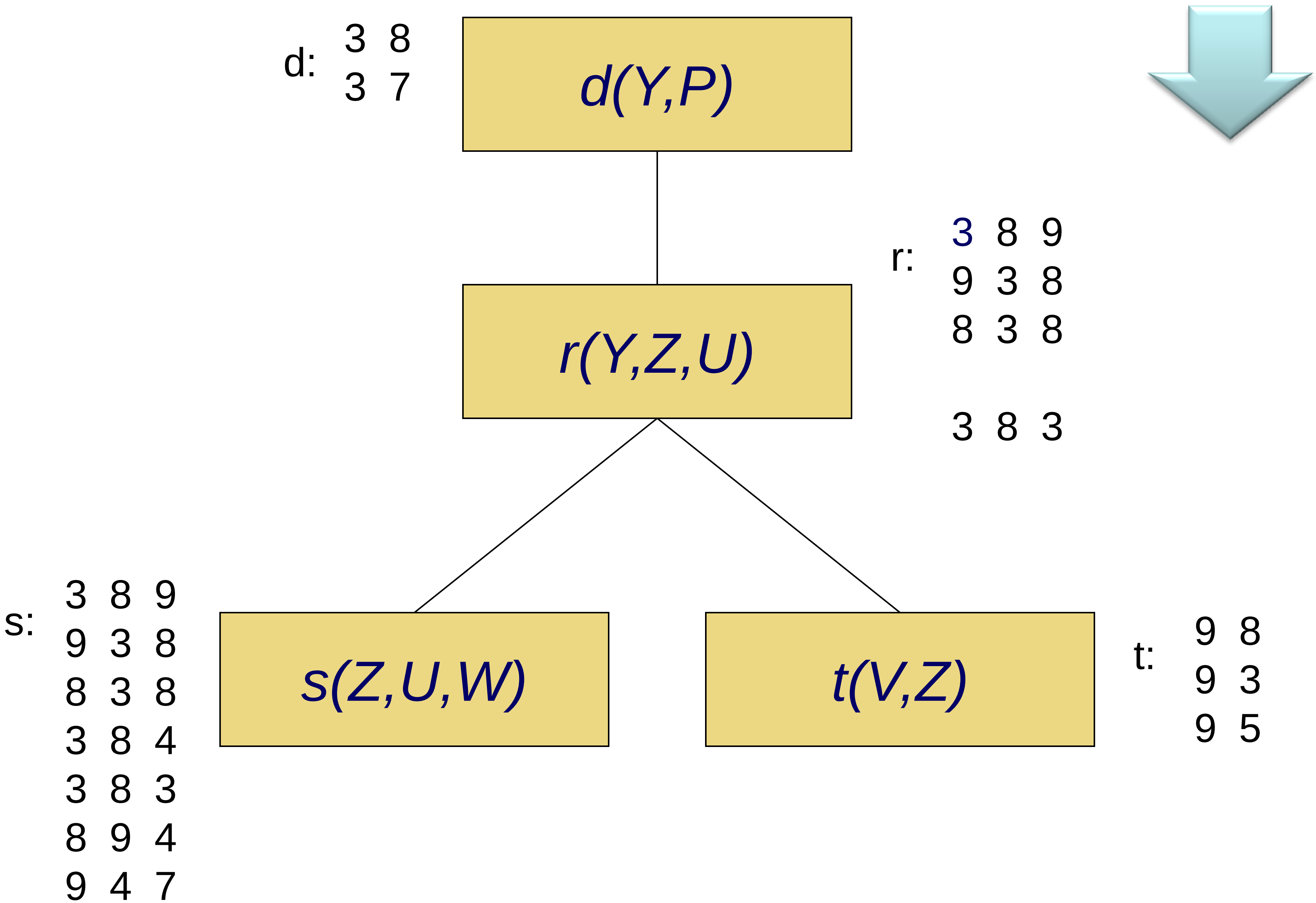
Enumeration With Polynomial Delay

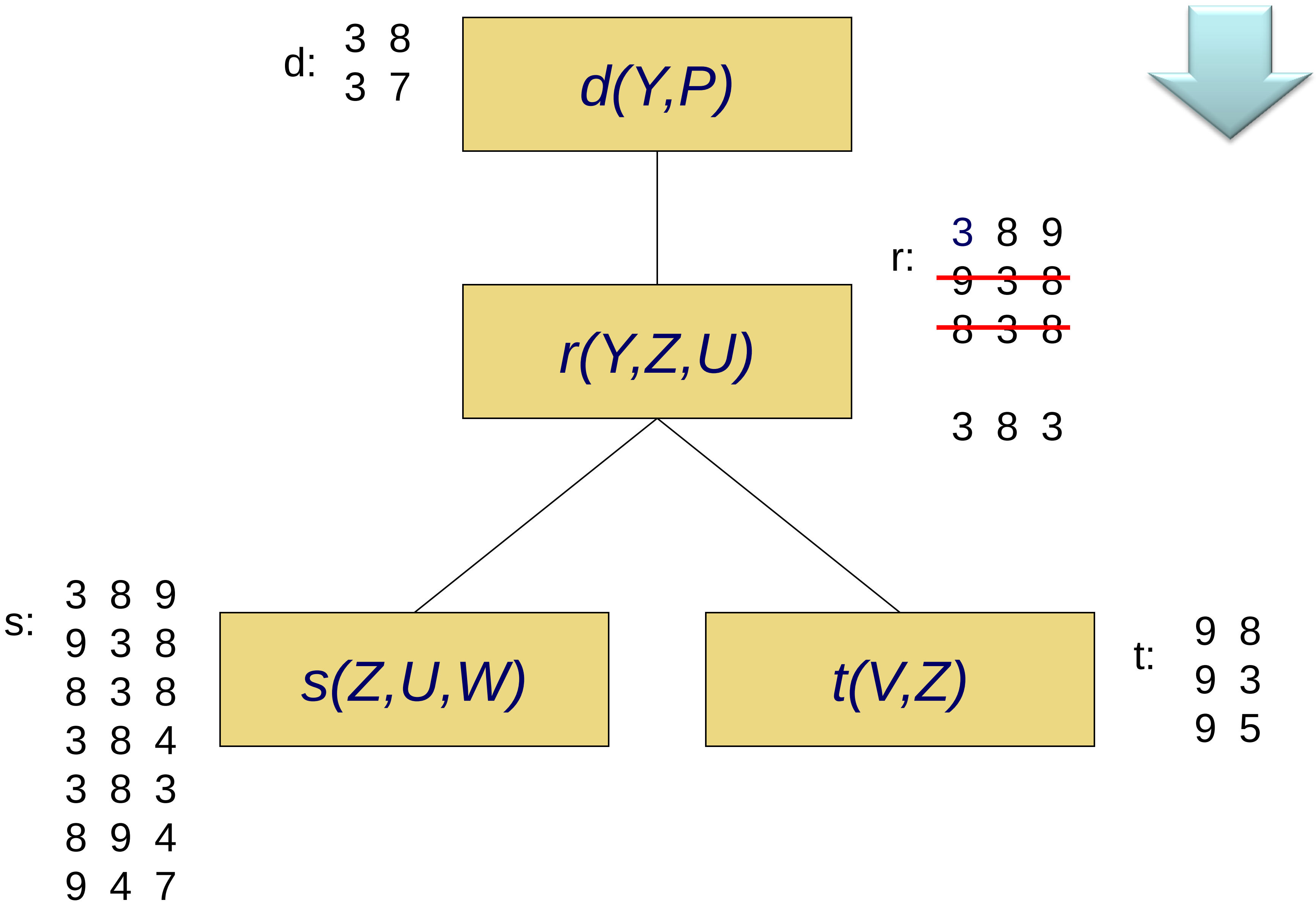
- All the following tasks are in **POLYNOMIAL TIME**
  - Decide whetere there is no solution
  - Find the *first* solution
  - Given the current solution, find the *next* one
  - After the last solution, check that there are no further ones

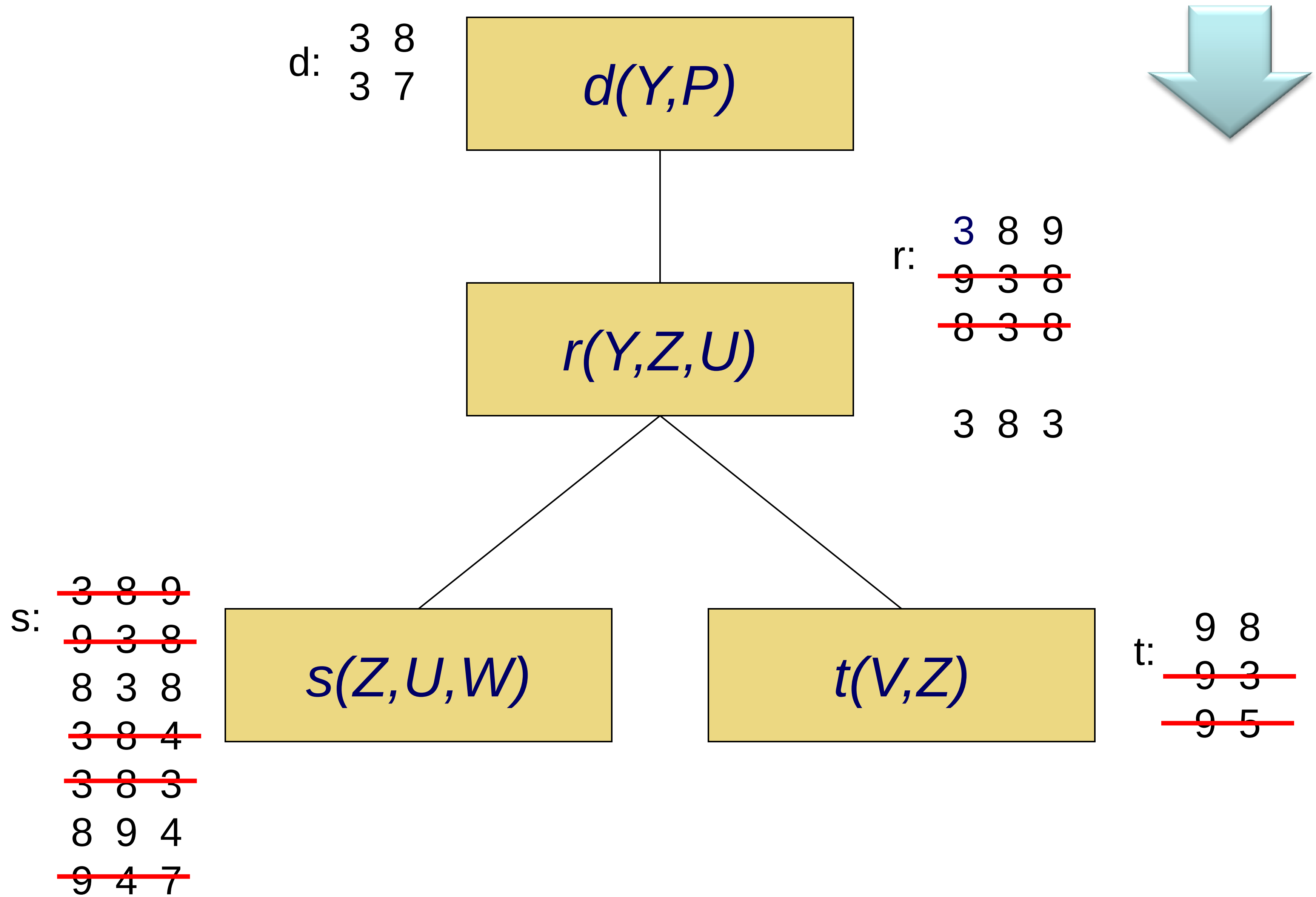



d(Y,P)
r(Y,Z,U)
s(Z,U,W)
t(V,Z)
d:
3 8
3 7
~~5 7~~
~~6 7~~
r:
3 8 9
9 3 8
8 3 8
~~3 8 4~~
3 8 3
~~8 9 4~~
~~9 4 7~~
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
t:
9 8
9 3
9 5

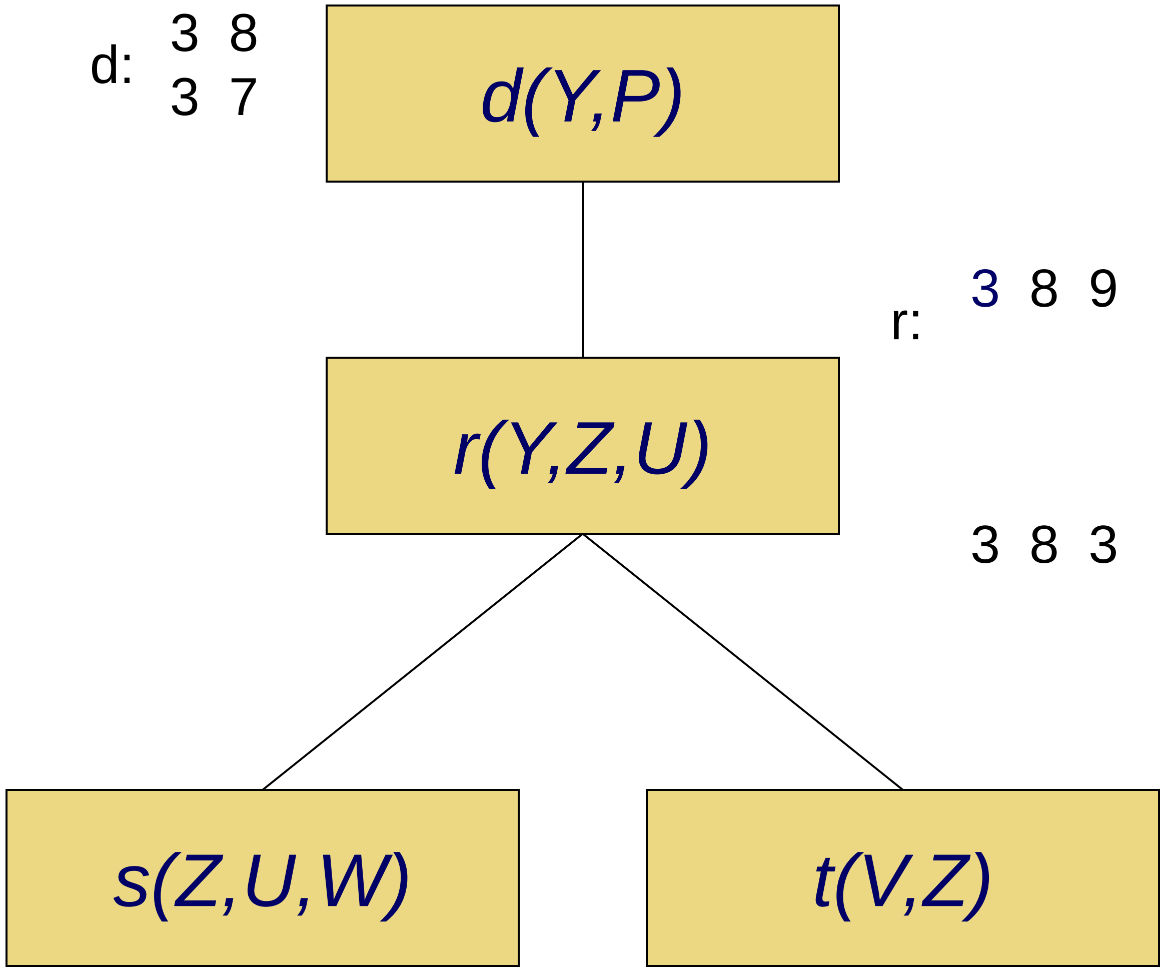

d(Y,P)
r(Y,Z,U)
s(Z,U,W)
t(V,Z)
d:
3 8
3 7
~~5 7~~
~~6 7~~
r:
3 8 9
9 3 8
8 3 8
~~3 8 4~~
3 8 3
~~8 9 4~~
~~9 4 7~~
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
t:
9 8
9 3
9 5


d:
3 8
3 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 3
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
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 3
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
d(Y,P)
r: 3 8 9
~~9 3 8~~
~~8 3 8~~
3 8 3
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~~

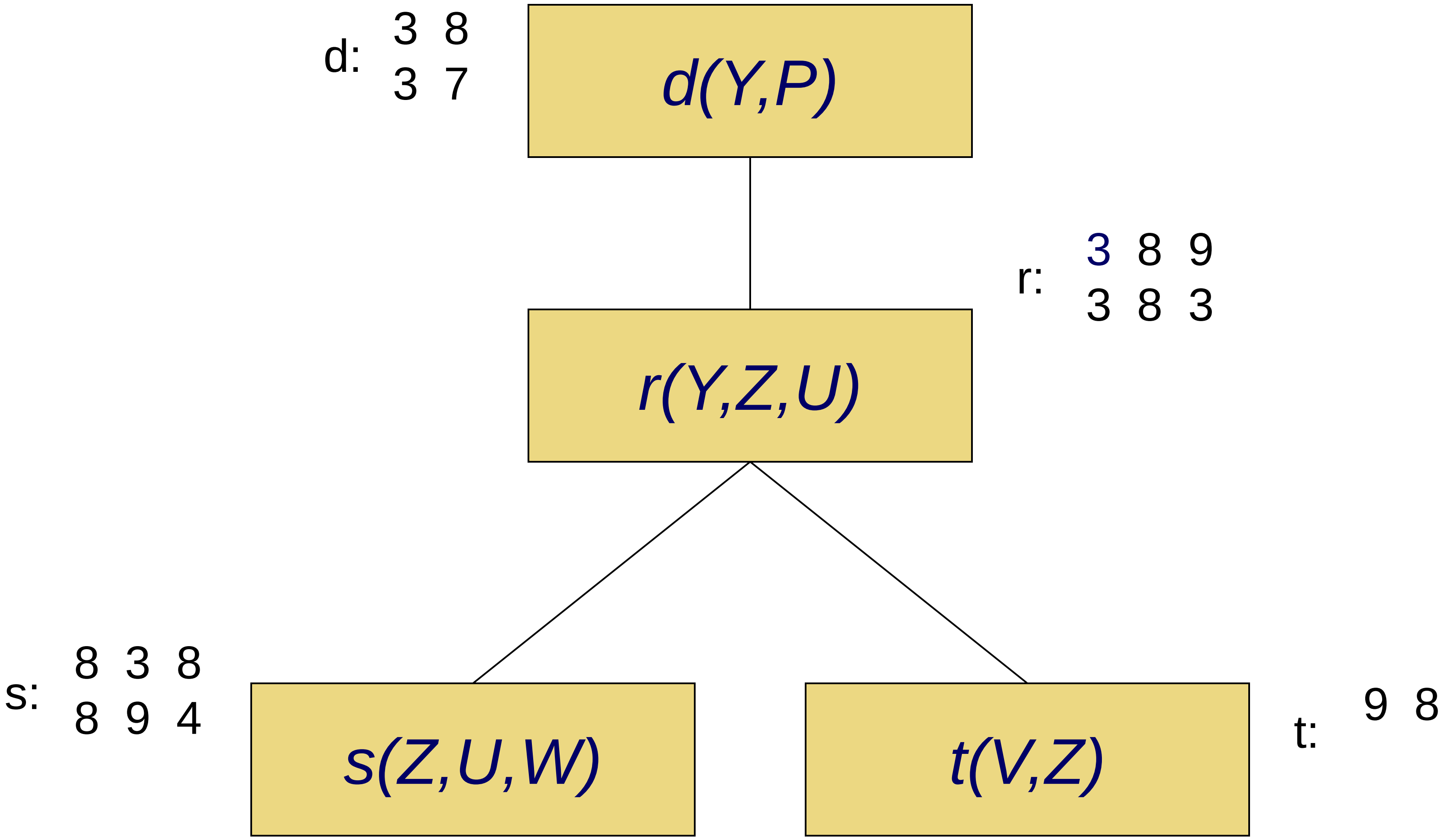
d:
3 8
3 7

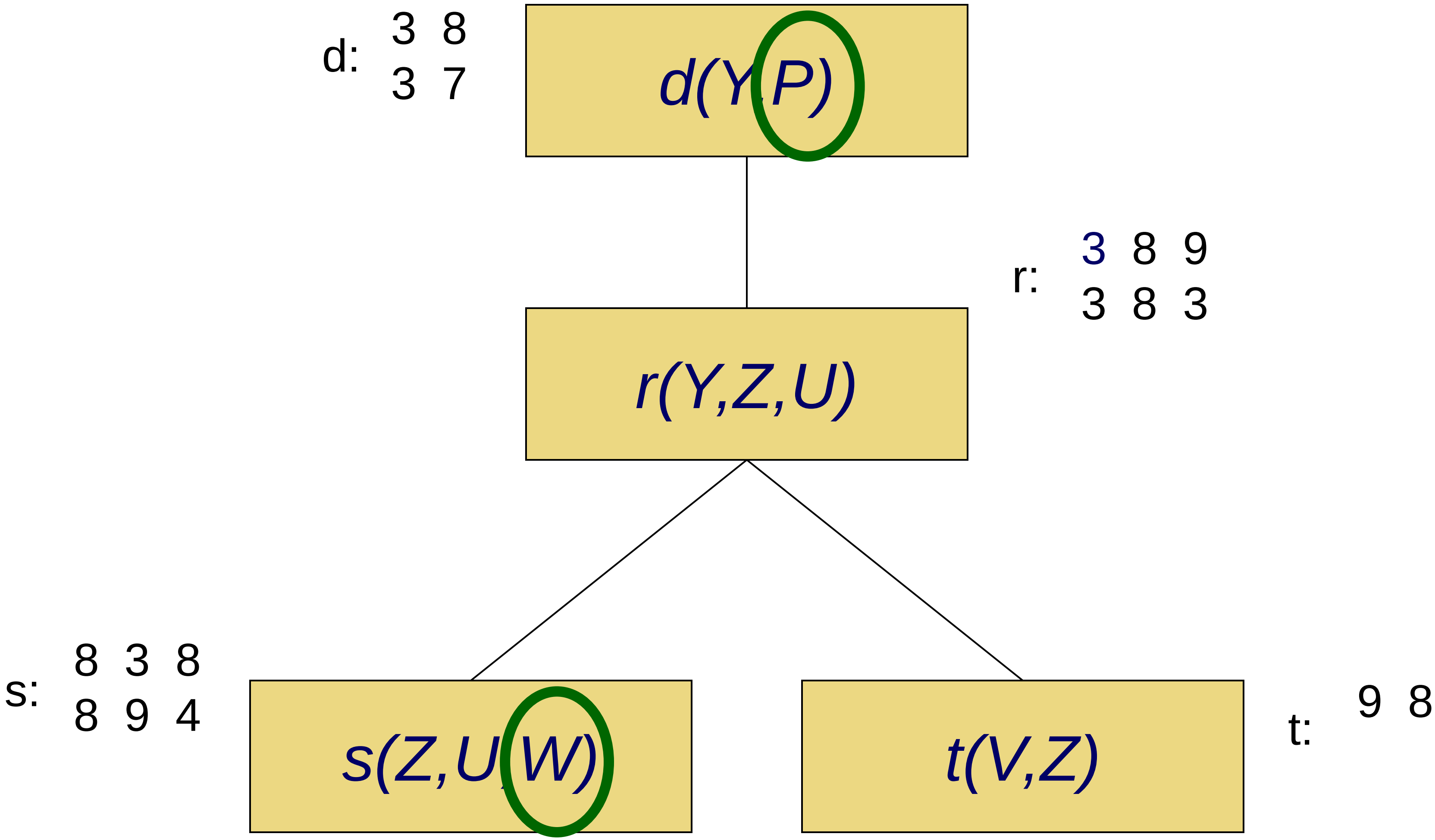


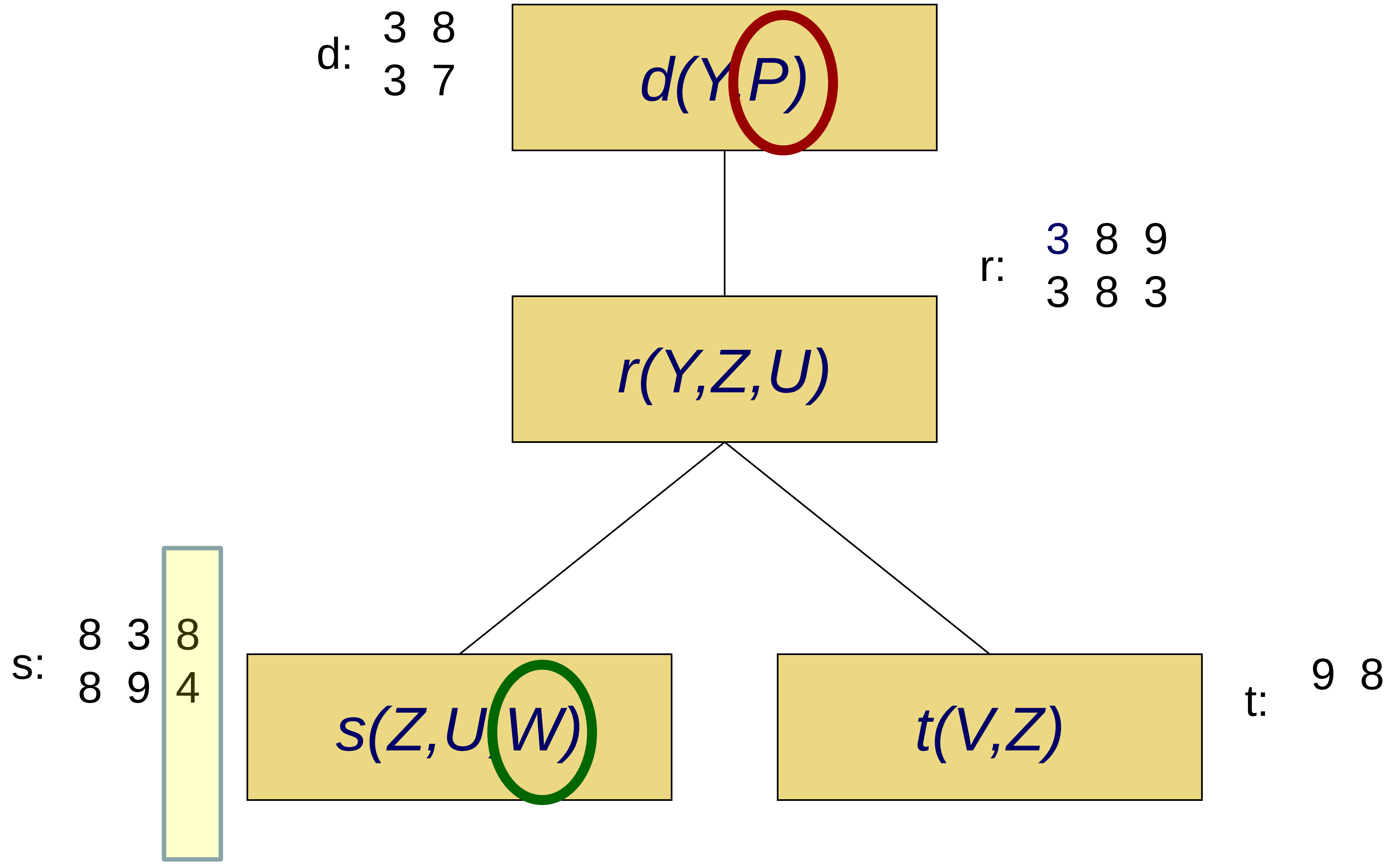
r:
3 8 9
3 8 3

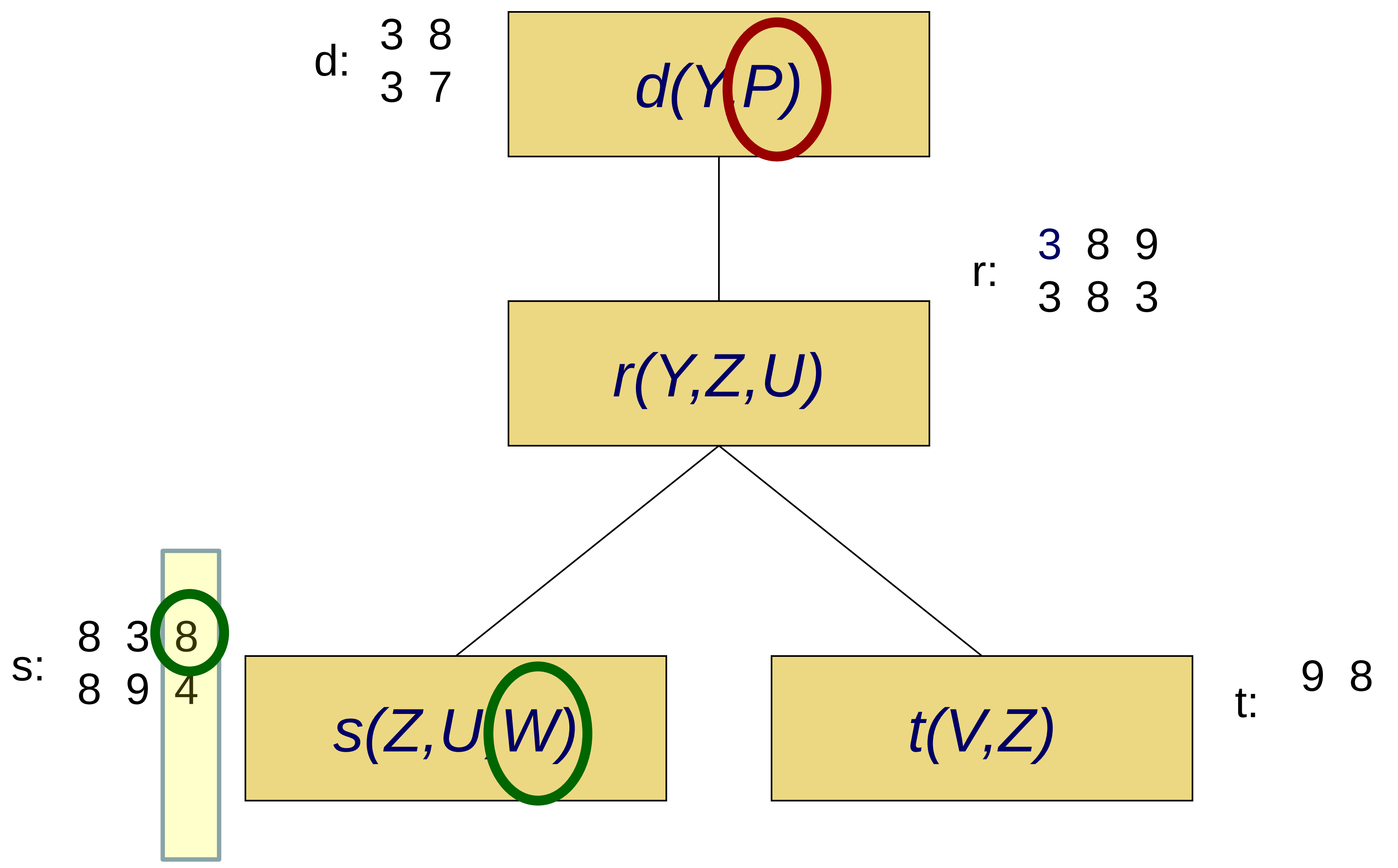
s:
8 3 8
8 9 4

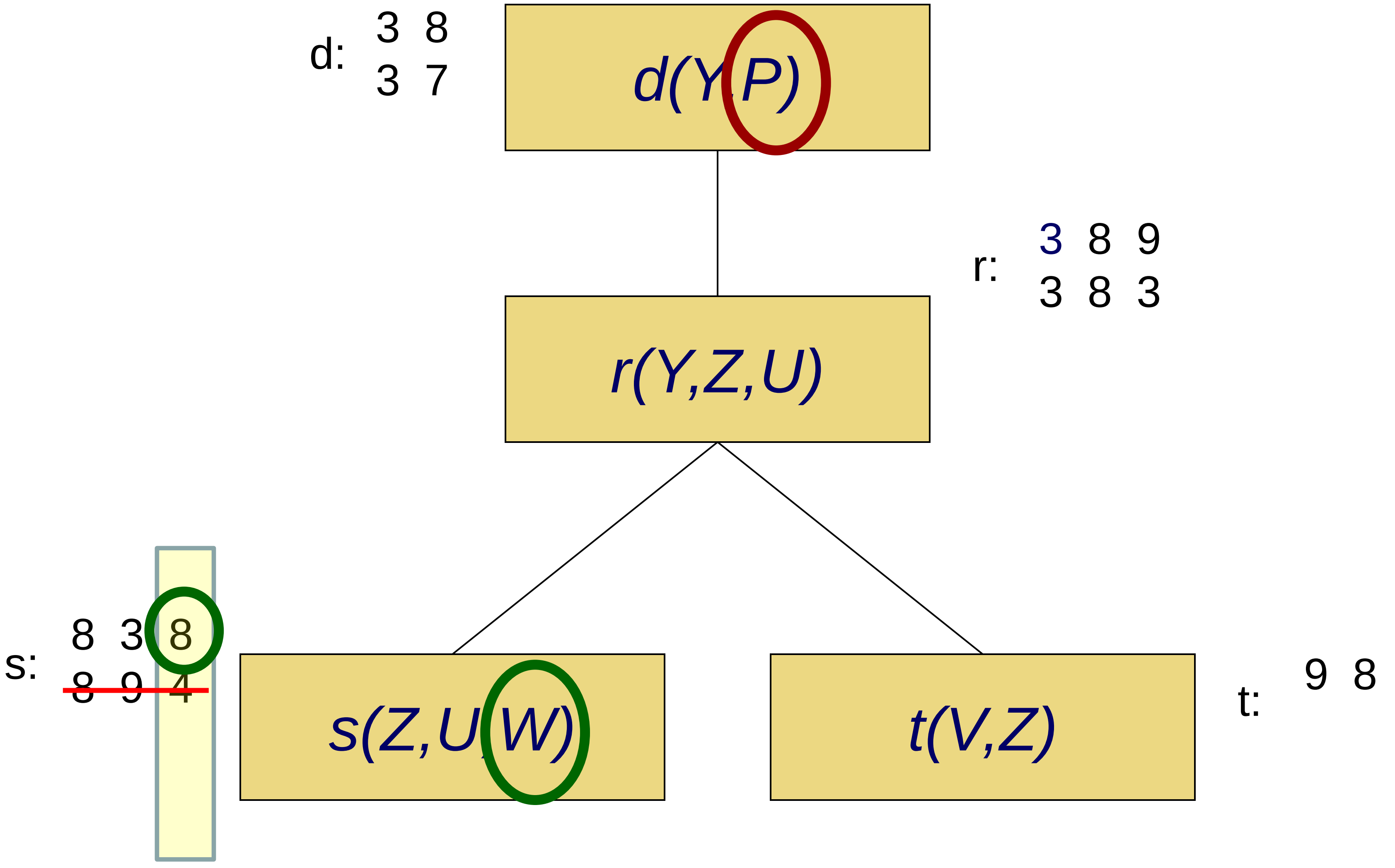
t:
9 8


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

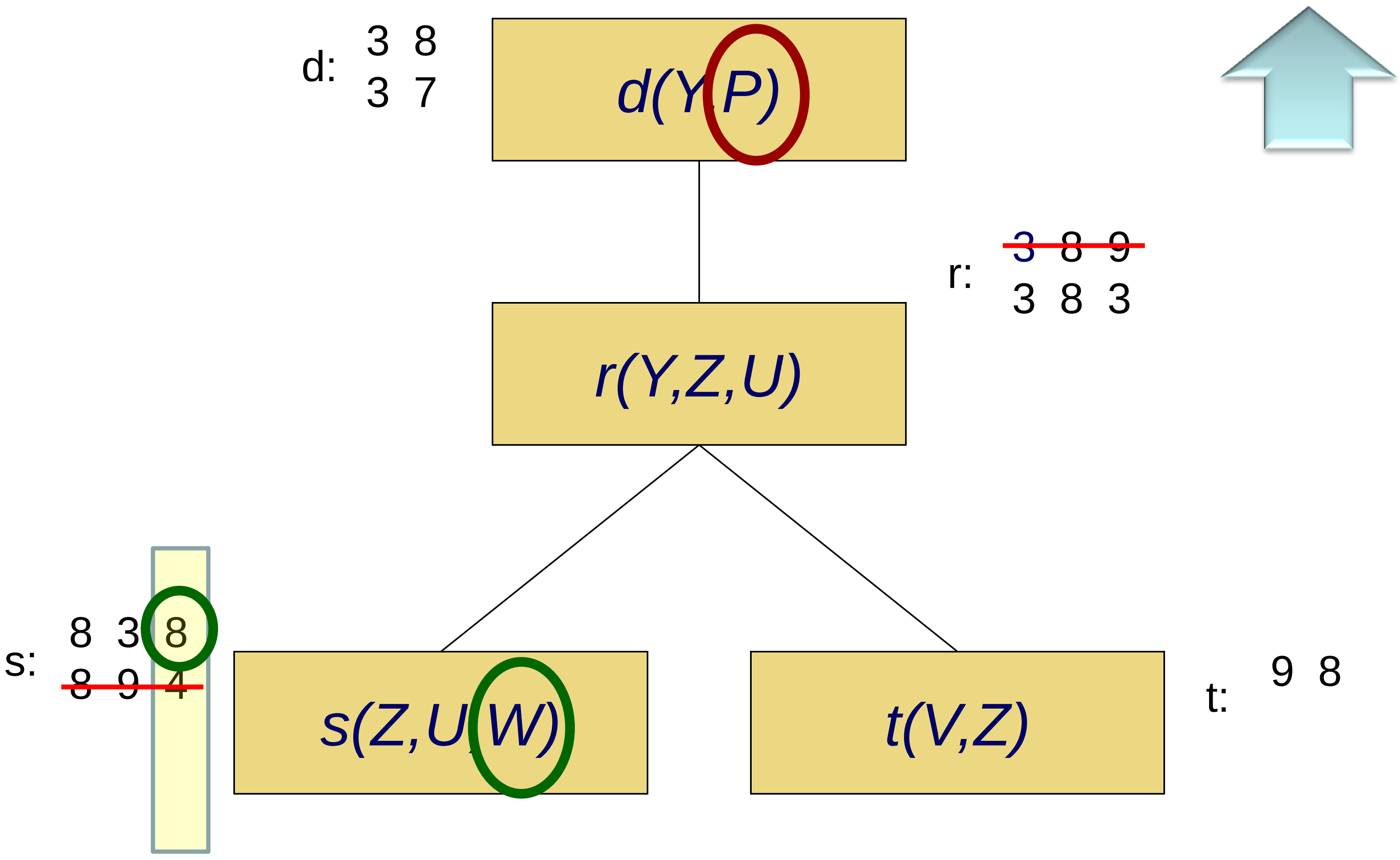

d(Y,P)
d:
3 8
3 7
r(Y,Z,U)
r:
3 8 9
3 8 3
s(Z,U,W)
s:
8 3 8
8 9 4
t(V,Z)
t:
9 8


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

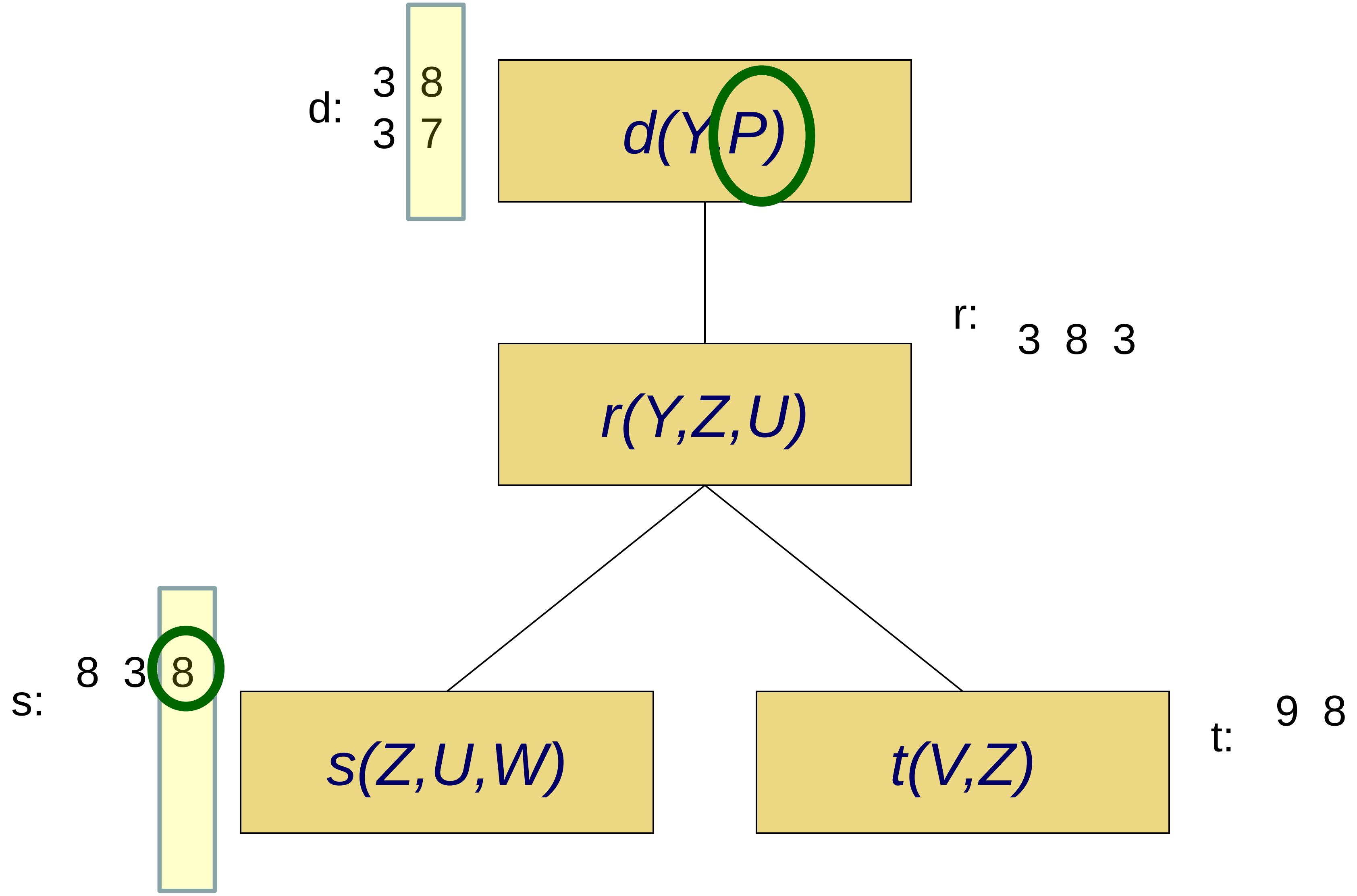

d:
3 8
3 7
d(Y,P)
r:
3 8 9
3 8 3
r(Y,Z,U)
s:
8 3 8
8 9 4
s(Z,U,W)
t(V,Z)
t:
9 8



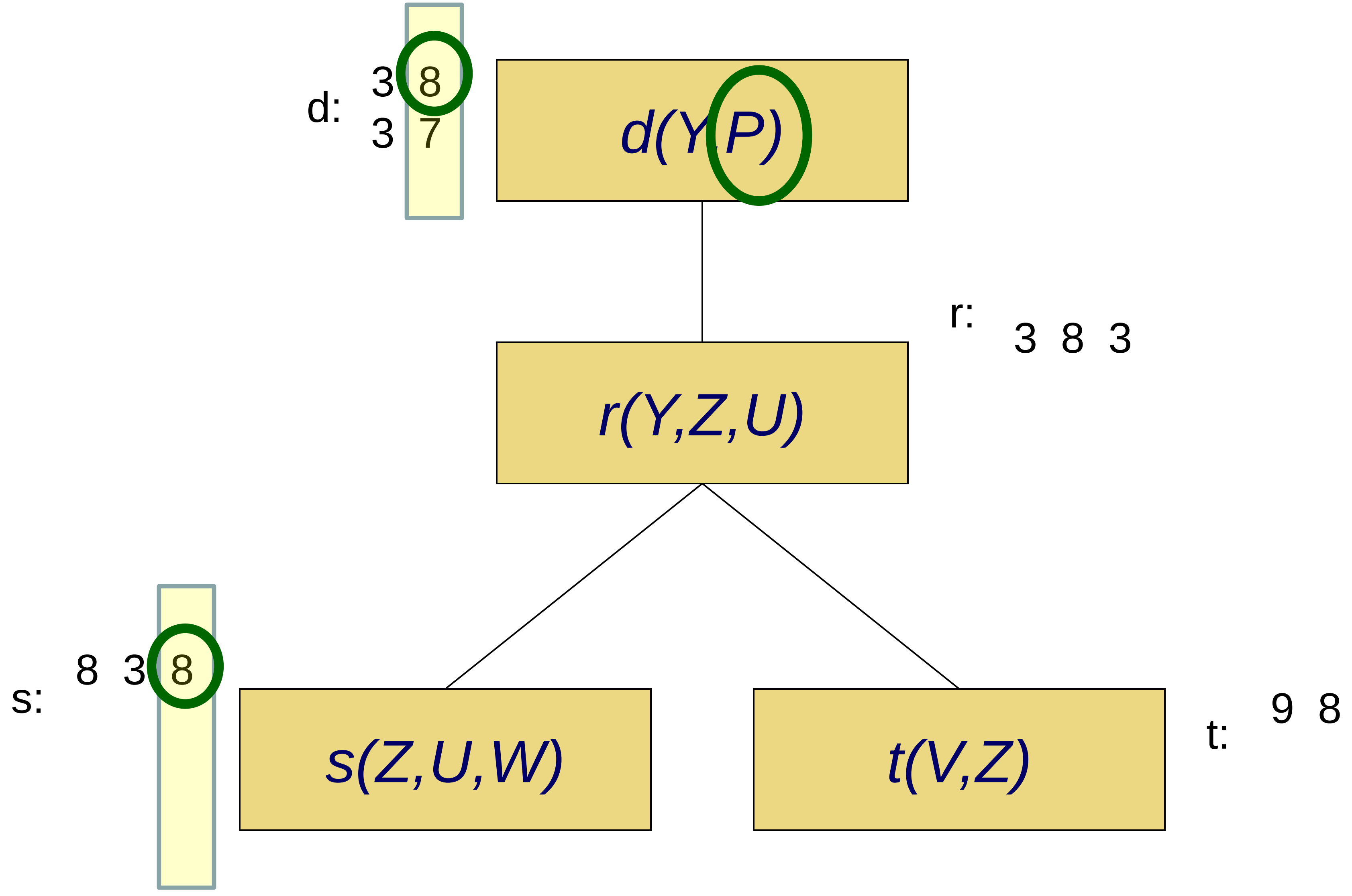
Fix a value, and propagate



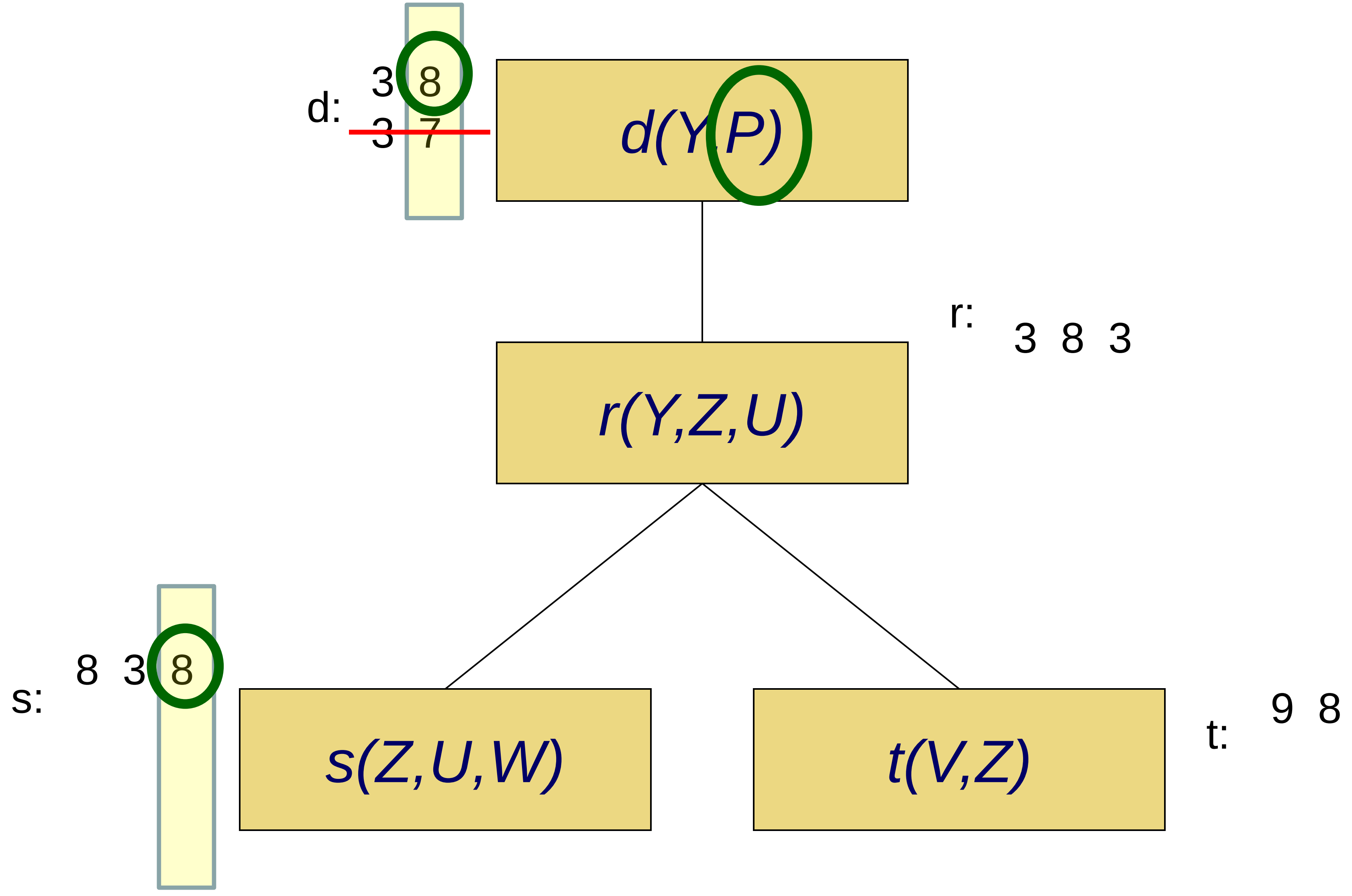
Fix a value, and propagate



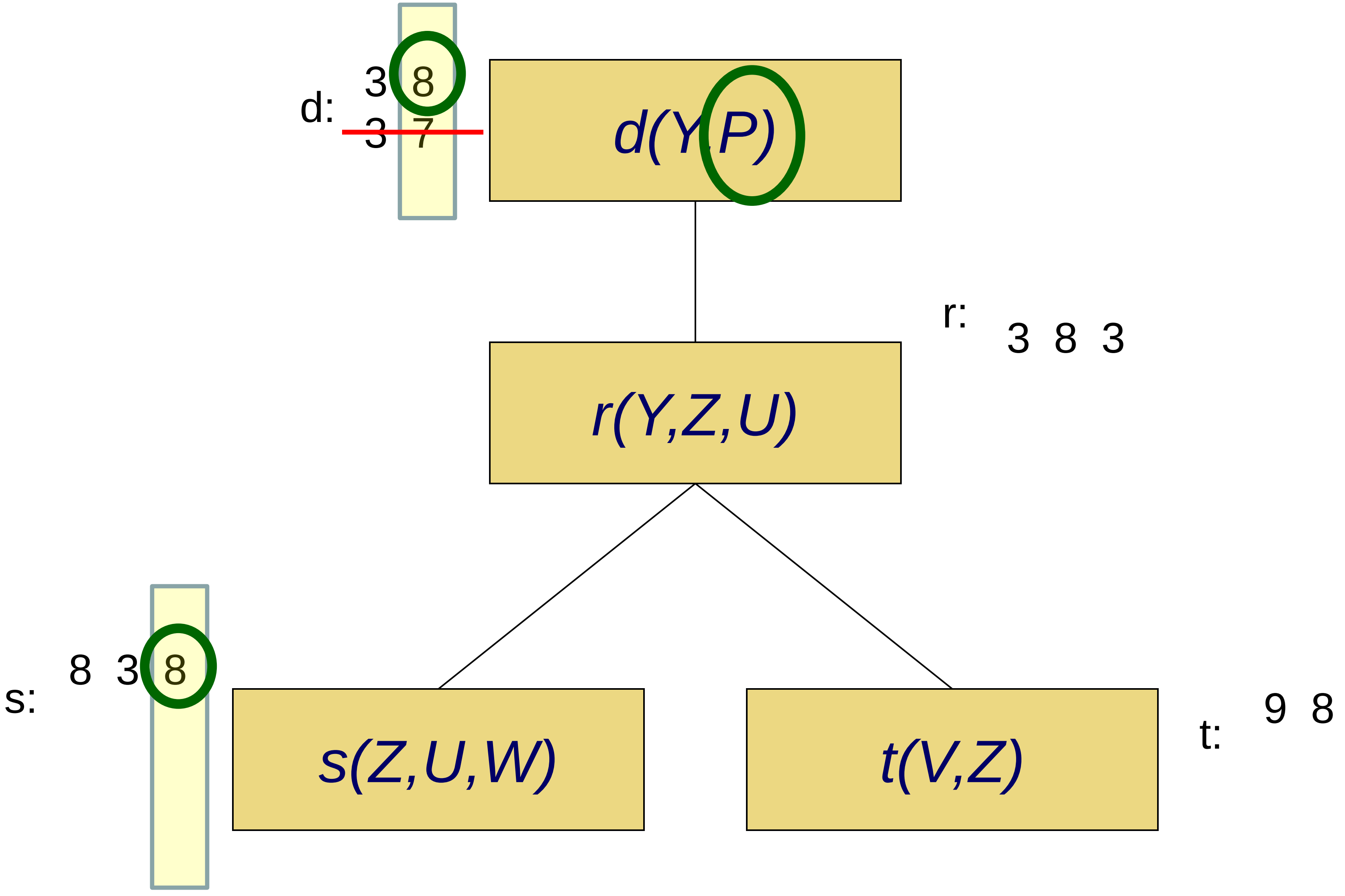
Focus on the next variable



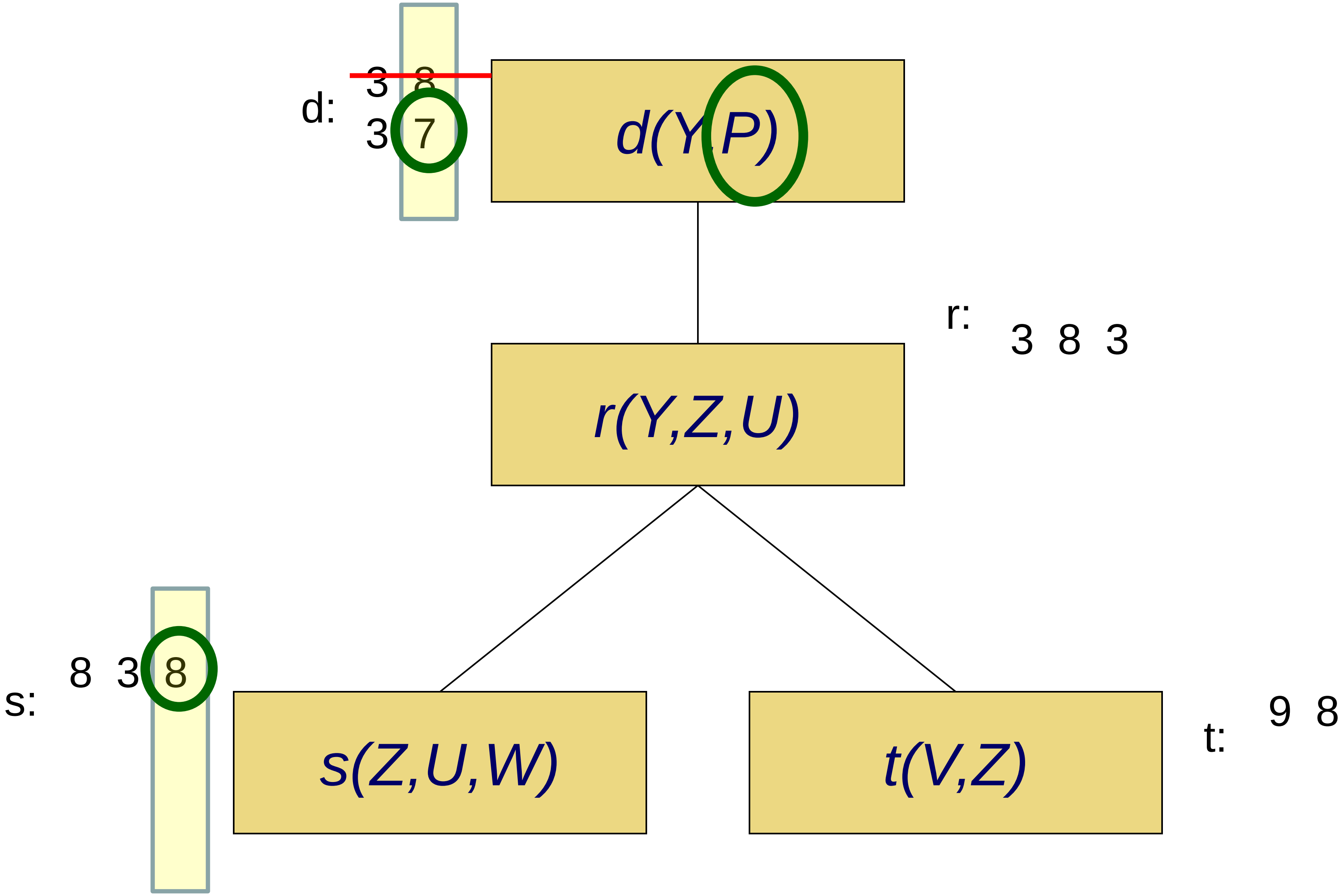
Fix a value, and propagate



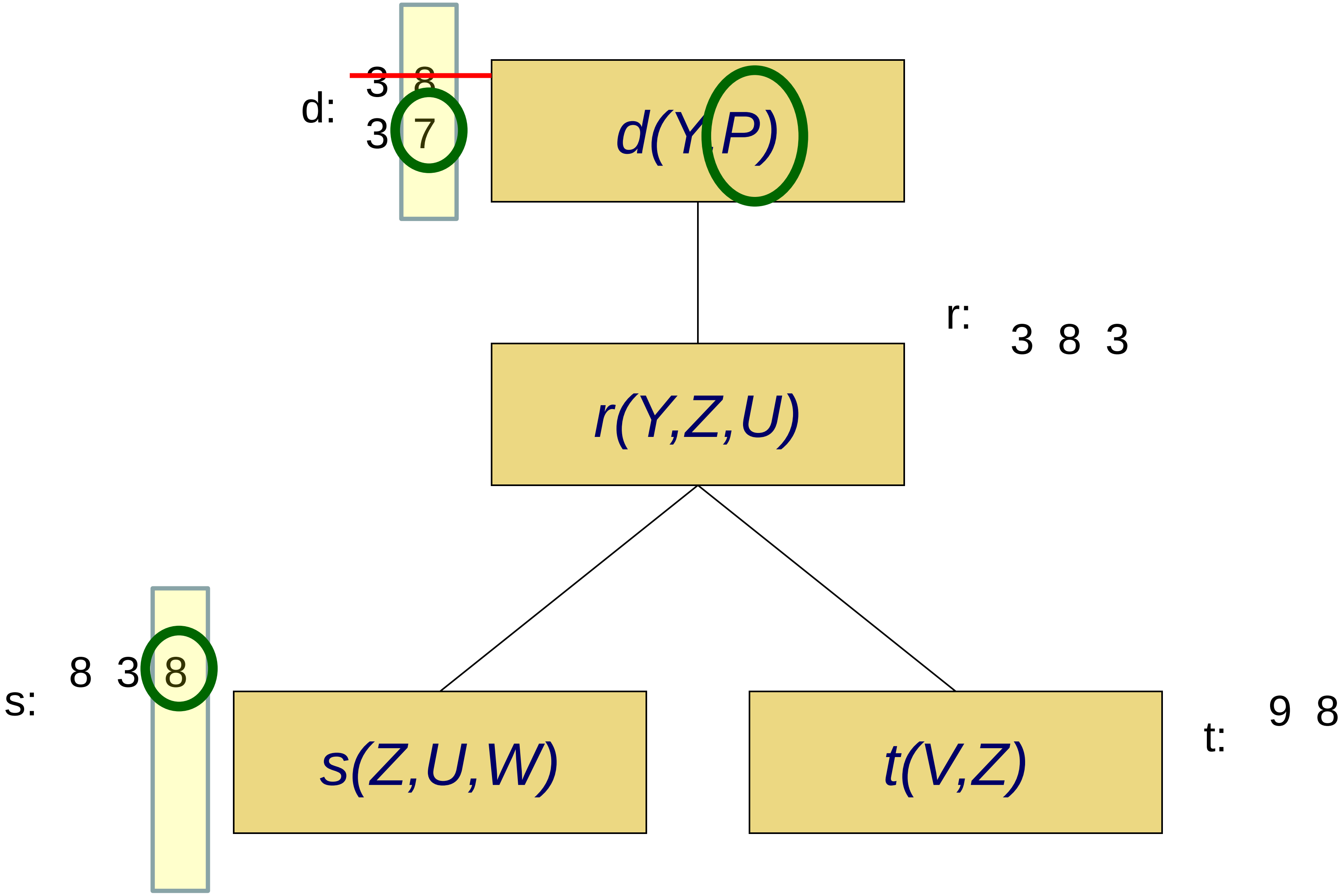
Fix a value, and propagate



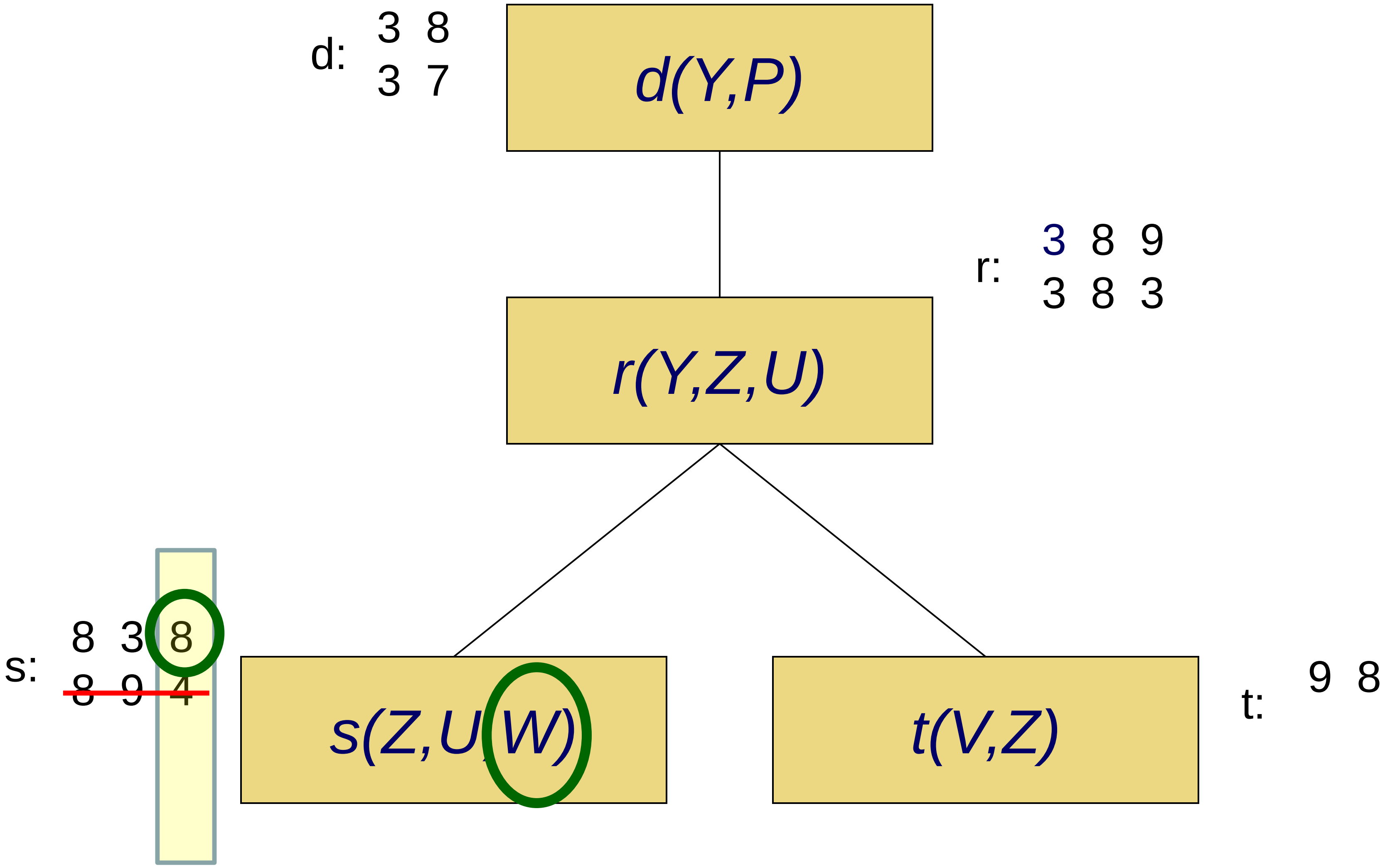
Iterate with a different value



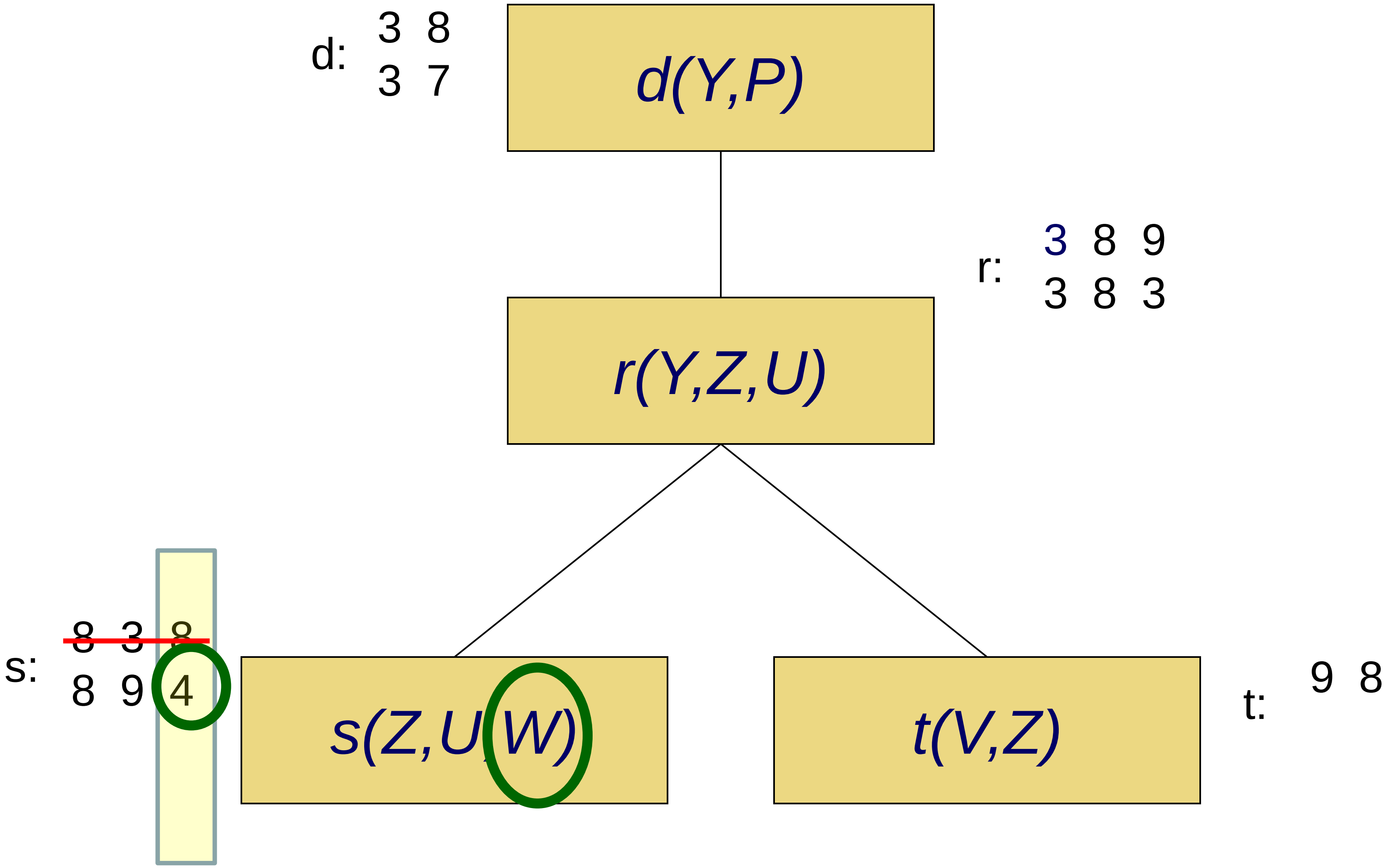
Iterate with a different value



Explore the next value on the previous variable



Explore the next value on the previous variable



Explore the next value on the previous variable

Putting It All Toghether

- Bottom-Up + Top-Down propagation
- Fix X_1 to the next value and *propagate*
  - Fix X_2 to the next value and *propagate*

    …
    - Fix X_n to the next value and *propagate*

Putting It All Toghether

- Bottom-Up + Top-Down propagation
- Fix X_1 to the next value and *propagate*
  - Fix X_2 to the next value and *propagate*

    …
    - Fix X_n to the next value and *propagate*

⬇

Output the given solution

Putting It All Toghether

- Bottom-Up + Top-Down propagation
- Fix X_1 to the next value and *propagate*
  - Fix X_2 to the next value and *propagate*
    - …
    - Fix X_n to the next value and *propagate*

Putting It All Toghether

- Bottom-Up + Top-Down propagation
- Fix X_1 to the next value and *propagate*
  - Fix X_2 to the next value and *propagate*

    …
    - Fix X_n to the next value and *propagate*

After the propagation phase,

every remaining tuple participates in at least one solution

[Yannakakis, VLDB'81]

Putting It All Toghether

- Bottom-Up + Top-Down propagation
- Fix X_1 to the next value and *propagate*
  - Fix X_2 to the next value and *propagate*
    - …
    - Fix X_n to the next value and *propagate*

Backtracking with no wrong choices

↓

Enumeration WPD

Outline

- Identification of “Easy” Classes
- Applications of Tree Decompositions
- Beyond Tree Decompositions
- Decision/Computation Problems
- Optimization Problems
- Enumeration Problems

Appendix: The Fronteer of Tractability

The Core

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

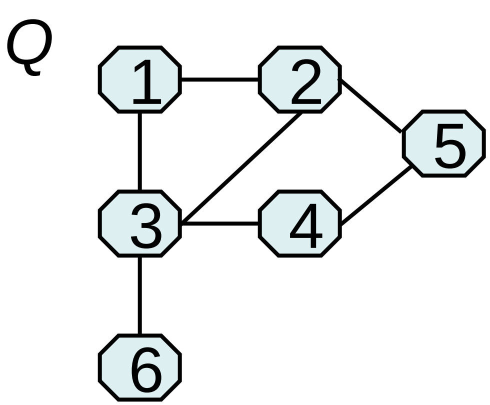
1. $atoms(Q') \subseteq atoms(Q)$
2. There is a mapping $h: var(Q) \rightarrow var(Q')$ s.t., $\forall\, r(\boldsymbol{X}) \in atoms(Q),\;\; r(h(\boldsymbol{X})) \in atoms(Q')$
3. There is no query Q'' satisfying 1 and 2 and such that atoms(Q'') $\subset$ atoms(Q')

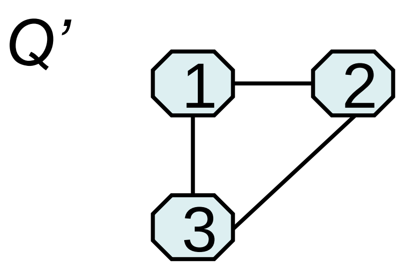
The Core

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:



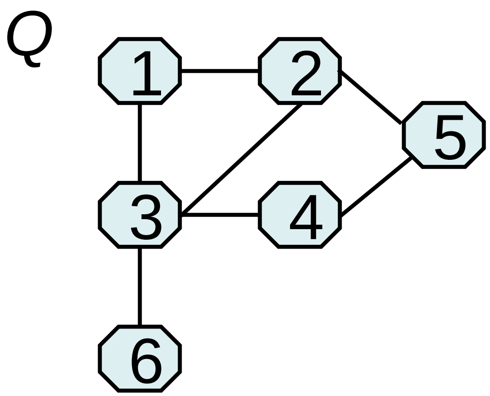


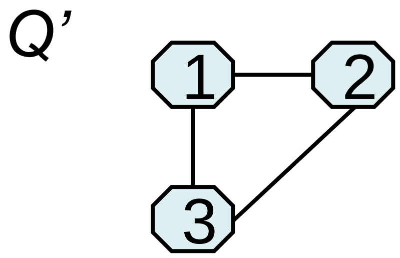
The Core

Cores are isomorphic → The "Core"

Cores are equivalent to the query

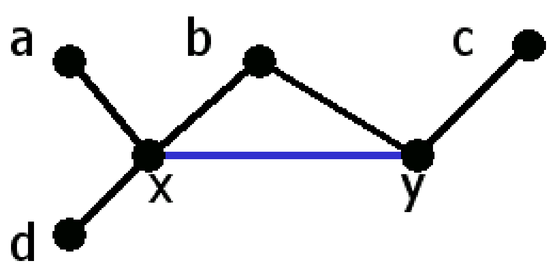
Example:

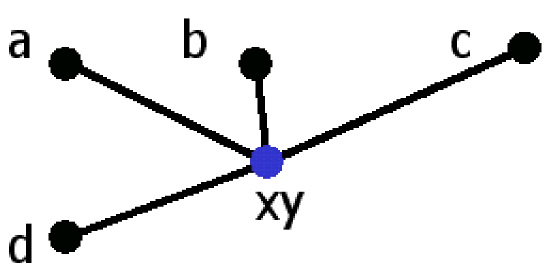




Graph Minors

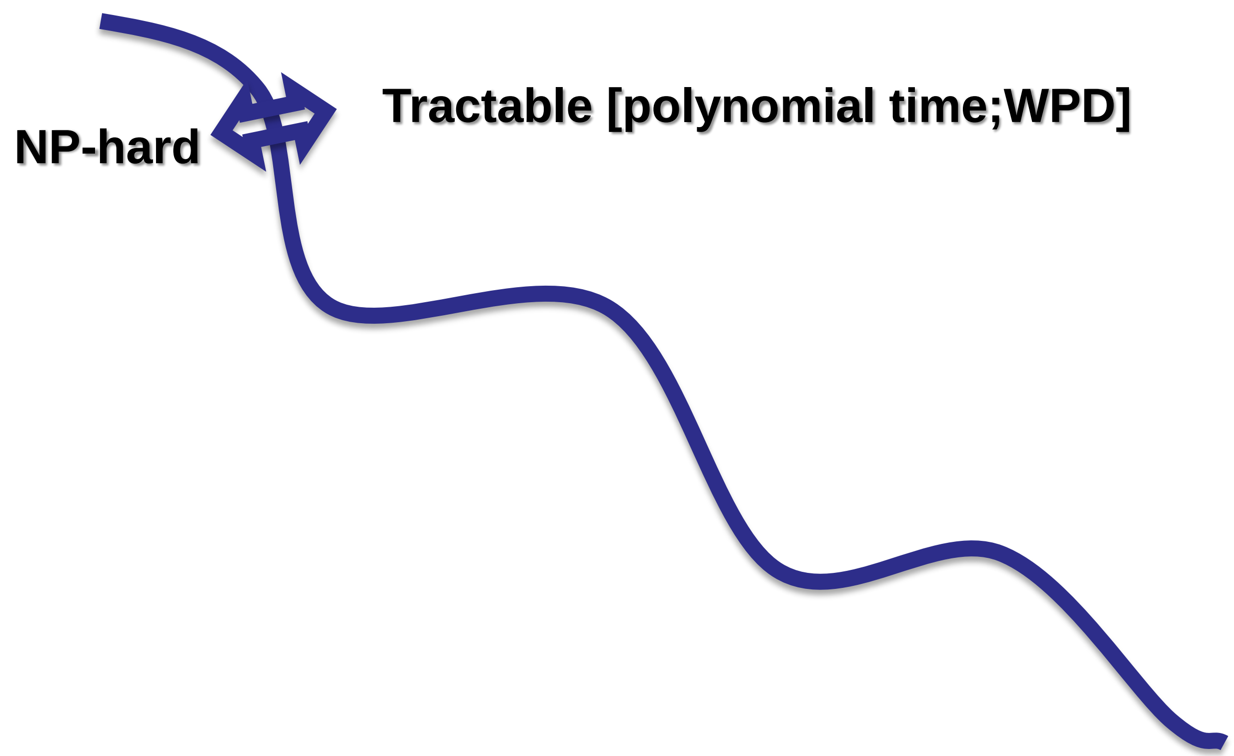
- H is a minor of G if it can be obtained by repeatedly applying:
  - Edge deletion
  - Vertex deletion
  - Edge contraction





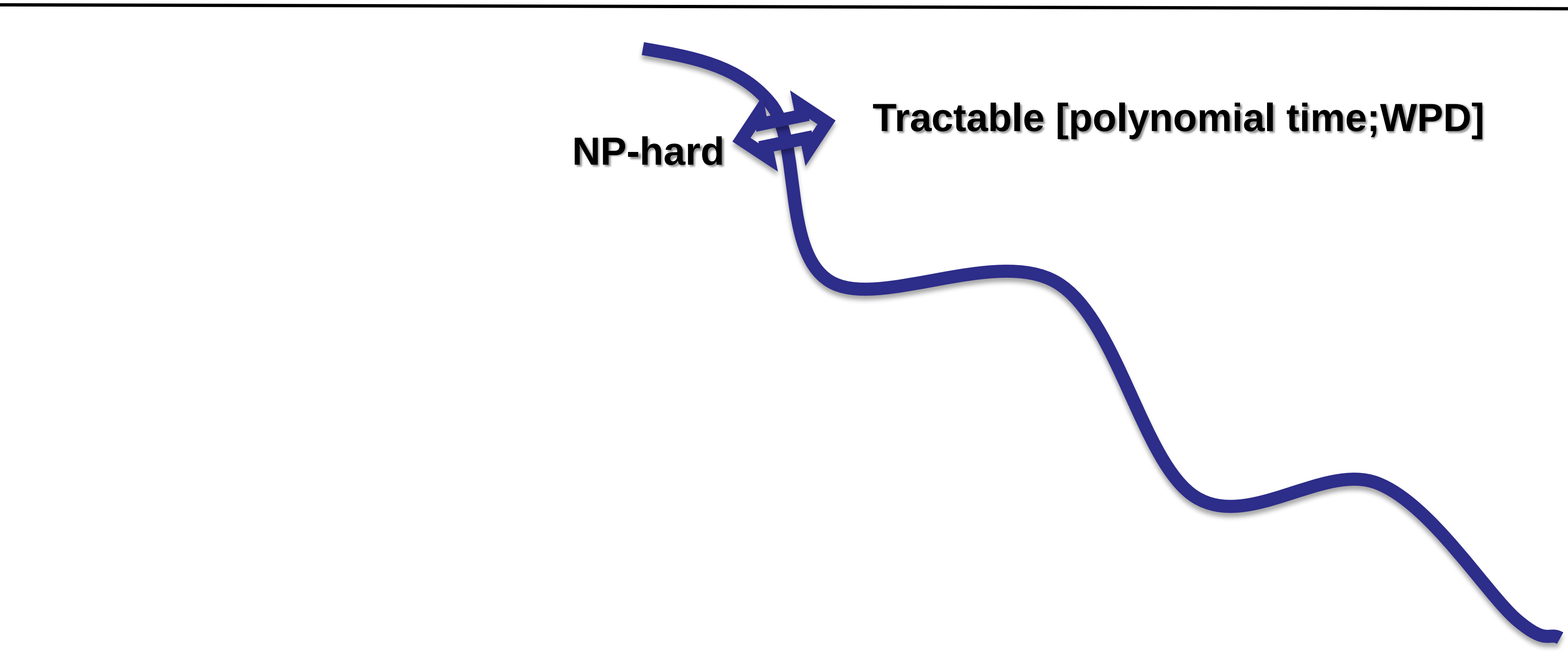
The Fronteer of Tractability

- Let **A** be a class of structures:



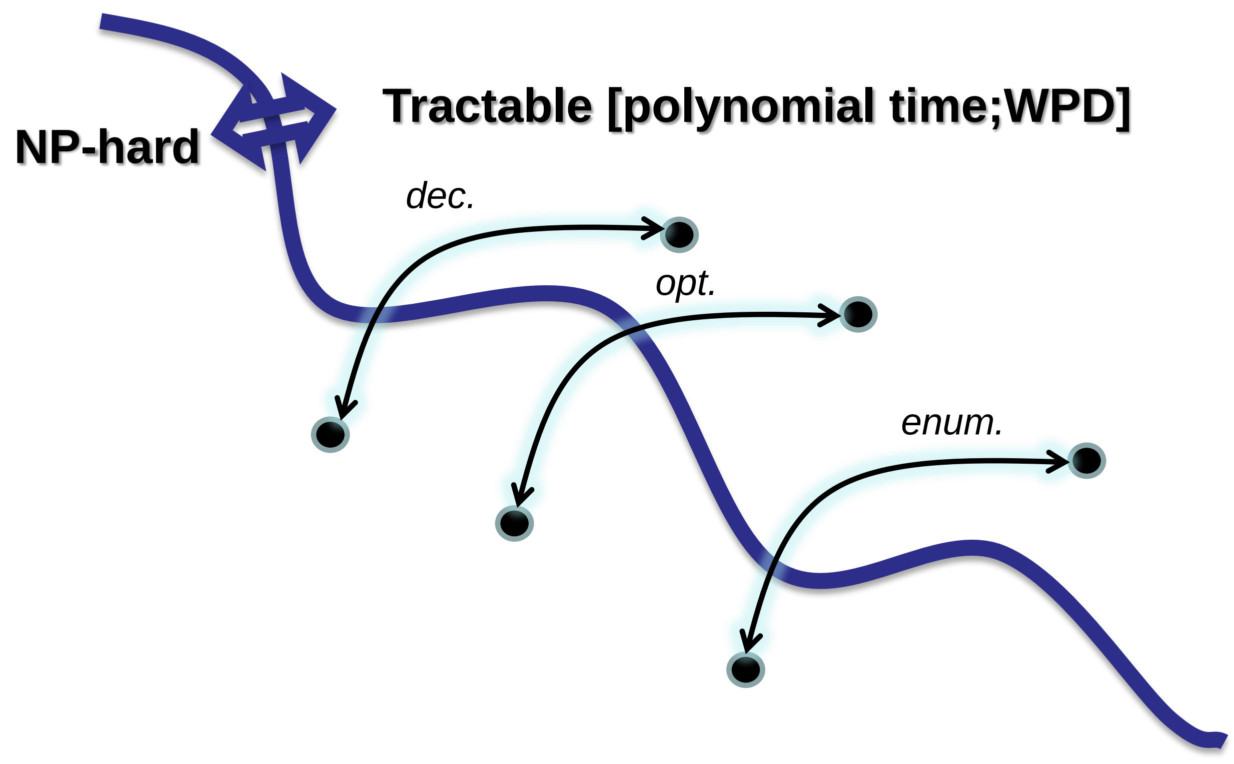
The Fronteer of Tractability

- Let **A** be a class of structures:
  - Assume FPT≠ WP[1]



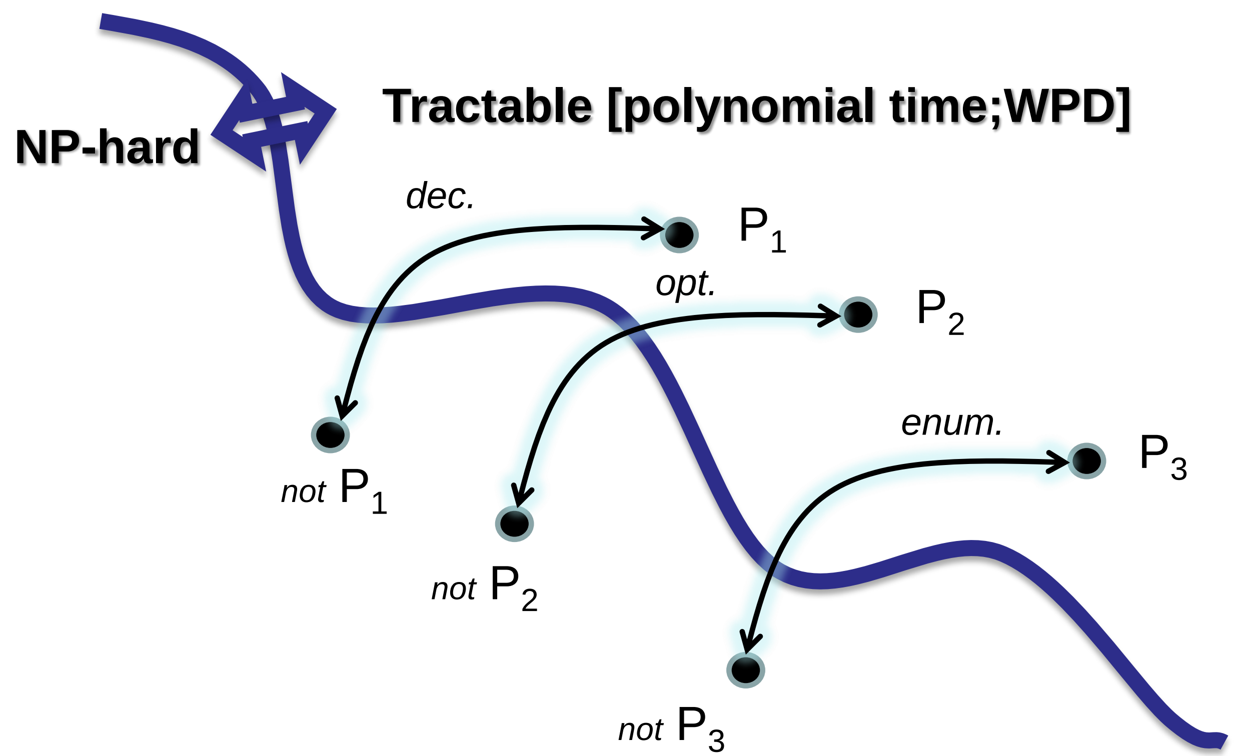
The Fronteer of Tractability

- Let **A** be a class of structures:
  - Assume FPT≠ WP[1]



The Fronteer of Tractability

- Let **A** be a class of structures:
  - Assume FPT≠ WP[1]

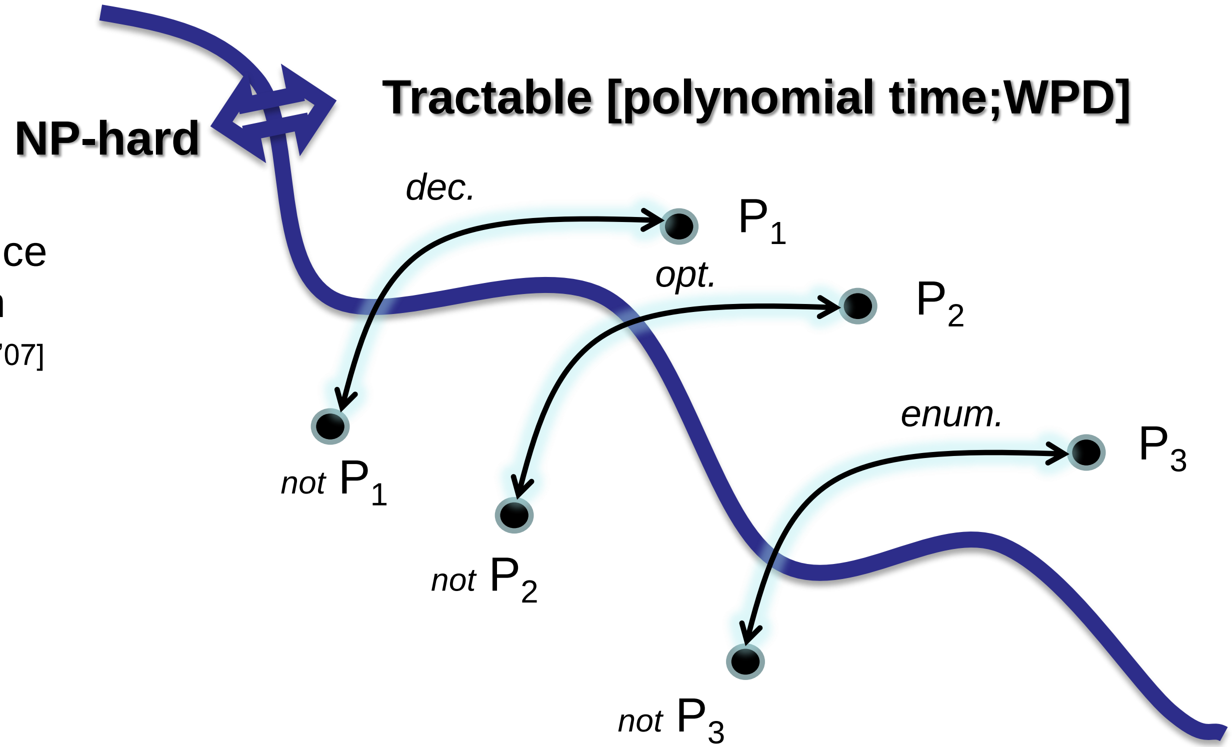


The Fronteer of Tractability

- Let **A** be a class of structures:
  - Assume FPT≠ WP[1]

- **P_1**: The core of each instance in **A** has bounded treewidth

[Grohe, '07]



The Fronteer of Tractability

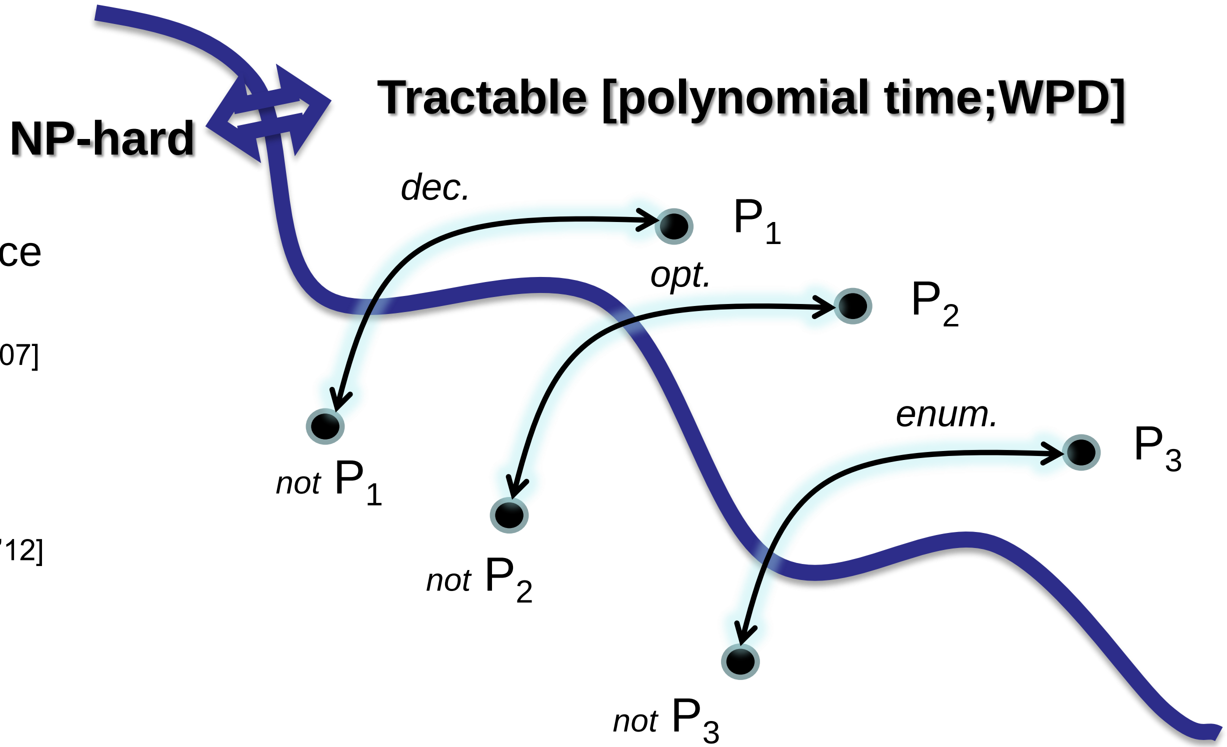
- Let **A** be a class of structures:
  - Assume FPT≠ WP[1]

- **P_1**: The core of each instance in **A** has bounded treewidth

  [Grohe, '07]

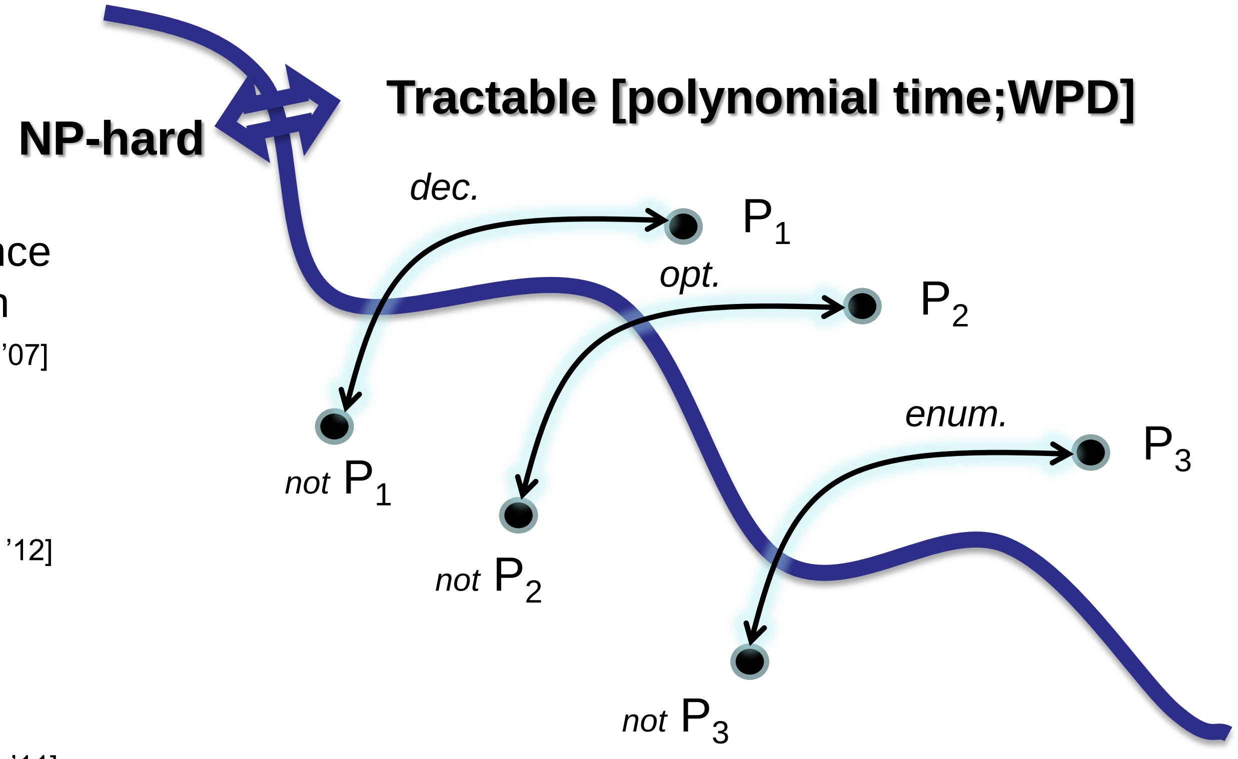
- **P_2**: Each instance in **A** has bounded treewidth

  [Greco & Scarcello, '12]



The Fronteer of Tractability

- Let **A** be a class of structures:
  - Assume FPT≠ WP[1]
- Assume **A** is *closed under taking minors*



- $\mathbf{P_1}$: The core of each instance in **A** has bounded treewidth

  [Grohe, '07]

- $\mathbf{P_2}$: Each instance in **A** has bounded treewidth

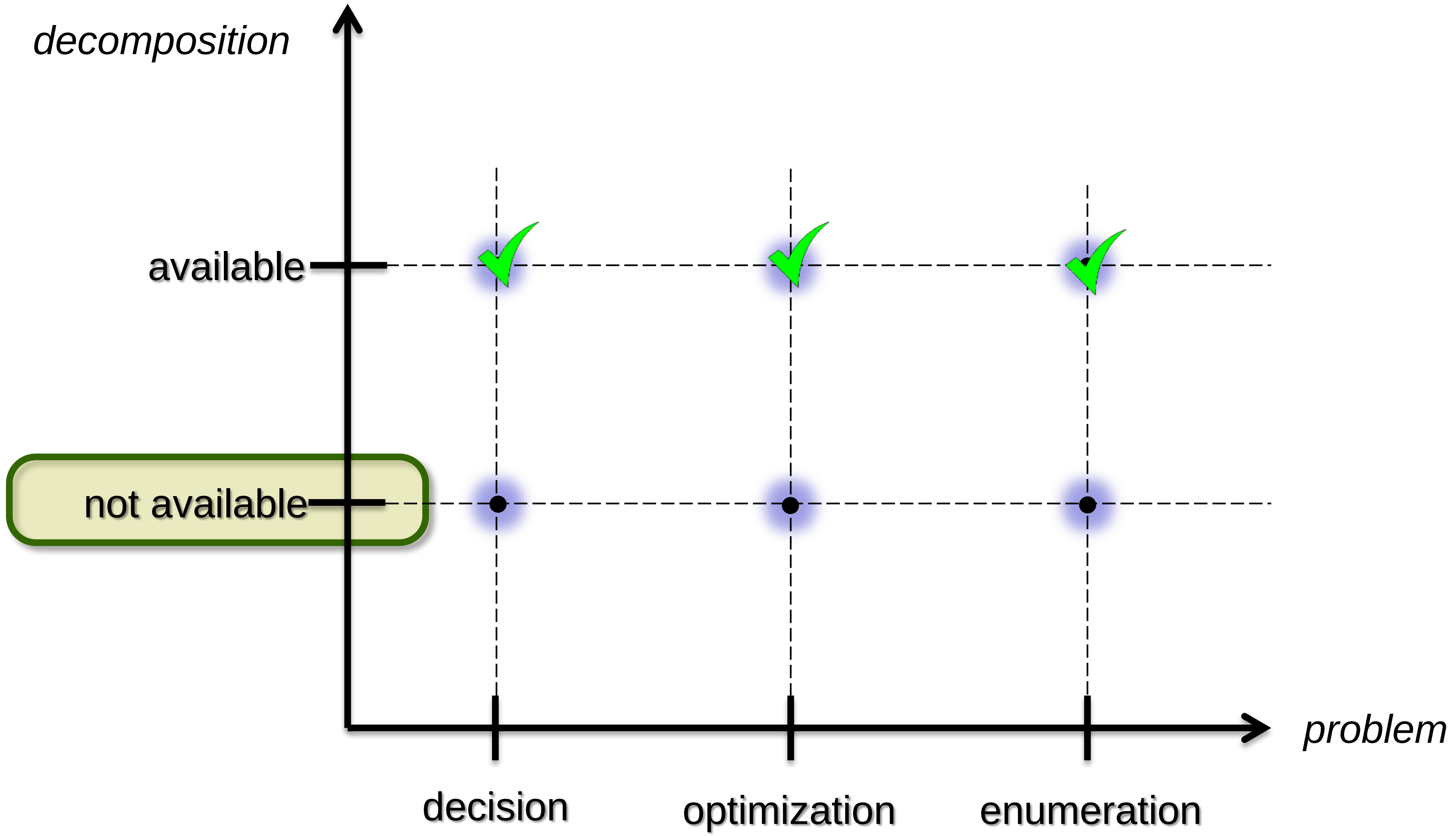
  [Greco & Scarcello, '12]

- $\mathbf{P_2}$: Each instance in **A** has bounded treewidth

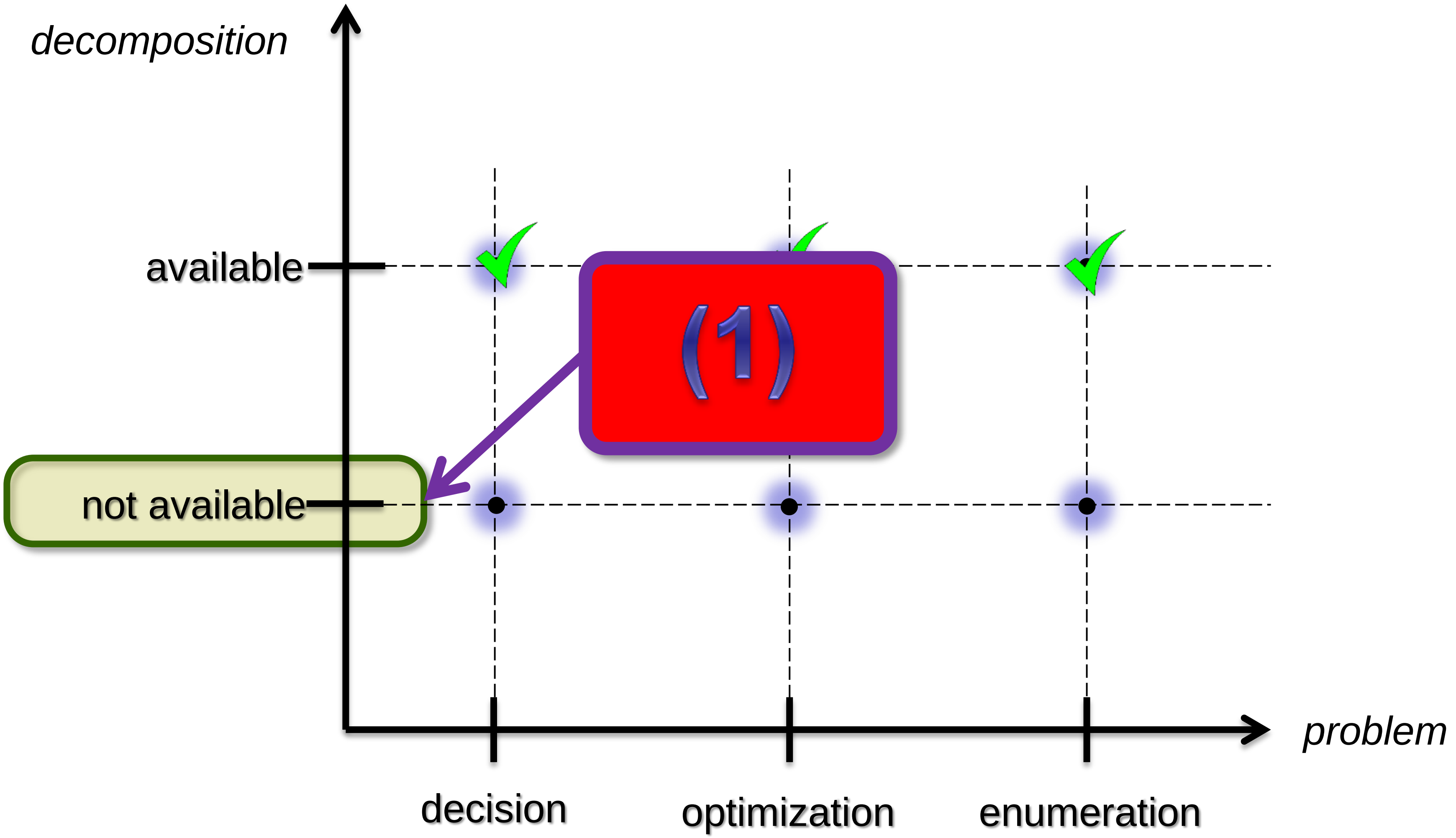
  [Greco & Scarcello, '11]

Appendix: Methods without Decompositions

Overview

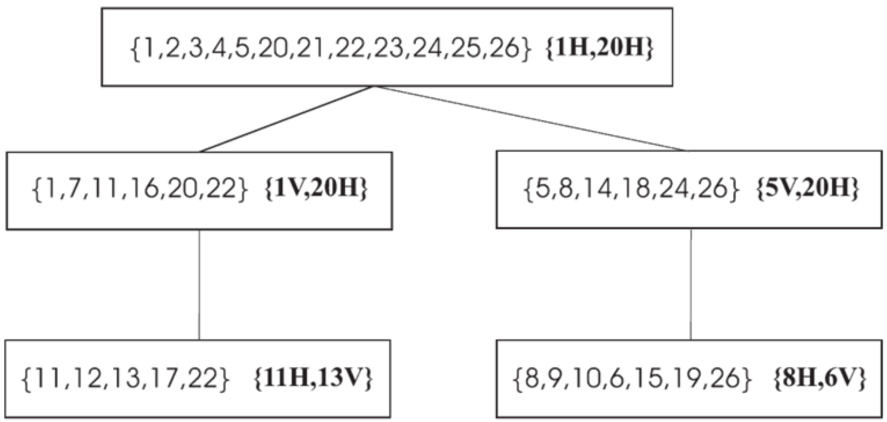

decomposition
available
not available
problem
decision
optimization
enumeration

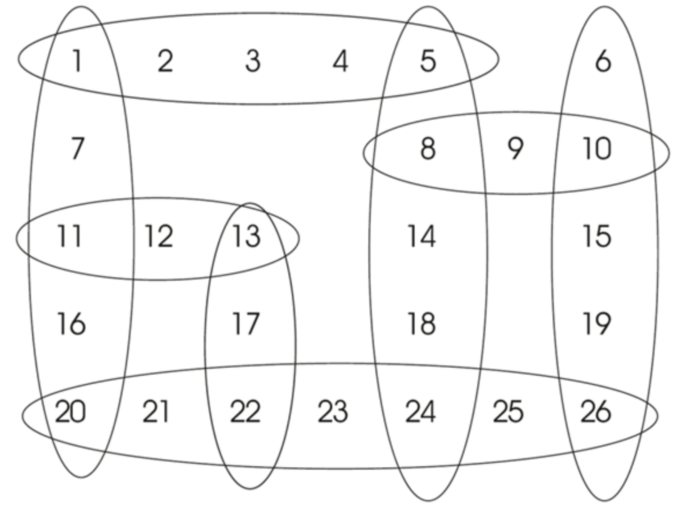
Overview



decomposition
available
not available
(1)
problem
decision
optimization
enumeration

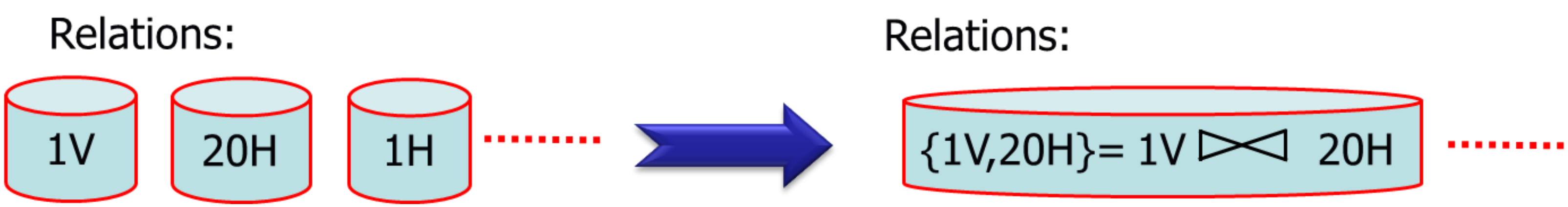
Revisiting Decomposition Methods



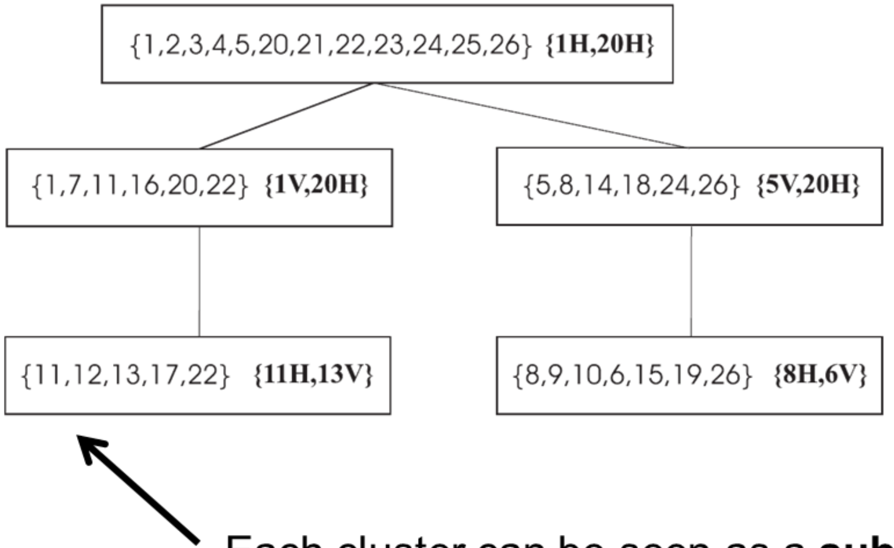


Each cluster can be seen as a **subproblem**

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

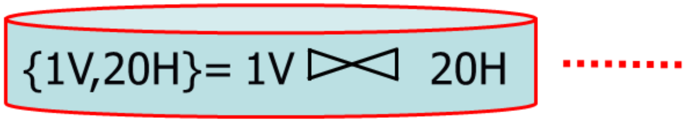


Revisiting Decomposition Methods

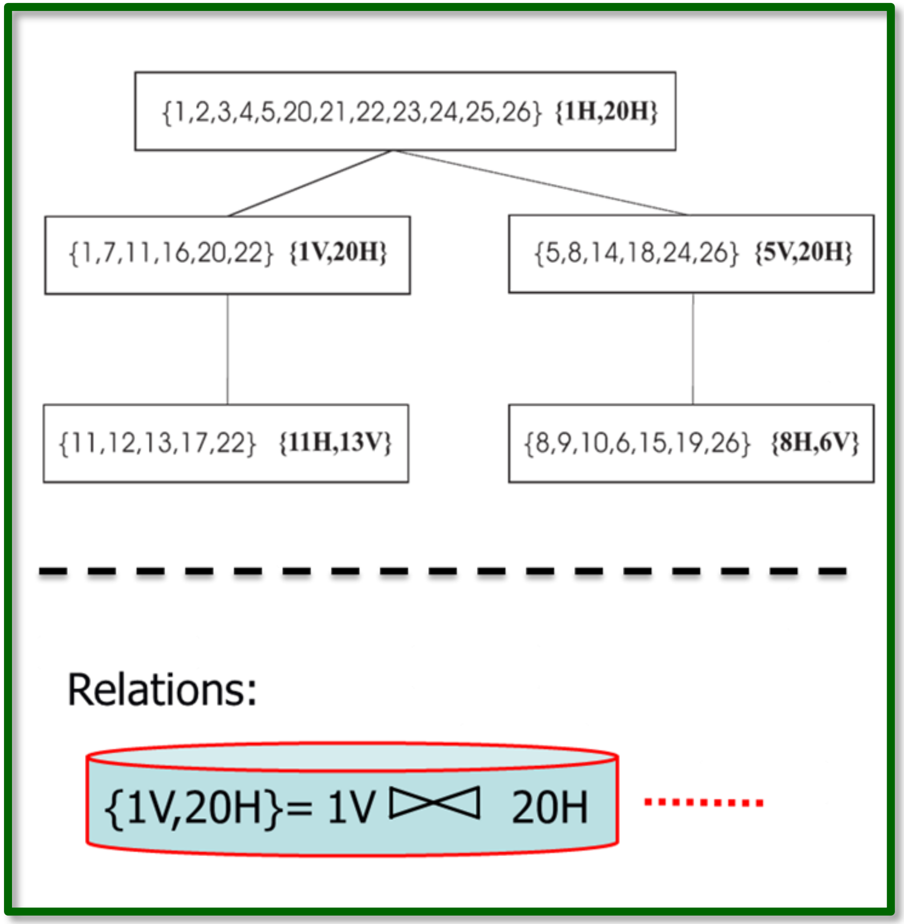


Each cluster can be seen as a **subproblem**

Relations:



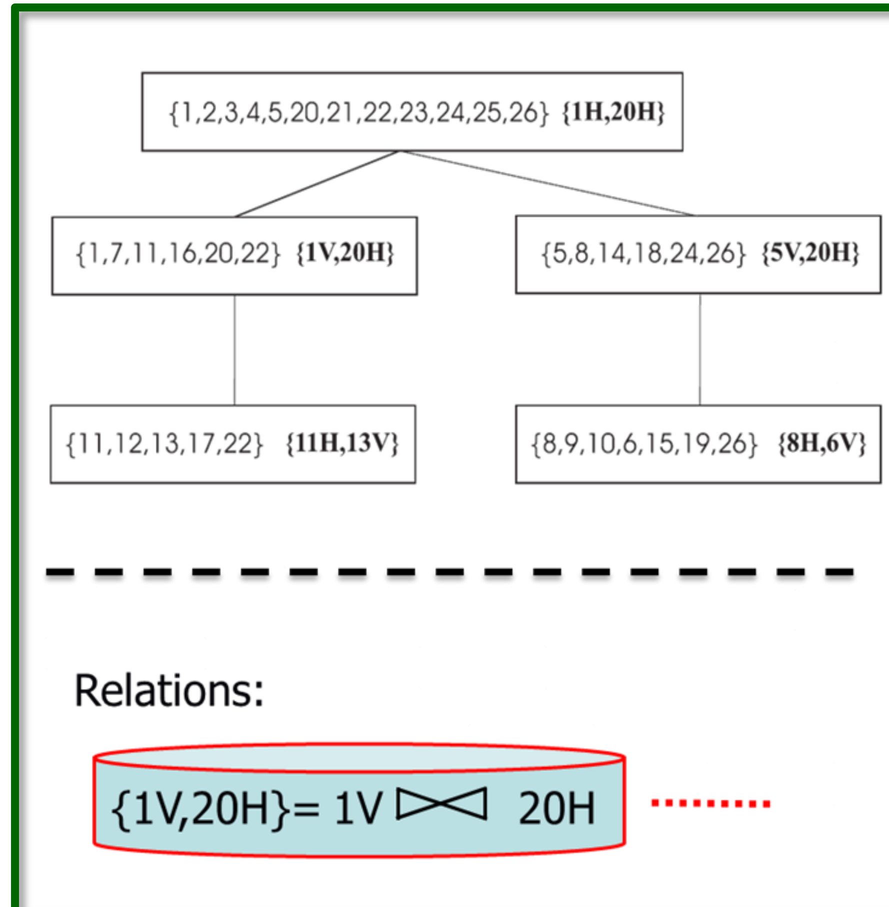
Revisiting Decomposition Methods



Revisiting Decomposition Methods

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

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



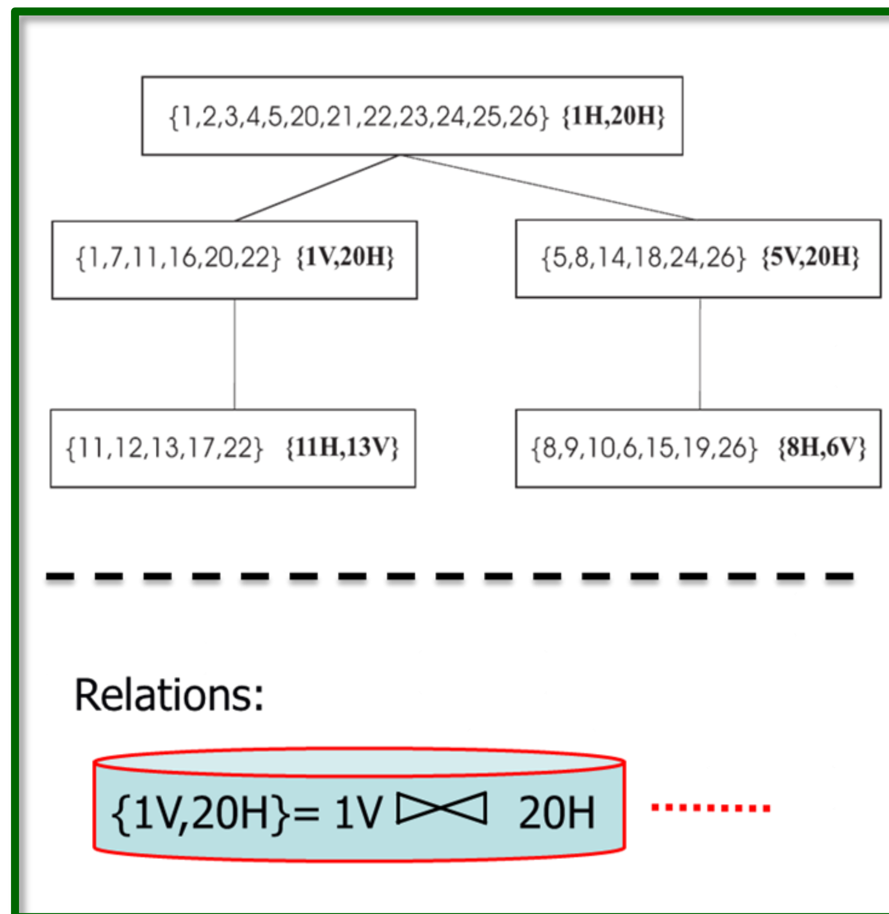
Revisiting Decomposition Methods

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

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

Scopes — Solutions

Work on subproblems



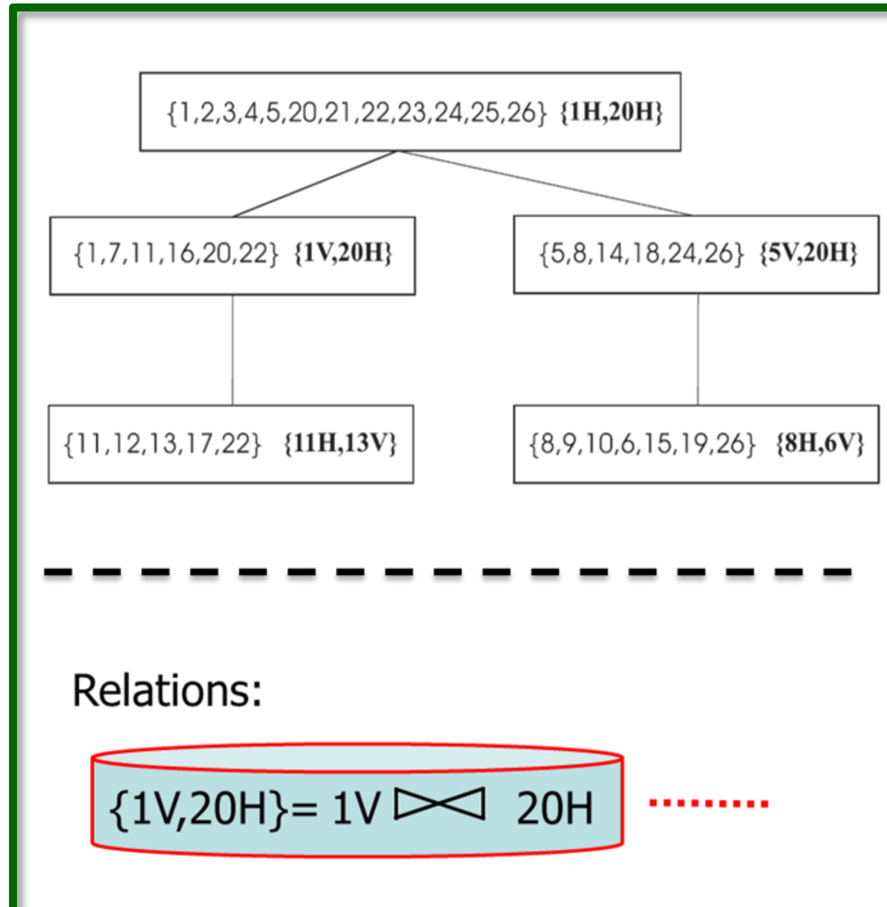
Revisiting Decomposition Methods

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

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

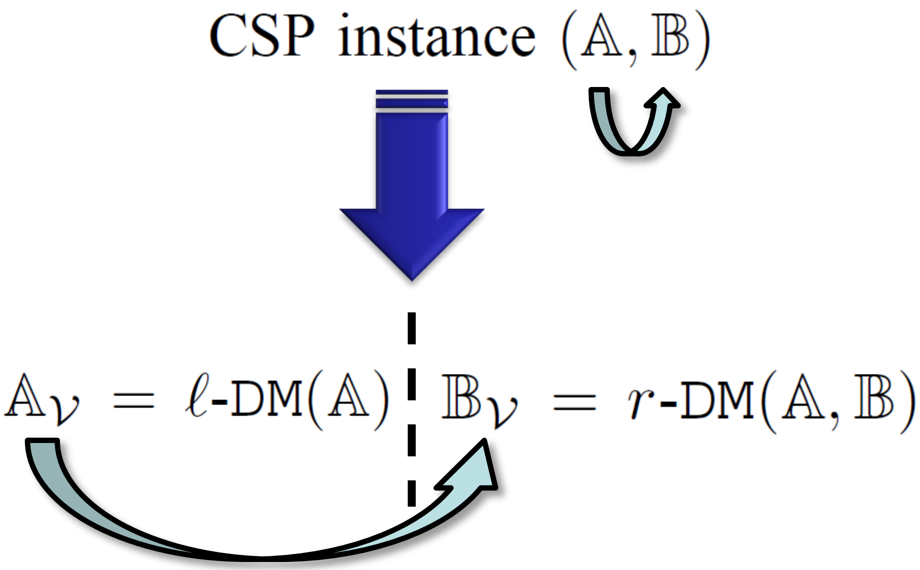
Scopes | Solutions

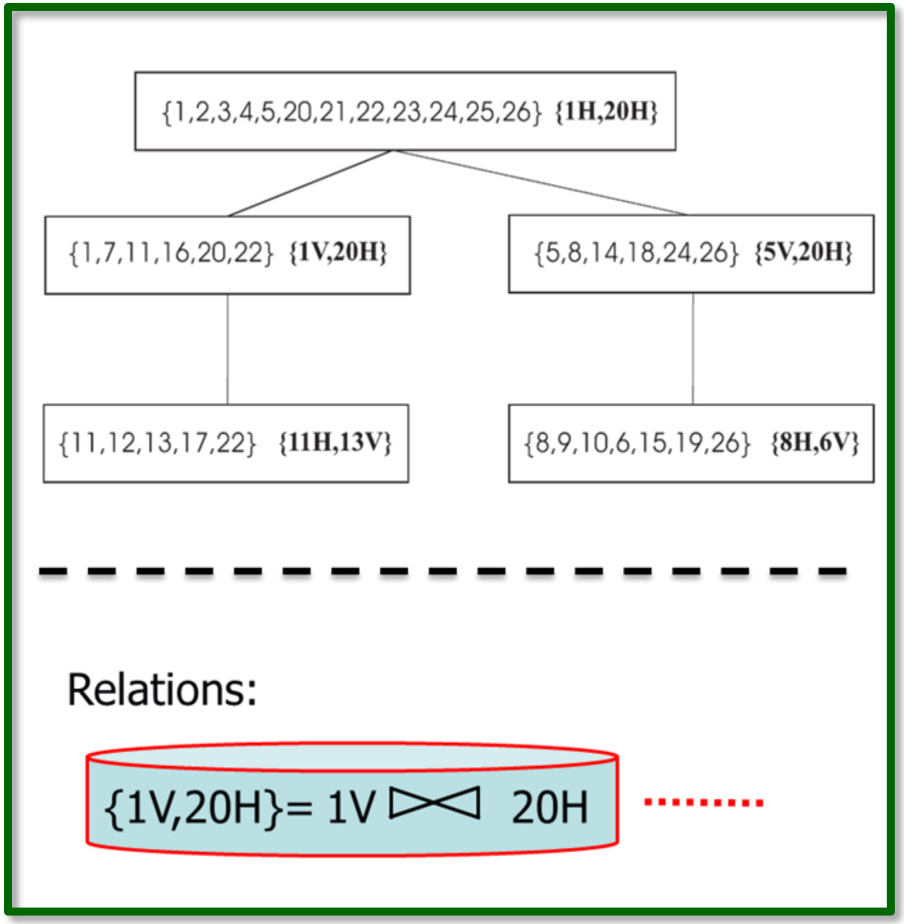
Work on subproblems



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

Revisiting Decomposition Methods

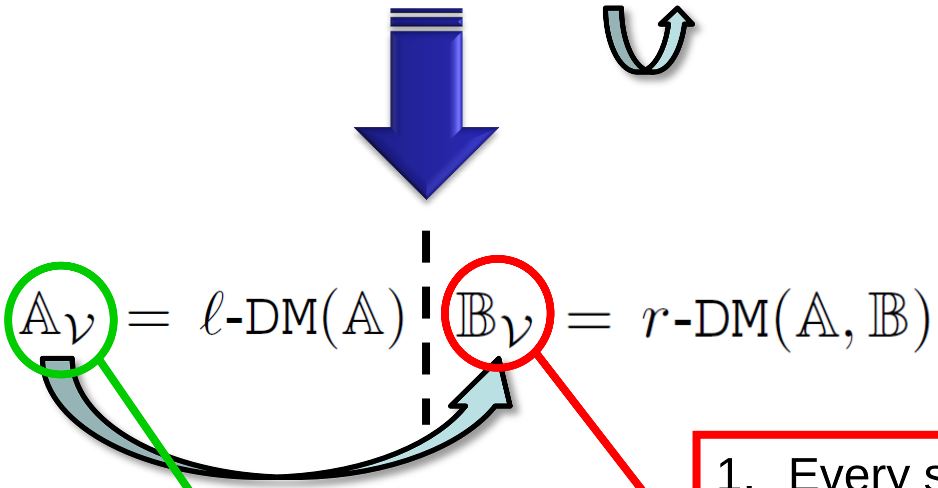




- *Generalized hypertree width*: take all views that can be computed by joining at most k atoms (k query views)

Requirements on Subproblem Definition

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



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

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

Working on subproblems is not necessarily beneficial…

Acyclicity in Decomposition Methods

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

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

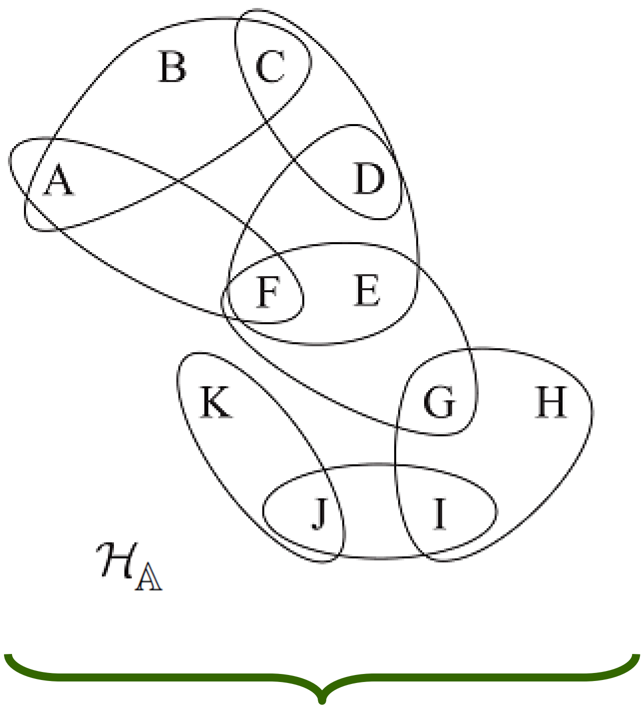
Working on subproblems is not necessarily beneficial…

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)

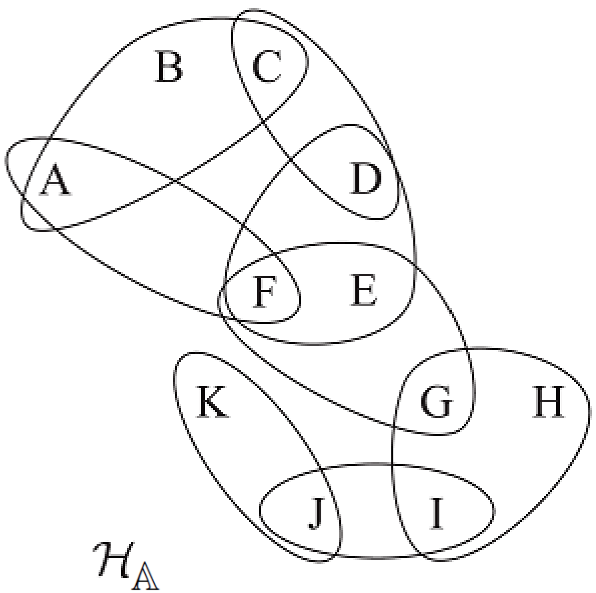
$\mathbb{A}$: $r_1(A,B,C)$ $r_2(A,F)$ $r_3(C,D)$ $r_4(D,E,F)$
$r_5(E,F,G)$ $r_6(G,H,I)$ $r_7(I,J)$ $r_8(J,K)$

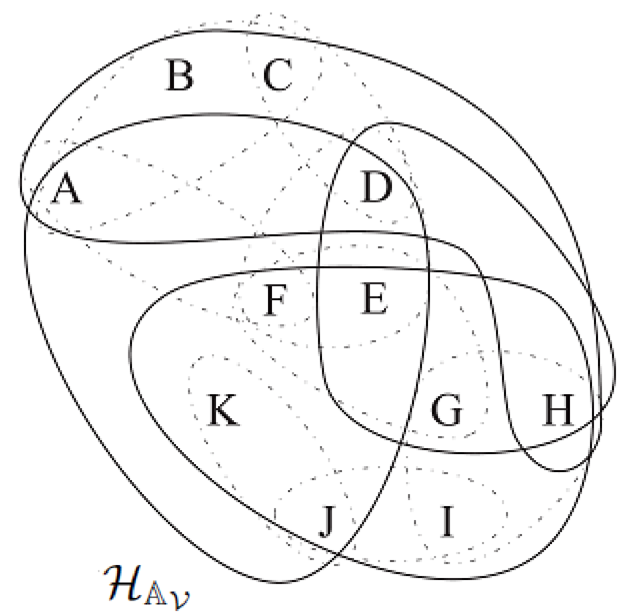


Structure of the CSP

Tree Projections (by Example)

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



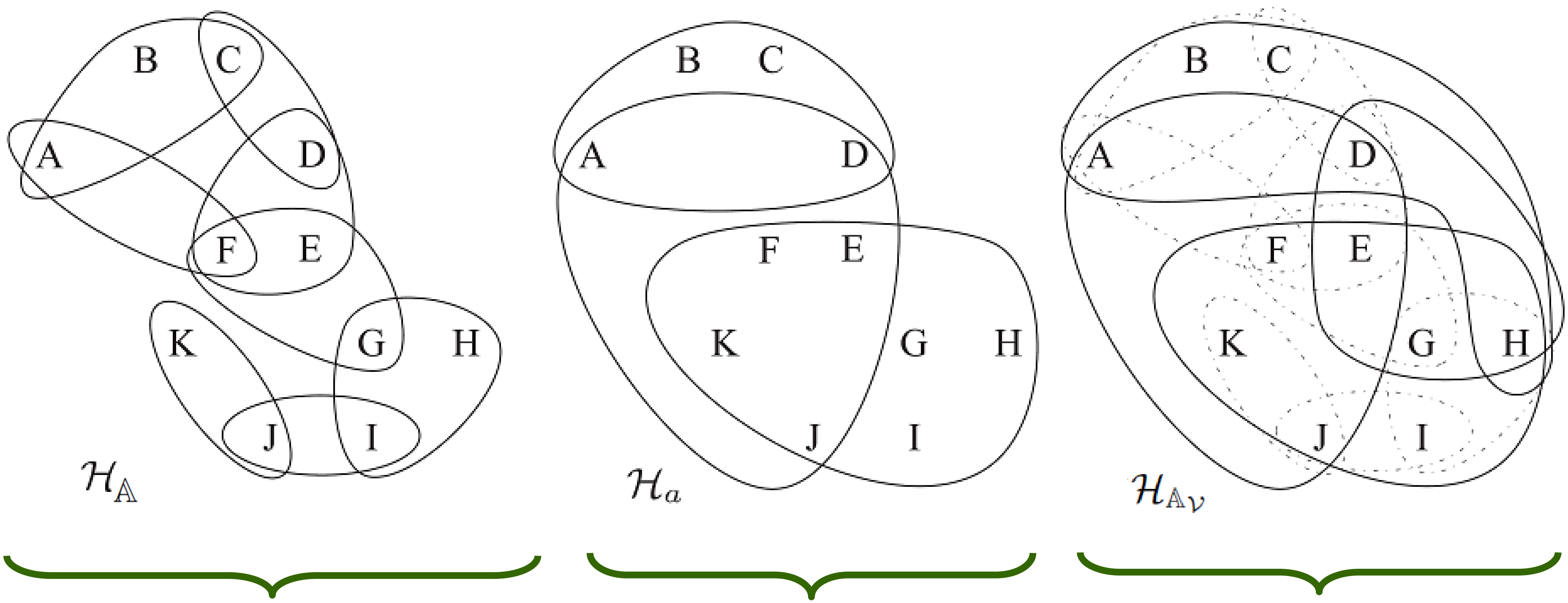


Structure of the CSP

Available Views

Tree Projections (by Example)

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



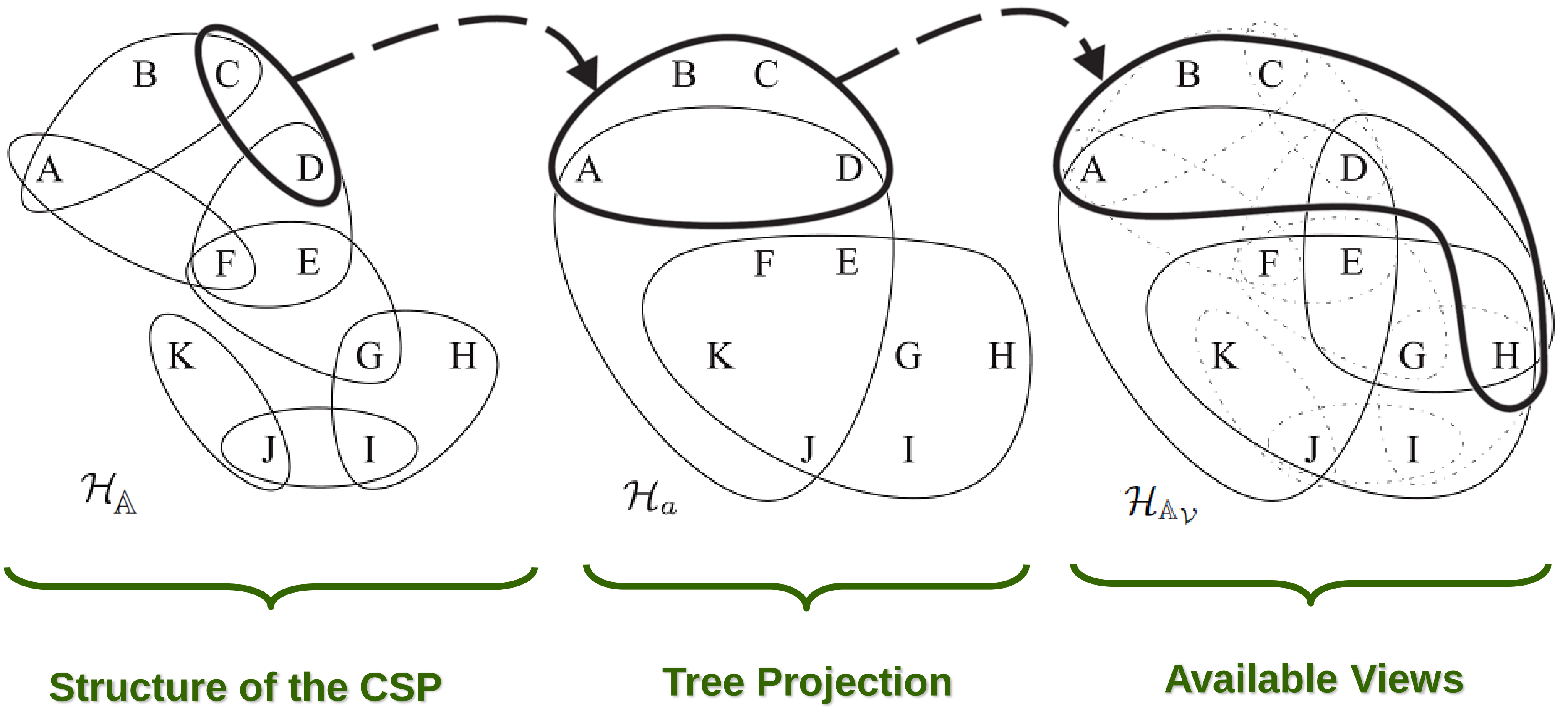
Structure of the CSP

Tree Projection

Available Views

Tree Projections (by Example)

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



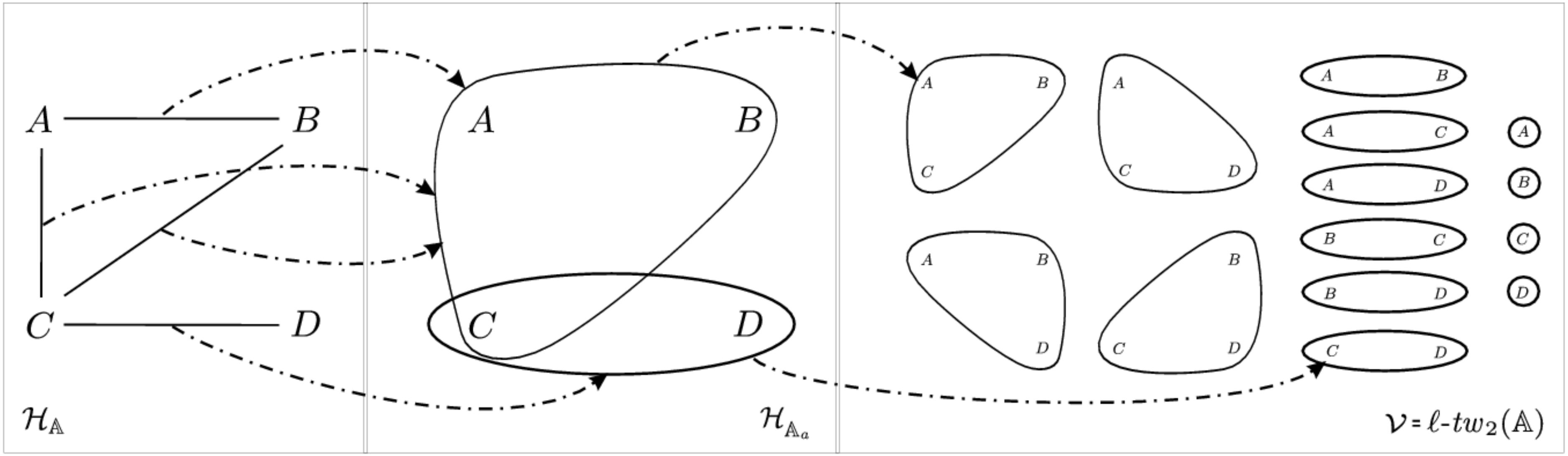
(Noticeable) Examples

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 General Framework, but

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

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

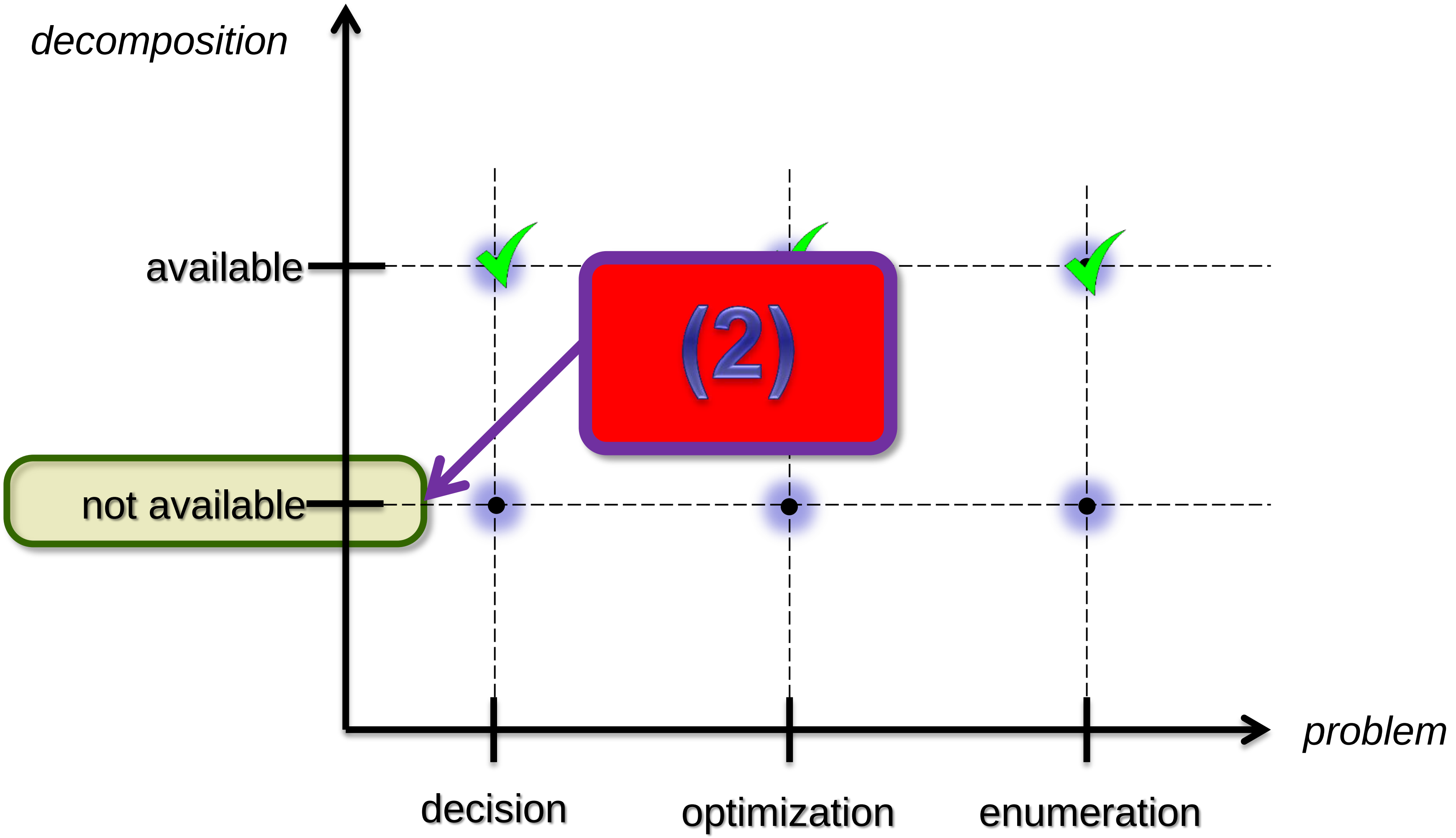
A General Framework, but

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

Hold on generalized hypertree width too.

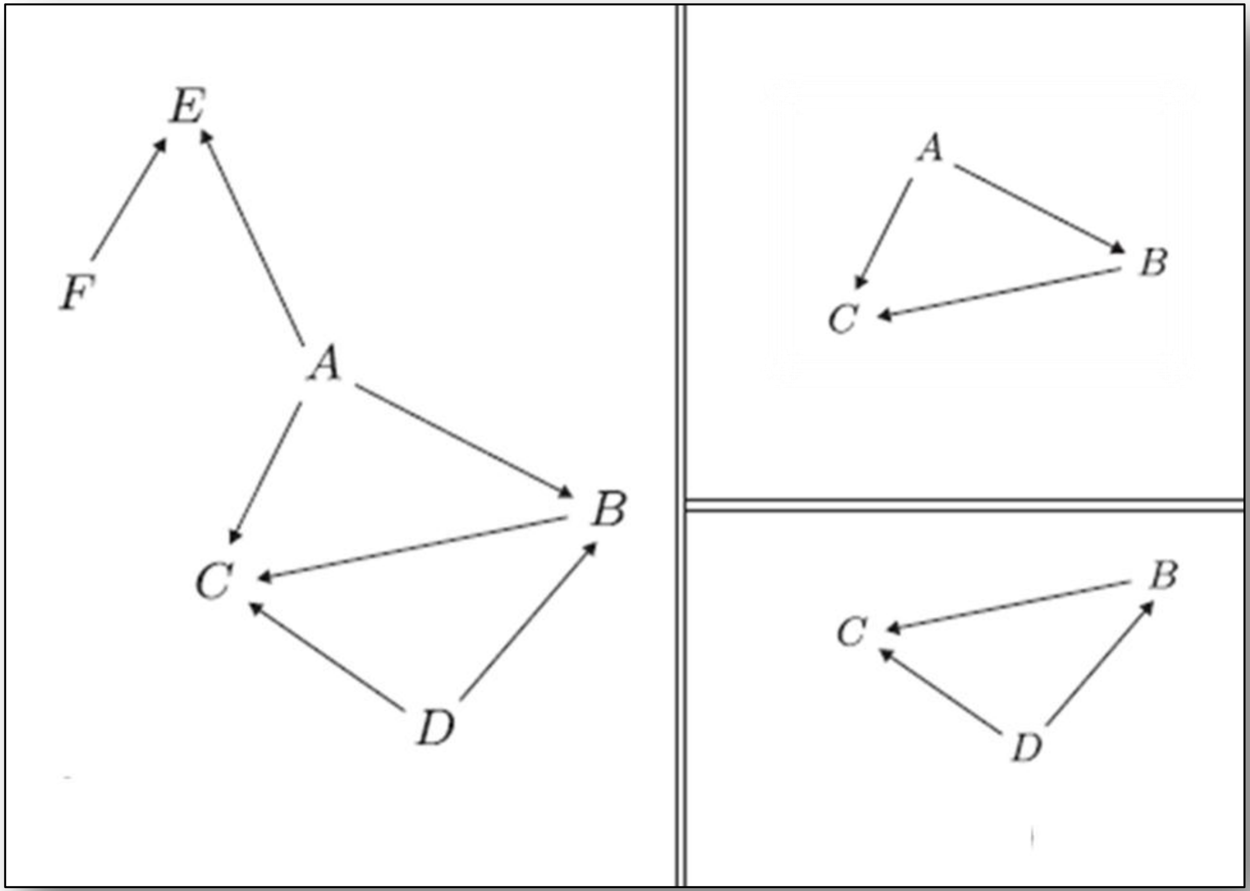
[Gottlob, Miklos, and Schwentick, JACM'09]

Overview


decomposition
available
not available
(2)
problem
decision
optimization
enumeration

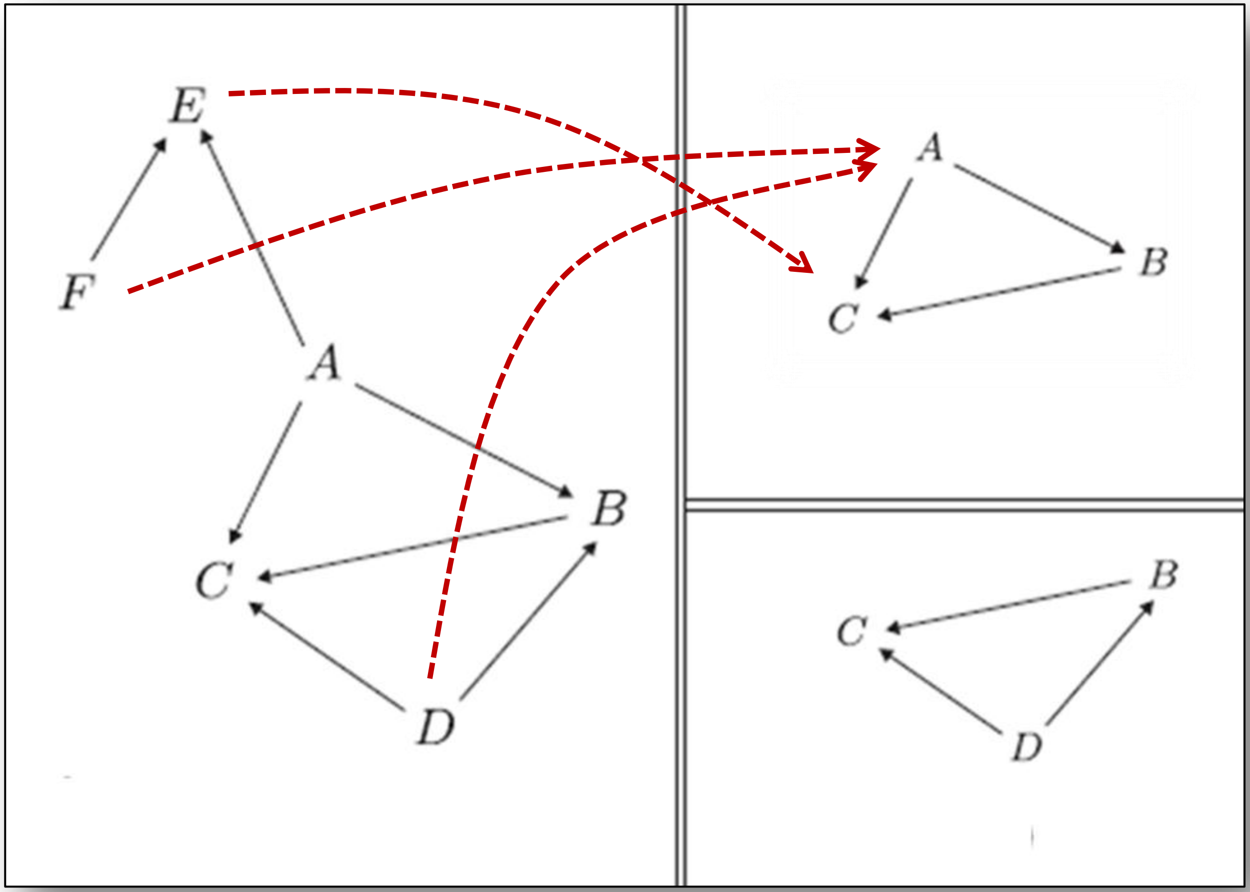
Cores and Tree Projections

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

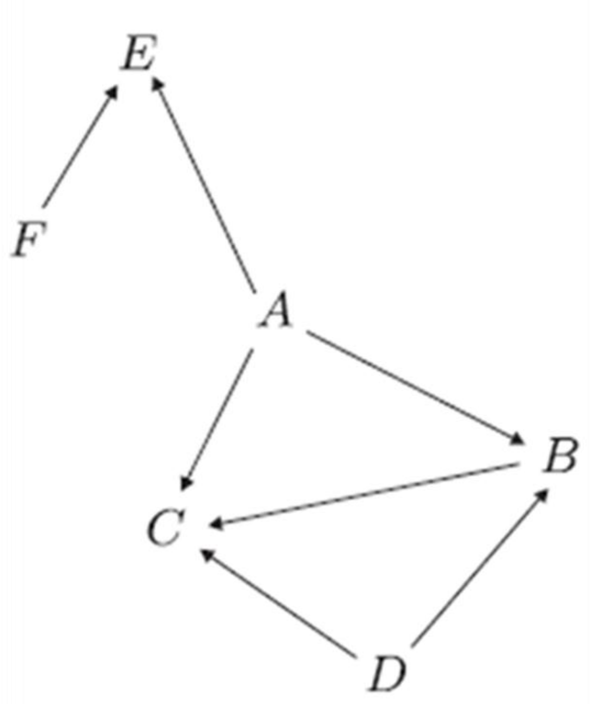


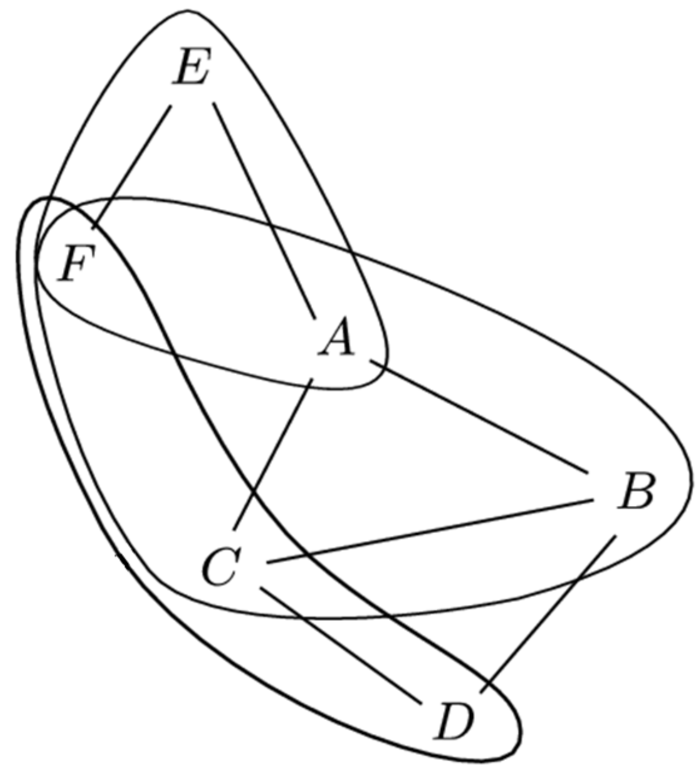
Cores and Tree Projections

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



Cores and Tree Projections



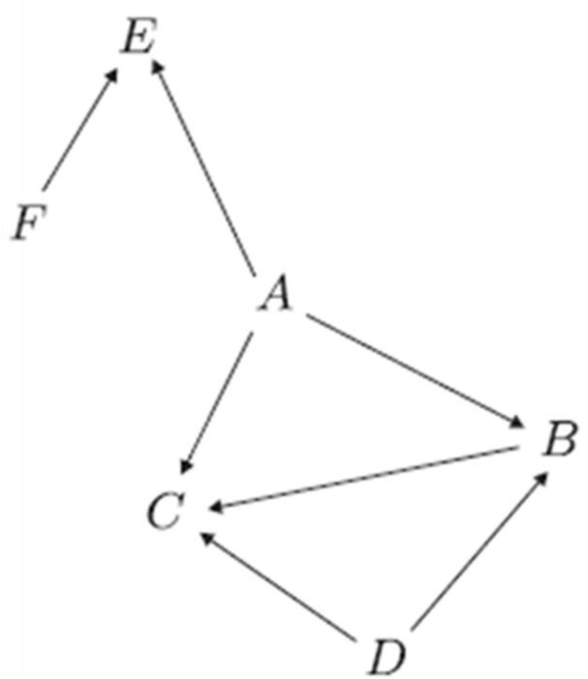


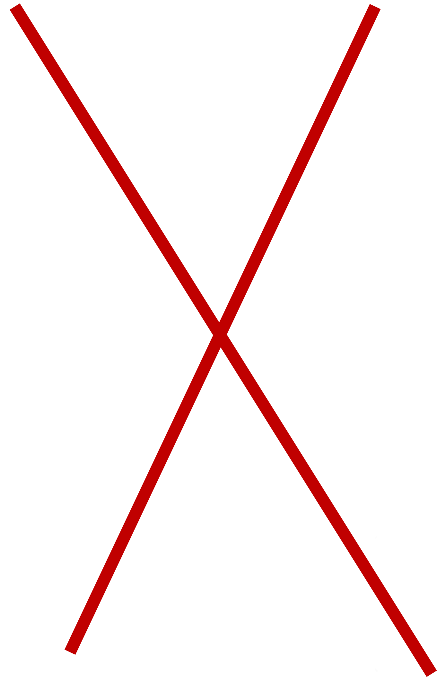
Structure of the CSP

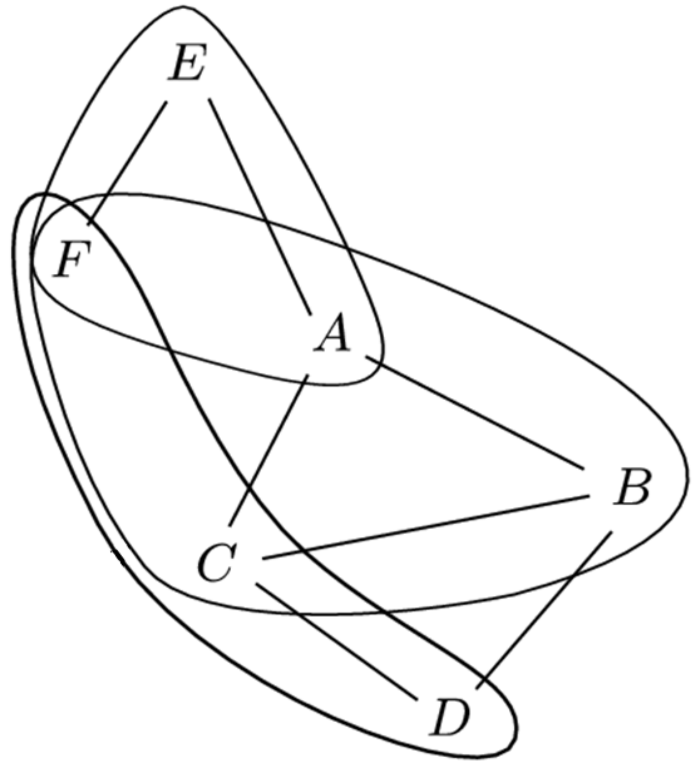
Tree Projection

Available Views

Cores and Tree Projections





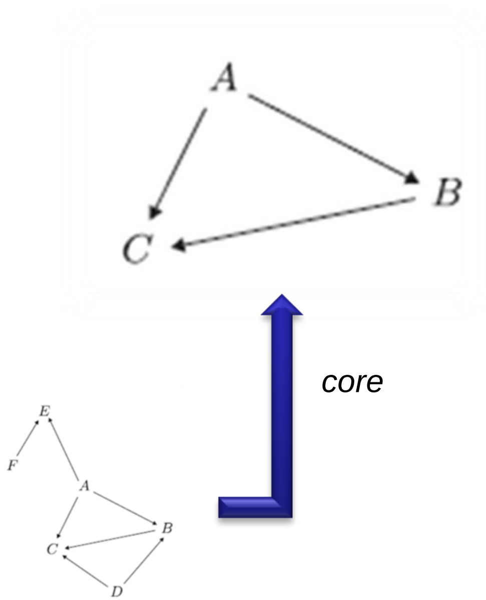


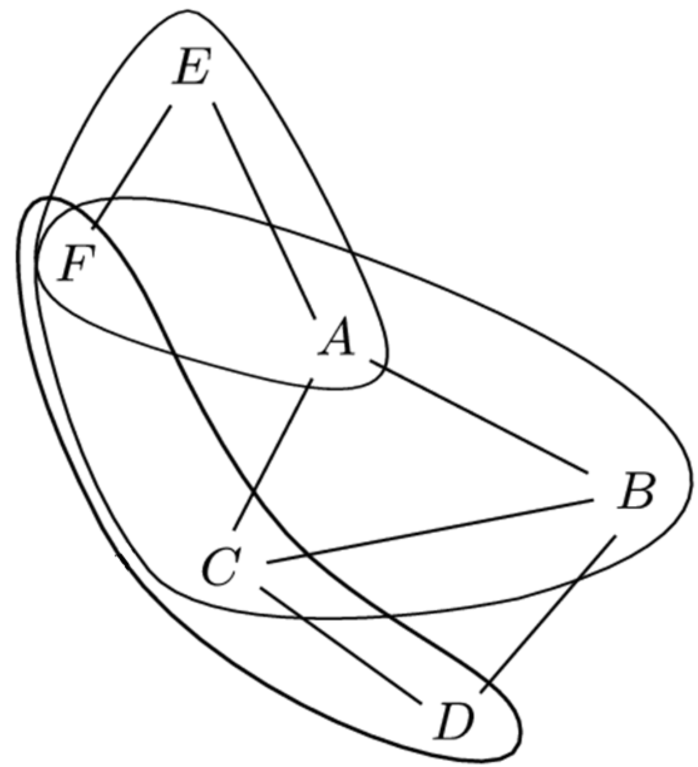
Structure of the CSP

Tree Projection

Available Views

Cores and Tree Projections



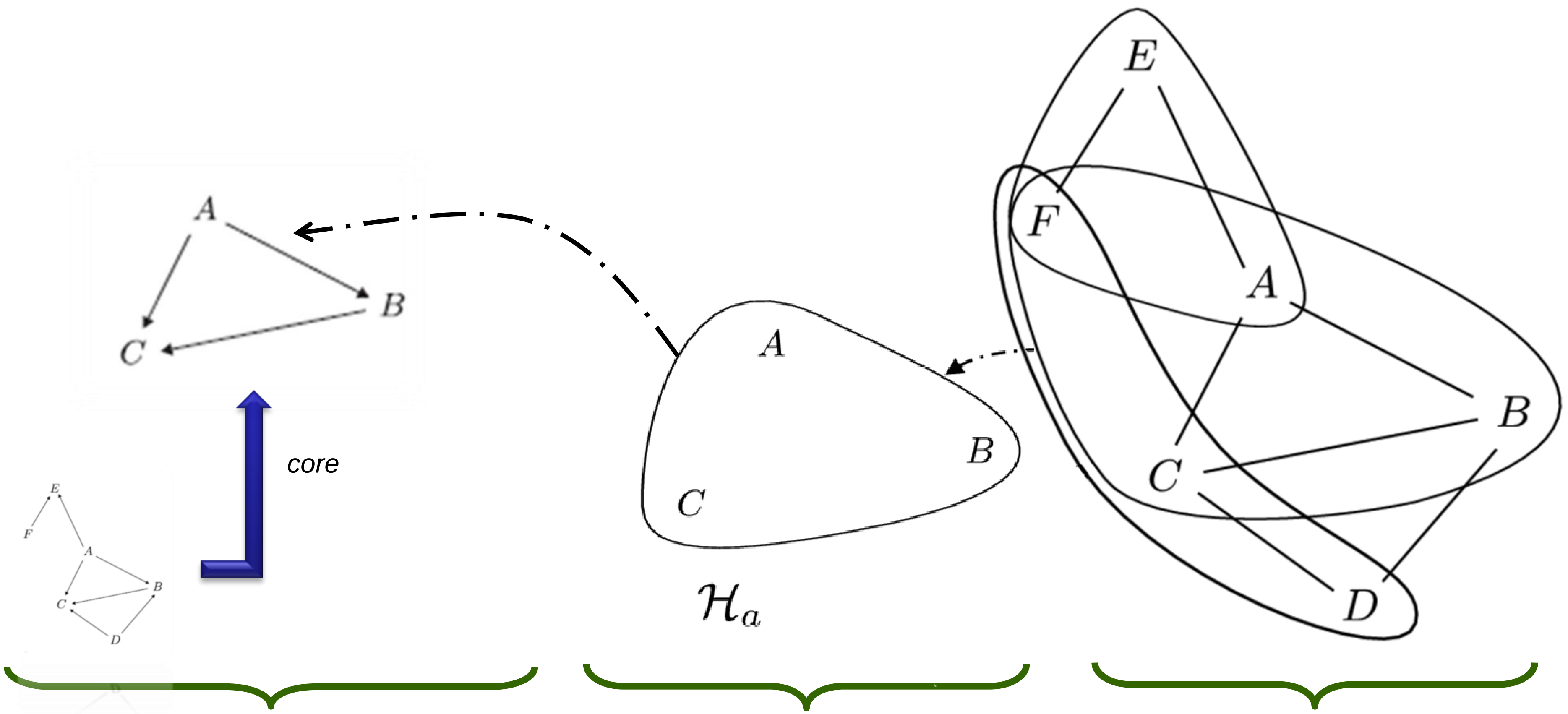


Structure of the CSP

Tree Projection

Available Views

Cores and Tree Projections

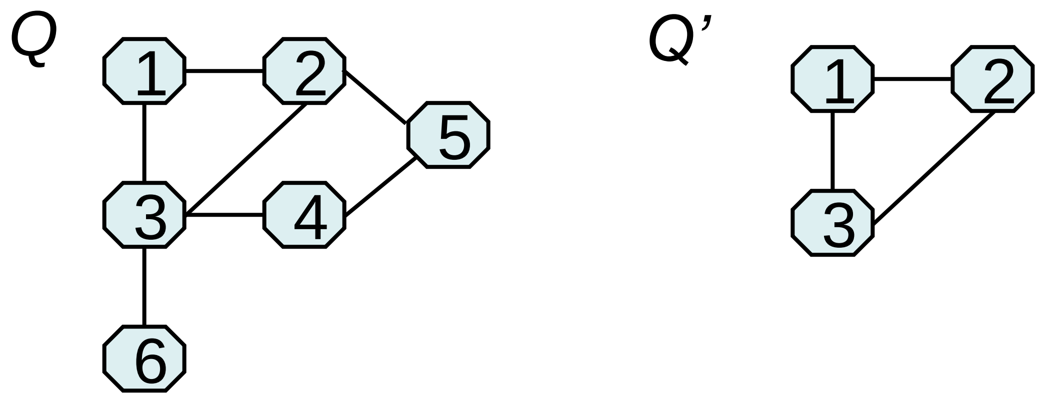


Structure of the CSP | **Tree Projection** | **Available Views**

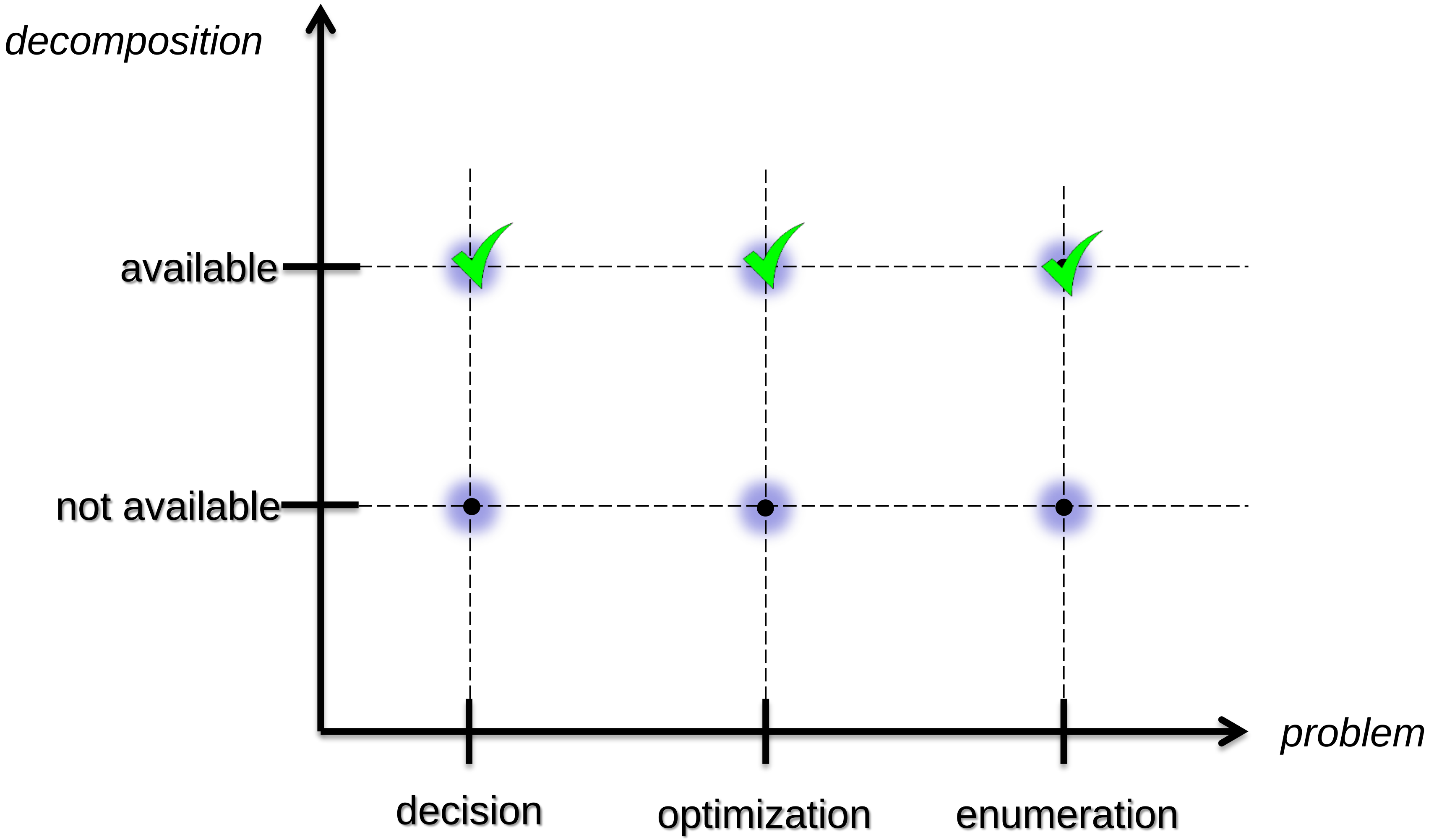
CORE is NP-hard

- 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∈3COL is NP-hard
- It is easy to see that G∈3COL ⇔ K_3 is the core of G

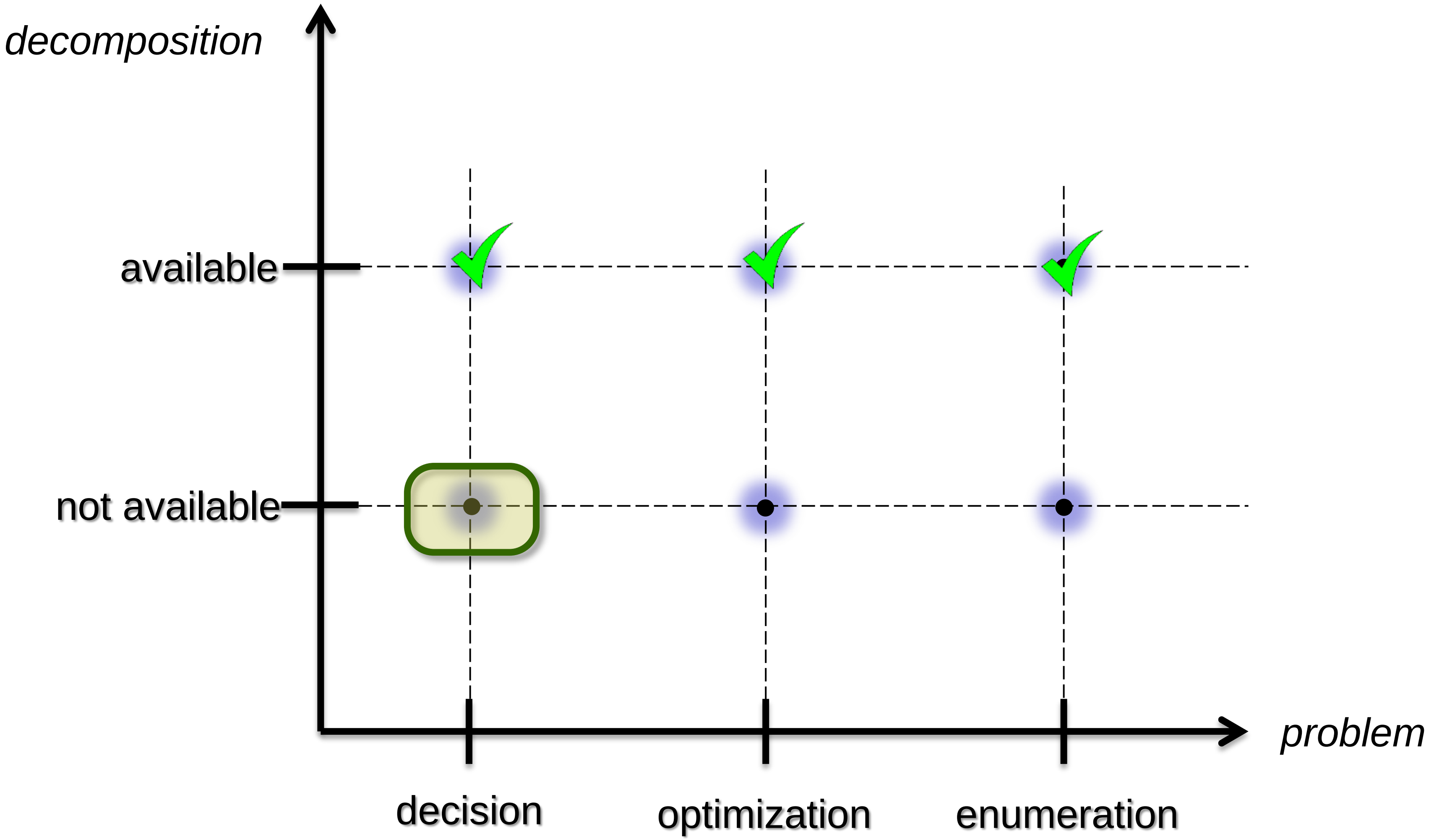
Example:



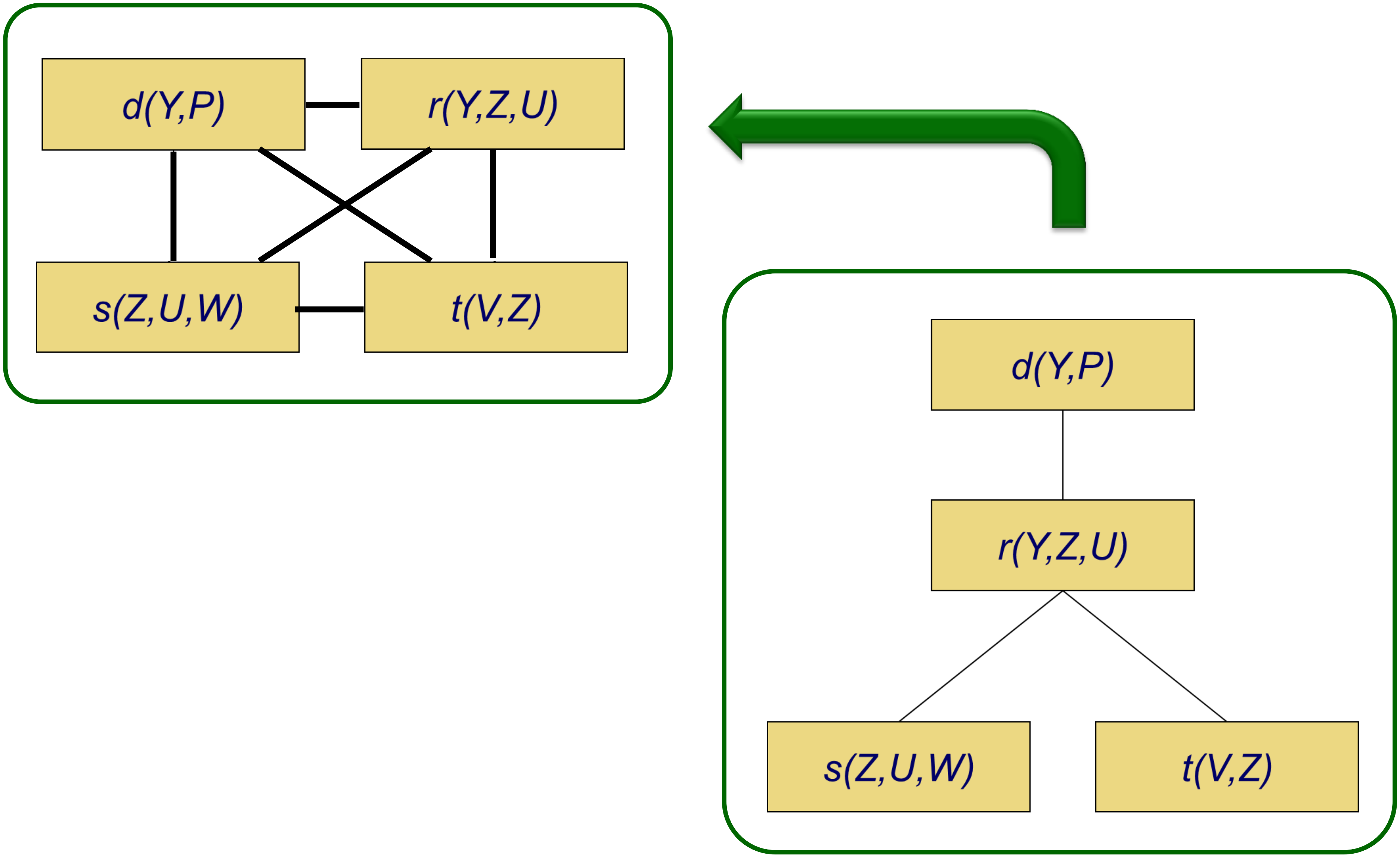
Overview


decomposition
available
not available
problem
decision
optimization
enumeration

Overview


decomposition
available
not available
problem
decision
optimization
enumeration

Enforcing Local Consistency (Acyclic)



Enforcing Local Consistency (Decomposition)

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

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

Enforcing Local Consistency (Decomposition)

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

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

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

[Sagiv & Smueli, ‘93]

Enforcing Local Consistency (Decomposition)

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

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

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

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

There is a polynomial-time algorithm that:

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

Even Better

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

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

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

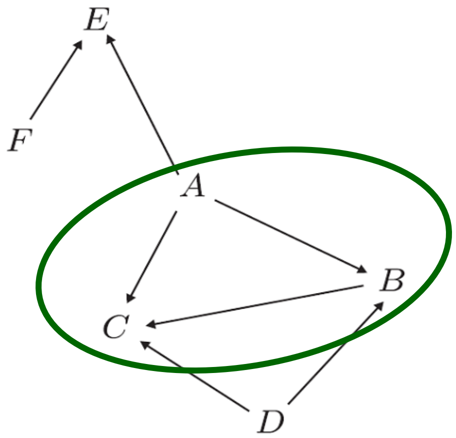
- The followings are equivalent:
  - Local consistency solves the decision problem
  - There is *a core* of the query having a tree projection

[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

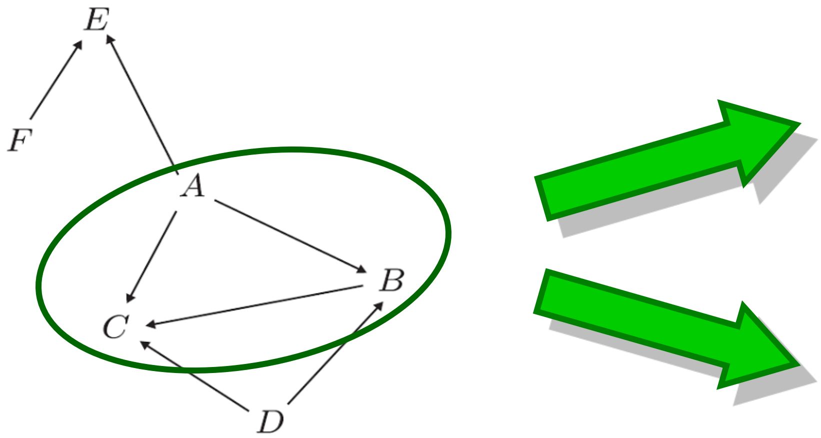
$$Q \;:\; r(A,B) \wedge r(B,C) \wedge r(A,C) \wedge r(D,C) \wedge r(D,B) \wedge r(A,E) \wedge r(F,E),$$

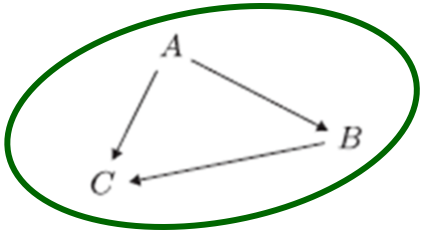


The Precise Power of Local Consistency

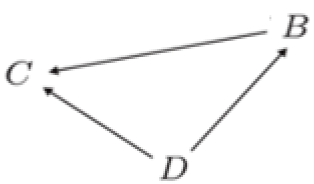
- The followings are equivalent
  - Local consistency solves the decision problem
  - There is *a core* of the query having a tree projection

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





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]

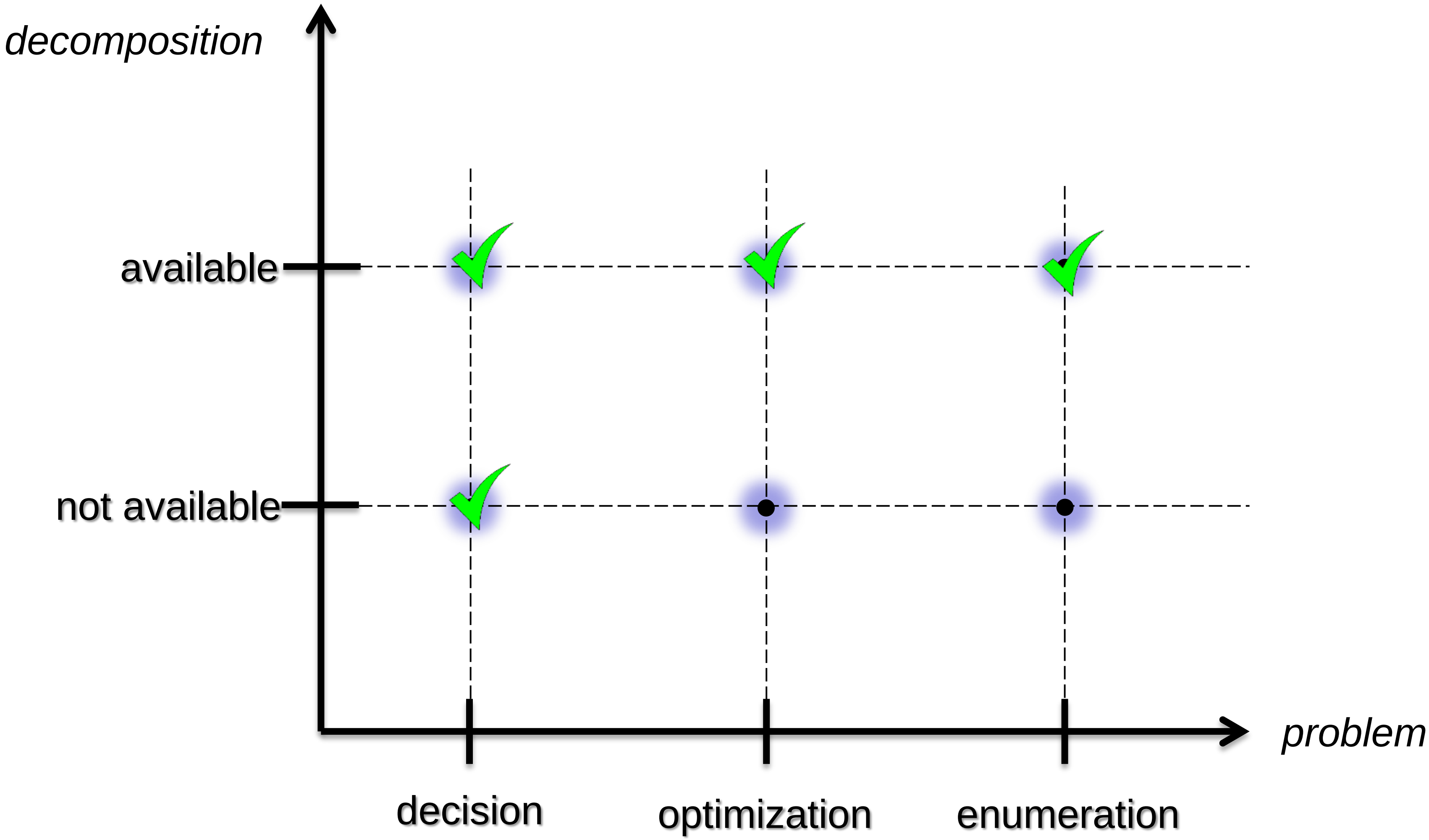
Back on the Result

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

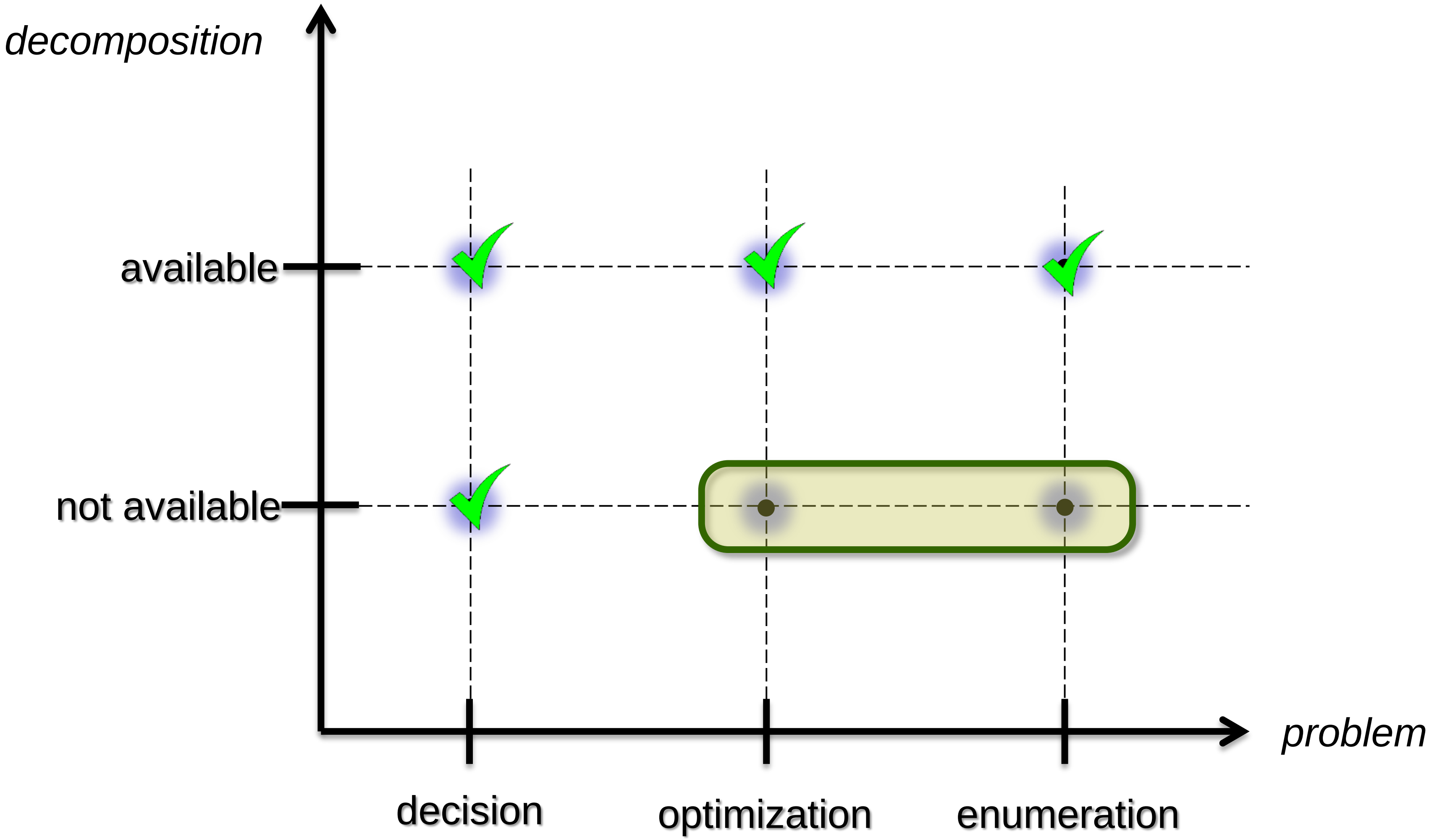
«Promise» tractability

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

Overview


decomposition
available
not available
problem
decision
optimization
enumeration

Overview



[Greco & Scarcello, CP‘11]

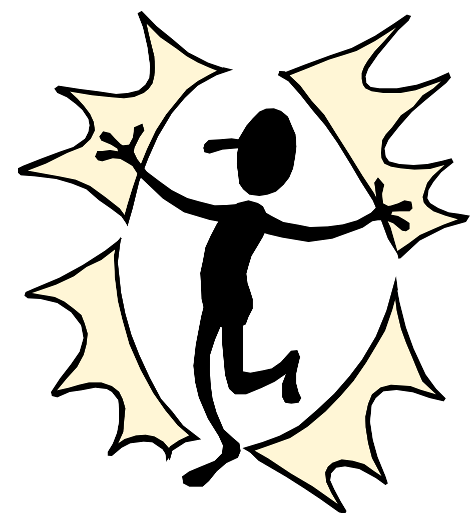
Recall This Approach

- Bottom-Up + Top-Down propagation
- Fix X_1 to the next value and *propagate*
  - Fix X_2 to the next value and *propagate*
    …
    - Fix X_n to the next value and *propagate*

Backtracking with no wrong choices

↓

Enumeration WPD



Recall This Approach

- Bottom-Up + Top-Down propagation
- Fix X_1 to the next value and *propagate*
  - Fix X_2 to the next value and *propagate*
    - …
    - Fix X_n to the next value and *propagate*

If there is a tree projection, then
the algorithm solves the enumeration problem

Recall This Approach

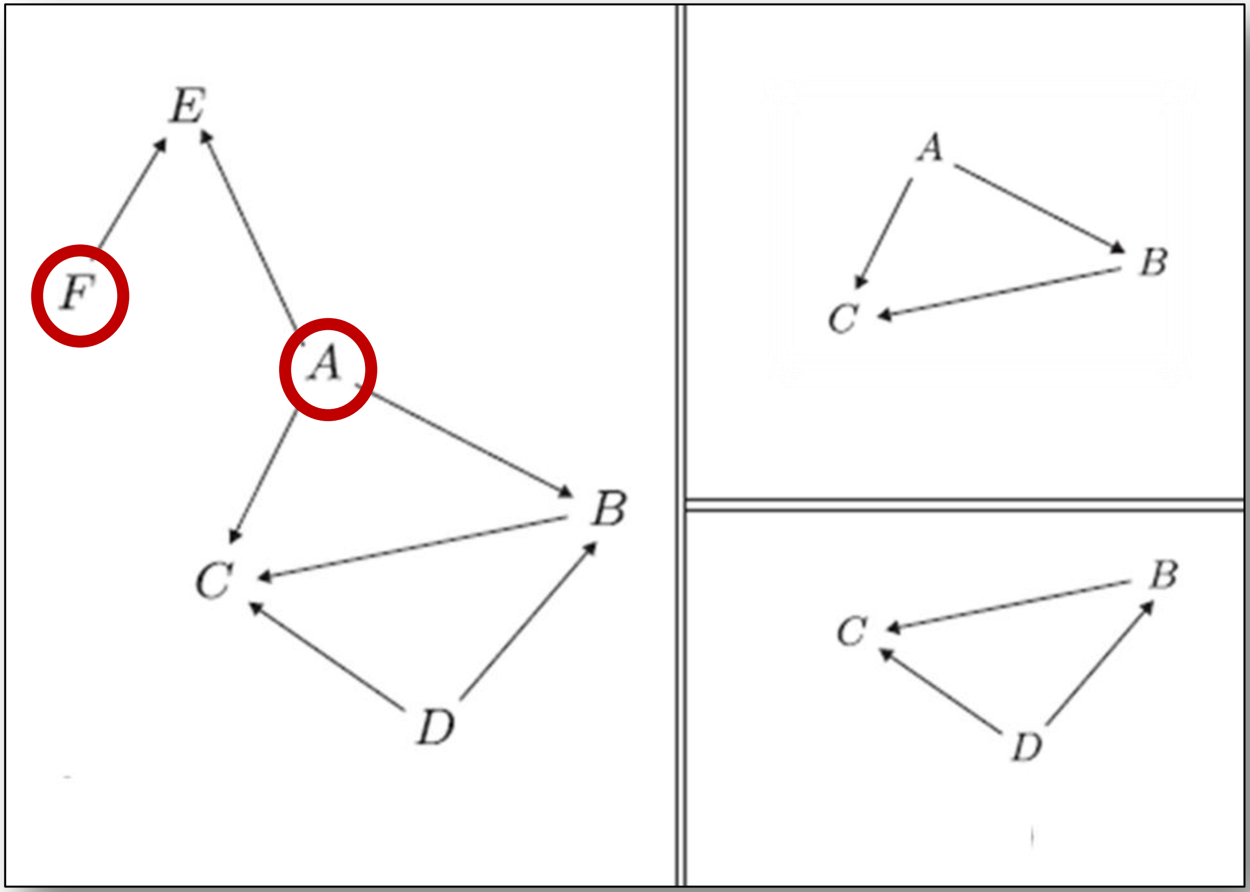
- Bottom-Up + Top-Down propagation
- Fix X_1 to the next value and *propagate*
  - Fix X_2 to the next value and *propagate*
    - …
    - Fix X_n to the next value and *propagate*

If there is a tree projection, then the algorithm solves the enumeration problem

but more can be done…

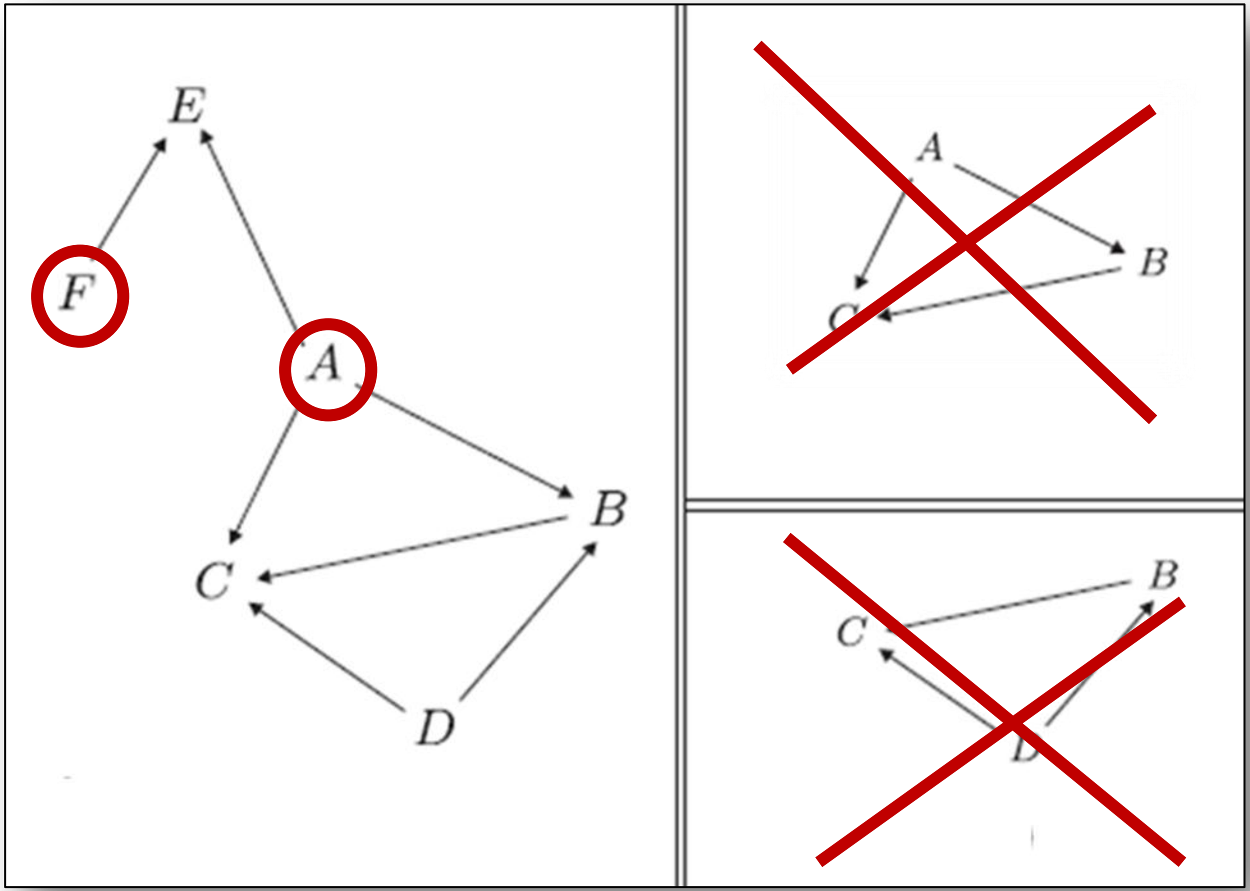
Tp-covered

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



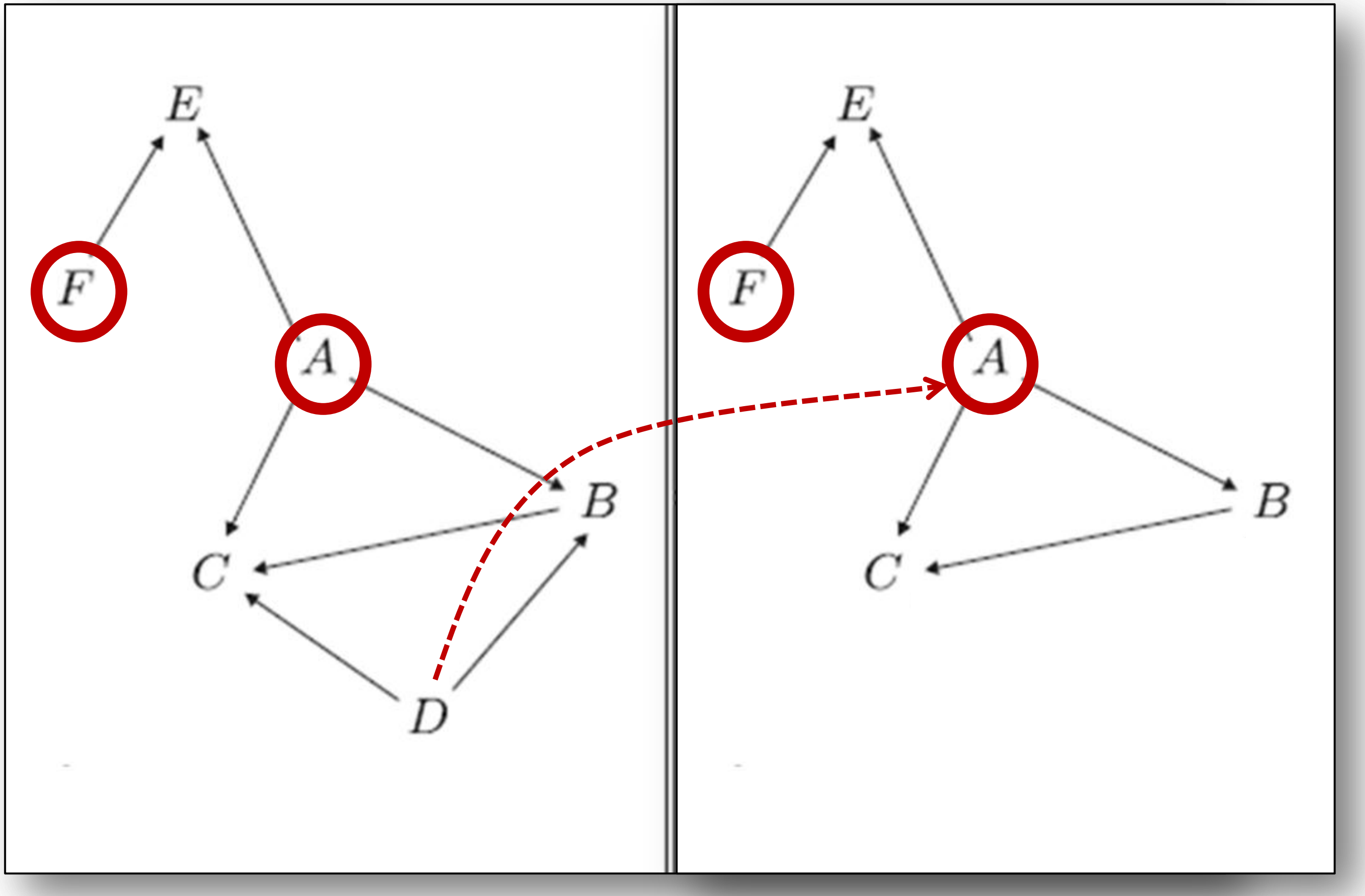
Tp-covered

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



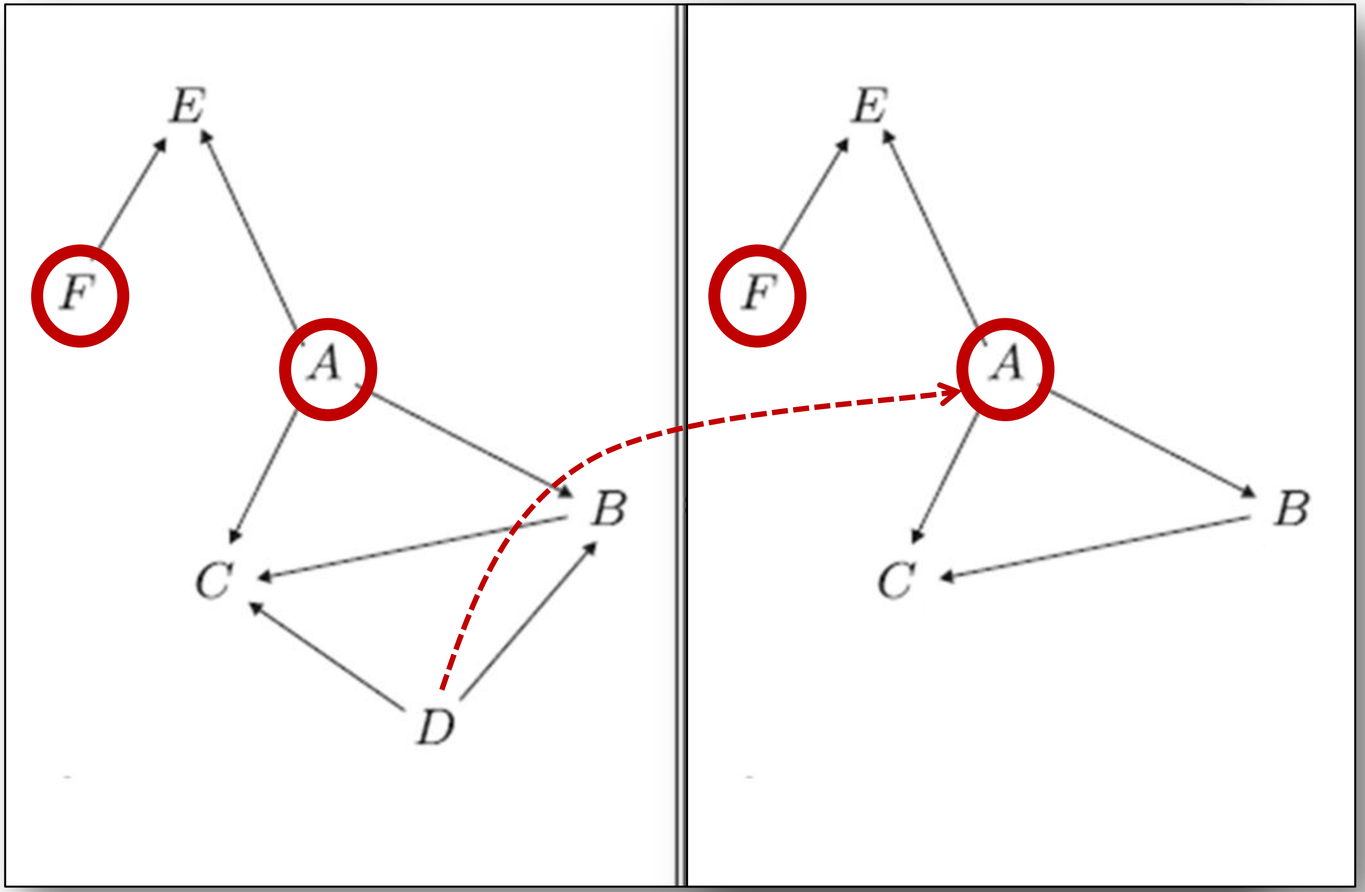
Tp-covered

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



Tp-covered

- {F,A} is tp-covered, if there is a tree projection covering an «output»-aware core



Results on Enumeration

There is a tree projection

↓

Enumeration over O is feasible WPD

[Greco & Scarcello, CP'11]

Results on Enumeration

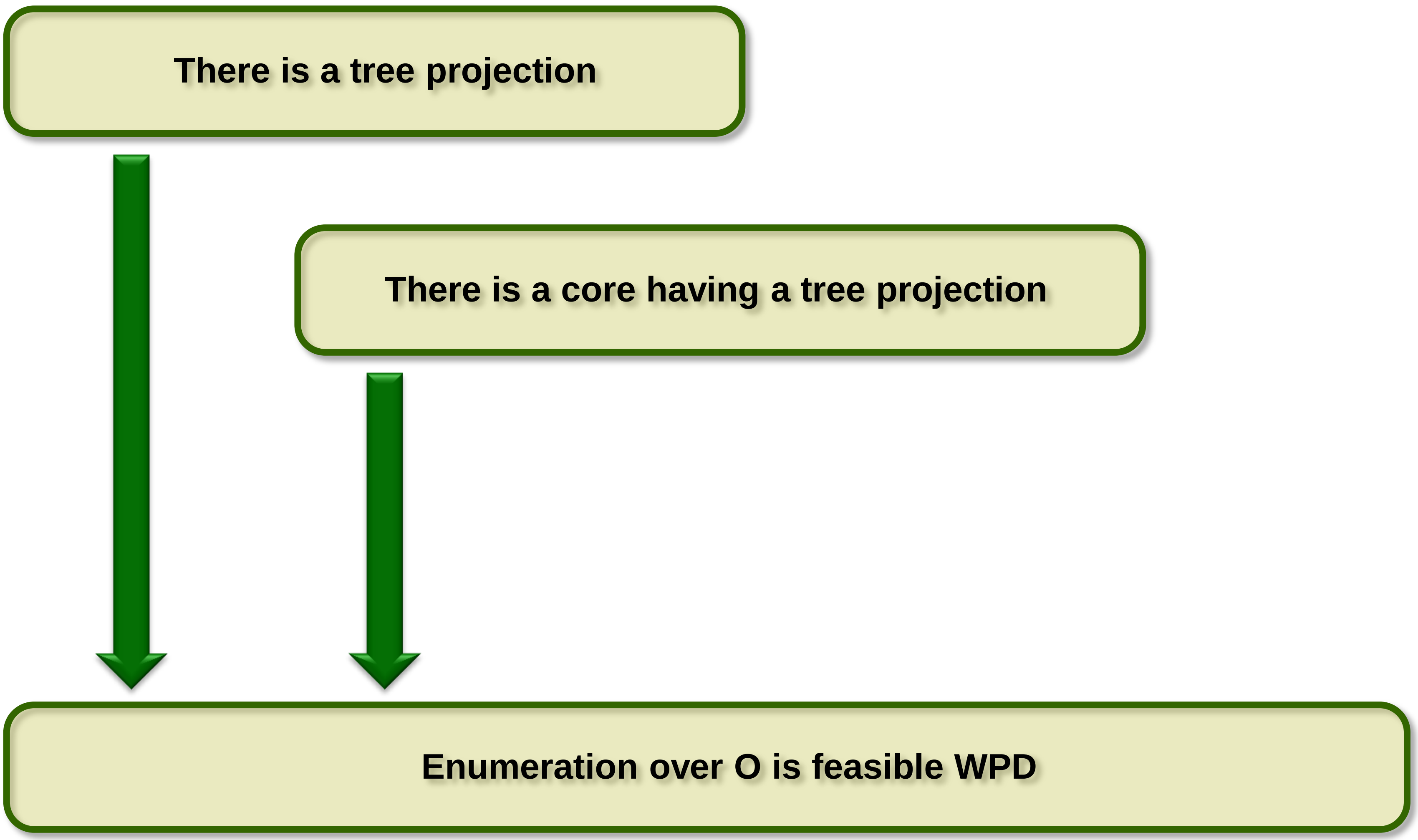
There is a tree projection

Does not need to be computed

Enumeration over O is feasible WPD

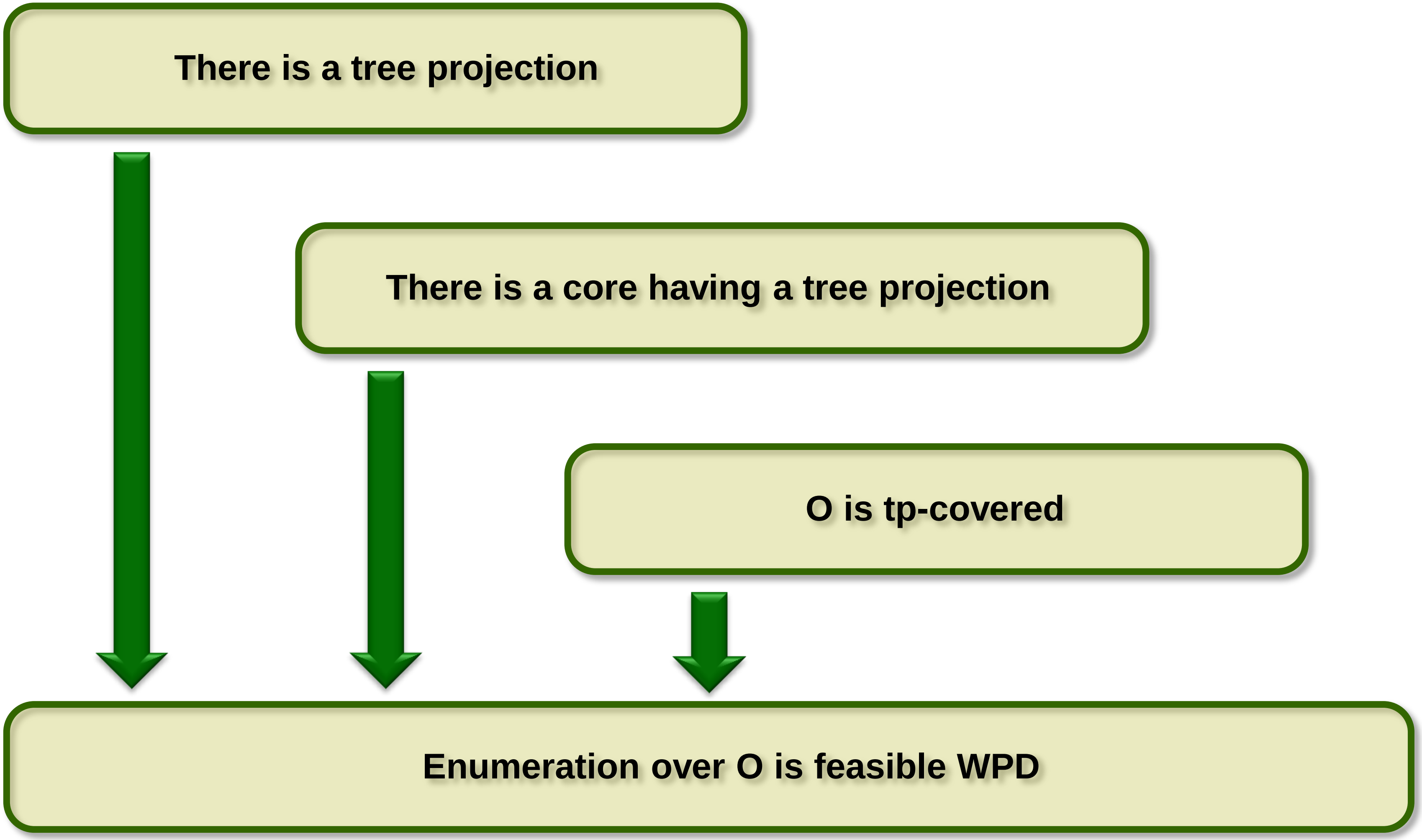
[Greco & Scarcello, CP'11]

Results on Enumeration



[Greco & Scarcello, CP'11]

Results on Enumeration



[Greco & Scarcello, CP‘11]

Results on Enumeration

There is a tree projection

O is tp-covered

Results on Enumeration

There is a tree projection

- The algorithm might return FAIL
  - Solutions so far computed are correct
  - There is no tree projection

O is tp-covered

«Promise» tractability

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

[Greco & Scarcello, PODS'10]

Thank you!

Appendix: LCFL Results

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, and Scarcello, JACM'01]

LOGCFL

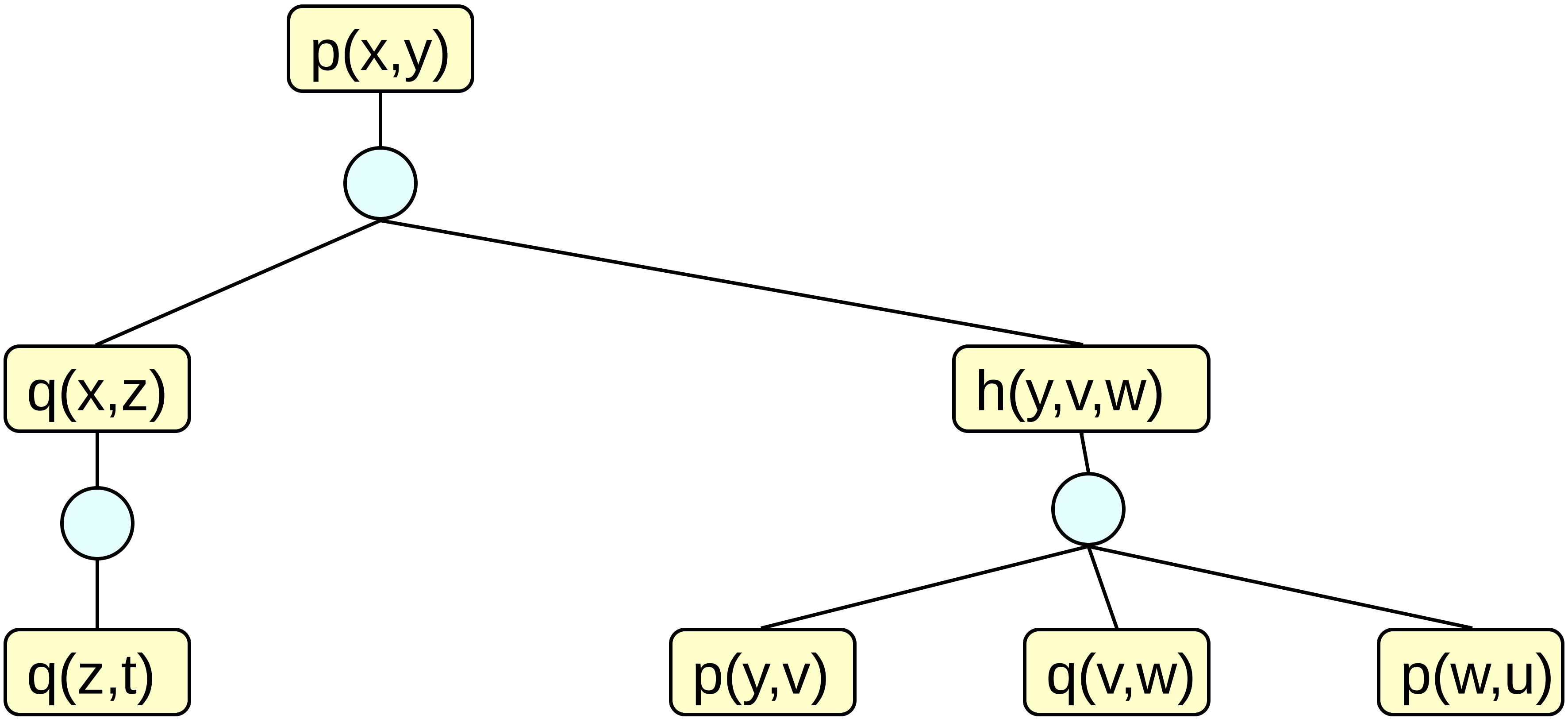
- LOGCFL: class of problems/languages that are logspace-reducible to a CFL
- Admit efficient parallel algorithms

$$AC_0 \subseteq NL \subseteq \text{LOGCFL} = SAC_1 \subseteq AC_1 \subseteq NC_2 \subseteq \cdots \subseteq NC = AC \subseteq P \subseteq NP$$

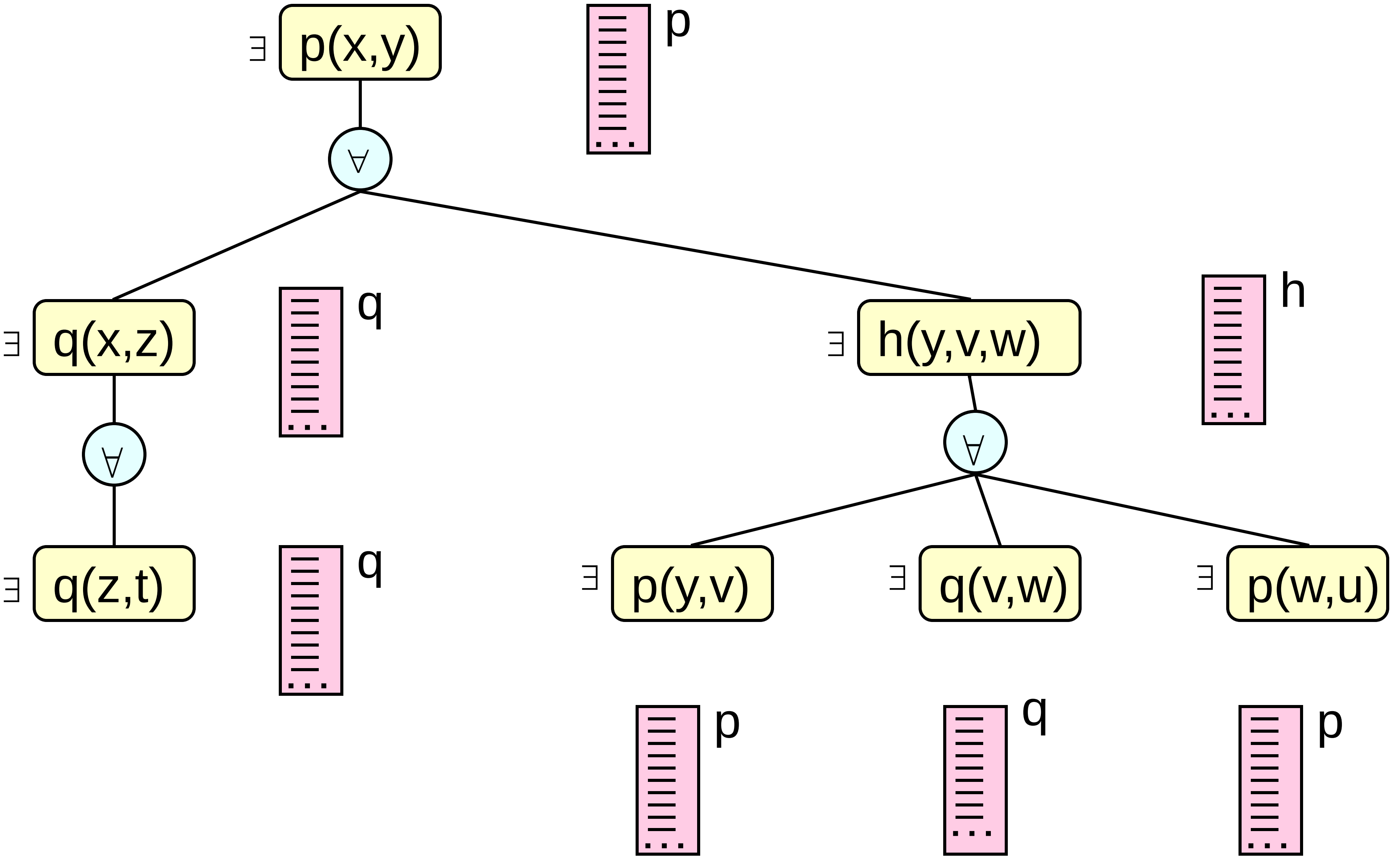
Characterization of LOGCFL [Ruzzo '80]:

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

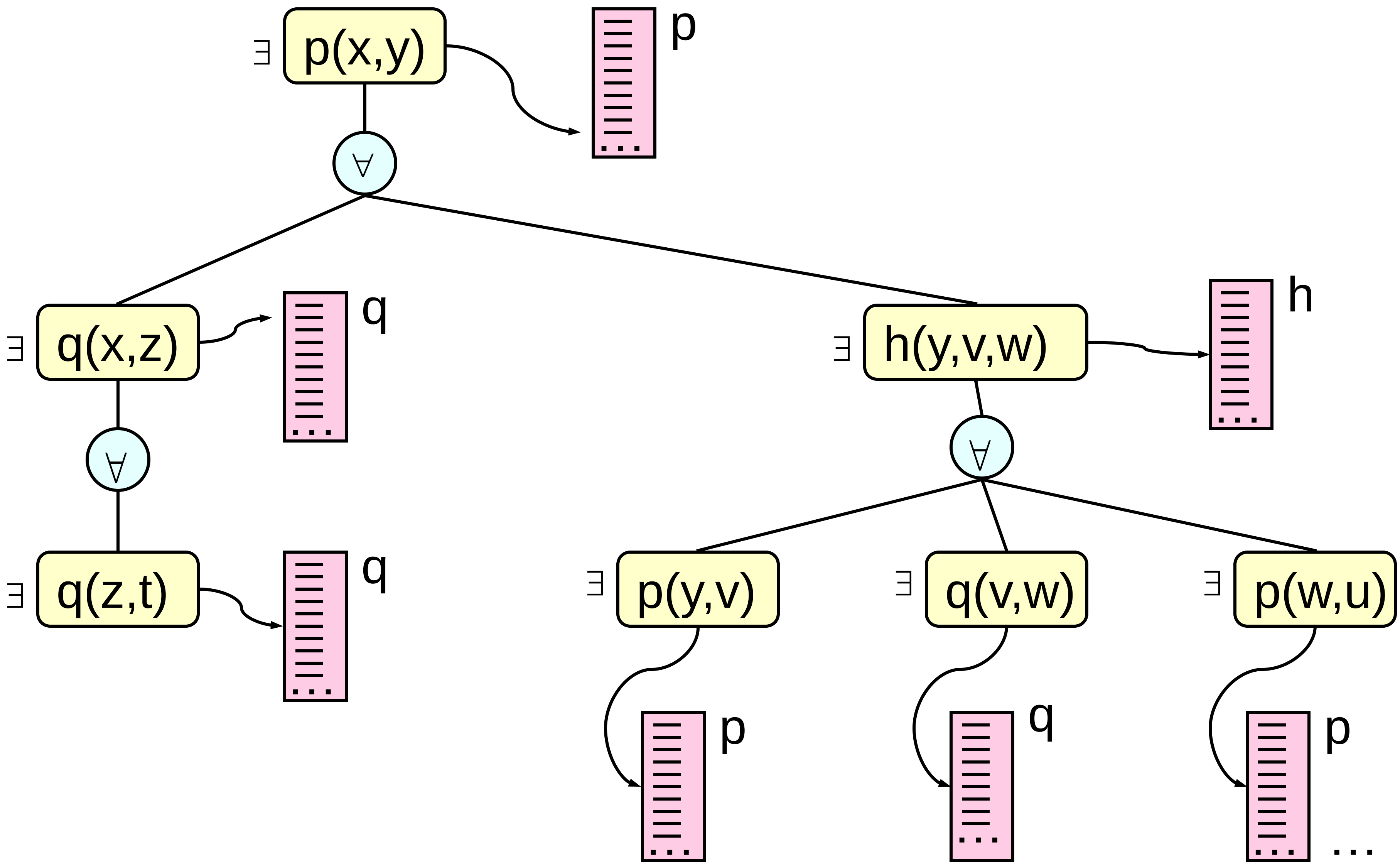
ABCQ is in LOGCFL



ABCQ is in LOGCFL



ABCQ is in LOGCFL



CSOP Extensions: Formal Framework

- Evaluation Functions

  - $\mathbb{D}$ domain of values, $\succeq$ a total order over it
  - *evaluation function* $\mathcal{F}$: a tuple $\langle w, \oplus \rangle$ with $w : \mathit{Var} \times \mathcal{U} \mapsto \mathbb{D}$
  - $\oplus$ commutative, associative, and closed binary operator with an identity element over $\mathbb{D}$
  - $\mathcal{F}(\theta) = \bigoplus_{X/u \in \theta} w(X, u)$ (with $\mathcal{F}(\emptyset)$ being the identity w.r.t. $\oplus$)

- Monotone Functions

$$\mathcal{F}(\theta) \succeq \mathcal{F}(\theta') \quad \Rightarrow \quad \mathcal{F}(\theta) \oplus \mathcal{F}(\theta'') \succeq \mathcal{F}(\theta') \oplus \mathcal{F}(\theta''), \quad \forall \theta''$$

[Greco & Scarcello, CP‘11]

CSOP Extensions: Multi-Objective Optimization

- We often want to express more preferences, e.g.,
  - minimize cost, then minimize total time, or
  - maximize the profit, then minimize the number of different buyers, or transactions

- Formally,

  – $L = [\mathcal{F}_1, ..., \mathcal{F}_m]$
  – $L(\theta)$ denotes $(\mathcal{F}_1(\theta), ..., \mathcal{F}_m(\theta)) \in \mathbb{D}_1 \times \cdots \times \mathbb{D}_m$

- Compare vectors by the lexicographical precedence relationship (Cascade of preferences)

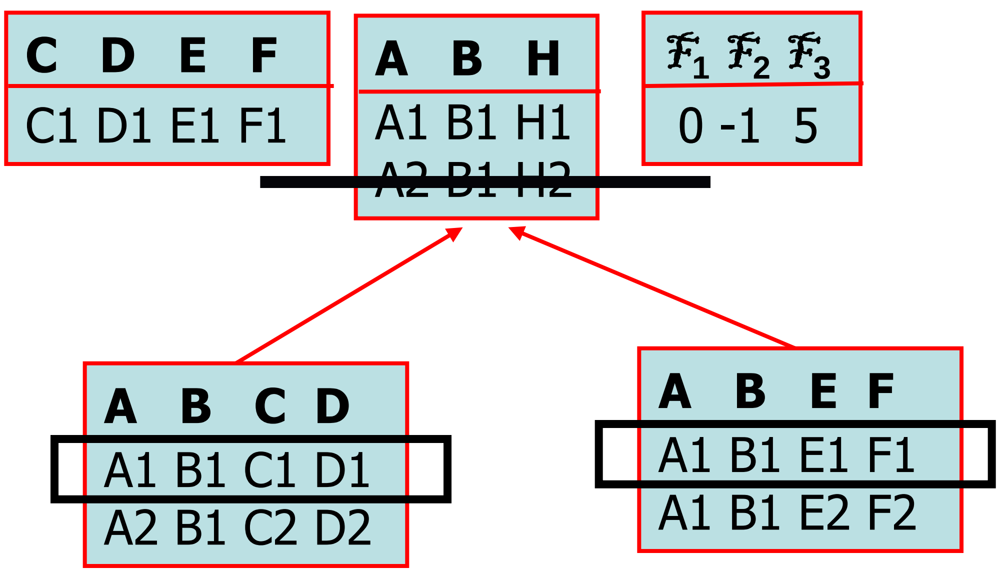
Linearization (following [Brafman et al'10])

- $\succeq_{\mathcal{U}}$ an arbitrary total order defined over $\mathcal{U}$
- $\ell = [X_1, ..., X_n]$ a list including all the variables in Var

- Define the total order $\succeq_L^\ell$
  - ties in $\succeq_L$ are resolved according to the lexicographical precedence relationship ℓ over variables and the total order $\succeq_{\mathcal{U}}$ over $\mathcal{U}$
  - $\succeq_L^\ell$ is a refinement of $\succeq_L$

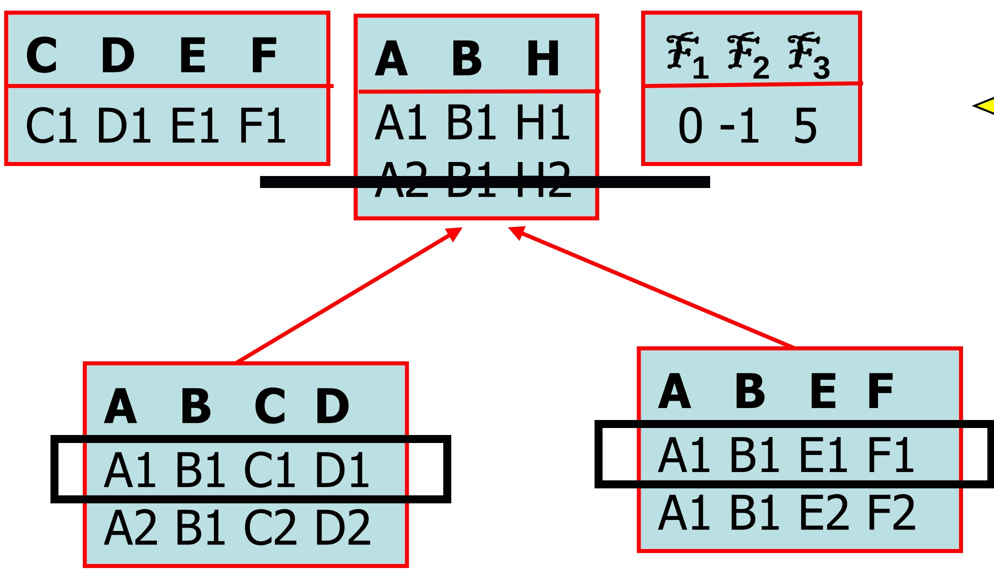
Hints (motonone lists)

- Extend the dynamic programming approach
- Because of linearization we have a total order
- The algorithm exploits an extended list of evaluation functions (still monotone) $[\mathcal{F}_1, \ldots, \mathcal{F}_m, \mathcal{F}_\ell]$, $\mathcal{F}_\ell = \langle w_\ell, + \rangle$
- where $w_\ell(X_i, u) = |\mathcal{U}|^{n-i} \times r_{\mathcal{U}}(u)$



Hints (motonone lists)

- Extend the dynamic programming approach
- Because of linearization we have a total order
- The algorithm exploits an extended list of evaluation functions (still monotone) $[\mathcal{F}_1, \ldots, \mathcal{F}_m, \mathcal{F}_\ell]$, $\mathcal{F}_\ell = \langle w_\ell, + \rangle$
- where $w_\ell(X_i, u) = |\mathcal{U}|^{n-i} \times r_{\mathcal{U}}(u)$



For each tuple, manage an Additional vector with the best Values for the m+1 functions

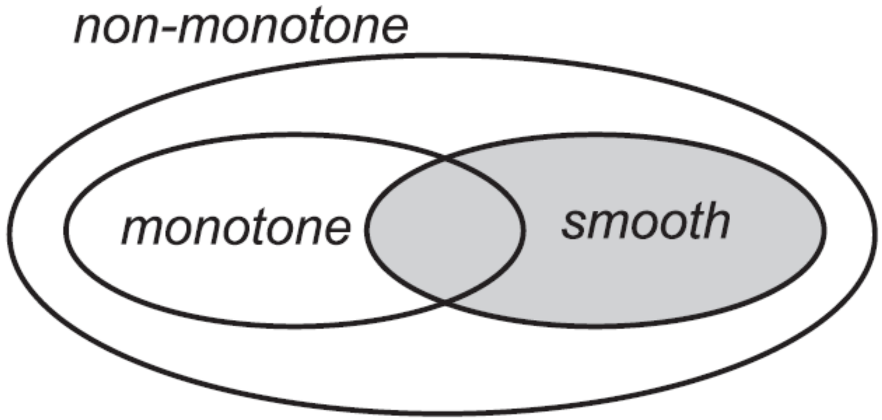
Note that we have no ties, because of the additional function

CSOP Extensions: Smooth Functions

- $\mathcal{F}$ is *smooth* (w.r.t. Φ and DB) if, $\forall\theta$, the value $\mathcal{F}(\theta)$ is polynomially-bounded by the size of Φ, DB, and $\mathcal{F}$
- a list L of evaluation functions is *smooth* if it consists of a constant number of smooth evaluation functions

CSOP Extensions: Smooth Functions

- Manipulate small (polynomially bounded) values
- Occur in many applications (for instance, in counting-based optimizations)
- May be non-monotonic



Examples of Smooth Functions

1. Finding solutions minimizing the number of variables mapped to certain domain values
   - It is smooth and monotone

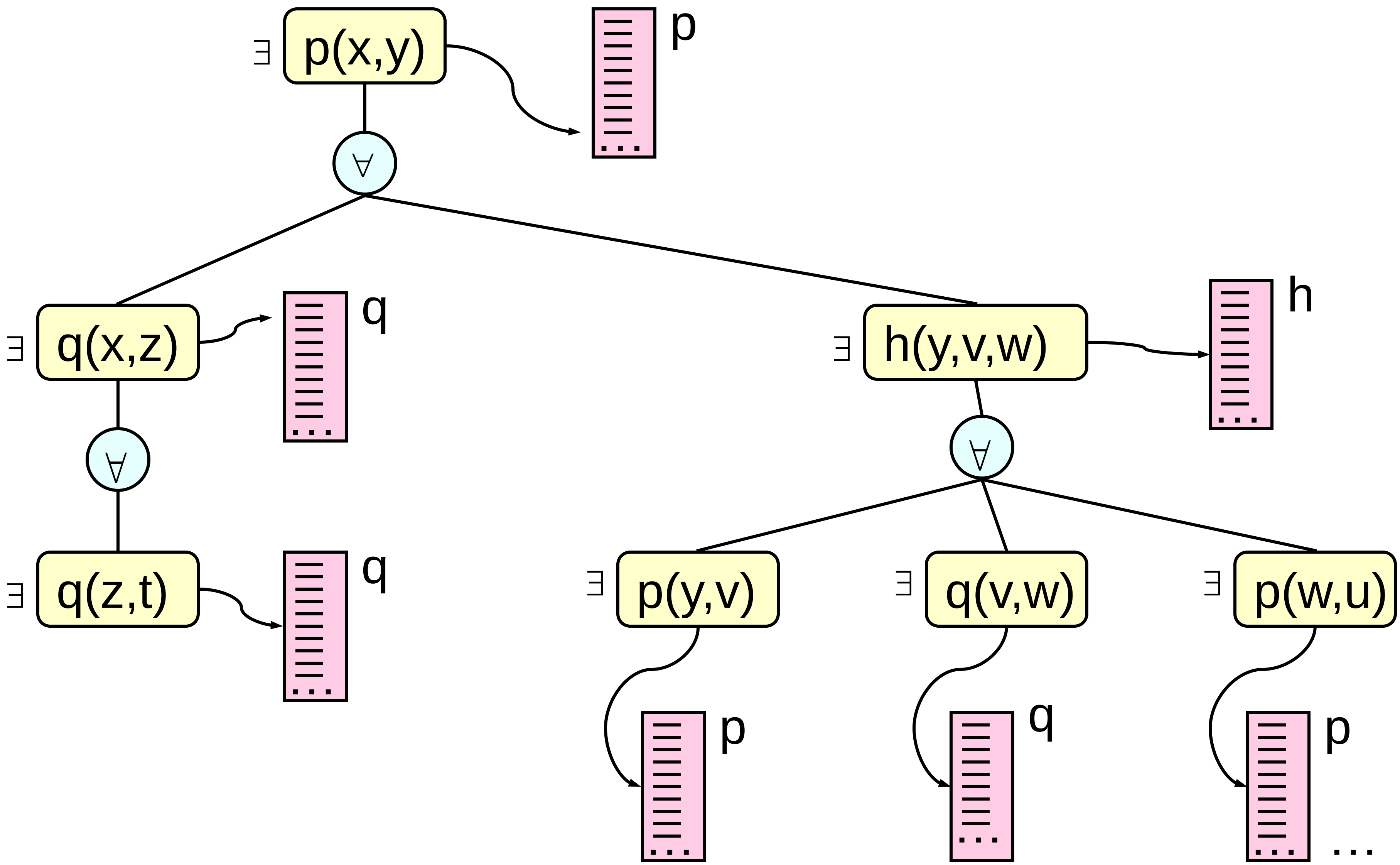
2. Finding solutions with an odd number of variables mapped to certain values (e.g. switch variables)
   - It is smooth and non-monotone

3. [2,1] (or viceversa) is a smooth list of evaluation functions

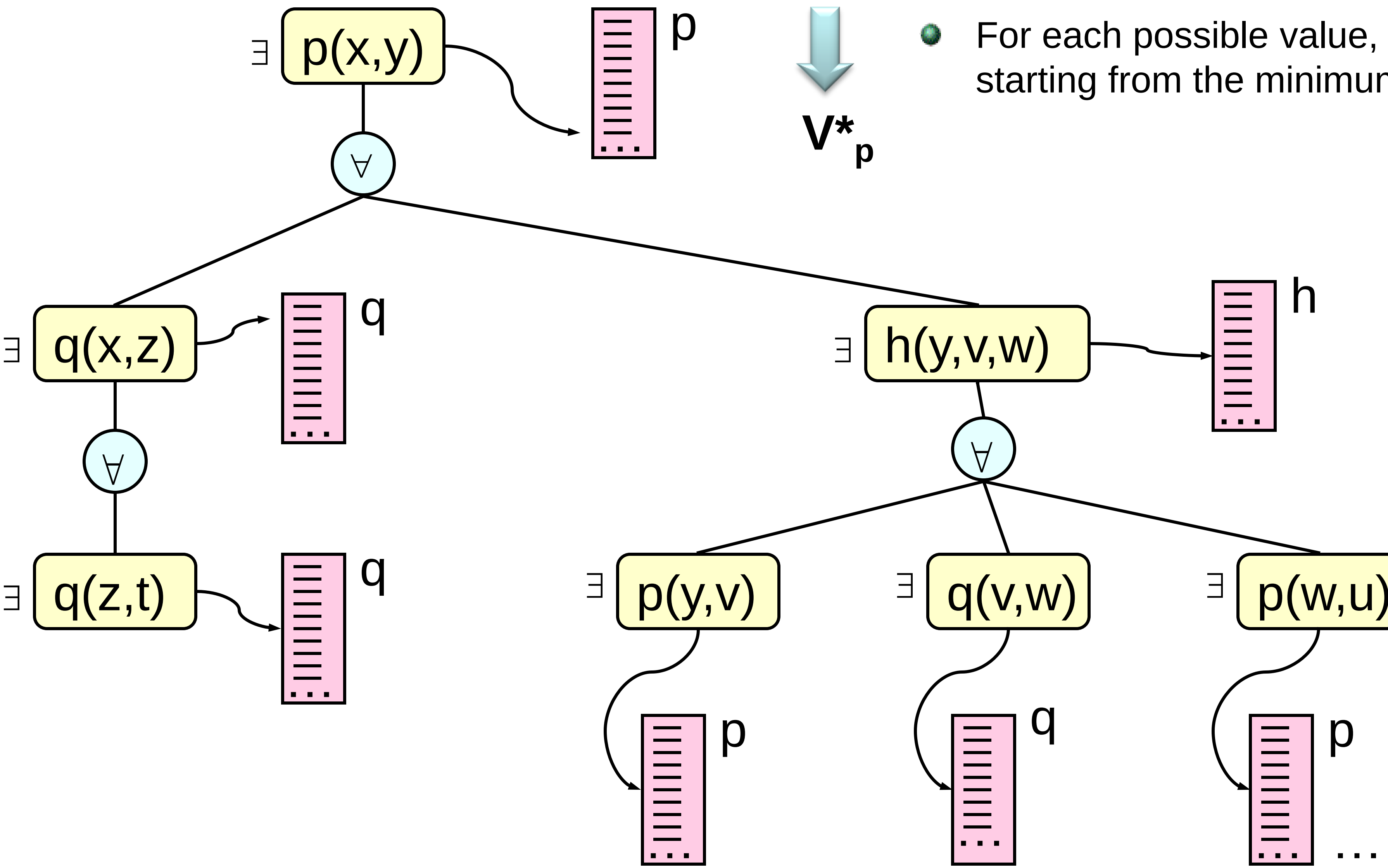
Tractability of Non-Monotone (smooth) Functions

- The classical dynamic programming approach does not work, with non-monotone functions
  - Good (partial) solutions in the subtree may lead to bad final solutions

Tractability of Non-Monotone (smooth) Functions

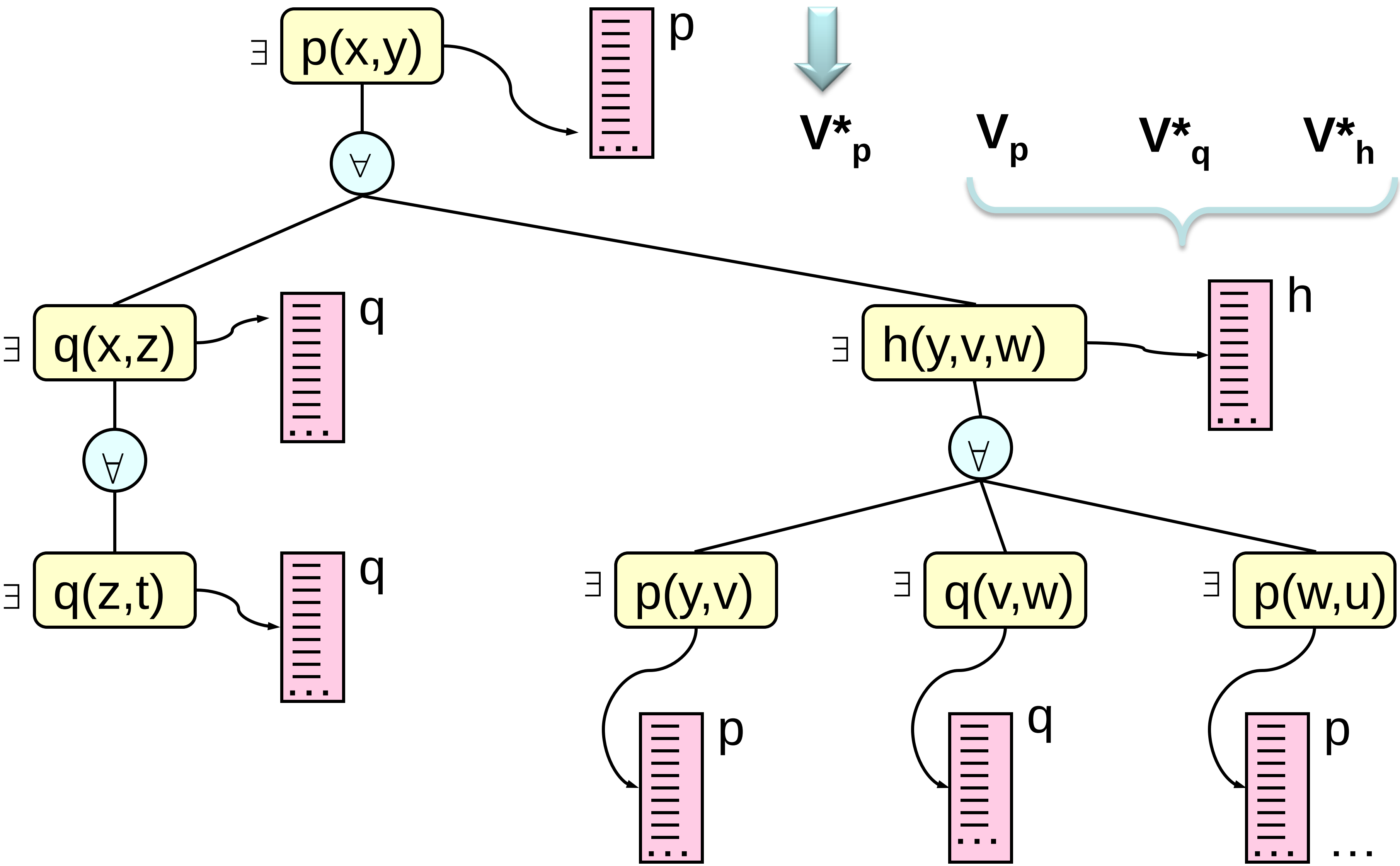


Tractability of Non-Monotone (smooth) Functions

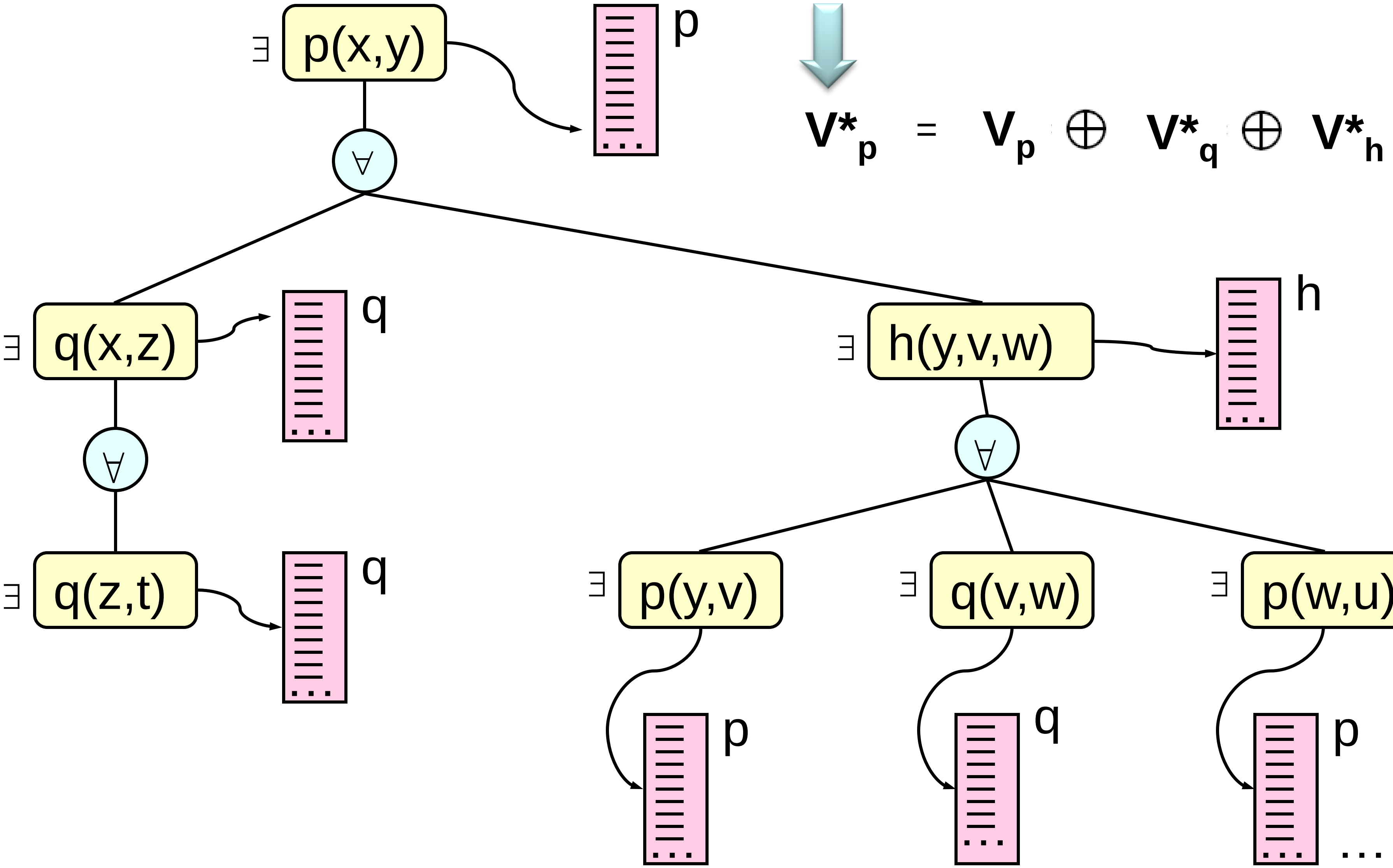


- For each possible value, starting from the minimum.

Tractability of Non-Monotone (smooth) Functions

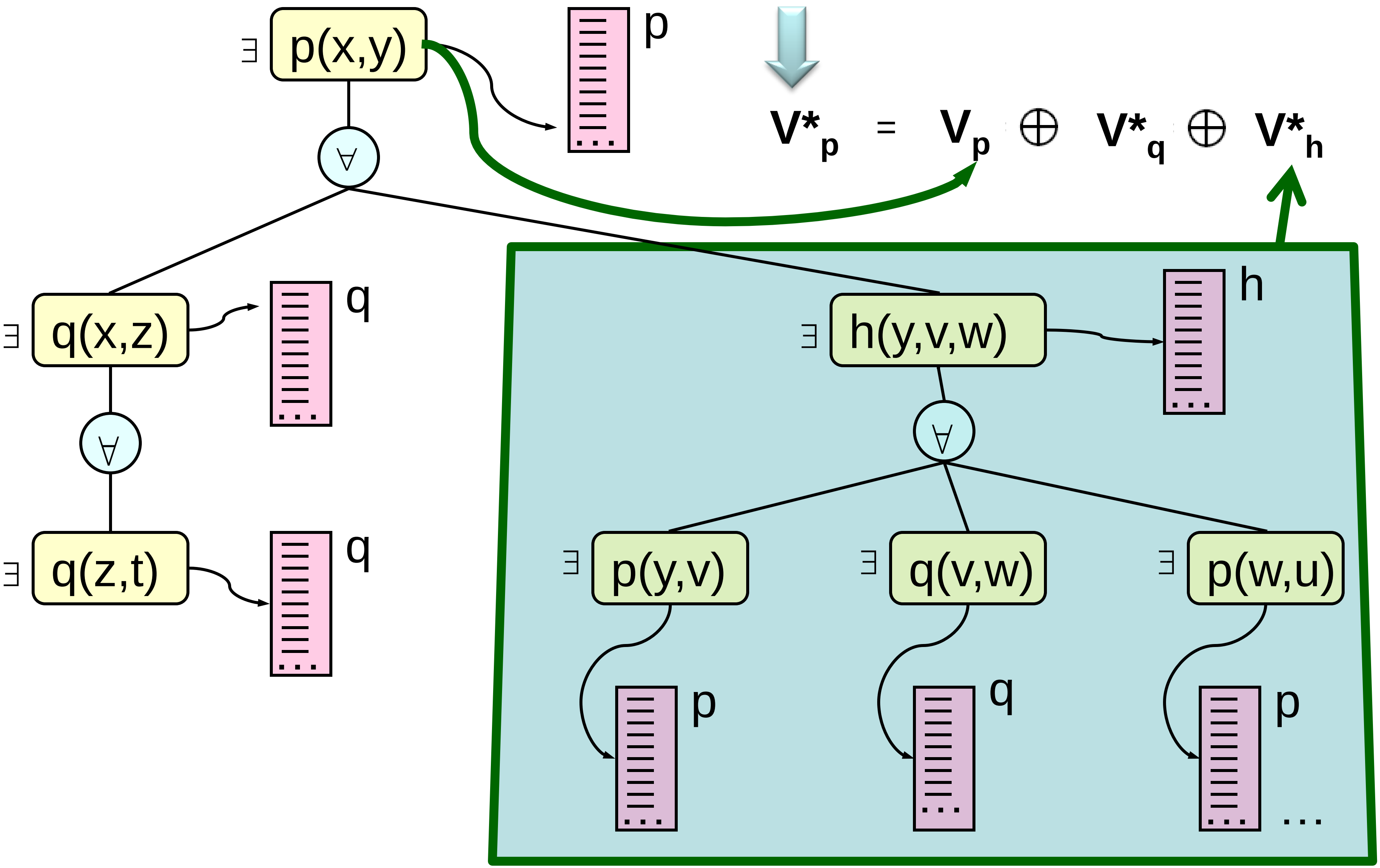


Tractability of Non-Monotone (smooth) Functions

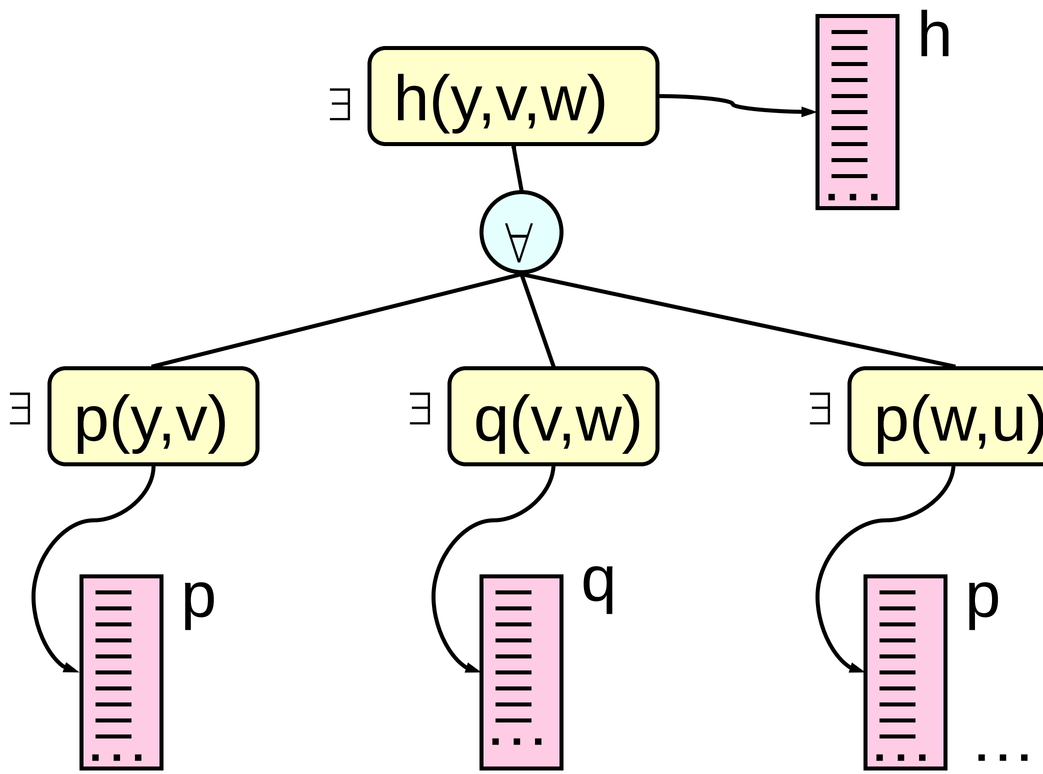


$$V^*_p = V_p \oplus V^*_q \oplus V^*_h$$

Tractability of Non-Monotone (smooth) Functions

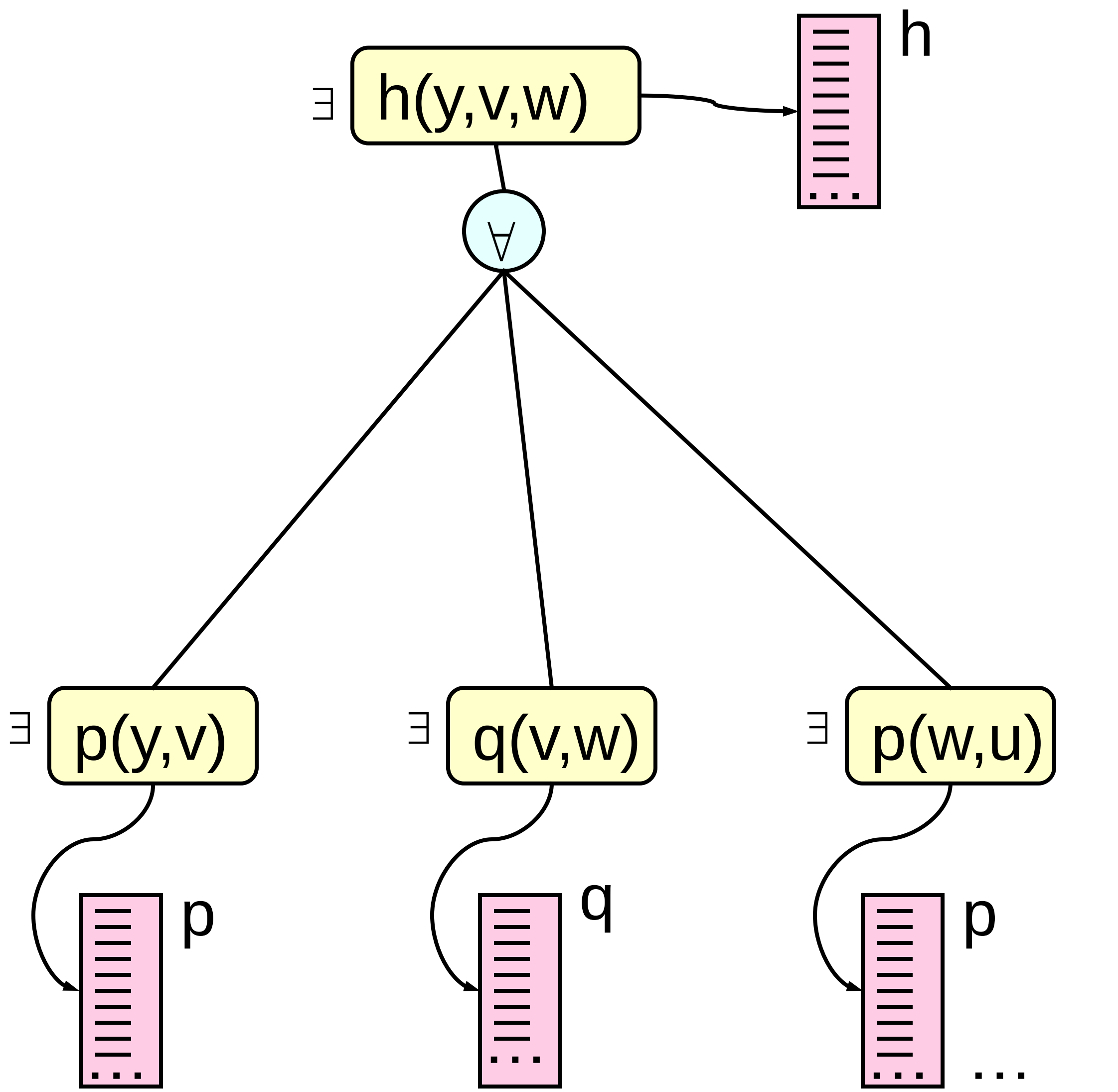


A Subtle Issue



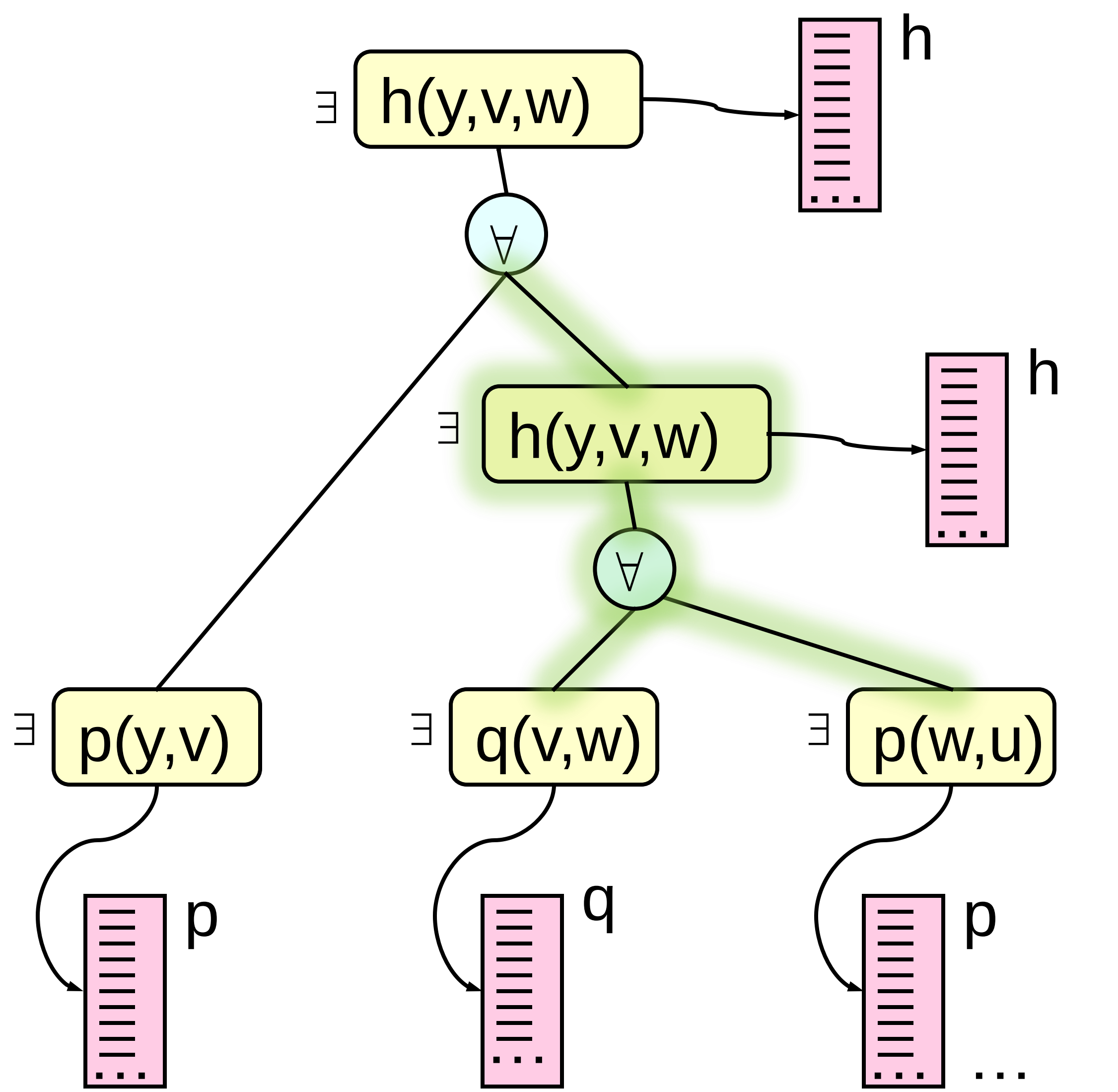
A Subtle Issue

- Binarization



A Subtle Issue

- Binarization



Appendix: TP-coverings