Conjunctive Queries

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.

But actually: HOM = BCQ = CSP

The Homomorphism Problem

Given two relational structures

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

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

Decide whether there exists a homomorphism h from A to B

$$h : U \longrightarrow V$$

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

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

The Homomorphism Problem

Given two relational structures

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

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

Decide whether there exists a homomorphism h from A to B

$$h: U \longrightarrow V$$

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

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

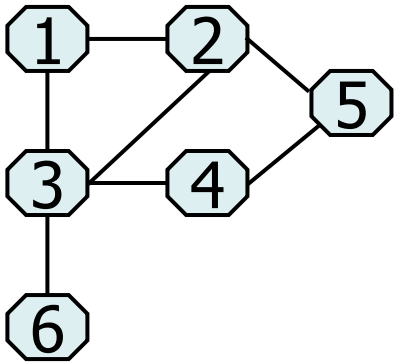
Feder & Vardi 93: Relationship to CSP, restrictions on B
Kolaitis & Vardi, 98: Relationship to Query Containment, restrictions on A, B

HOM is NP-complete

(well-known)

Membership: Obvious, guess *h*.

Hardness: Transformation from 3COL.



A

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

B

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

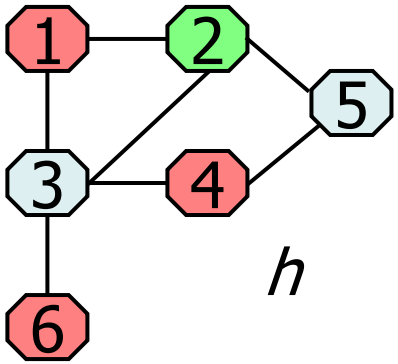
Graph 3-colourable iff HOM(*A*,*B*) yes-instance.

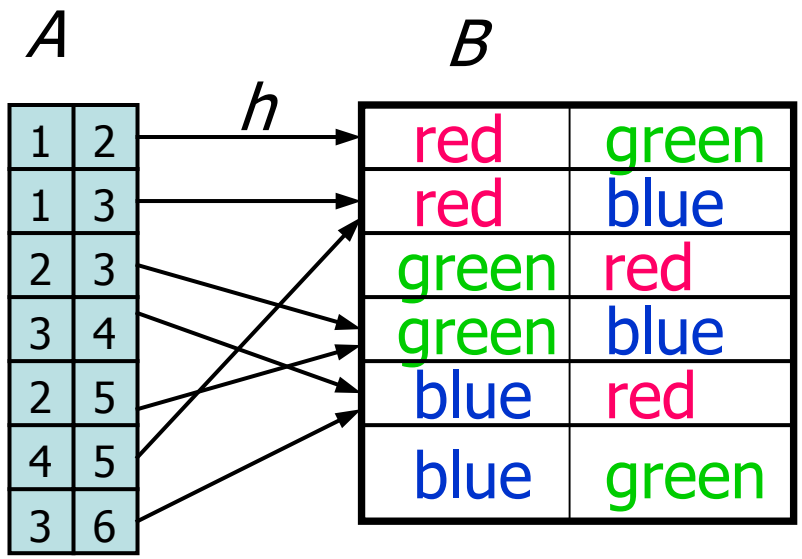
HOM is NP-complete

(well-known)

Membership: Obvious, guess *h*.

Hardness: Transformation from 3COL.

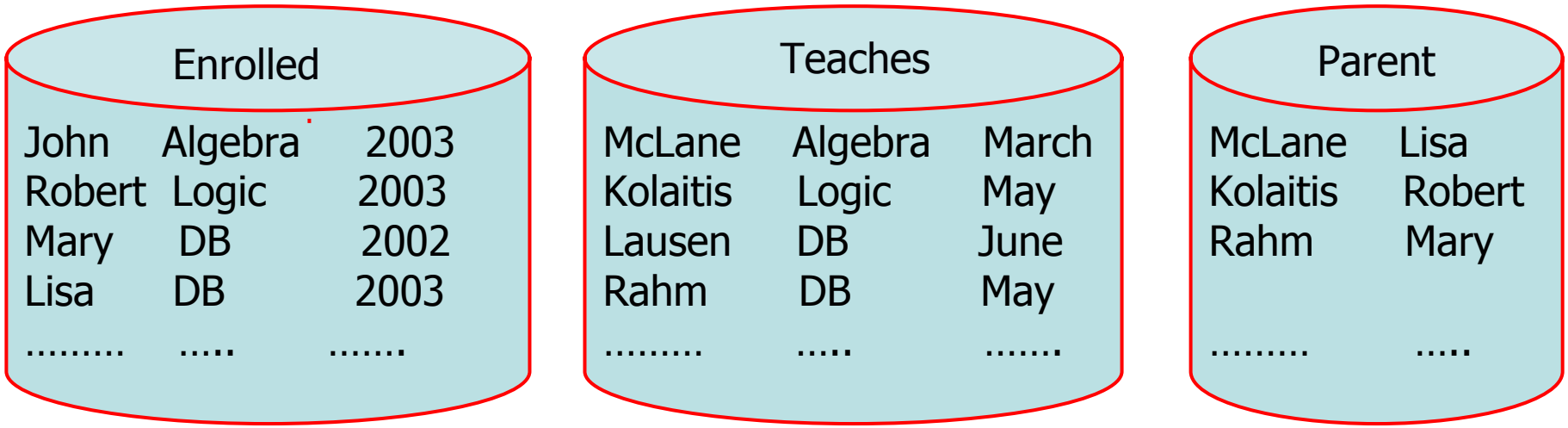




Graph 3-colourable iff HOM(*A*,*B*) yes-instance.

Boolean Conjunctive Query

DATABASE:



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

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

Boolean Conjunctive Query (2)

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

- Is there any teacher whose child attend some course?

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

Boolean Conjunctive Queries

The problem BCQ:

Instance: < **DB, Q**>

Question: Has **Q** a nonempty result over **DB?**

Conjunctive Query with Output

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

- Output all students x enrolled in a course taught by a parent of x

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

Conjunctive Queries and RA, SQL

Conjunctive queries correspond to Select-Project-Join queries,and to simple "SELECT-FROM-WHERE SQL statements with a conjunctive WHERE clause.

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

Q: Ans(S) ← Enrolled(S,C,R) ∧ Teaches(P,C,A) ∧ Parent(P,S)

```
SELECT Enrolled.Pers#
FROM Enrolled Teaches Parent
WHERE Enrolled Pers# = Parent.Pers2
        and  Enrolled.Course = Teaches.Course
        and  Teaches.Pers# = Parent.Pers1
```

Logical View of CQs

A conjunctive query is a FO query of the form

$$\{r(\mathbf{x}) \mid \exists \mathbf{y}\, (r_1(\mathbf{x_1},\mathbf{y_1}) \wedge r2(\mathbf{x_2},\mathbf{y_2}) \wedge \ldots \wedge r2(\mathbf{x_k},\mathbf{y_k}))\}$$

Where:

- $\mathbf{x}$, $\mathbf{y}$ are disjoint lists of variables,
- all variables of x occur free in the formula $\exists \mathbf{y}\, (r_1(\mathbf{x_1},\mathbf{y_1}) \wedge r2(\mathbf{x_2},\mathbf{y_2}) \wedge \ldots \wedge r2(\mathbf{x_k},\mathbf{y_k}))$
- each x_i is a list of variables from $\mathbf{x}$,
- each y_i is contained in $\mathbf{y}$

Written: $r(\mathbf{x}) \leftarrow r_1(\mathbf{x_1},\mathbf{y_1}) \wedge r2(\mathbf{x_2},\mathbf{y_2}) \wedge \ldots \wedge r2(\mathbf{x_k},\mathbf{y_k})$.

BCQ = HOM

View query *Q* as a relational structure

Universe: Variables of the query
Relations: sets of query-atoms for each database relation.

The database *D* is itself a relational structure.

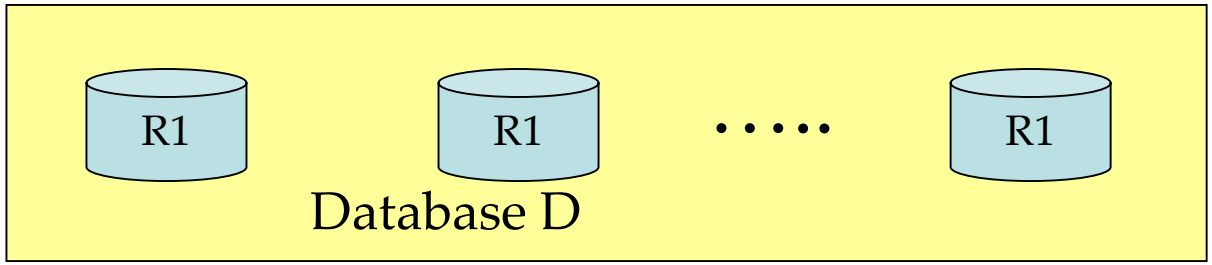
By the definition of the semantics of FO queries, the Boolean conjunctive query evaluates to true iff there is a homomorphism h: Q→D.

$$DB=Q \quad \text{iff} \quad HOM(Q,D)\neq\emptyset$$

Vive-versa, every HOM instance can be reformulated as a Boolean conjunctive query.

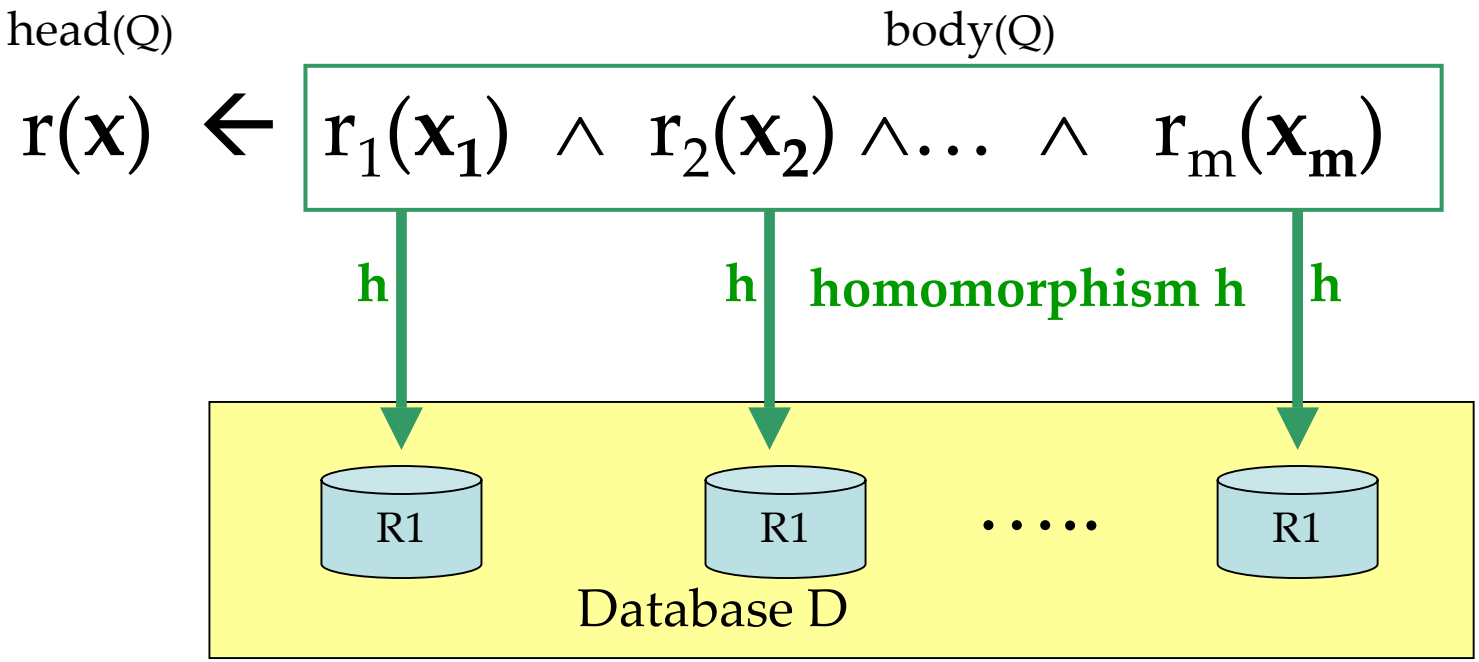
Non-Boolean CQs

$$Q: \quad r(x) \leftarrow r_1(\mathbf{x_1}) \wedge r_2(\mathbf{x_2}) \wedge \ldots \wedge r_m(\mathbf{x_m})$$



How do we formally define Q(D)?

Non-Boolean CQs



$$Q(D) = \{h(head(Q)) \mid h \in HOM(body(Q), D)\}$$

Problems Equivalent to BCQ

- Conjunctive Query Containment

$$Q_1 \subseteq Q_2 \;\Leftrightarrow\; \forall db \;\; Q_1(db) \subseteq Q_2(db)$$

- Query of Tuple Problem

$$t \in Q(db) \quad ?$$

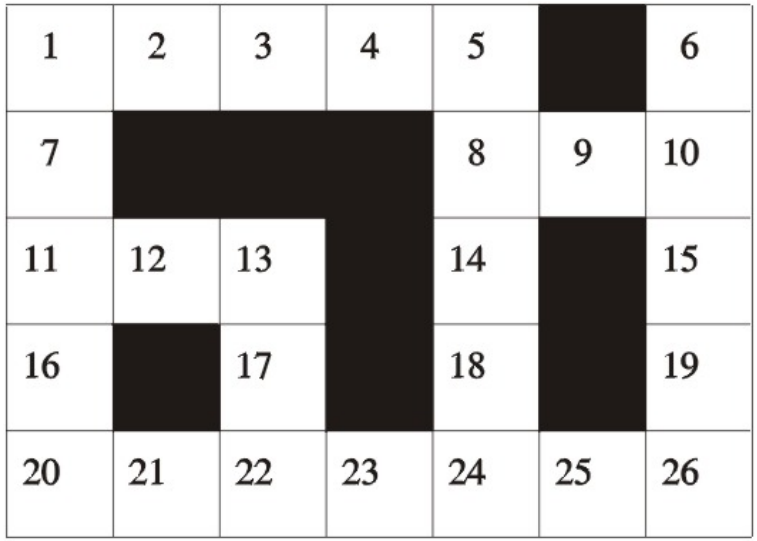
- Constraint Satisfaction in AI

$$\text{CSP} \;\approx\; \text{BCQ}$$

- Clause Subsumption in Theorem Proving

$$\exists \vartheta \text{ s.t. } C\vartheta \subseteq D \quad ?$$

Combinatorial Crossword Puzzle



All known general solution algorithms rely on backtracking

We are looking for a homomorphism f:[1..26]→[A..Z] that translates 1H ={<1,2,3,4,5>} into 1h, and so on.

1h:

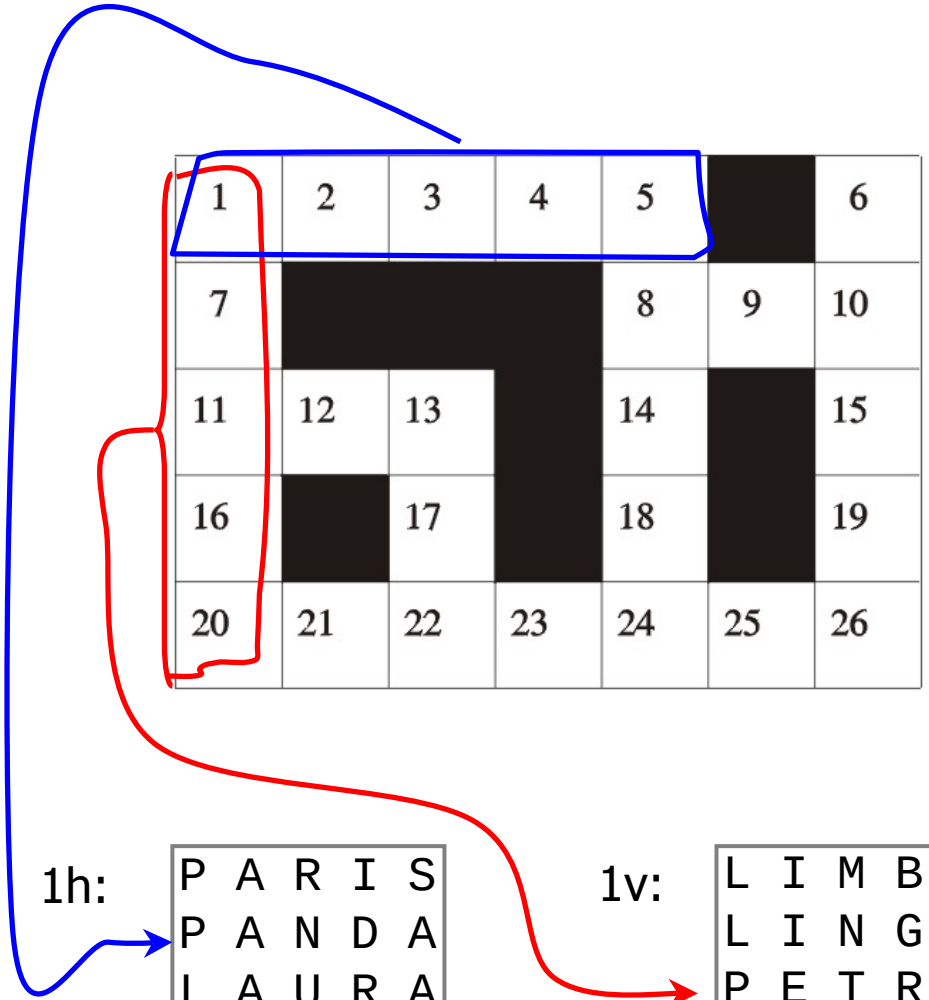
P	A	R	I	S
P	A	N	D	A
L	A	U	R	A
A	N	I	T	A

1v:

L	I	M	B	O
L	I	N	G	O
P	E	T	R	A
P	A	M	P	A
P	E	T	E	R

and so on

Combinatorial Crossword Puzzle



All known general solution algorithms rely on backtracking

We are looking for a homomorphism f:[1..26]→[A..Z] that translates 1H ={<1,2,3,4,5>} into 1h, and so on.

and so on

Containment of CQ's

- Given conjunctive queries Q1 and Q2, we say that Q1 is **contained** in Q2, denoted as Q1 $\subseteq$ Q2, iff for **all** databases D, Q1(D) $\subseteq$ Q2(D).

- Example:
  - Q1: p(X,Y) $\leftarrow$ arc(X,Z) , arc(Z,Y)
  - Q2: p(X,Y) $\leftarrow$ arc(X,Z) , arc(W,Y)

- DB is a graph; Q1 produces paths of length 2, Q2 produces pairs of nodes with an arc out and in, respectively.

Why CQ Containment?

Containment tests used for equivalence tests.

$$Q1 = Q2 \quad \text{iff} \quad Q1 \subseteq Q2 \text{ and } Q2 \subseteq Q1$$

- Equivalence is fundamental to query optimization

- Answering queries using *views*.

- Data integration: collecting data from different sources (e.g. on Internet), answering your specific query $\rightarrow$ need to know whether the sources can (fully, or partially) answer your query.

Query Homomorphisms

(aka Containment Mappings)

A mapping from the **variables** of CQ Q2 to the **variables** of CQ Q1, such that:

- The head of Q2 is mapped to the head of Q1.
- Each subgoal (= body atom) of Q2 is mapped to some subgoal of Q1 with the same predicate.

Homomorphism Theorem:

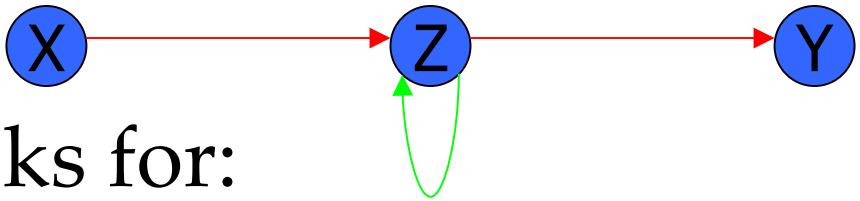
There is a homomorphism from Q2 to Q1 if and only if Q1 $\subseteq$ Q2.

Example

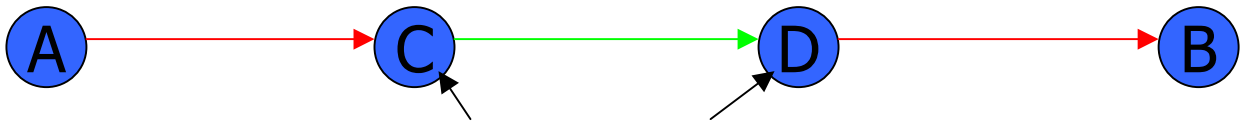
Q1: p(X,Y) ← r(X,Z), g(Z,Z), r(Z,Y)

Q2: p(A,B) ← r(A,C), g(C,D), r(D,B)

Q1 looks for:



Q2 looks for:

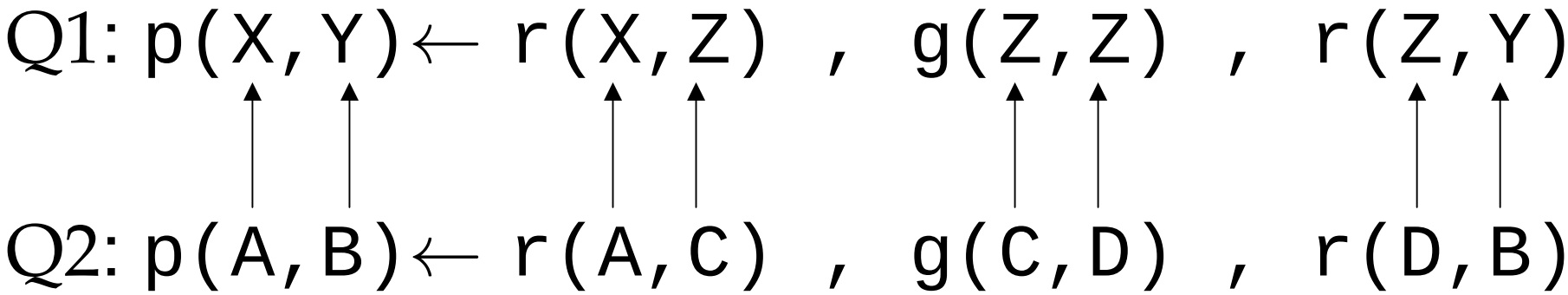


Since C=D is possible,
expect Q1 ⊆ Q2.

Example --- Continued

- Whenever there is a path from X to Y, it must be that X has an arc out, and Y an arc in.
- Thus, every fact (tuple) produced by Q1 is also produced by Q2.
- That is, Q1 $\subseteq$ Q2.

Example --- Continued



Homomorphism: h(A)=X; h(B)=Y; h(C)=h(D)=Z.

Example ---Concluded

Q1: `p(X,Y)` ← `r(X,Z) , g(Z,Z) , r(Z,Y)`

Q2: `p(A,B)` ← `r(A,C) , g(C,D) , r(D,B)`

- No containment mapping from Q1 to Q2.
  - g(Z,Z) can only be mapped to g(C,D).
    - No other *g* subgoals in Q2.
  - But then Z must map to both C and D --- impossible.
- Thus, Q1 properly contained in Q2.

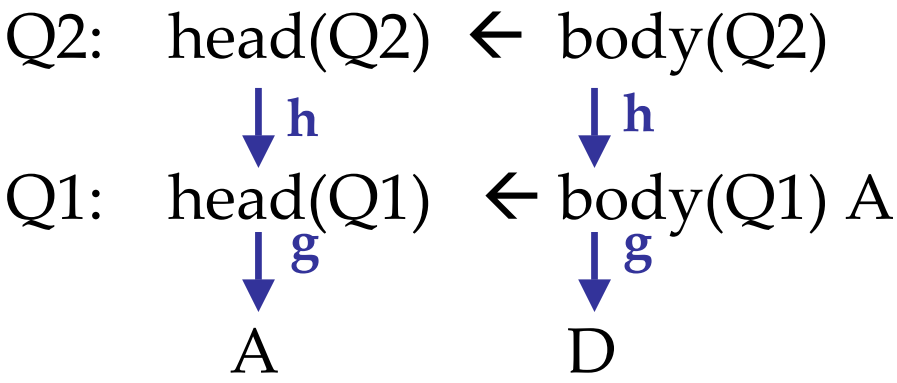
Proof of the Homomorphism Theorem

Hom(Q2,Q1)≠∅ iff Q1 ⊆ Q2.

a) ∃h∈HOM(Q2,Q1) ➔ Q1 ⊆ Q2 (this is the *only-if* direction)

Let D be a database and A∈Q1(D).

Then ∃ g∈HOM(body(Q1),D), such that A=g(head(Q1)).



But then g∘h is a homomorphism in HOM(body(Q2,D)),

and g∘h(head(Q2))= g(h(head(Q2))=g(head(Q1)) =A

Thus A ∈ Q2(D) .

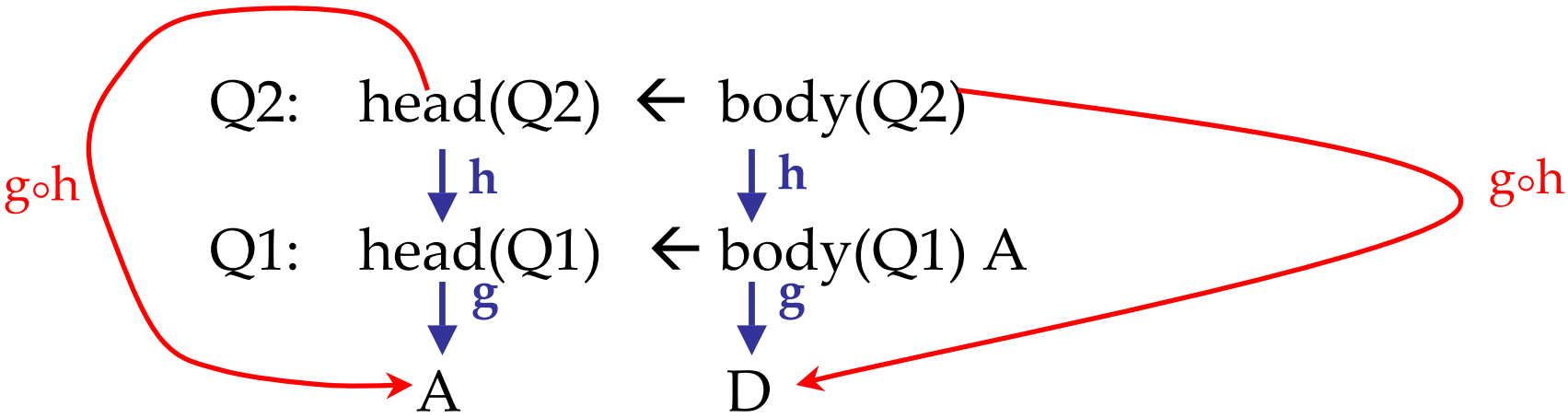
In summary, ∀A∈Q1(D), A ∈Q2(D), thus Q1 ⊆ Q2.

Proof of the Homomorphism Theorem Hom(Q2,Q1)≠∅ iff Q1 ⊆ Q2.

a) ∃h∈HOM(Q2,Q1) ➔ Q1 ⊆ Q2 (this is the *only-if* direction)

Let D be a database and A∈Q1(D).

Then ∃ g∈HOM(body(Q1),D), such that A=g(head(Q1)).



But then g∘h is a homomorphism in HOM(body(Q2,D)),

and g∘h(head(Q2))= g(h(head(Q2))=g(head(Q1)) =A

Thus A ∈ Q2(D) .

In summary, ∀A∈Q1(D), A ∈Q2(D), thus Q1 ⊆ Q2.

b) $Q1 \subseteq Q2 \rightarrow \exists h \in HOM(Q2,Q1)$ (this is the *if* direction)

Q2: p(A,B)← r(A,C) , g(C,D) , r(D,B)
Q1: p(X,Y)← r(X,Z) , g(Z,Z) , r(Z,Y)

Canonical Database Can_{Q1}:

Generate a new constant **f(U)=U** for each variable **U**

Can_{Q1} = f(atoms(body(Q))) = {**r(X,Z), g(Z,Z), r(Z,Y)**}

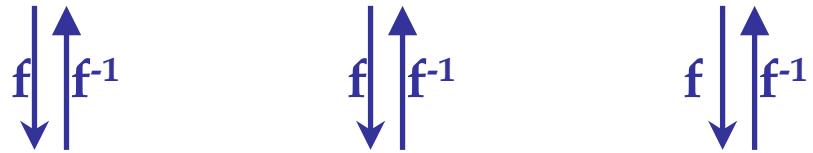
$dom(Can_{Q1})$ = {**X,Y,Z**}

Note: The mapping f:var(Q1)→ {new constants} is an **isomorphism** where

$f(atoms(body(Q1))) = Can_{Q1}$ and $f^{-1}(Can_{Q1}) = atoms(body(Q1))$

Q2: **p(A,B)← r(A,C) , g(C,D) , r(D,B)**

Q1: **p(X,Y)← r(X,Z) , g(Z,Z) , r(Z,Y)**



{r(X,Z) , g(Z,Z) , r(Z,Y)}= Can_{Q1}:

Since f maps body(Q) into Can_{Q1},

A new fact A= f(head(Q1)) is generated

Q2: **p(A,B)← r(A,C) , g(C,D) , r(D,B)**

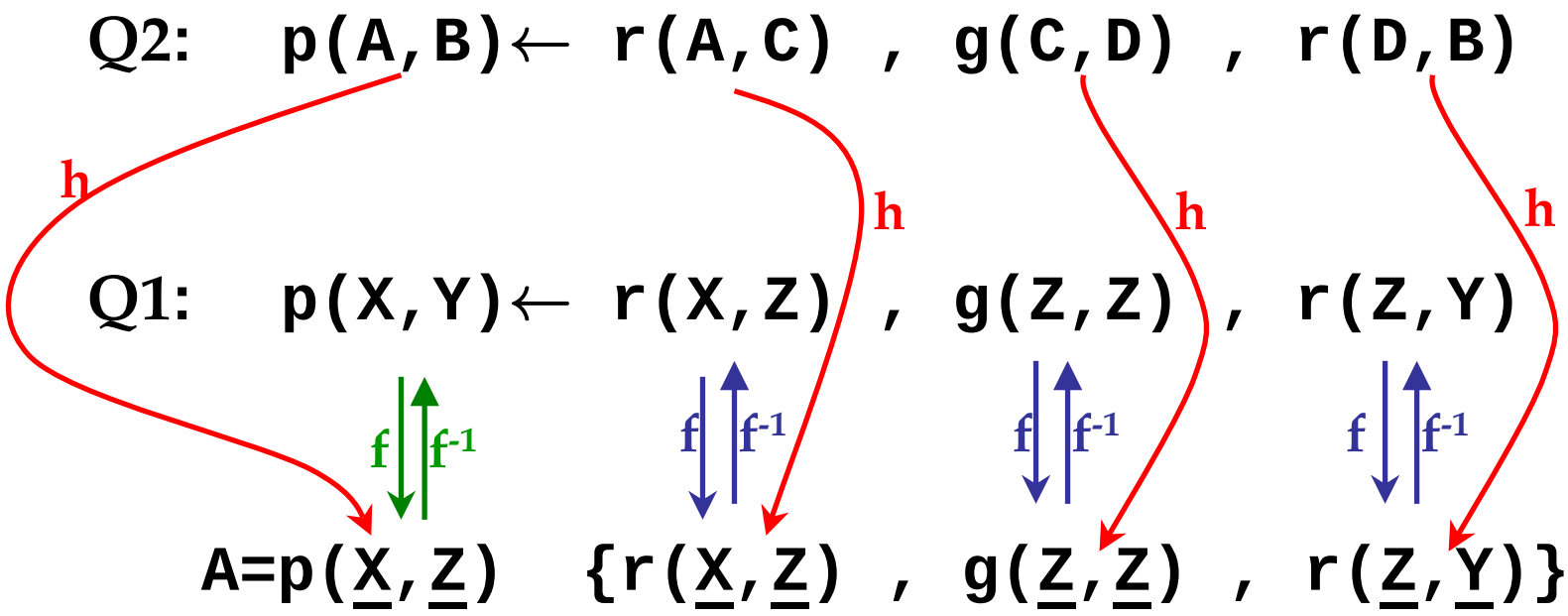
Q1: **p(X,Y)← r(X,Z) , g(Z,Z) , r(Z,Y)**

A=p(X,Z) {r(X,Z) , g(Z,Z) , r(Z,Y)}= Can_{Q1}:

A= f(head(Q1))∈Q1(Can_{Q1}).

Since Q1 ⊆ Q2, A ∈Q2(Can_{Q1}).

Therefore ∃ h ∈Hom(Q2, Can_{Q1} ∪{A}).



A= f(head(Q1))∈Q1(Can_{Q1}).

Since Q1 ⊆ Q2, A ∈Q2(Can_{Q1}).

Therefore ∃ h ∈Hom(Q2, Can_{Q1} ∪{A}).

Then f^{-1}∘h∈ HOM(Q2,Q1) is the desired homomorphism.

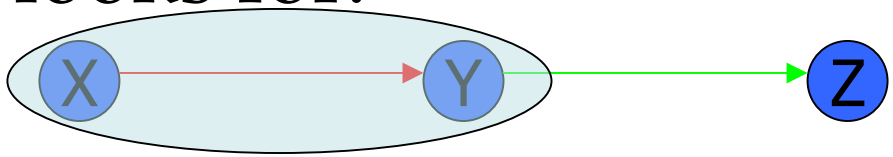
QED

Another Example

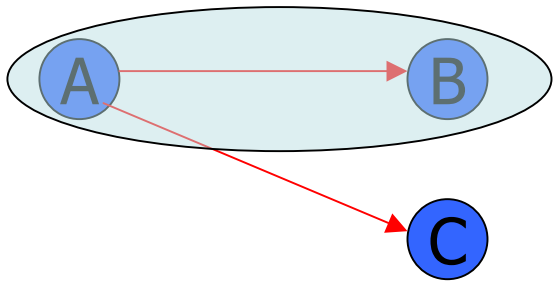
Q1: `p(X,Y)` ← `r(X,Y) , g(Y,Z)`

Q2: `p(A,B)` ← `r(A,B) , r(A,C)`

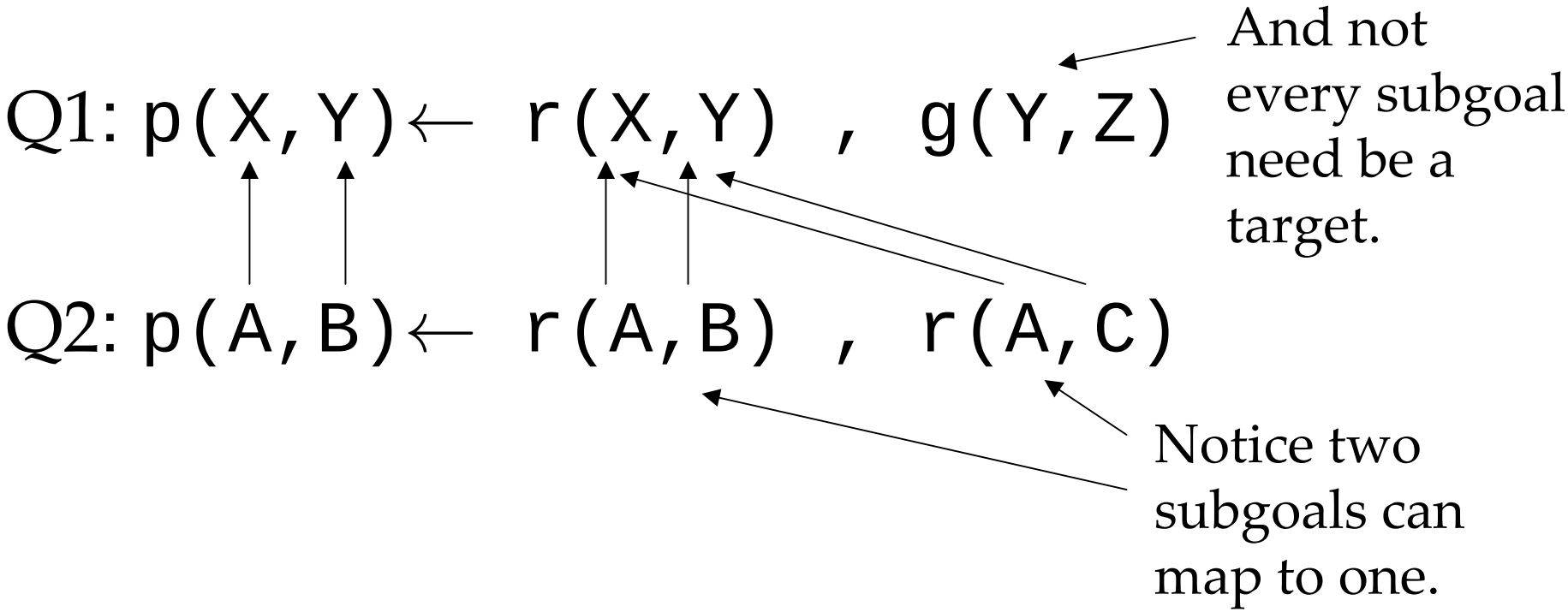
Q1 looks for:



Q2 looks for:



Example --- Continued



Containment mapping: m(A)=X; m(B)=m(C)=Y.

Example ---Concluded

Q1: `p(X,Y):-r(X,Y) & g(Y,Z)`
Q2: `p(A,B):-r(A,B) & r(A,C)`

- No containment mapping from Q1 to Q2.
  - g(Y,Z) cannot map anywhere, since there is no *g* subgoal in Q2.

- Thus, Q1 properly contained in Q2.

Query Containment $\Leftrightarrow$ BCQ

Let $Can^{+}_{Q1} = Can^{+}_{Q1} \cup f(head(Q1))$

Let $Q2^{+} = Q2 \cup head(Q2)$ ($Q2^{+}$ is a Boolen query)

Then,

$Q1 \subseteq Q2$ iff $Can^{+}_{Q1} \models Q2^{+}$

i.e., iff $Q2^{+}$ evaluates to true over Can^{+}_{Q1}

Therefore, all complexity results for BCQ also hold for the query containment problem.

(Boolean) Conjunctive Query evaluation

- {enc(I), enc(φ) | I $\models$ φ}
  - if φ is a conjunctive query, combined complexity?
  - $n = |I|$, $m = |\varphi|$
  - n^m possible assignments for vars(φ)
  - Given an assignment, the testing is in polynomial time
  - In NP

Conjunctive Query evaluation

THEOREM: BCQ is NP Complete (combined complexity)

Proof: Follows from the NP-completeness of HOM

Exercise: As an alternative proof of NP-hardness, give a reduction from 3SAT.

COROLLARY: Conjunctive Query Containment is NP complete

COROLLARY: Conjunctive Query Equivalence is NP complete

Note: The latter problems are undecidable for FO

CQ minimization

- A conjunctive query Q is *minimal*, if it contains minimal number of subgoals –
  (subgoal = body atom)

  This means: whenewer we drop an atom from the body of Q, we get a query $Q' \not\equiv Q$

- The task of the *CQ minimization*: minimize the number of subgoals of a given CQ.

Cores

BCQ:

P←p(X,Y), p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d)

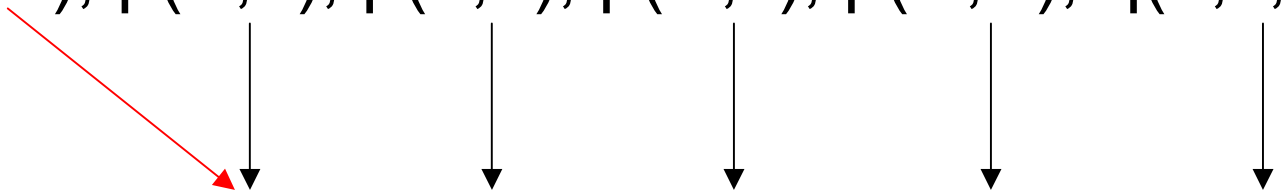
Logical meaning

∃ X, Y, U, V:
p(X,Y) & p(X,b) & p(a,b) & p(U,c) & p(U,V) & q(a,c,d)

Cores

endomorphism h: {Y → b}

B= { p(X,Y), p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d)}



{ p(X,Y), p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) }

Cores

endomorphism h: {Y → b}

B = { p(X,Y), p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) }

{ p(X,Y), p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) }

REDUNDANT!

$\exists X,Y\ p(X,Y) \ \&\ P(X,b)$

$\Uparrow\Downarrow$

$\exists X\ p(X,b)$

Cores

endomorphism h: {Y → b}

B = { p(X,Y), p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) }

{ ~~p(X,Y)~~, p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) }

REDUNDANT!

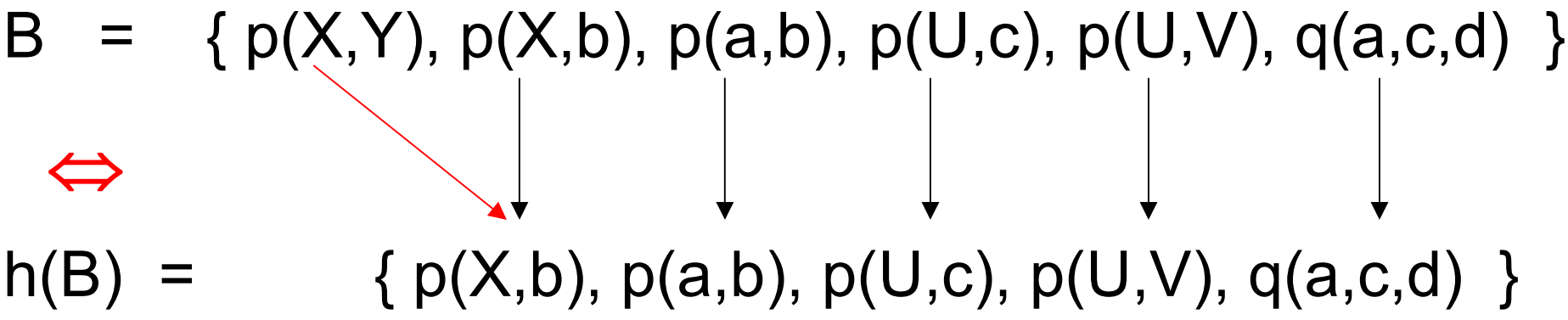
$\exists X,Y\ p(X,Y)$

$\Uparrow$

$\exists X\ p(X,b)$

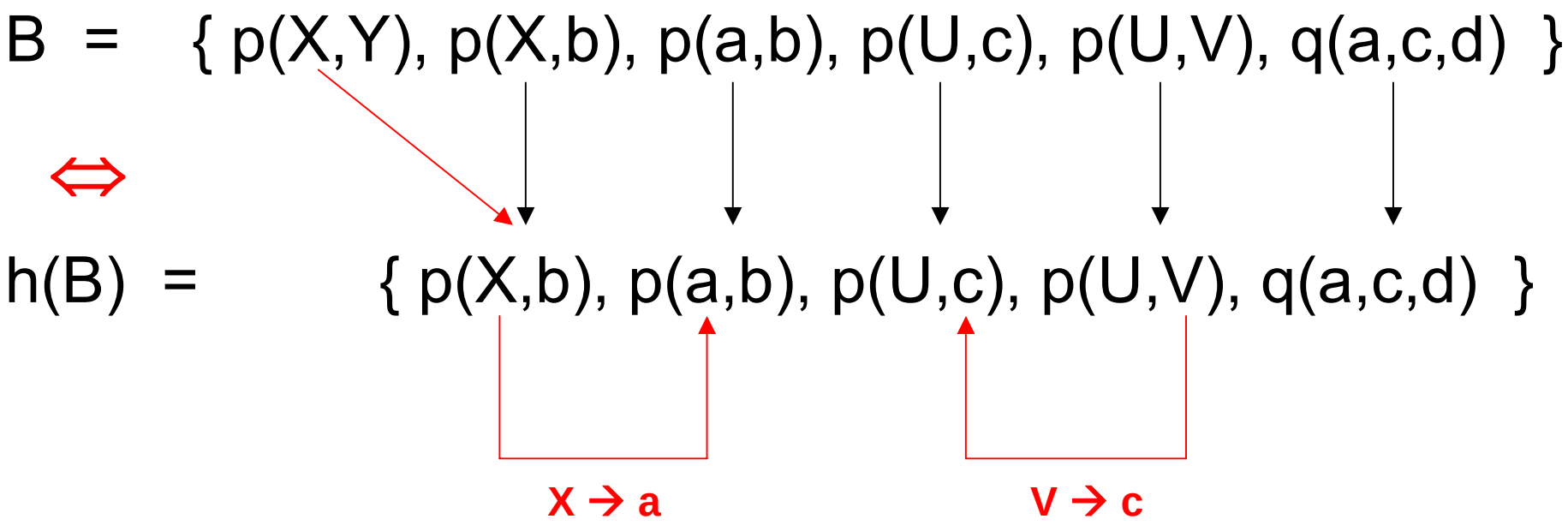
Cores

endomorphism h: {Y → b}



Cores

endomorphism h: {Y → b}



h(B) can be further reduced by endomorphism g: {X → a, V→ c}

Cores

endomorphism h: {Y → b}

B = { p(X,Y), p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) }

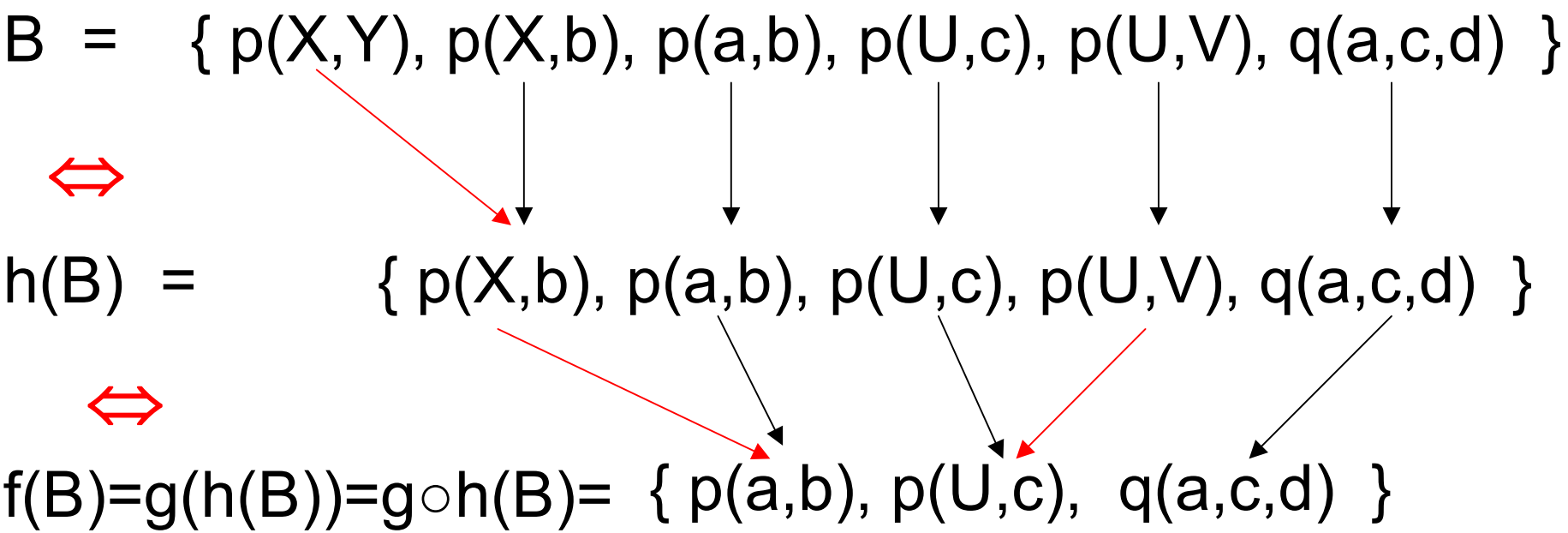
⇔

h(B) = { p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) }

{ p(a,b), p(U,c), q(a,c,d) }

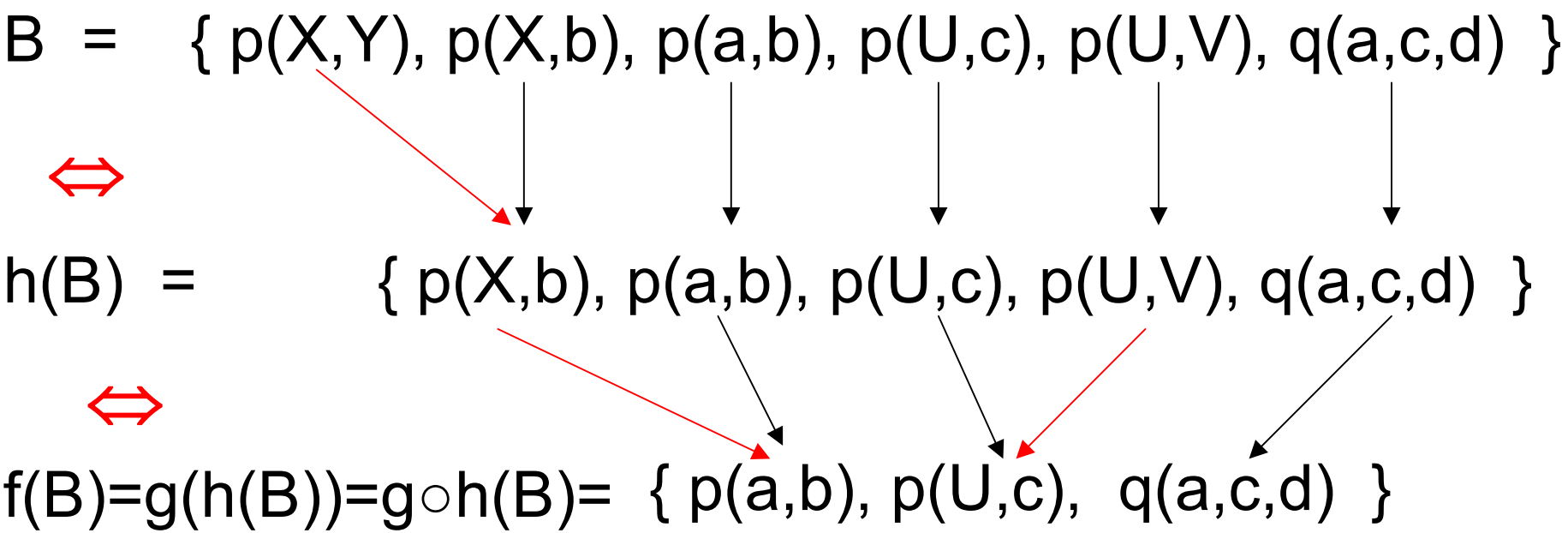
h(B) can be further reduced by endomorphism g: {X → a, V→ c}

Cores



endomorphism f: {X → a, Y→b, V→ c}

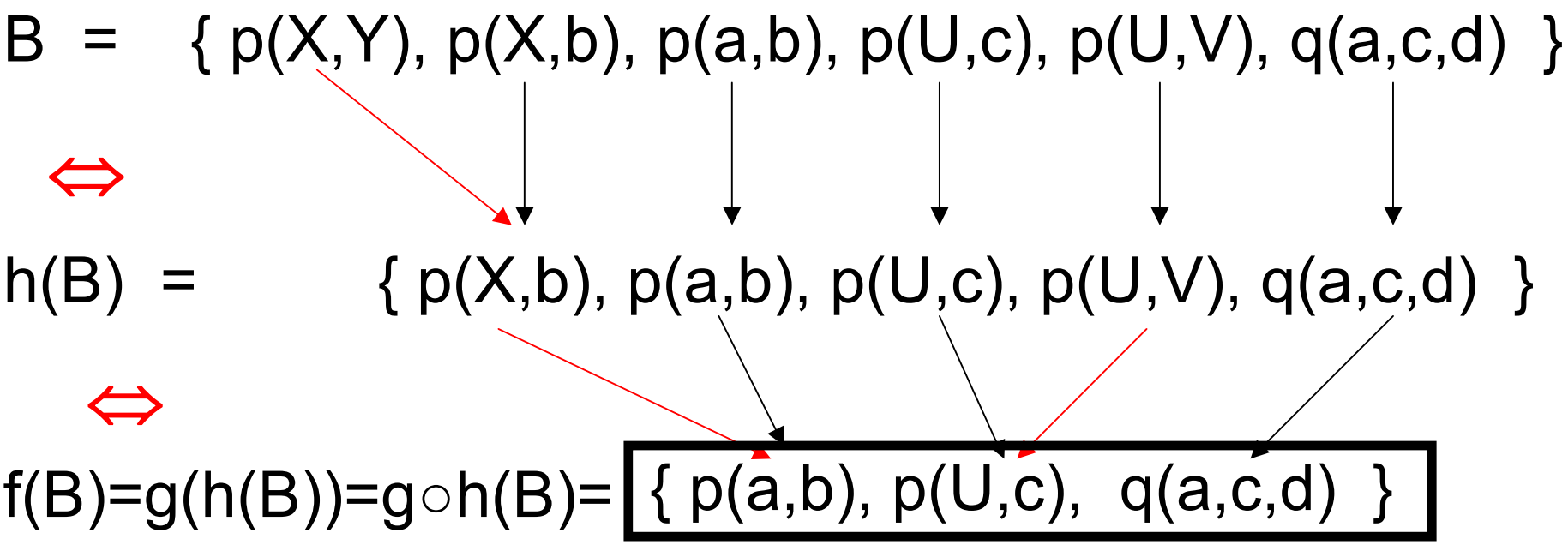
Cores



no refinement by endomorphisms possible !

endomorphism f: {X → a, Y→b, V→ c}

Cores



Core(I)

unique up to variable-renaming!

endomorphism f: {X → a, Y→b, V→ c}

CQ minimization

$$\begin{aligned} \text{Ans}(X,Y,Z) \leftarrow\ & R(X_2,Y_1,Z), \\ & R(X,Y_1,Z_1), \\ & R(X_1,Y,Z_1), \\ & R(X,Y_2,Z_2), \\ & R(X_2,Y_2,Z) \end{aligned}$$

Observations

- The goal: minimize the number of rows in the tableau.
  - The head atom cannot be eliminated!
  - Can we just check the subgoals one by one? (if one subgoal can not be removed at contain moment, could it be removed at a later moment?)
  - Does the sequence of the elimination count? (or, is the minimized query unique?)

Minimization algorithm

- Can we just check the rows one by one?

Yes. Take the subgoal u_i, in a CQ Q1, check whether there is a homomorphism h, such that $h(Q1) \subseteq (Q1 - u_i)$. If there exists such a homomorphism, then u_i can be removed. Otherwise, u_i can not be removed forever. Because Q1 is getting smaller.

- Does the sequence of the elimination count? (or, is the minimized query unique?)

-- homework

THEOREM: The Core is Unique up to isomorphism

Isomorphism here means: bijective variable renaming

PROOF: Homework.

CQ Containment vs. BCQ evaluation

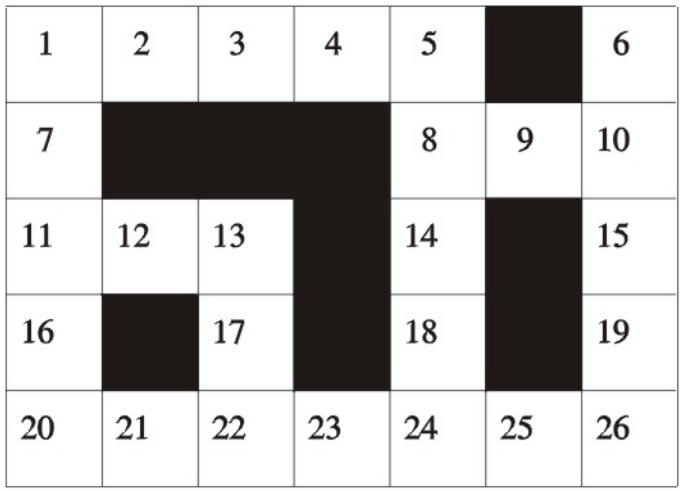
- Both are NP-complete, but:
  - the potential number of mappings is different: n^m: where n is
    - the size of Q1 for CQ containment (small, and normally does not hurt)
    - is the size of a database for BCQ evaluation (could be huge)

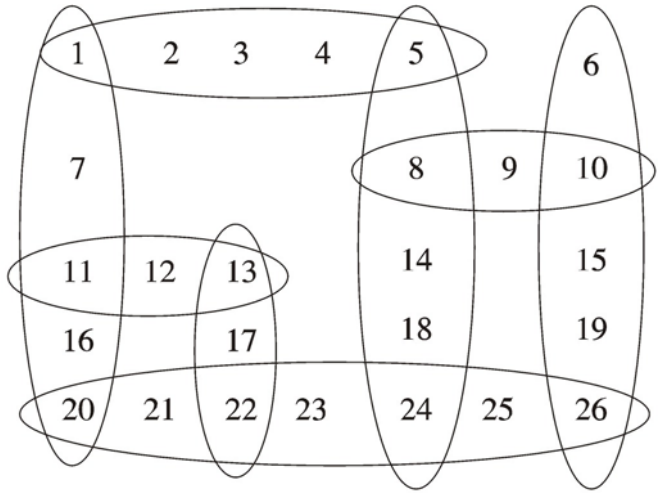
- Highly desirable: Identify fragments of BCQ, whose evaluation problem is tractable (in P).

BCQ evaluation

- Intuition: the problem with cycles in the query:
  - Consider the queries:
    - Q1: ans $\leftarrow$ e(A,B), e(B,C), e(C,D), e(D,A)
    - Q2: ans $\leftarrow$ e(A,B), e(B,C), e(C,D), e(D,E)
- In processing Q1, maintain intermediate join results e'(A,B,C,D) $\bowtie$ e(D,A)
- In processing Q2, all the join operations are only between two predicates (locally)

Hypergraph of Crossword Puzzle

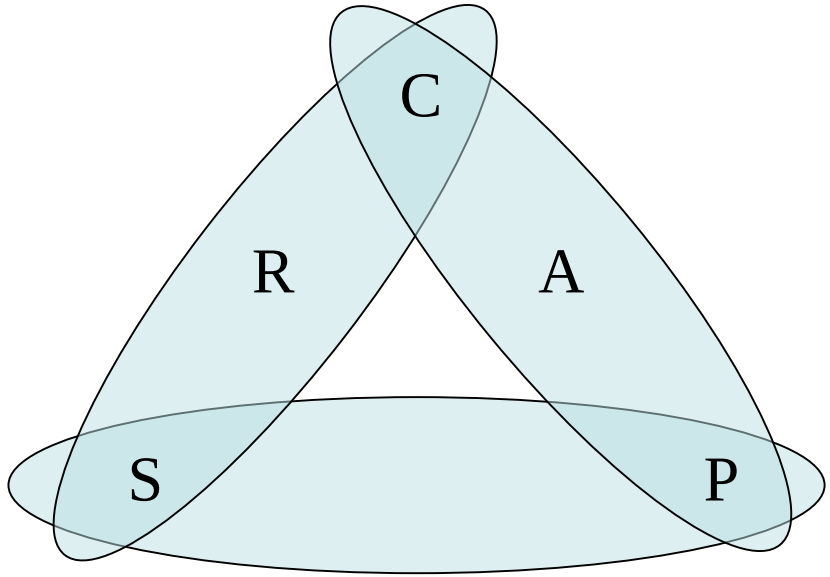




Theorem: The evil is in the cycles.

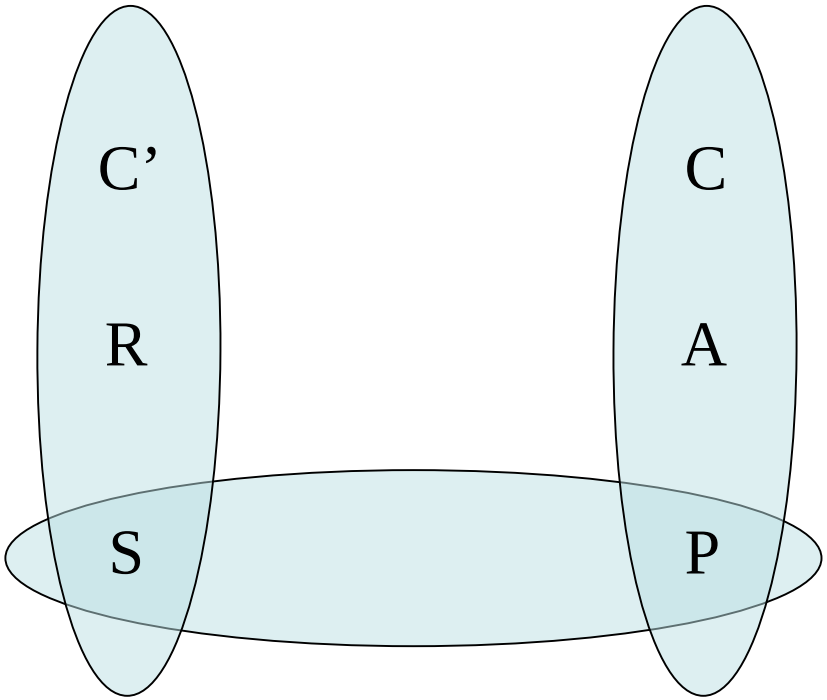
Queries, CSPs, and Hypergraphs

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



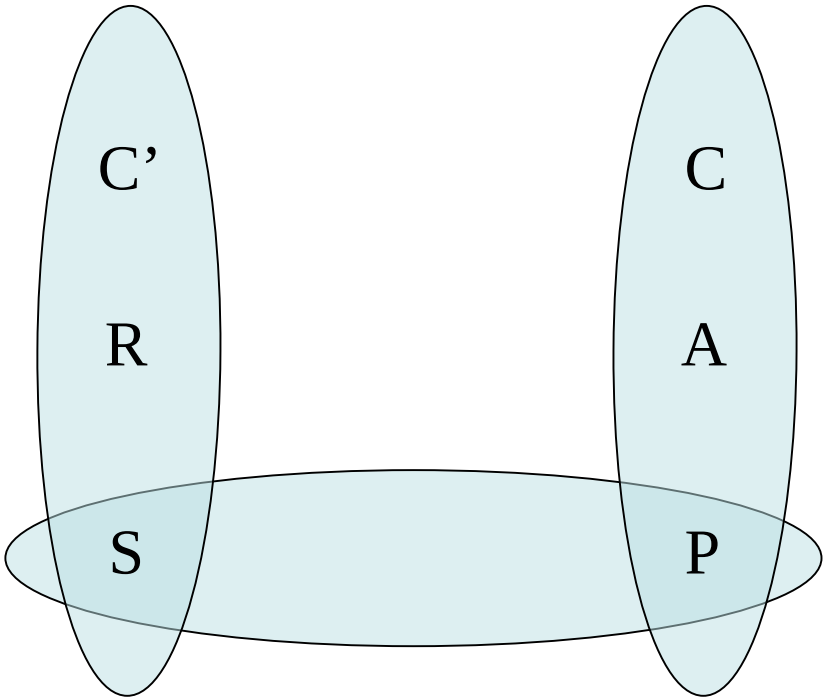
Acyclic CSP

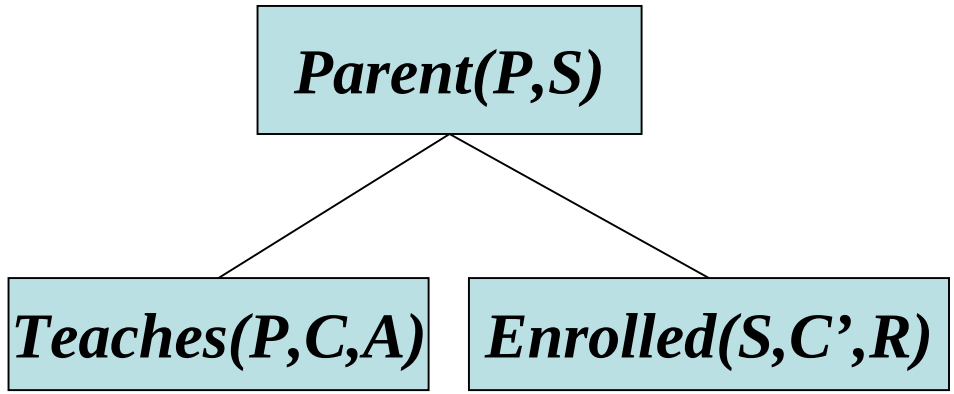
$$ans \leftarrow Enrolled(S,C',R) \wedge Teaches(P,C,A) \wedge Parent(P,S)$$



Acyclic queries or CSPs

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





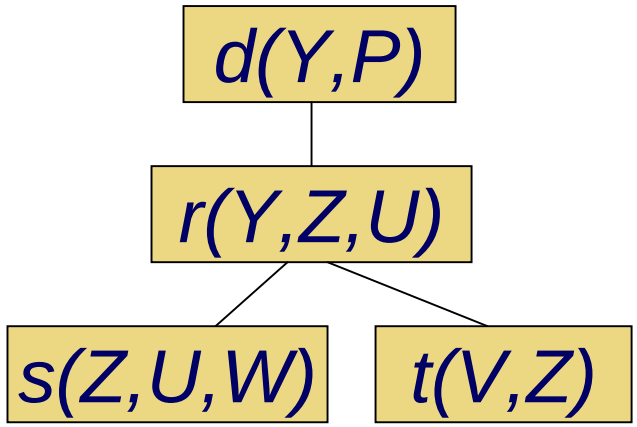
Join Tree

Join Tree

Definition: A Join tree of a BCQ is an organisation of its atoms into a tree T such that for each variable X, the nodes in which X occurs span a connected subtree of T. (Connectedness Condition)

Definition: A CQ is a *tree query* iff it has a join tree.

Example: Q= R(Y,Z,U) & s(Z,U,W) & d(Y,P) & t(V,Z)



Acyclicity

GYO-Reduction (Graham, Yu, and Ozsoyoglu):

Input: Hypergraph H

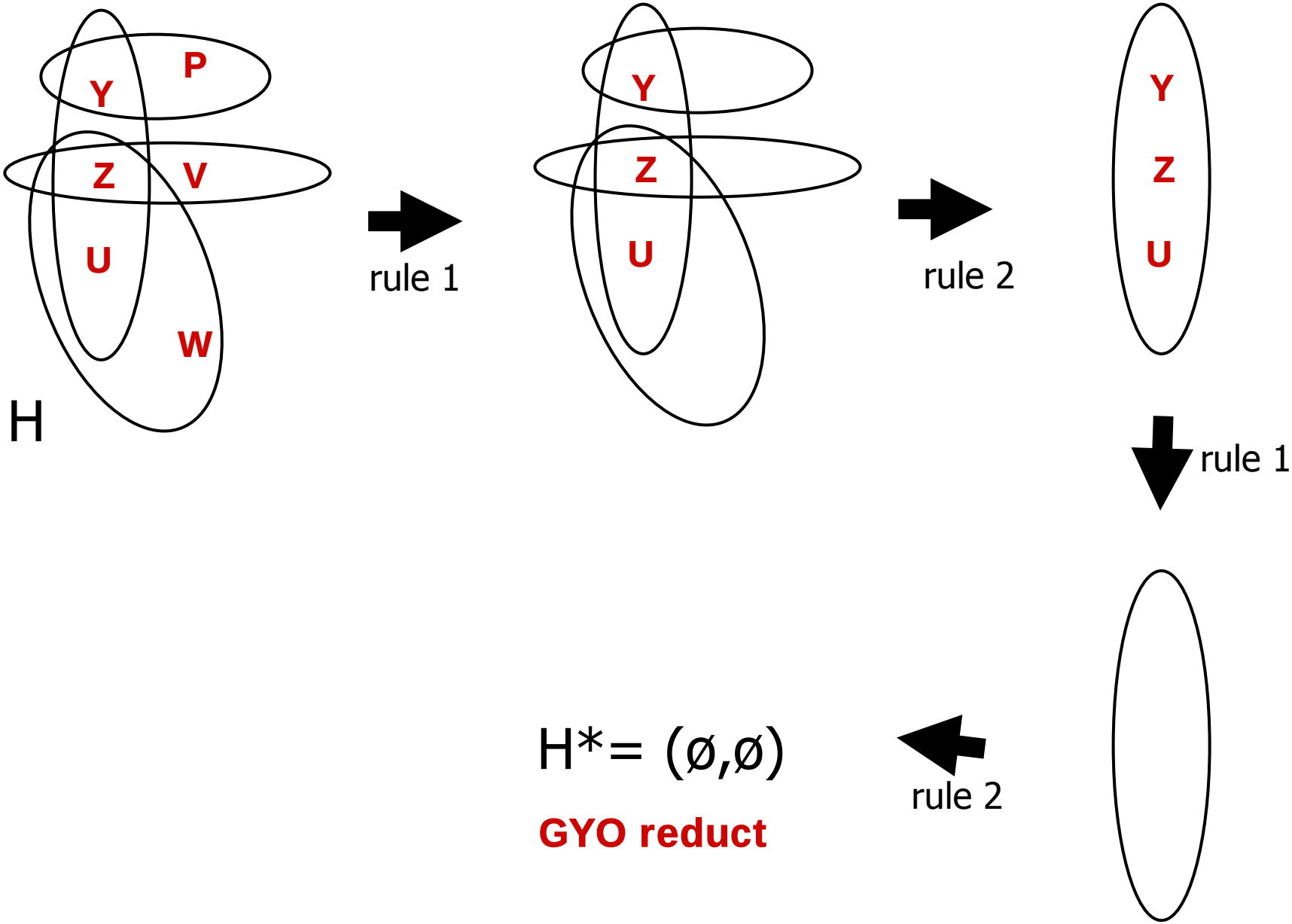
Output: GYO-reduct of H

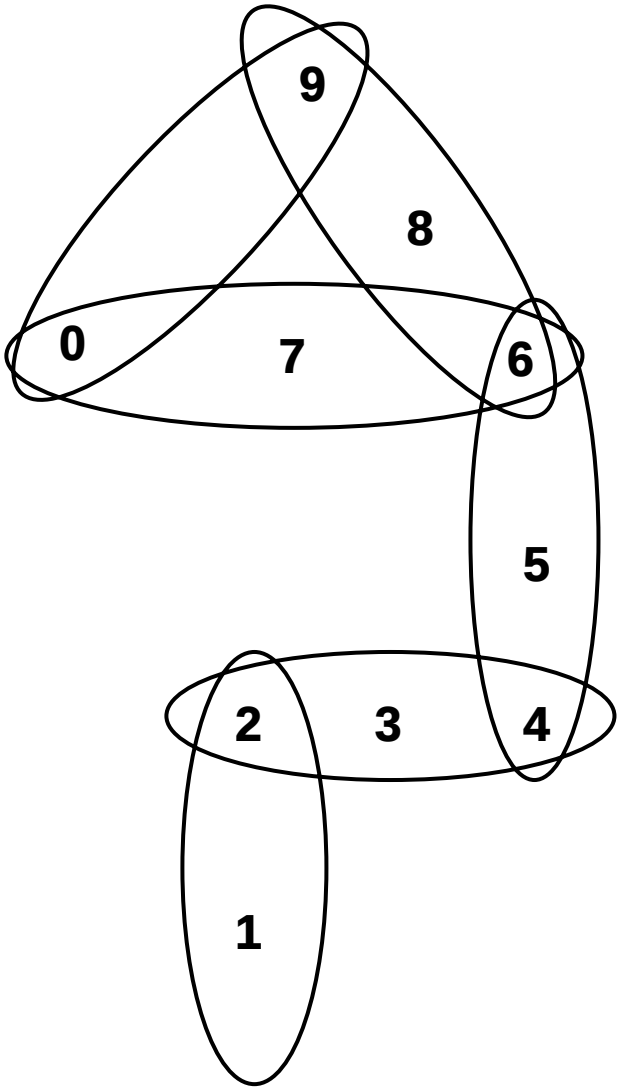
Apply the following rules as long as possible:

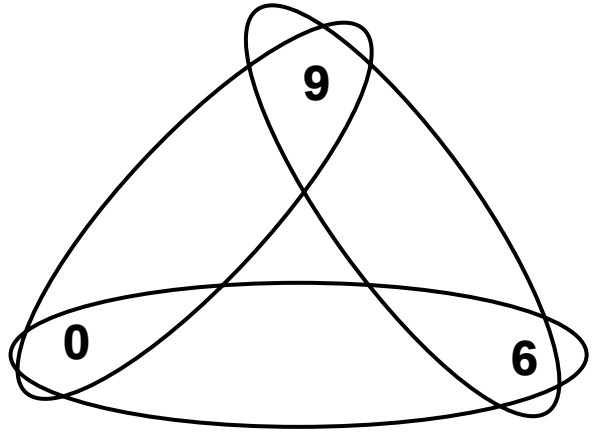
- Eliminate vertices that are contained in at most one hyperedge (ear vertices)
- Eliminate hyperedges that are empty or contained in other hyperedges

Definition: H is acyclic iff GYO-reduct(H)=(∅,∅).

Definition: Q is acyclic iff its hypergraph H(Q) is acyclic.


P
Y
Z
V
U
W
H
rule 1
Y
Z
U
rule 2
Y
Z
U
rule 1
H*= (ø,ø)
GYO reduct
rule 2





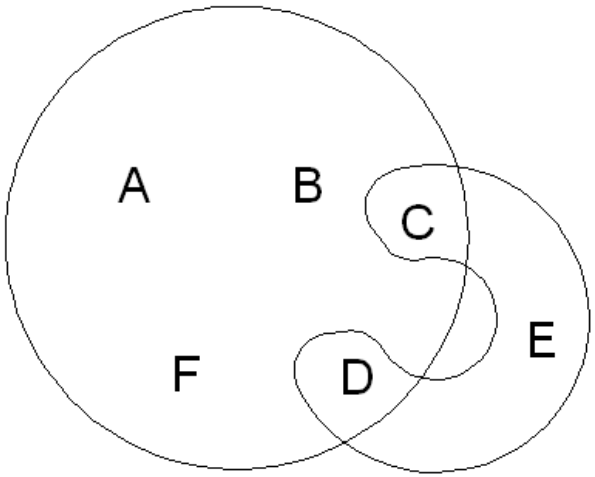
GYO-irreducible

Aternative Procedure

Definition: An *ear edge* (short: *ear)* of a hypergraph (V,E) is a hyperedge $e \in E$ such that one of the following conditions holds:

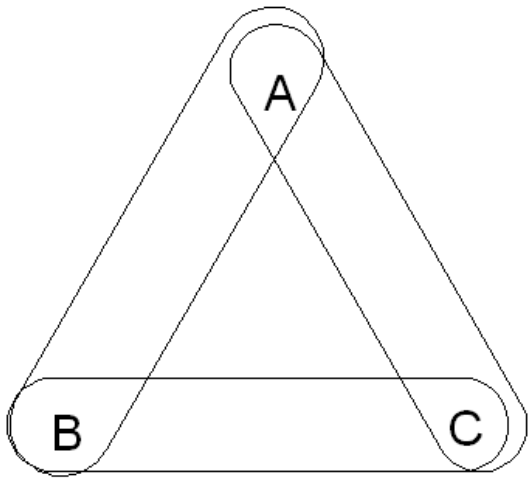
(1) There is a witness $e' \in E$, such that $e' \neq e$ and each vertex from e is either only in e or in e', or

(2) e has no intersection with any other hyperedge.

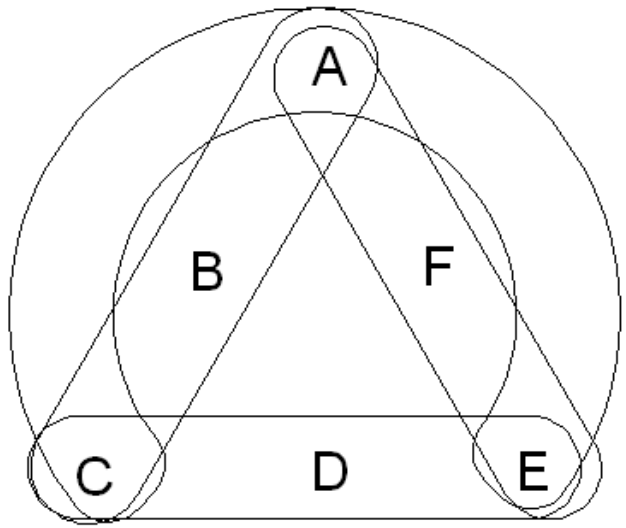


(C,D,E) is an ear

Hypergraph of BCQs



Ears?



Ears?

GYO′ algorithm – removing ears

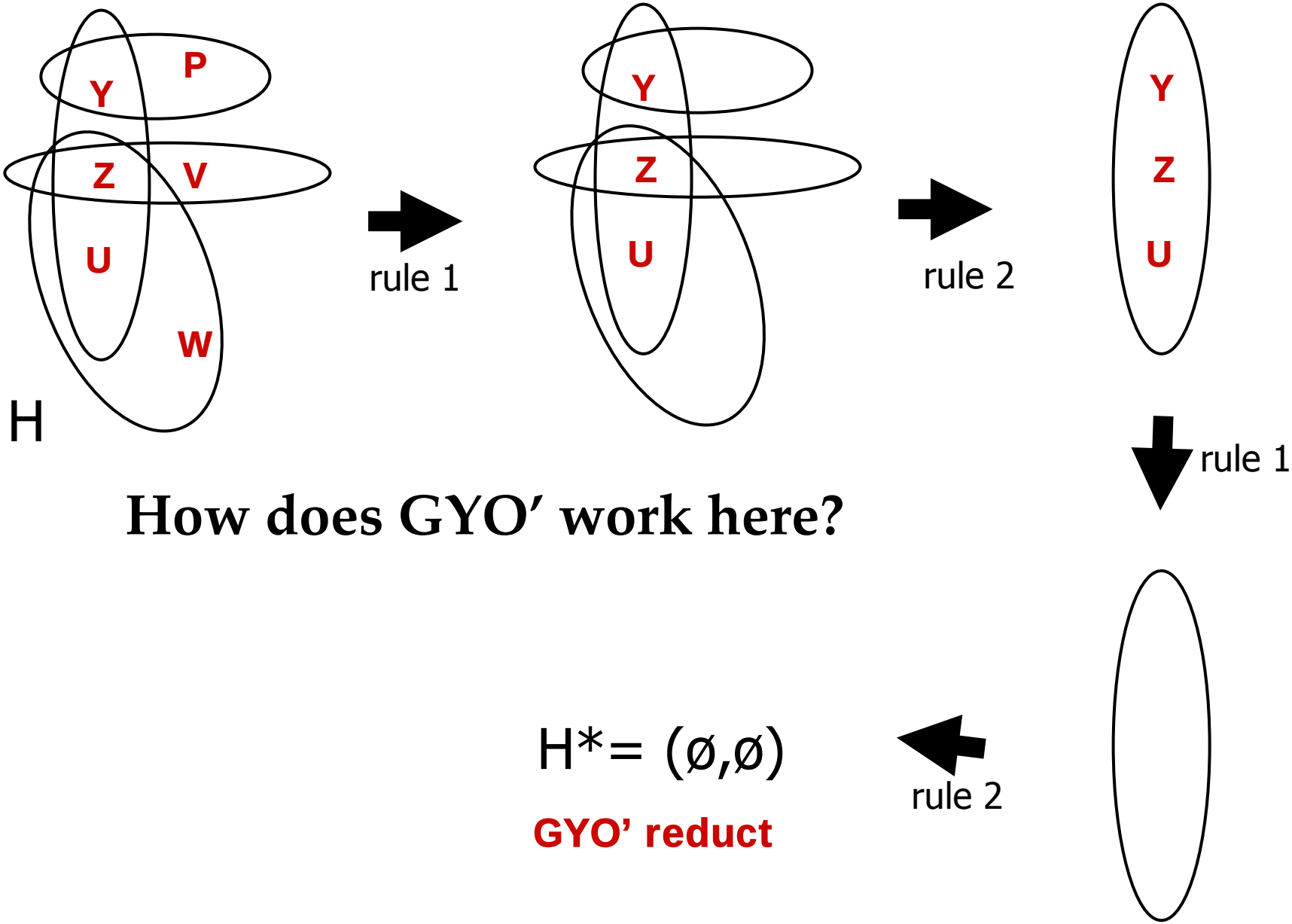
Algorithm (GYO′ Algorithm)

Input: Hypergraph $H = (V , E)$
Output: GYO′-reduct of H

Do until E has no ears:

Choose any ear e of E.

Set $H := (V' , E - \{e\})$, where V' =vertices$(E - \{e\})$.



How does GYO′ work here?

GYO' reduct

Acyclicity Theorems

Lemma: A hypergraph G is a GYO-reduct of H iff it is a GYO'-reduct of H.

THEOREM: A hypergraph is acyclic iff its only GYO'-reduct is (∅,∅).

THEOREM: A query has a join tree iff it is acyclic.

Easy algorithms exist fort checking whether A query is acyclic and for computing a join tree.

Summary

- The GYO (and GYO') algorithm takes linear time
- The hypergraph is acyclic iff the output of GYO (or equivalently, GYO') algorithm is empty
- A BCQ is acyclic if its hypergraph is acyclic
- If a BCQ is acyclic, then there is a join tree, obtained from the sequence of the ear-removing of the GYO algorithm.

It remains to show: The query evaluation of acyclic BCQ is in polynomial time

The acyclic query evaluation

- ans $\leftarrow$ d(Y,P), t(V,Z), s(Y,Z,U), s(Z,U,W)
- The polynomial algorithm of acyclic BCQ evaluation:
  - Bottom-up
  - Semi-join

Boolean conjunctive queries
CSPs
The Homomorphism Problem….

….polynomialloy solvable for on instances having acyclic hypergraphs

Semijoin Operation

Employee

Employee Name	Employee Salary	Department Name
Ackman	5000	Car
Baker	4500	Car
Carson	4800	Toy
Davis	5100	Toy

Department

Department Name	Department Budget
Car	450000000

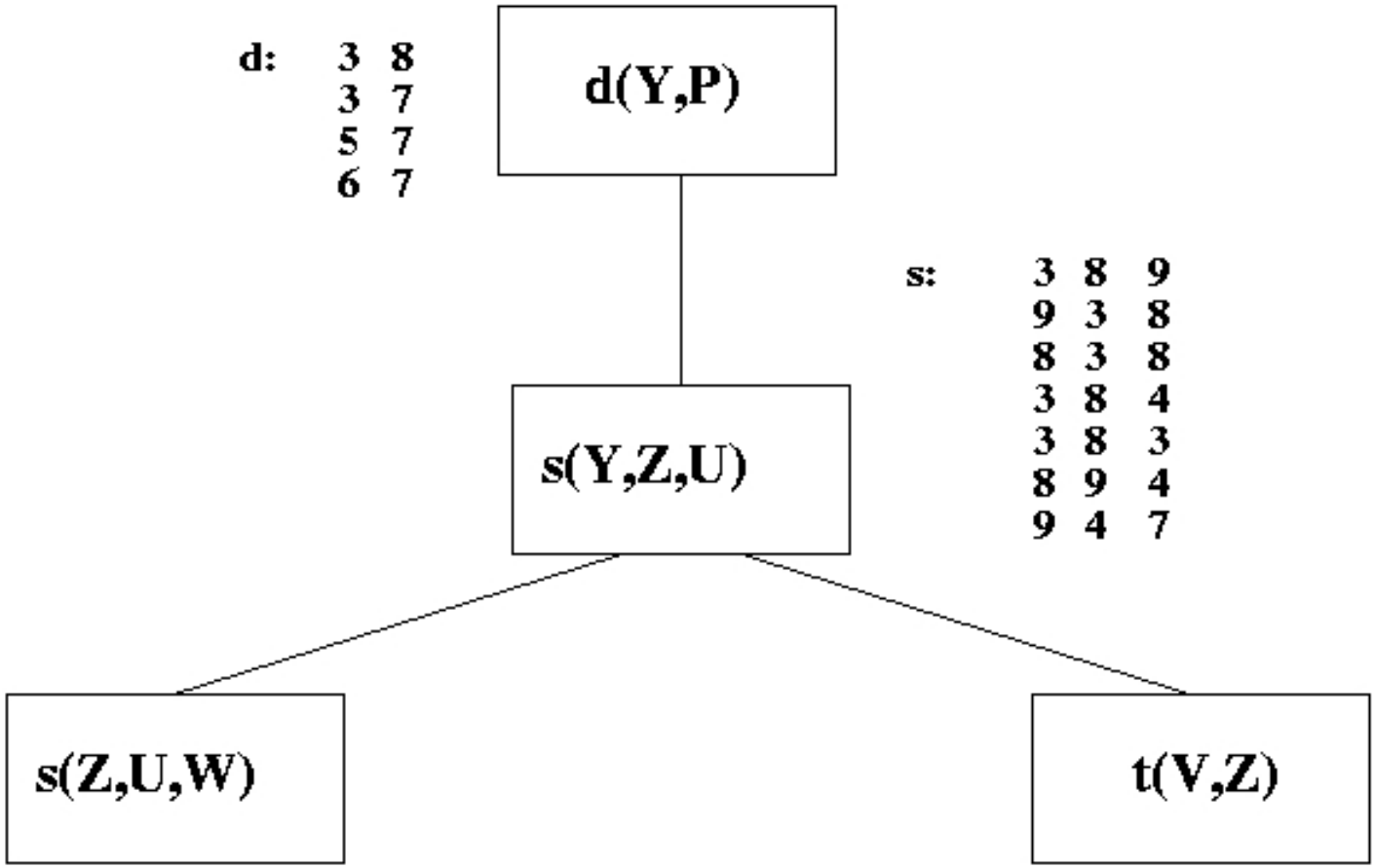
Temp=
Employee semijoin (DepartmentName=DepartmentName)
Department

Employee

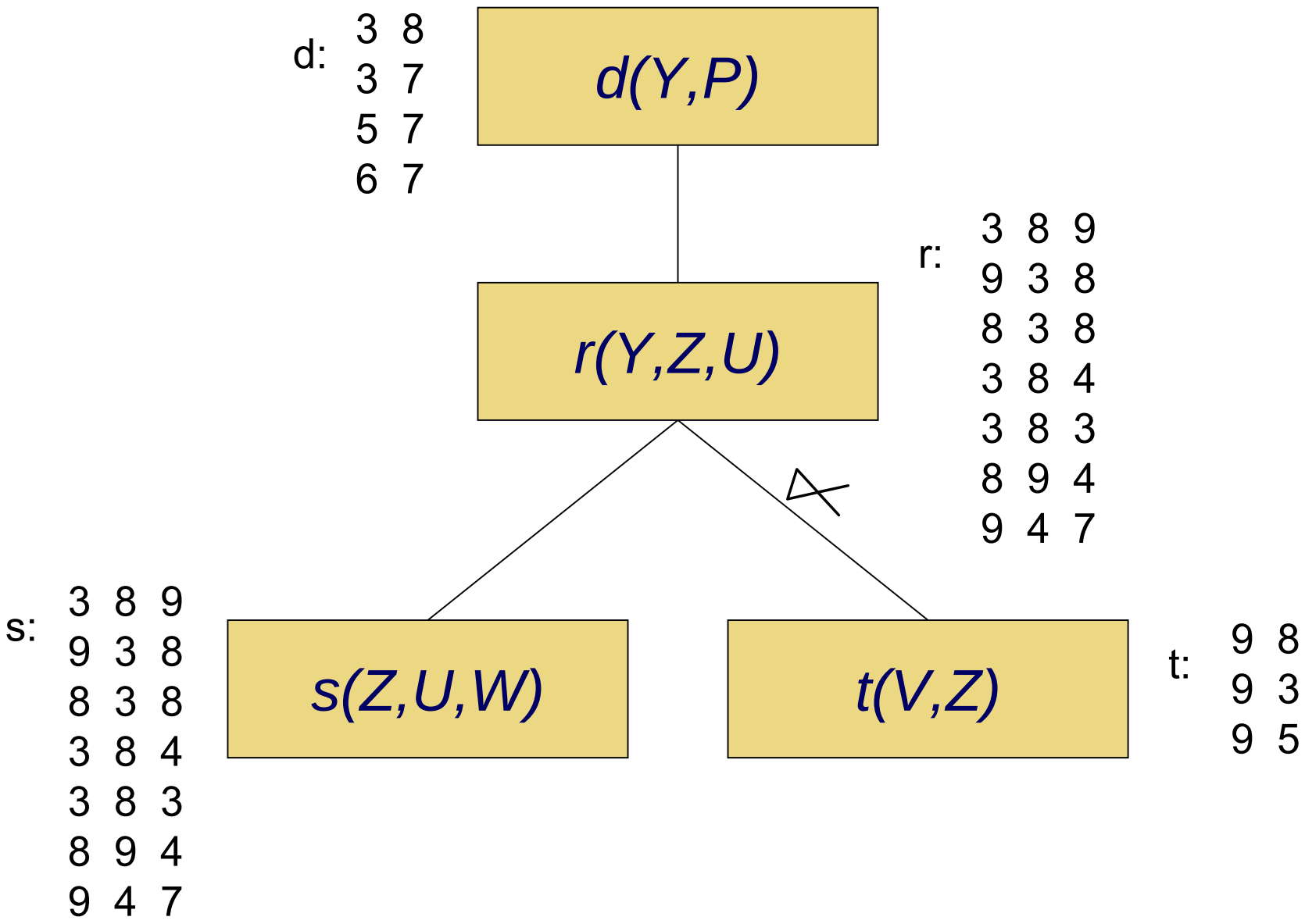
Employee Name	Employee Salary	Department Name
Ackman	5000	Car
Baker	4500	Car

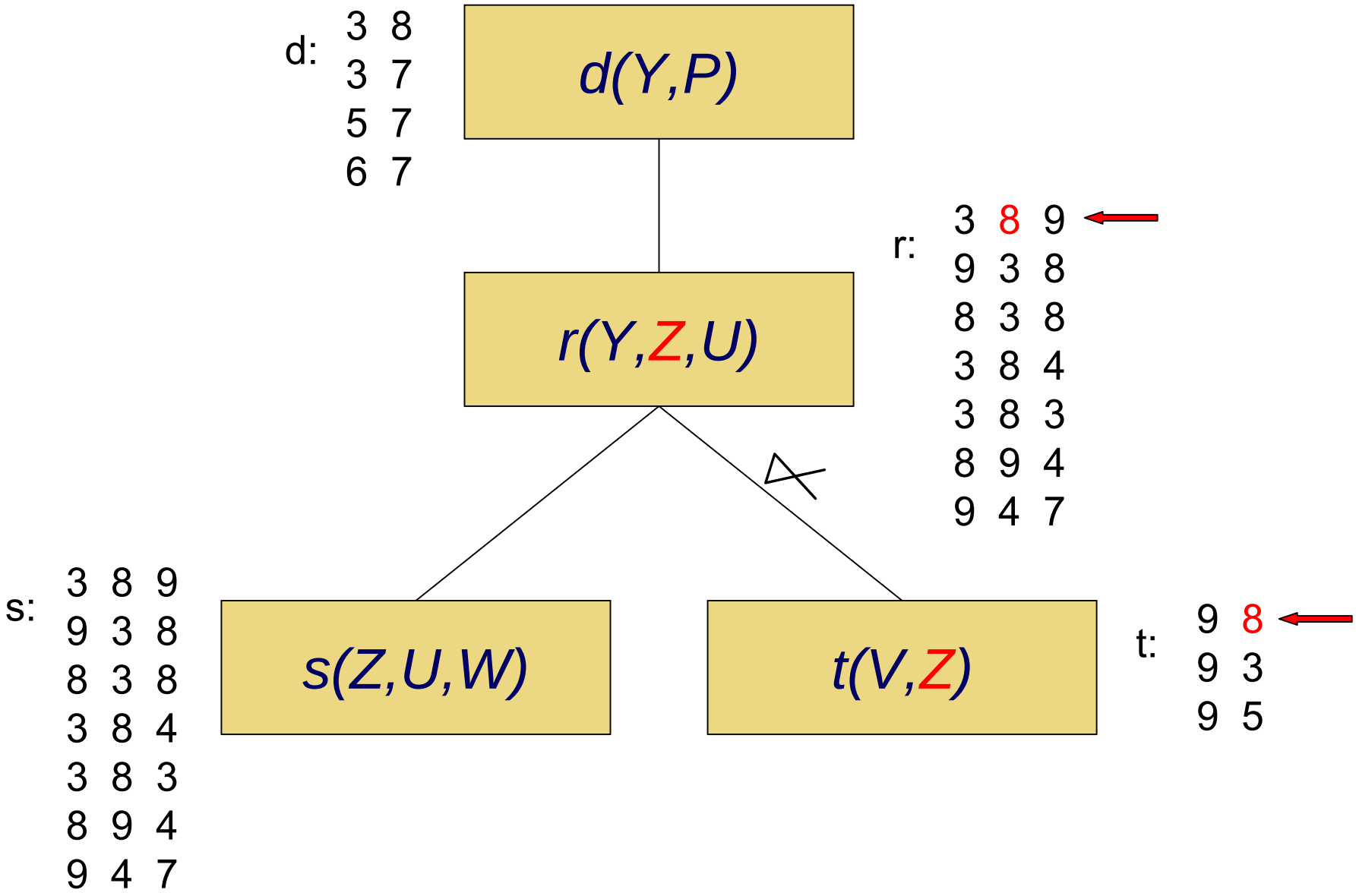
Figure by Dr. James A. Larson

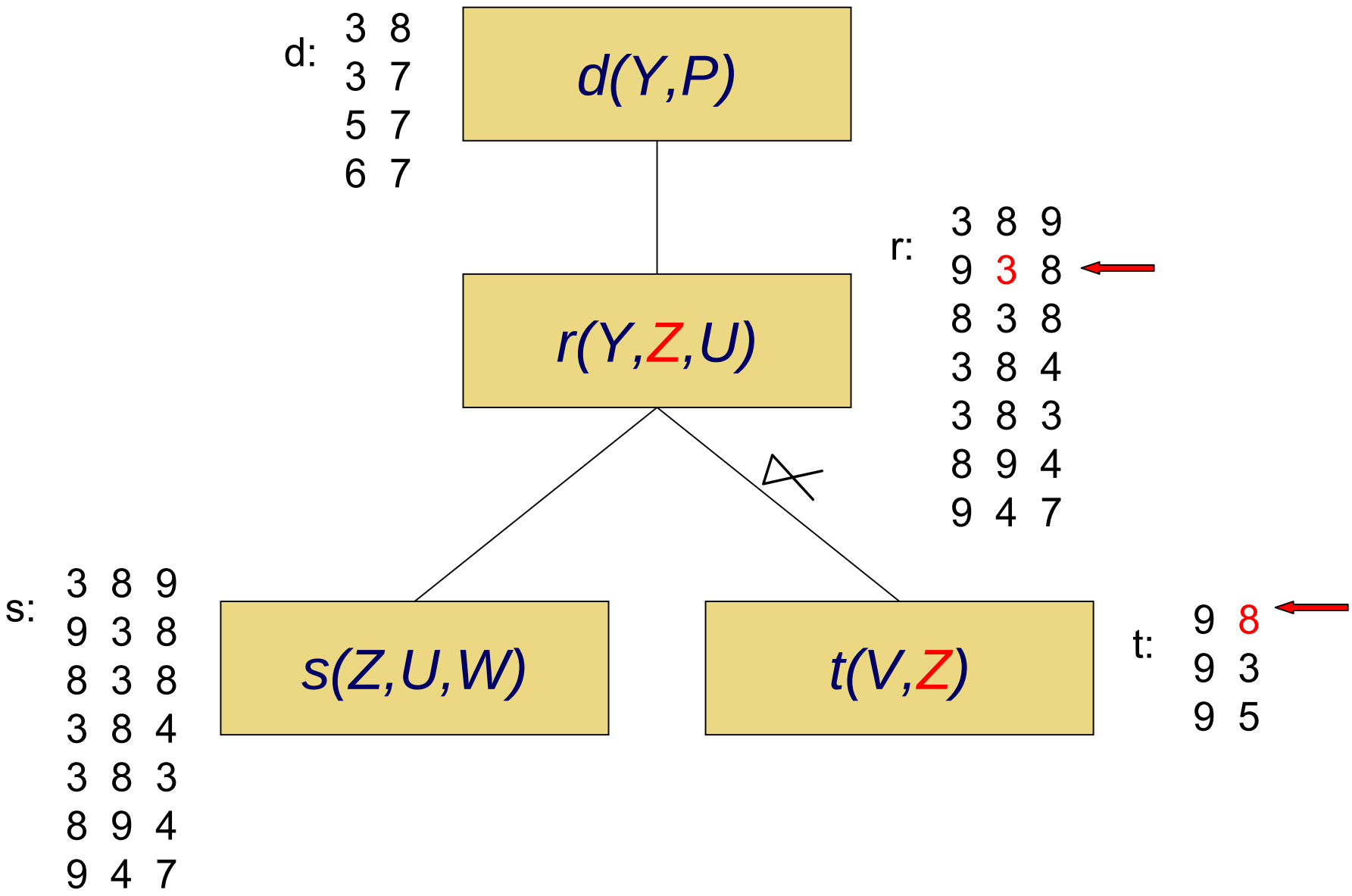
$$Emp \ltimes_{A=B} Dept = \pi_{attr(Emp)}(Emp \bowtie_{A=B} Dept)$$

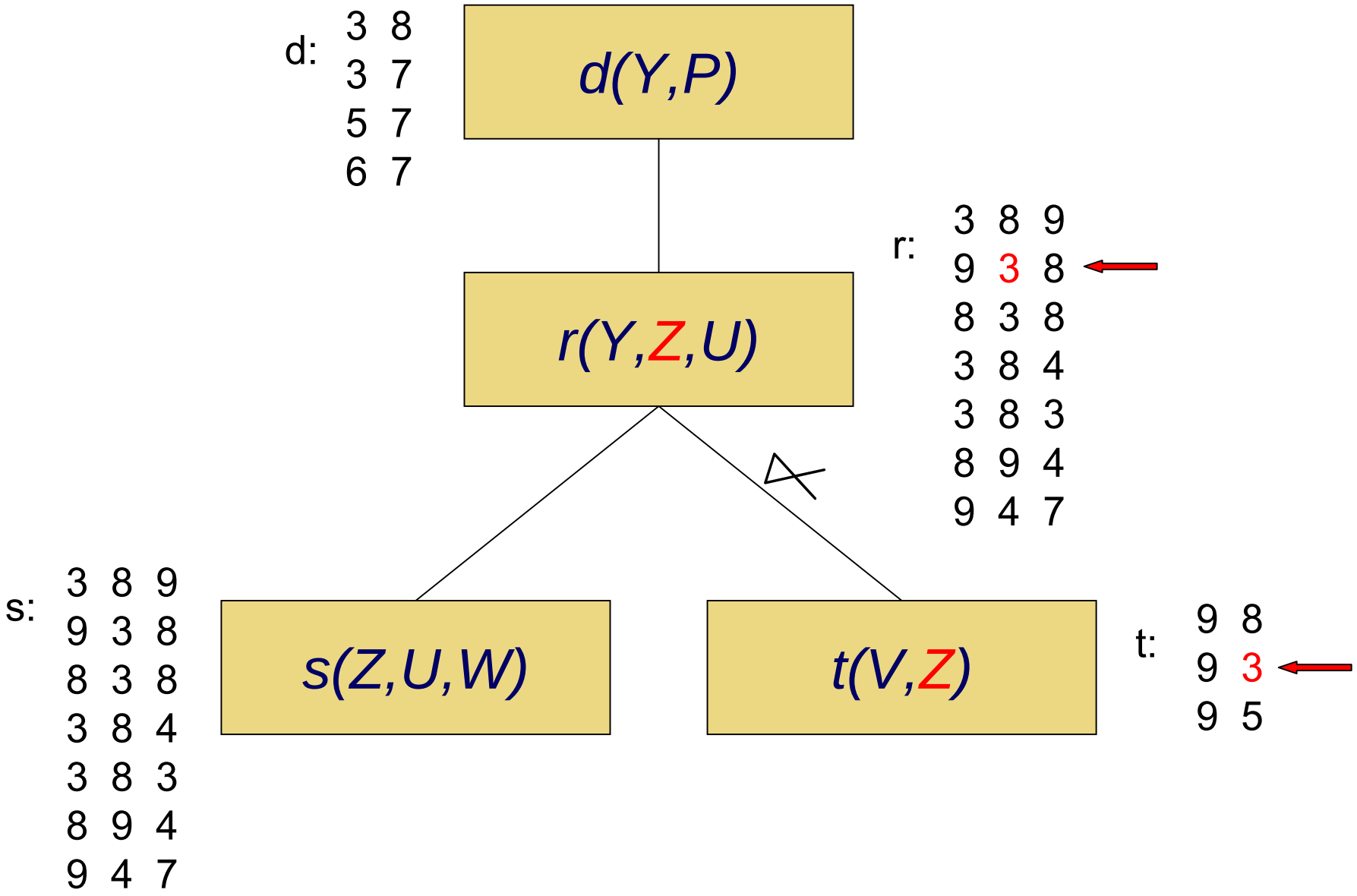

d:
3 8
3 7
5 7
6 7
d(Y,P)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Y,Z,U)
s(Z,U,W)
t(V,Z)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r:
9 8
9 3
9 5

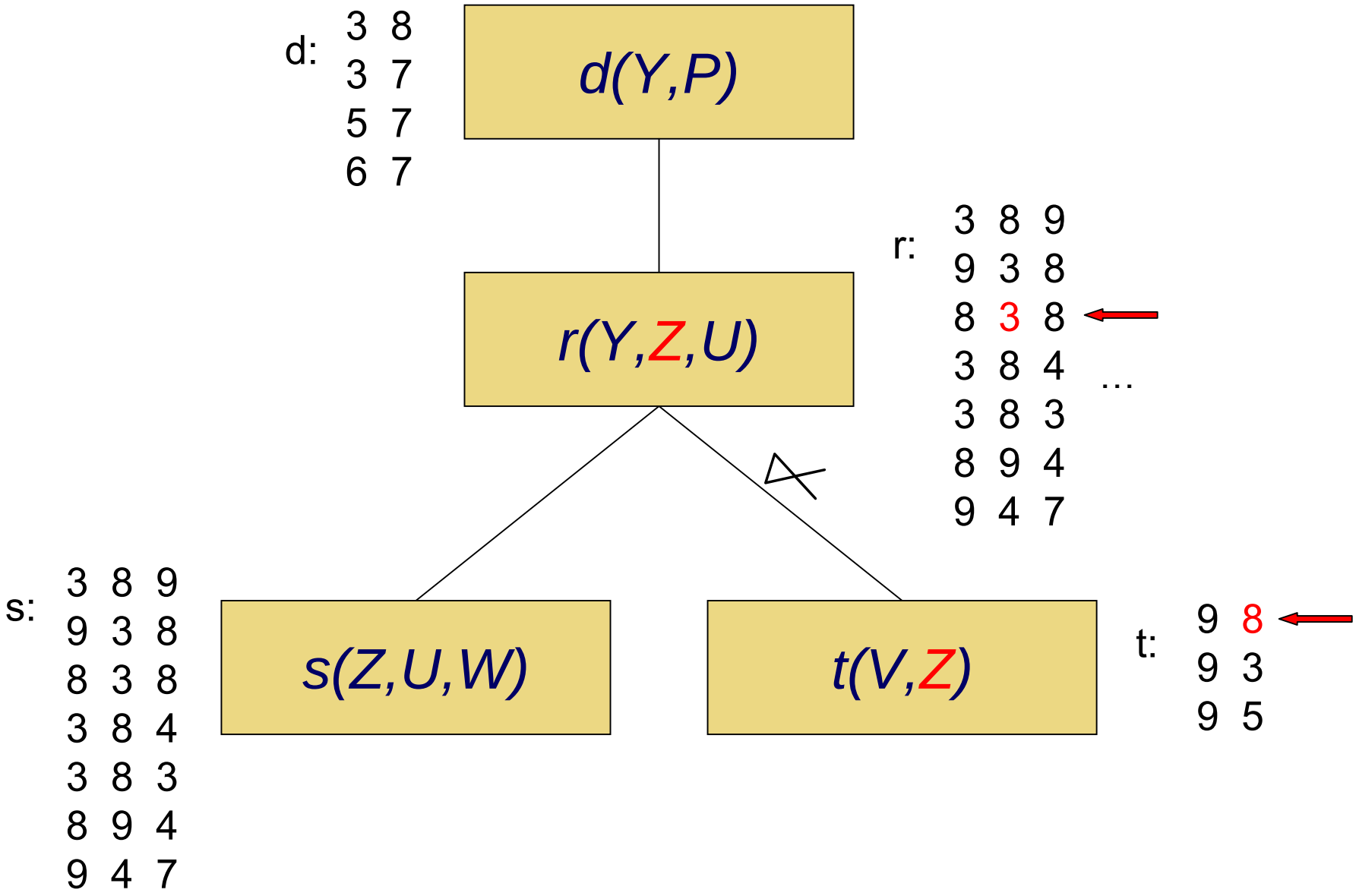
$n^2 \log n$

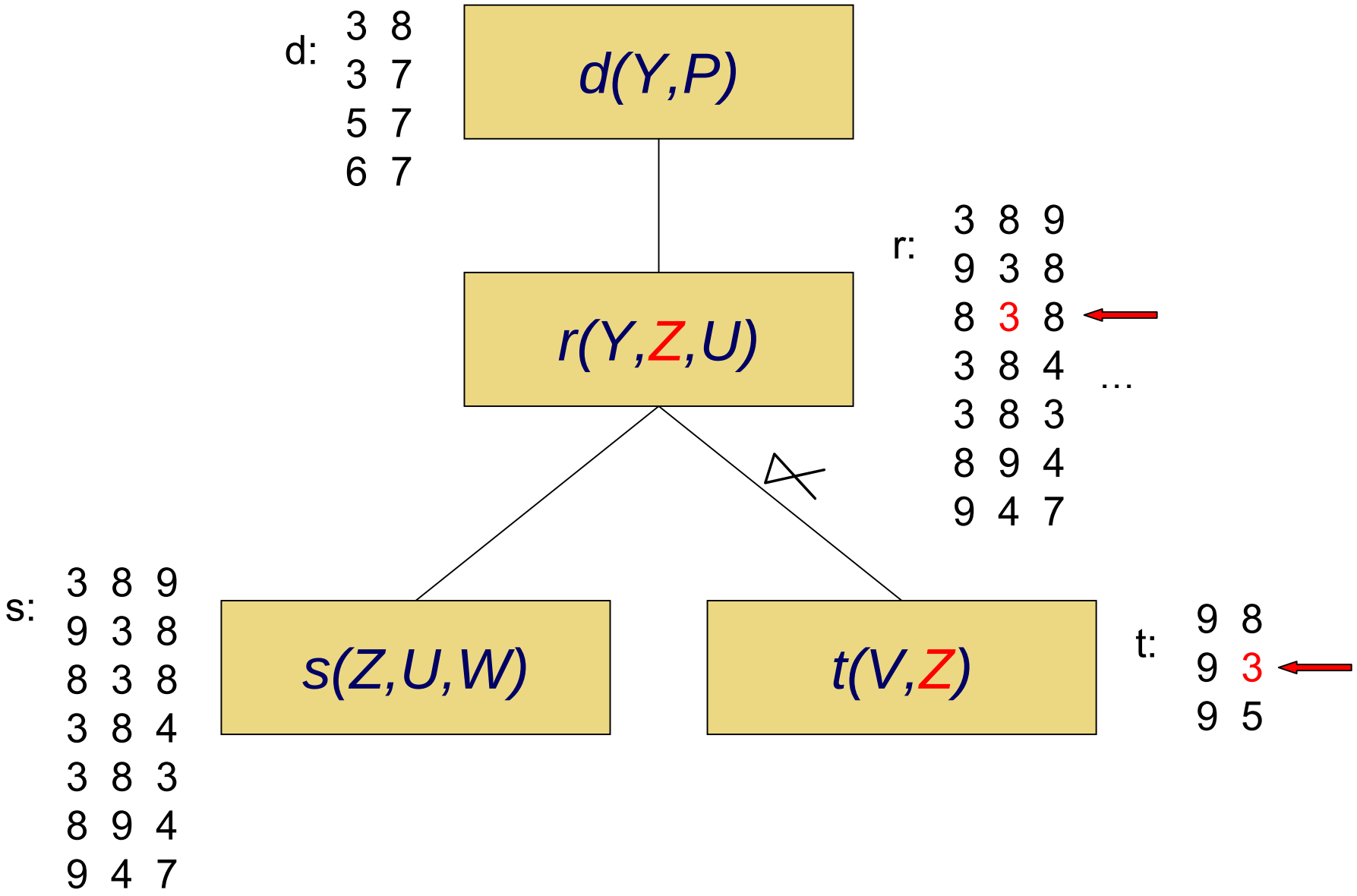

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

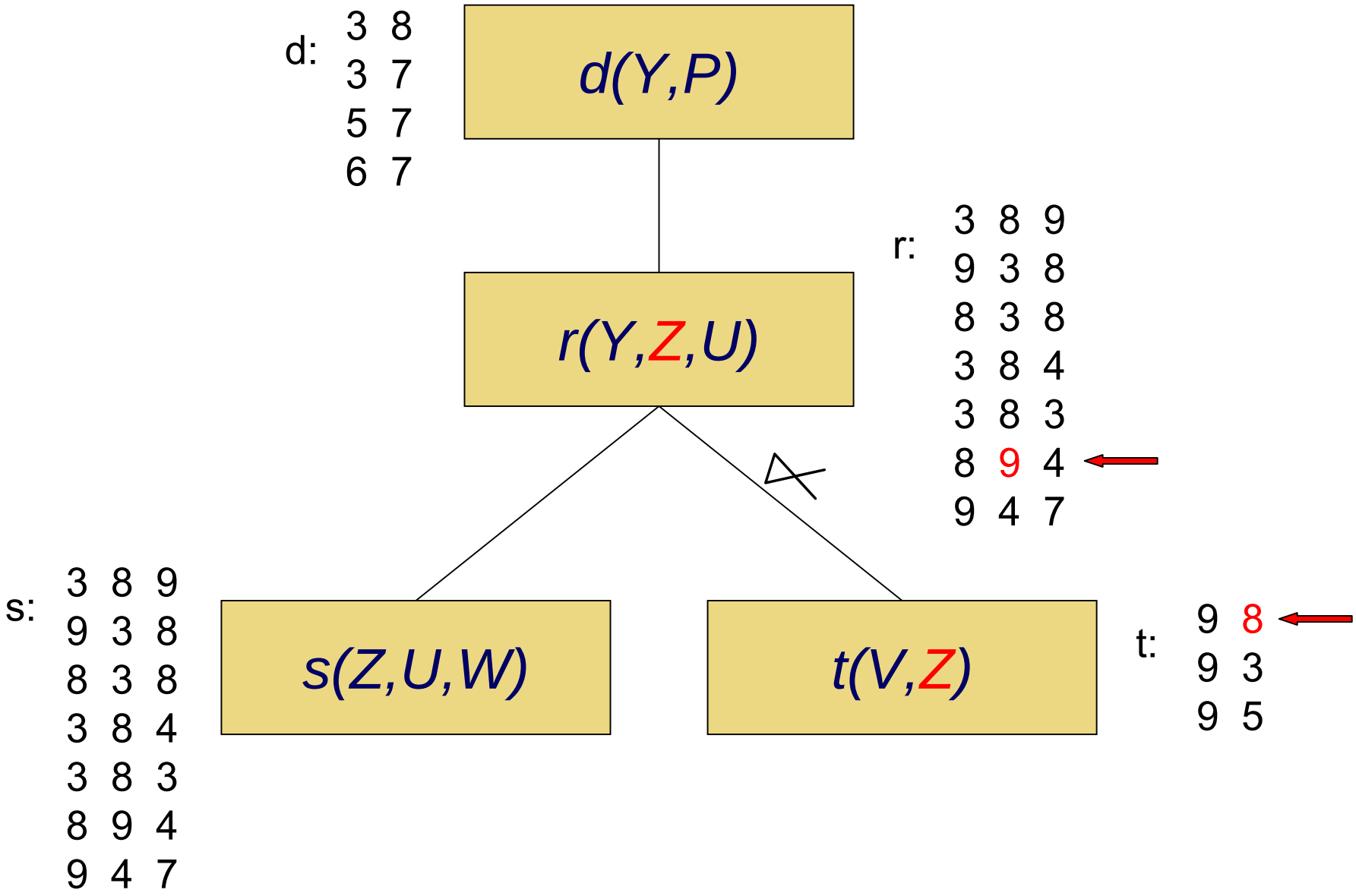

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

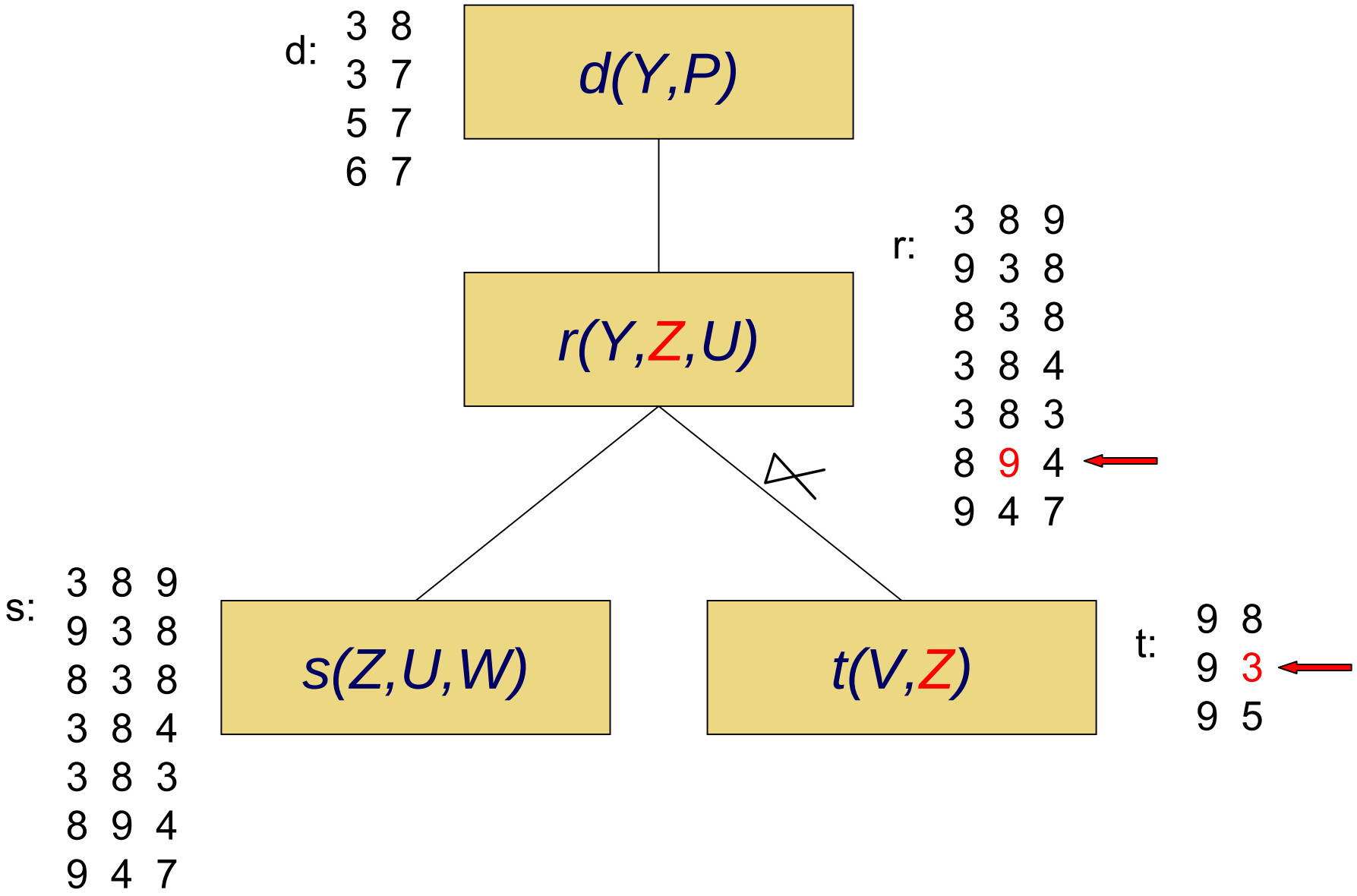

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

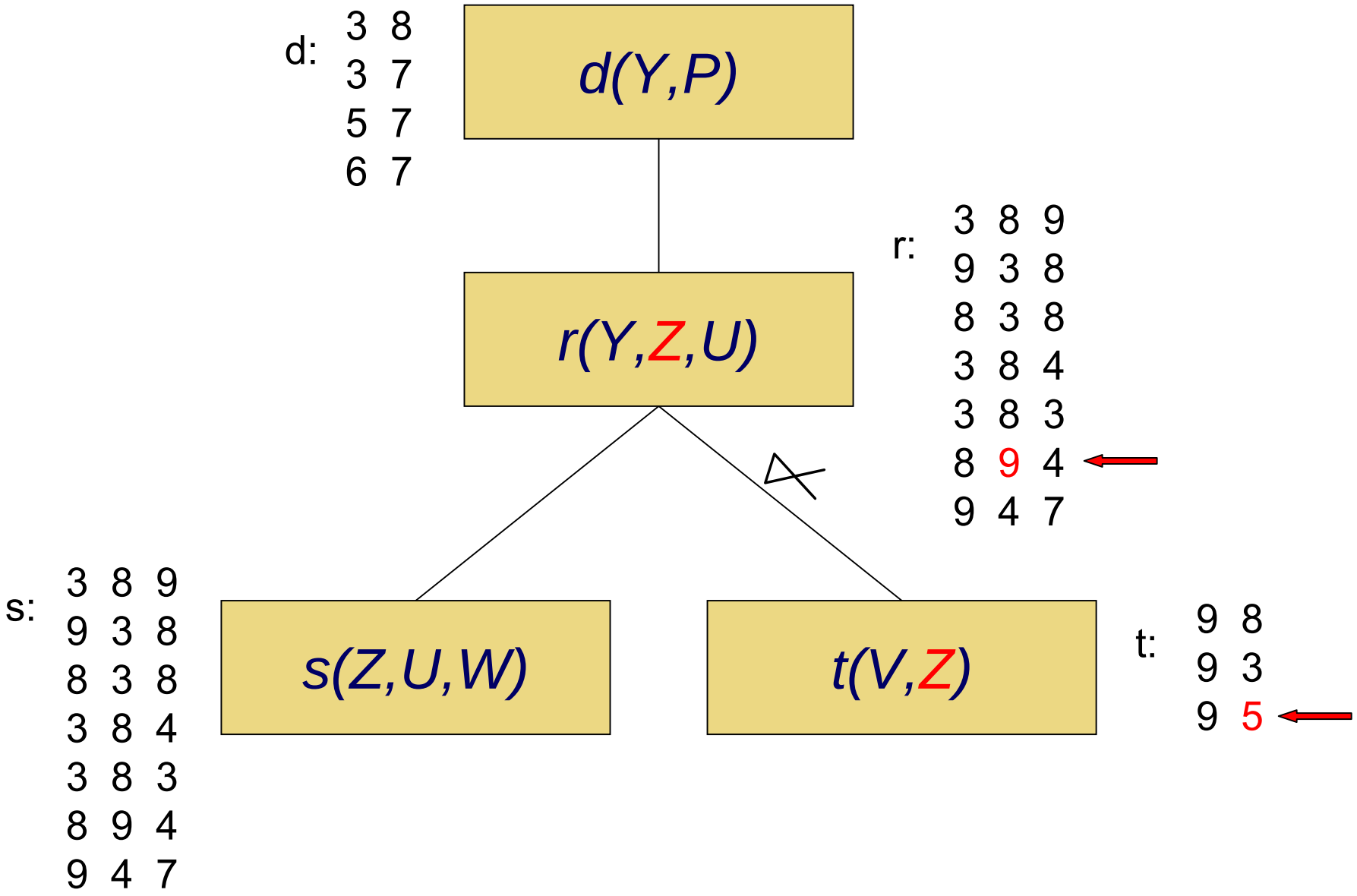

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

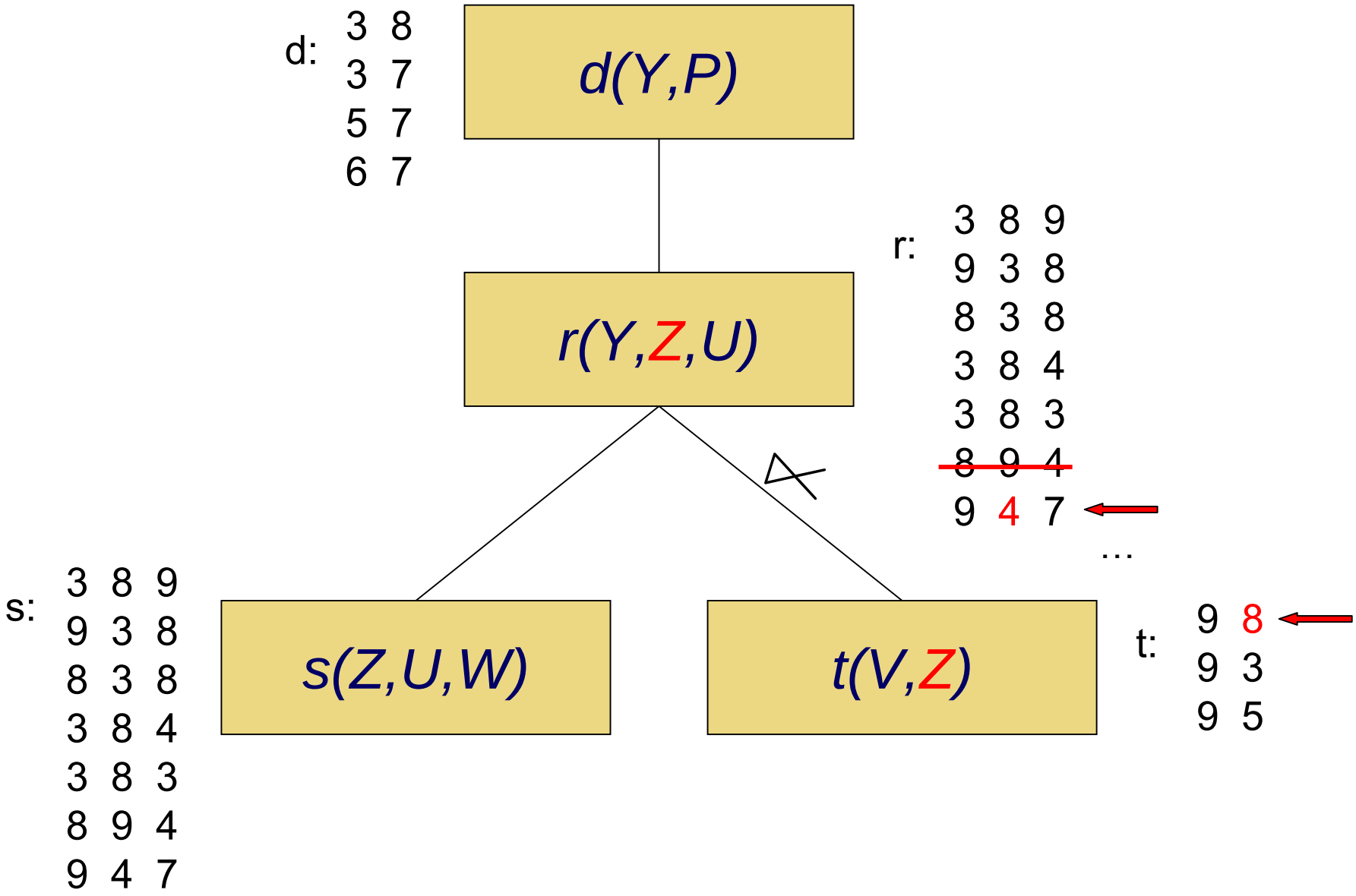

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
...
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

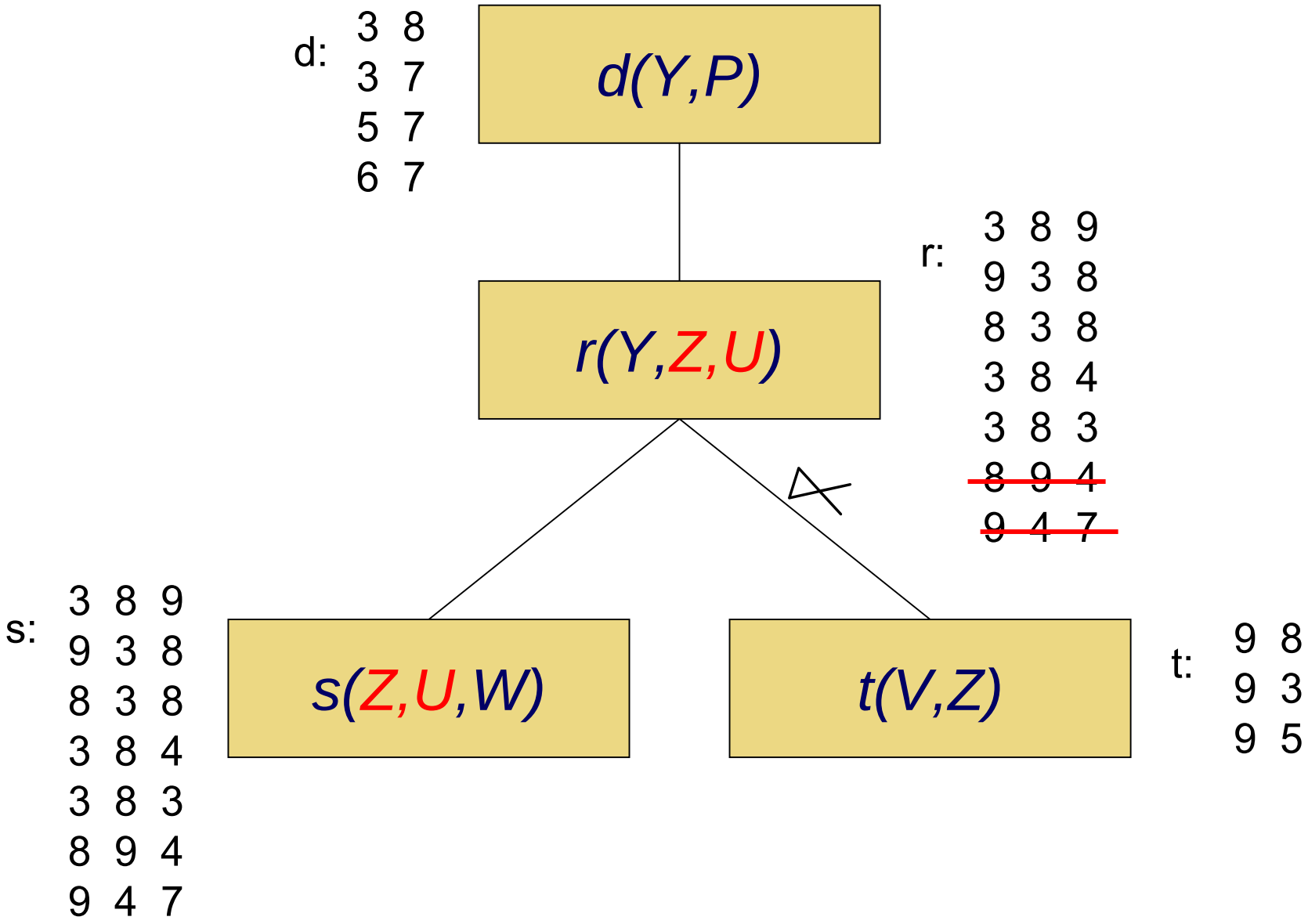

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
...
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

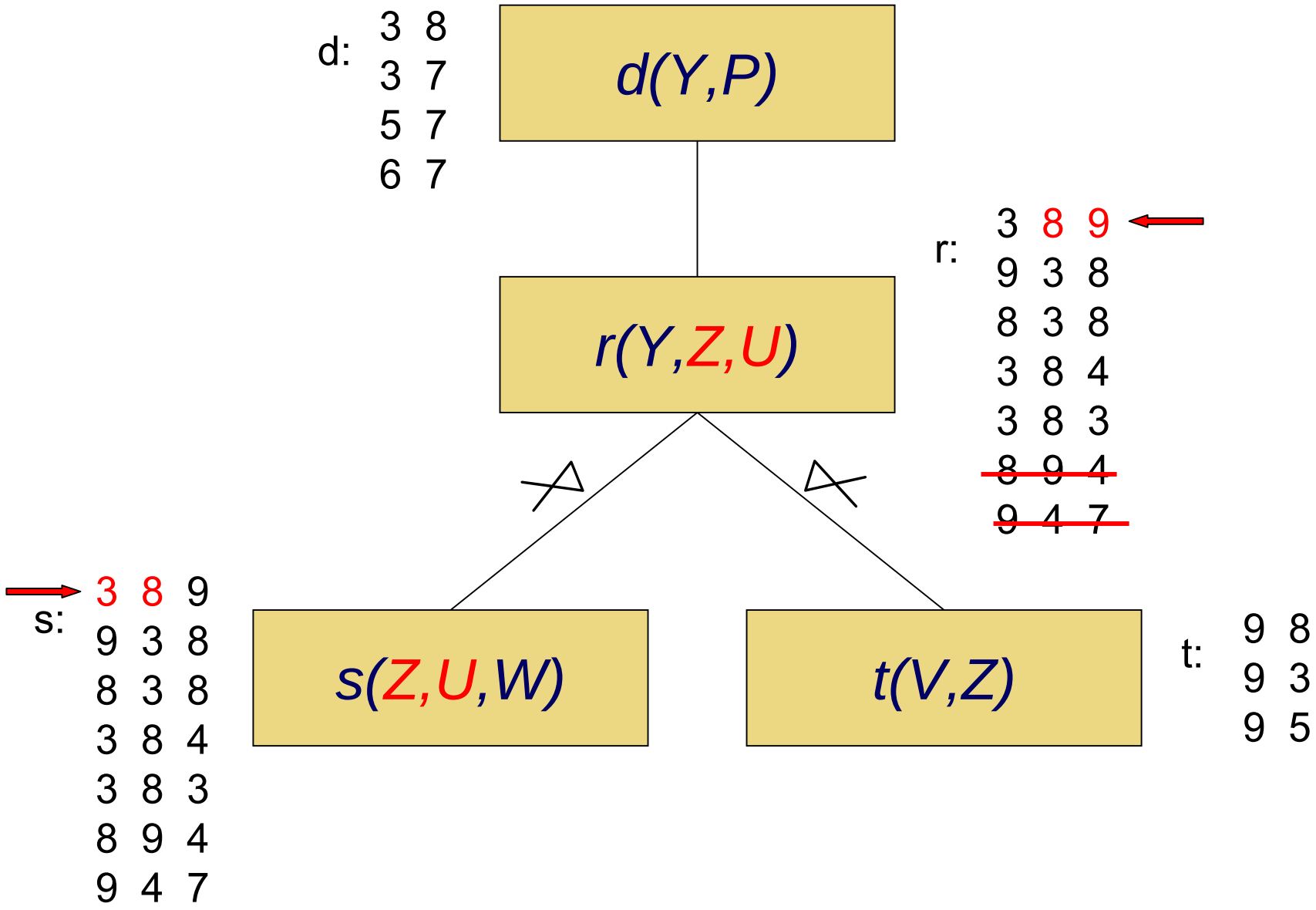

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

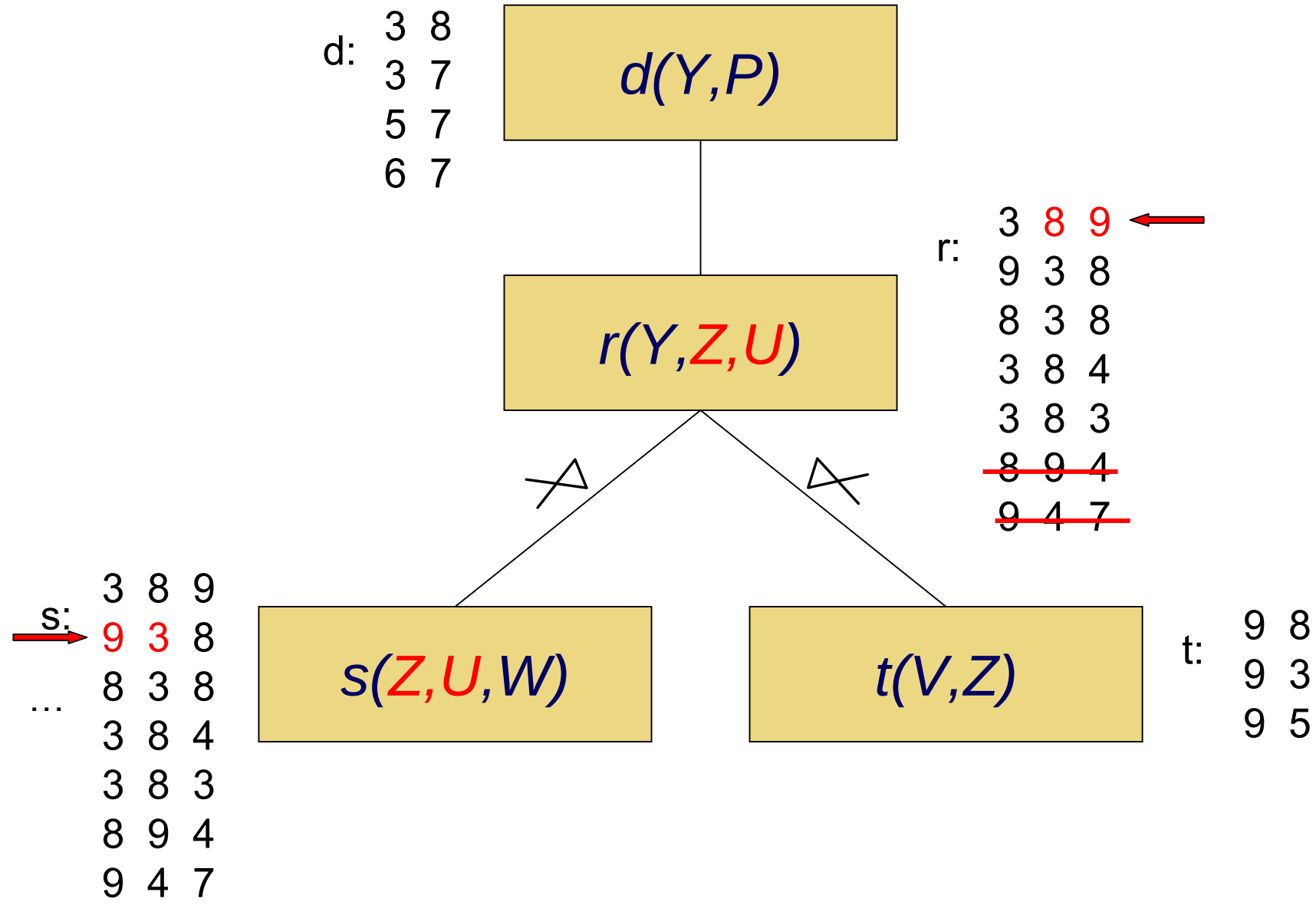

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

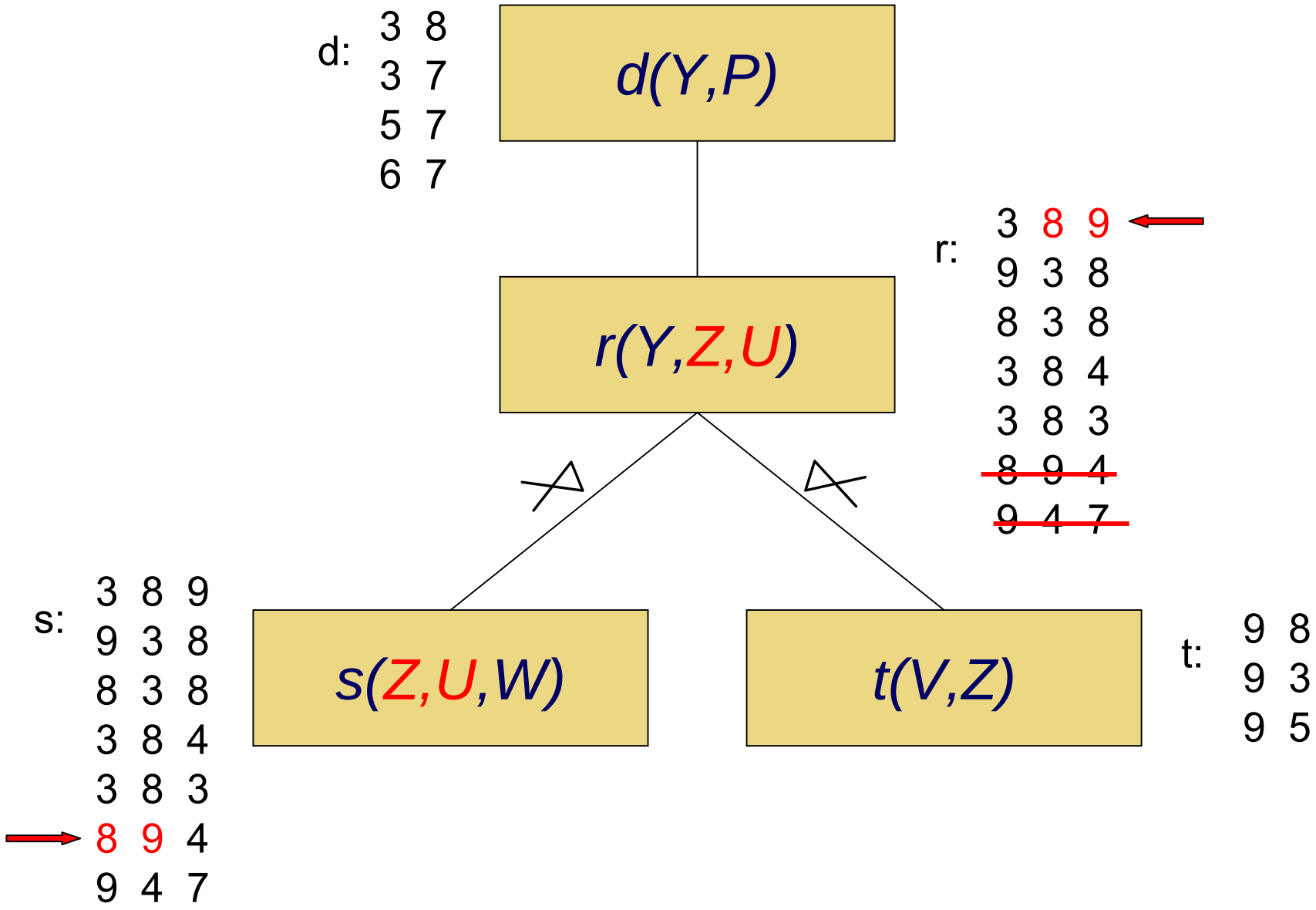

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

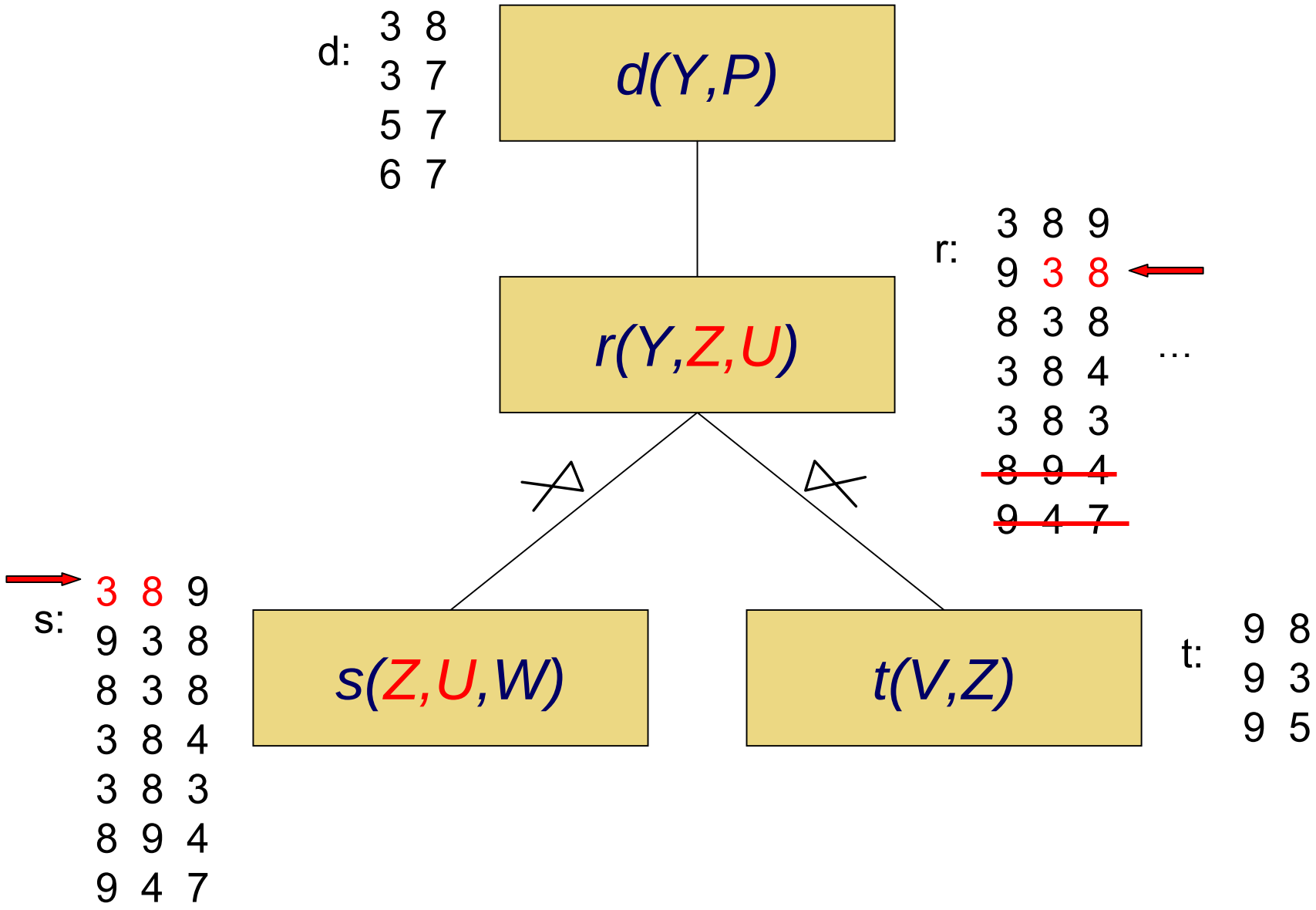

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
...
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

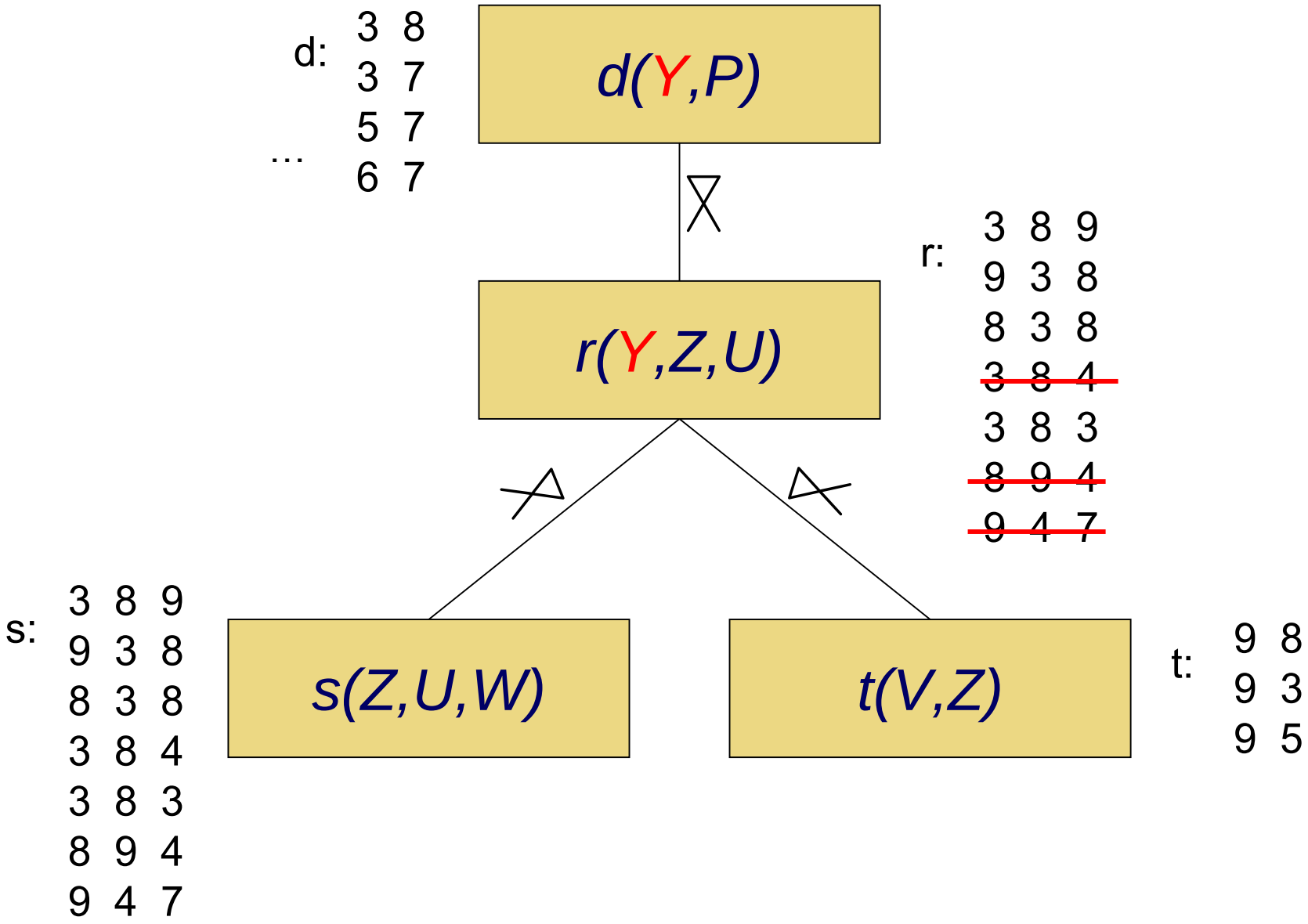

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

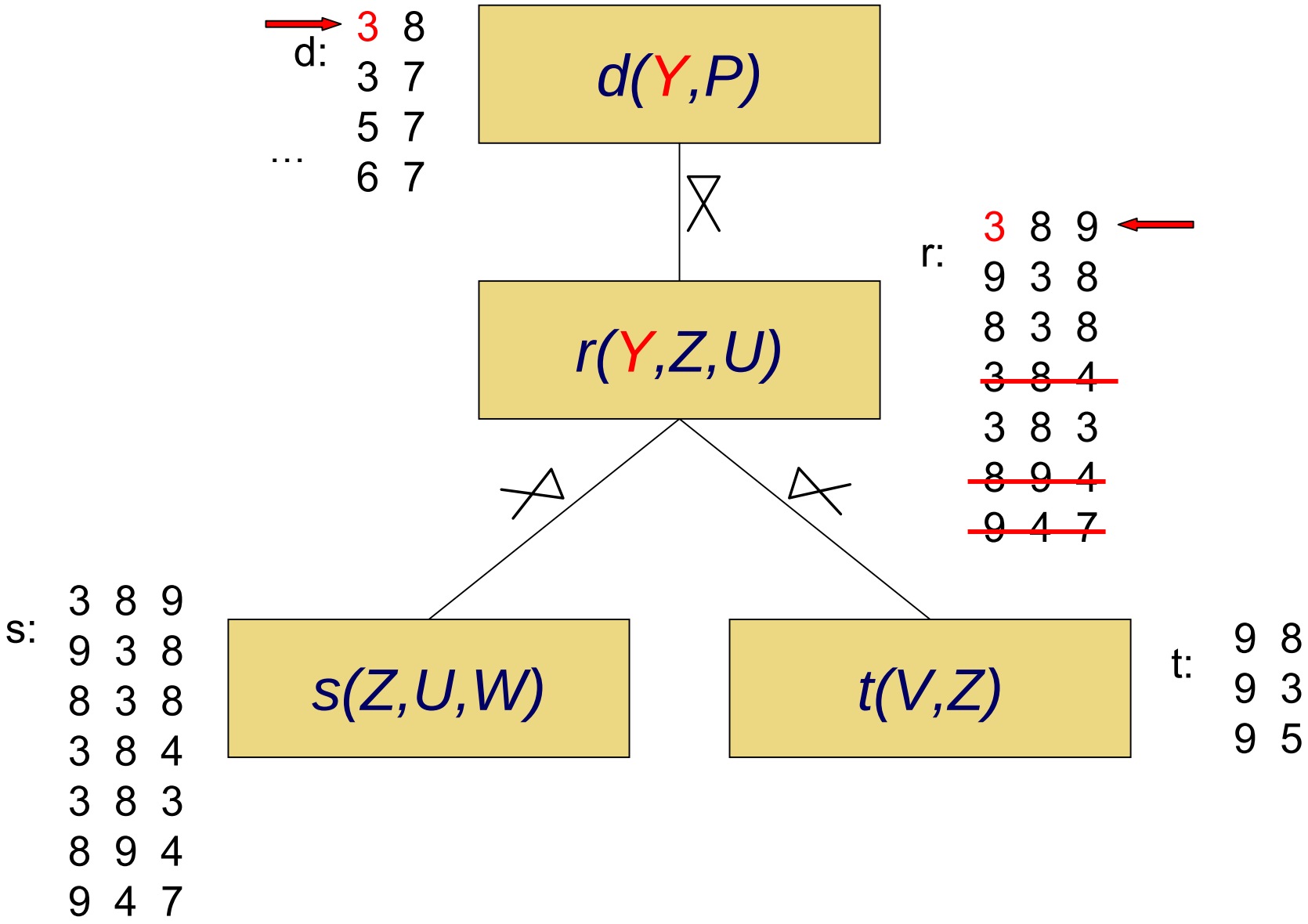

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

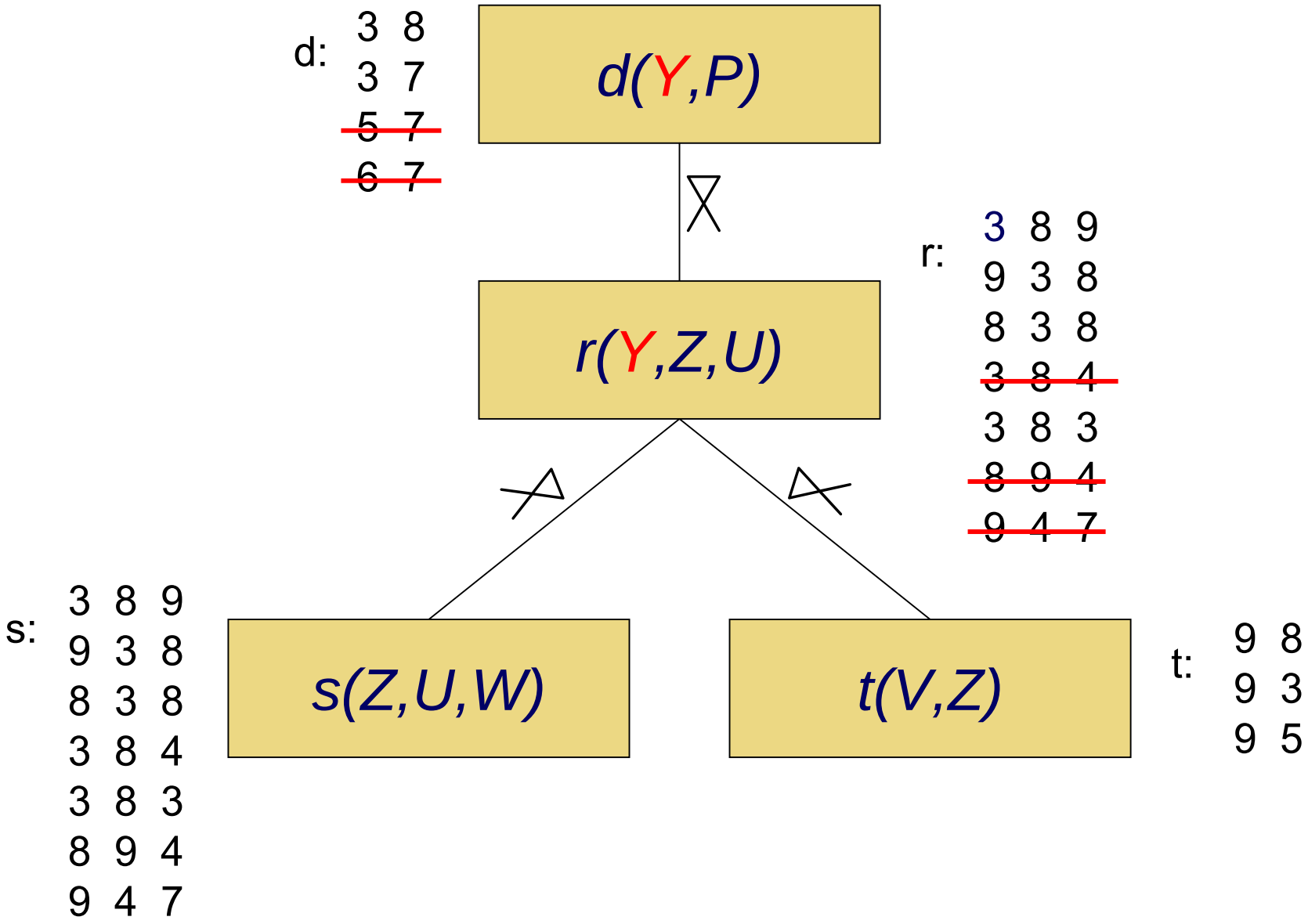

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
…
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

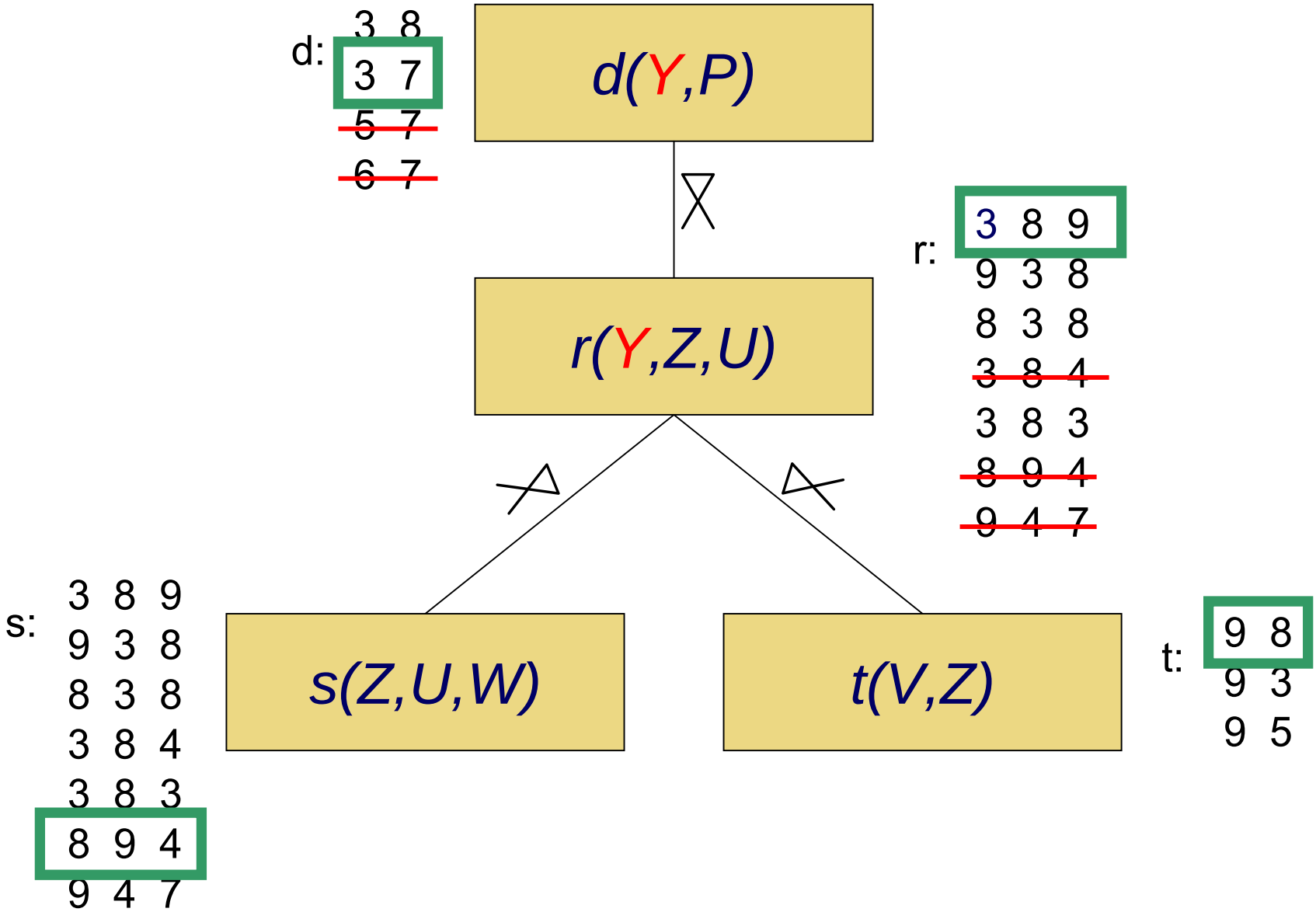

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5


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


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


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


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



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

Computing the result

- The result size can be exponential (even in case of ACQs).
- Even when the result is of polynomial size, it is in general hard to compute.
- In case of acyclic queries, the result can be computed in time polynomial in the result size (i.e., in output-polynomial time).
- This will remain true for the subsequent generalizations of ACQs.
- The result of ACQs can be computed by adding a top-down phase to Yannakakis' algorithm for ABCQs and by joining the partial results.

Precise complexity of acyclic Boolean queries ?

- Can be done better? Parallel algorithm?

Theorem: Answering acyclic BCQs is LOGCFL-complete

$AC0 \subset NC1 \subseteq L \subseteq NL \subseteq LOGCFL \subseteq NC2$

LOCFL: Class of all problems that are logspace-reducible to a context-free language.

Theorem [GLS99]: Acyclic CSP-solvability is LOGCFL-complete.
Answering acyclic BCQs is LOGCFL-complete

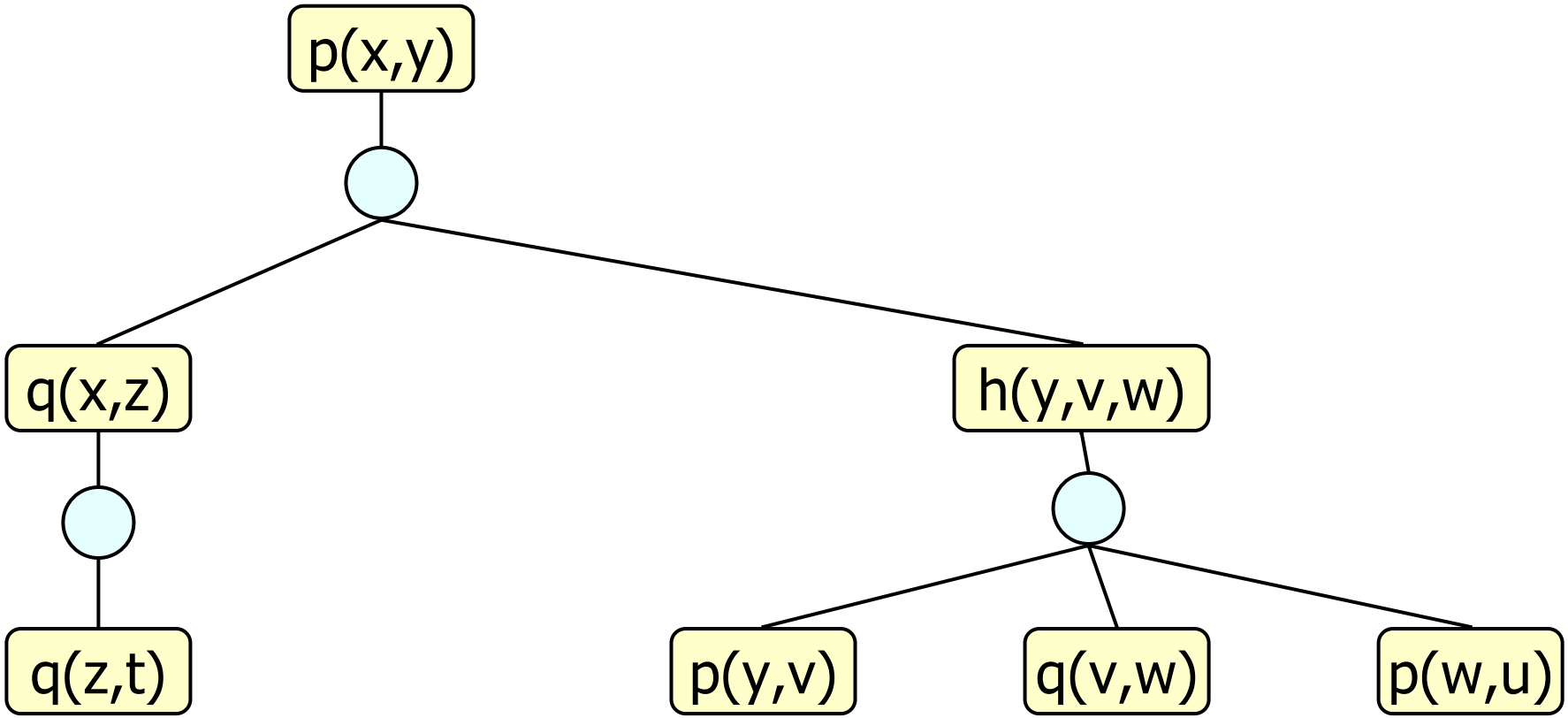
LOGCFL: class of problems/languages that are logspace-reducible to a CFL

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

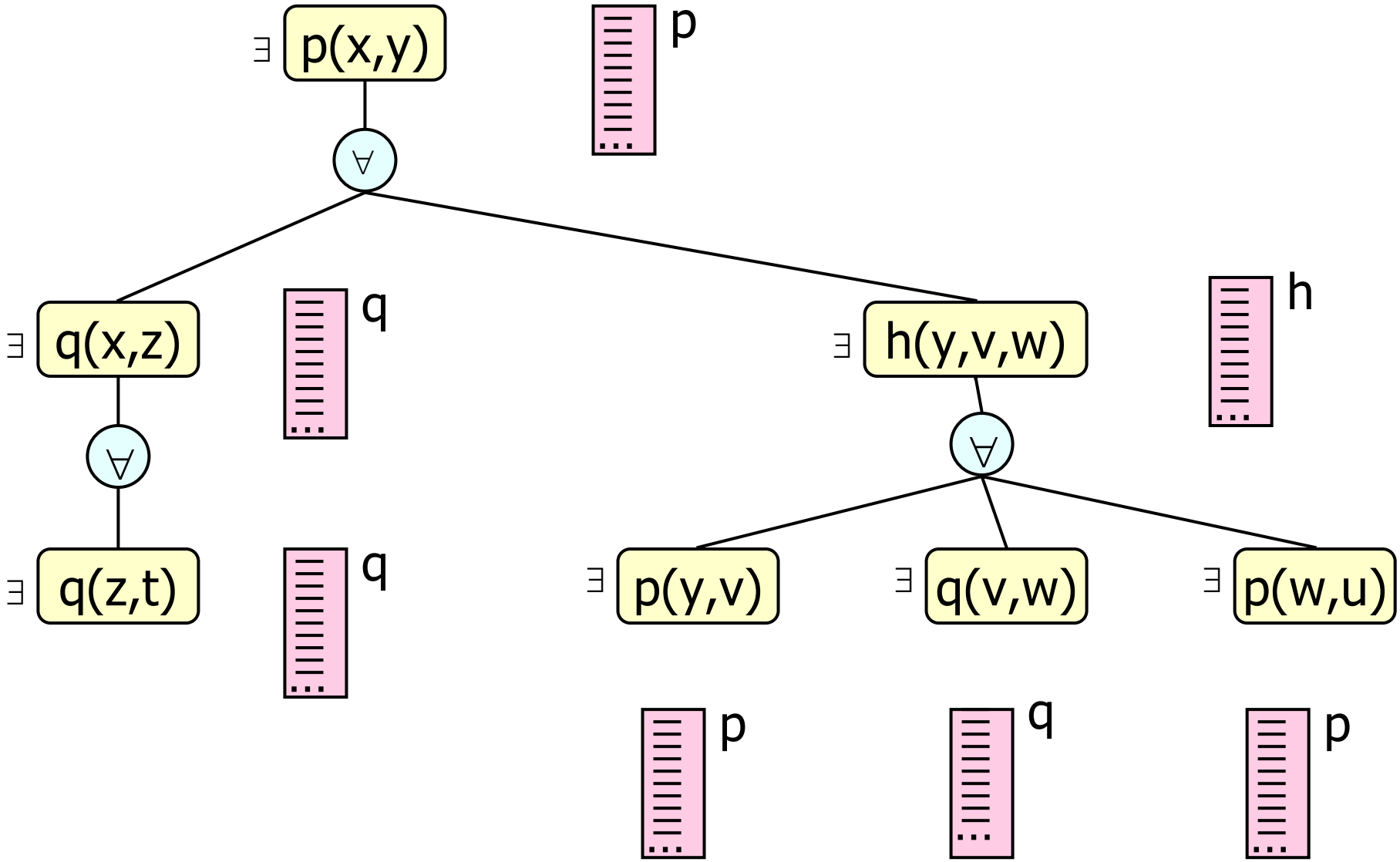
Characterization of LOGCFL [Ruzzo80]:

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

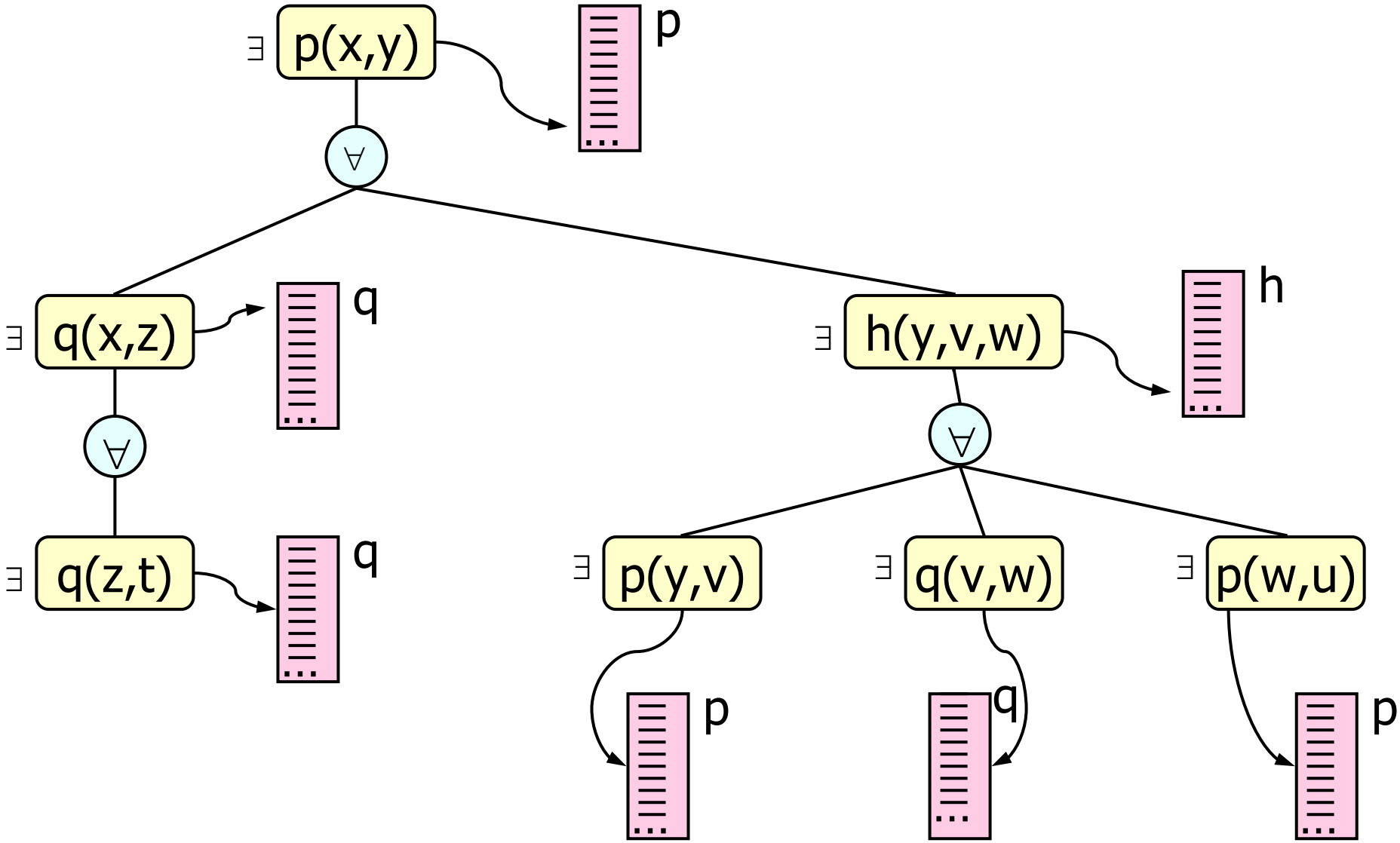
ABCQ is in LOGCFL



ABCQ is in LOGCFL



ABCQ is in LOGCFL



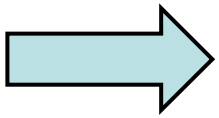
Is this query hard?

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

n size of the database
m number of atoms in the query

m = 11 !

- Classical methods worst-case complexity: $O(n^m)$
- Despite its apparence, this query is nearly acyclic



It can be evaluated in $O(m \cdot n^2 \cdot \log n)$

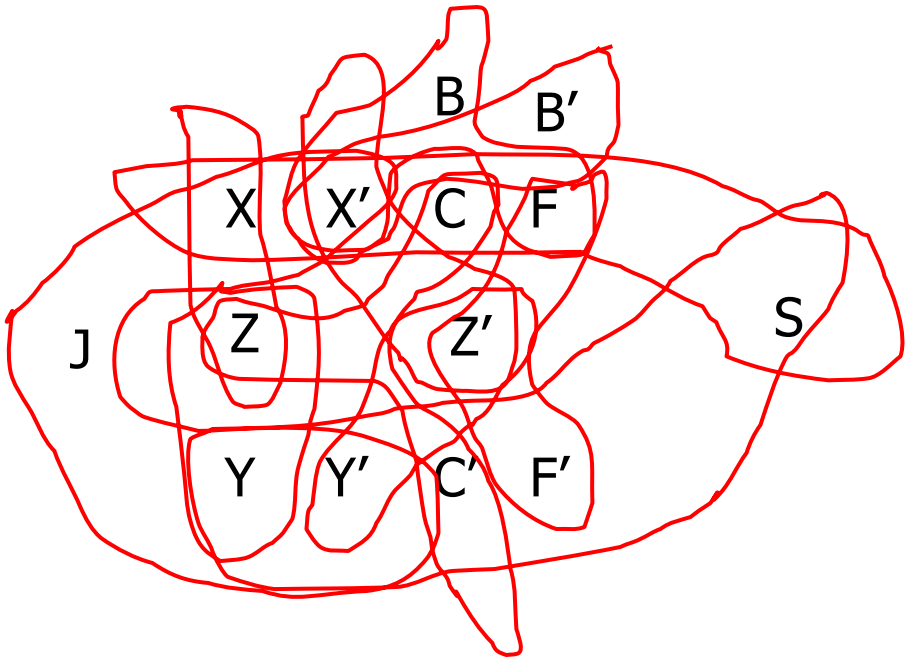
Nearly Acyclic Queries

In practice, many queries are acyclic,
But many others are "nearly acyclic".

Can we make this concept precise?

Can we develop efficient algorithms for
Nearly acyclic queries?

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



It can be evaluated in $O(m \cdot n^2 \cdot \log n)$

Nearly Acyclic Queries & CSPs

- Bounded Treewidth *(tw)*
  - a measure of the cyclicity of graphs
  - for queries: $tw(Q) = tw(G(Q))$
- For fixed k:
  - checking $tw(Q) \leq k$
  - Computing a tree decomposition

  } linear time (Bodlaender'96)

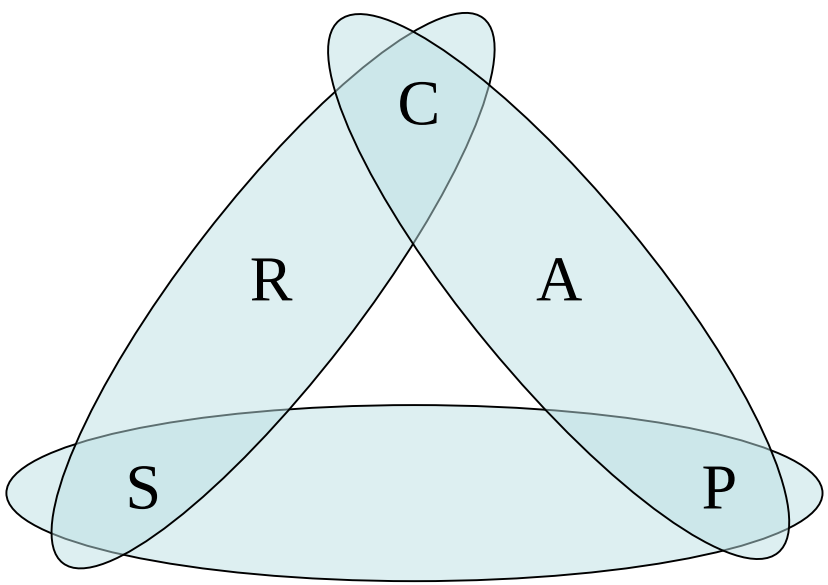
- Deciding CSP of treewidth k:

  $O(n^k \log n)$ [Dechter; Chekuri & Rajaraman'97; Kolaitis & Vardi, 98]

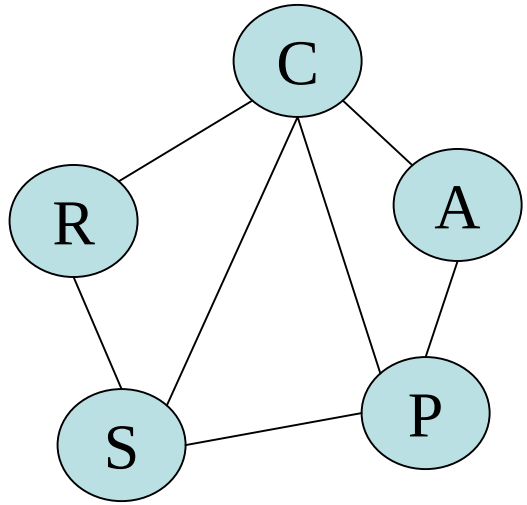
  LOGCFL-complete (G.L.S.'98)

Primal graphs of CSPs/Queries

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

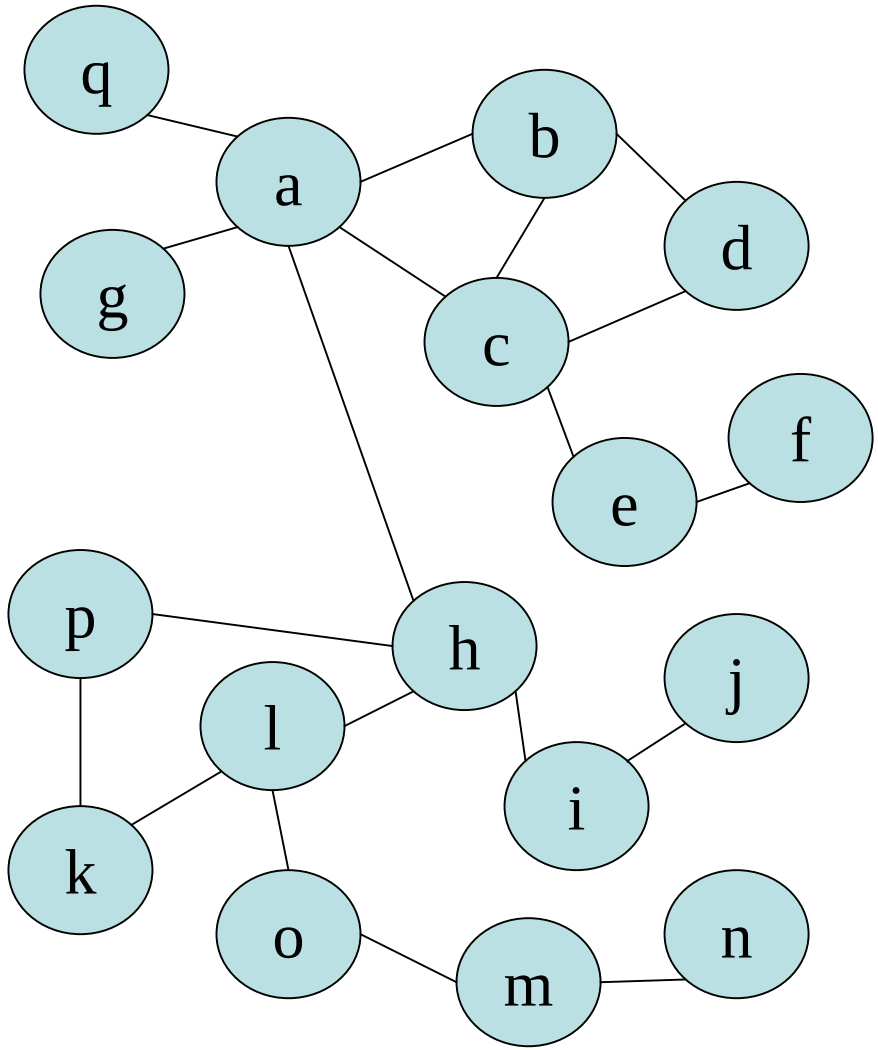


Hypergraph *H(Q)*

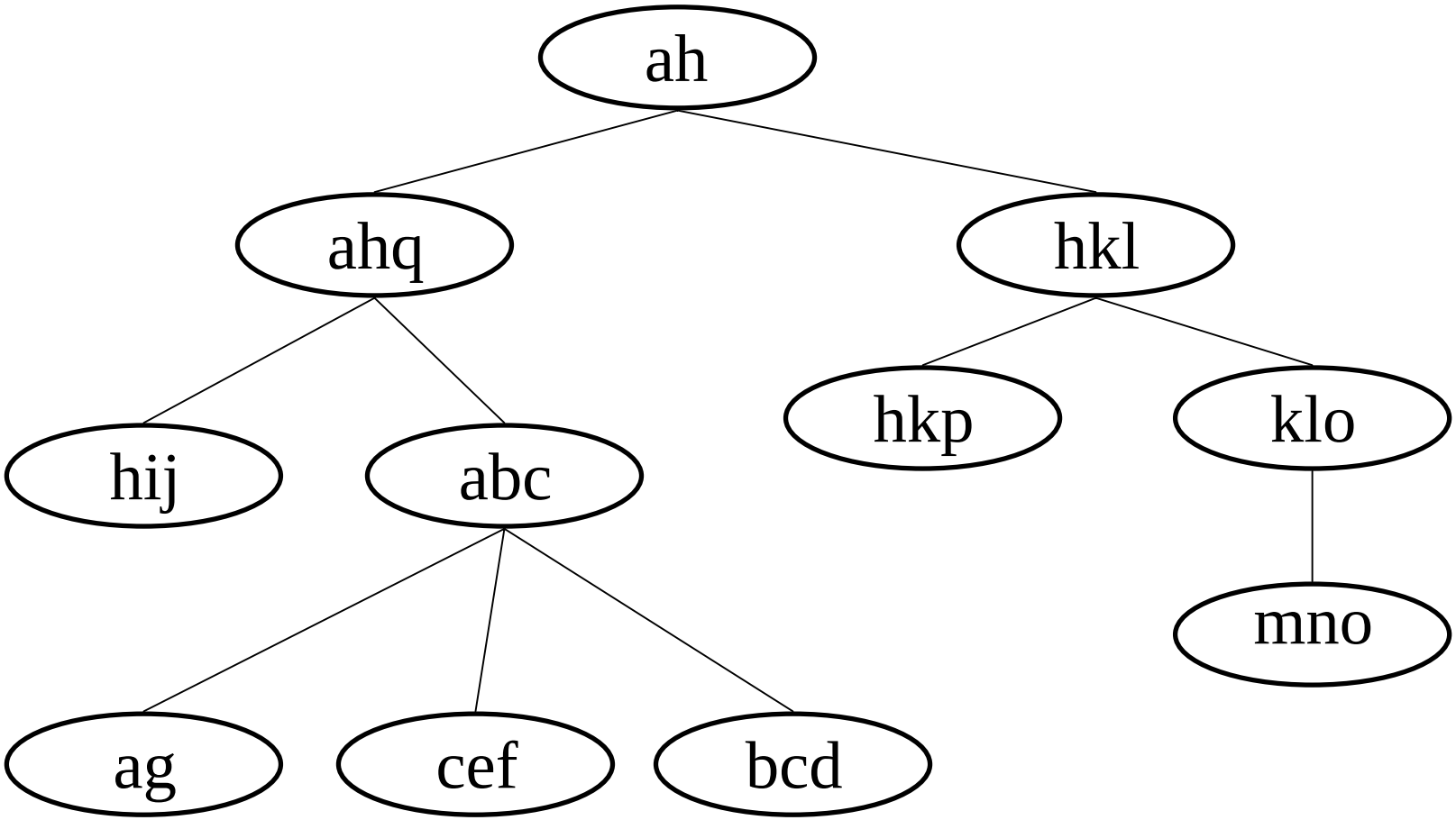


Primal graph *G(Q)*

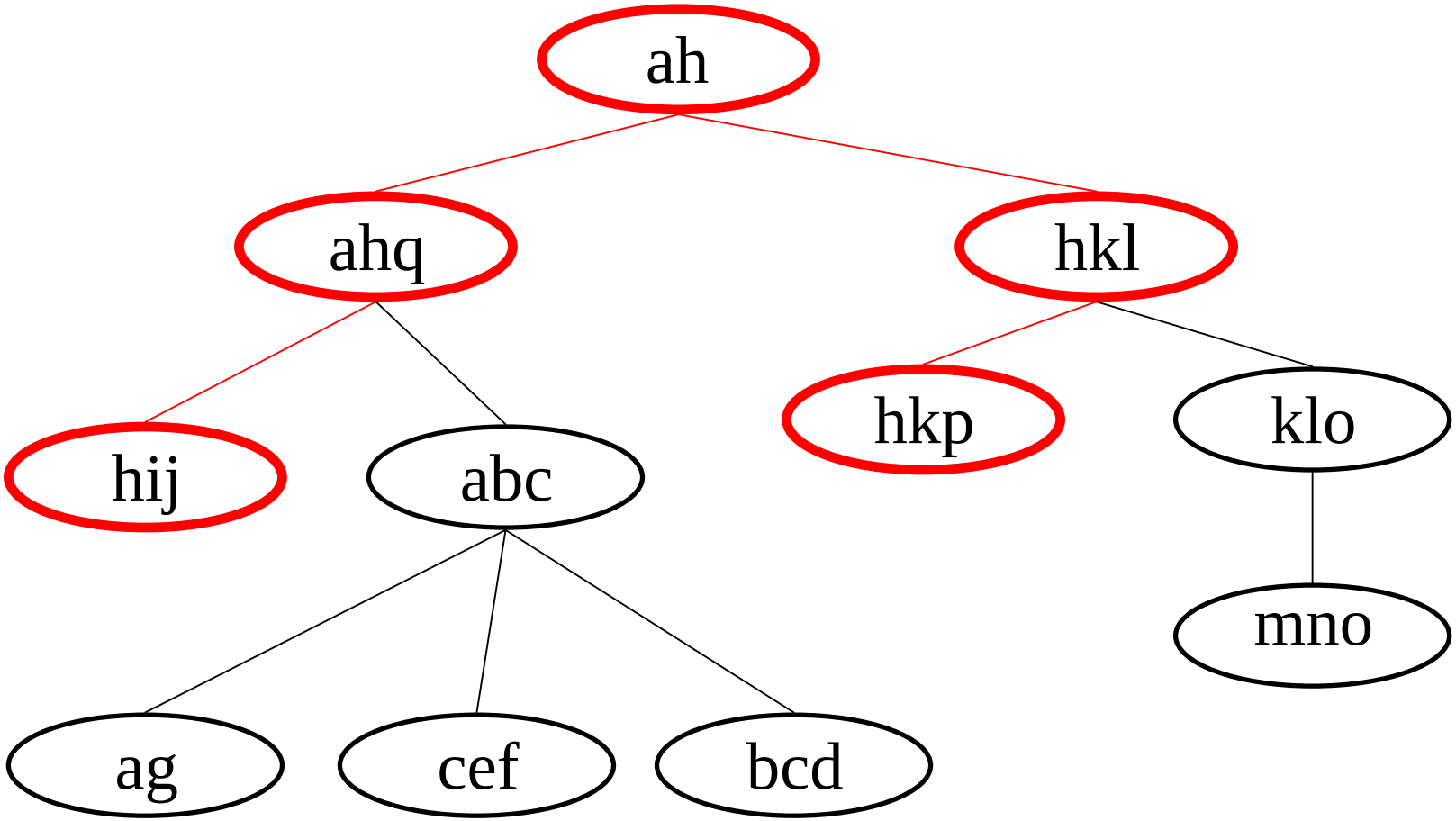
Example: a cyclic graph



A tree decomposition of width 2



Connectedness condition for *h*



Logical characterization of Treewidth

[Kolaitis & Vardi '98]

$$\text{Logic } \mathsf{L} \ : \ \text{FO based on } \exists, \wedge \quad (\neg, \vee, \forall \text{ disallowed})$$

$$\mathsf{L} = \exists \mathrm{FO}_{\wedge,+} \qquad \text{Basic Querying Logic}$$

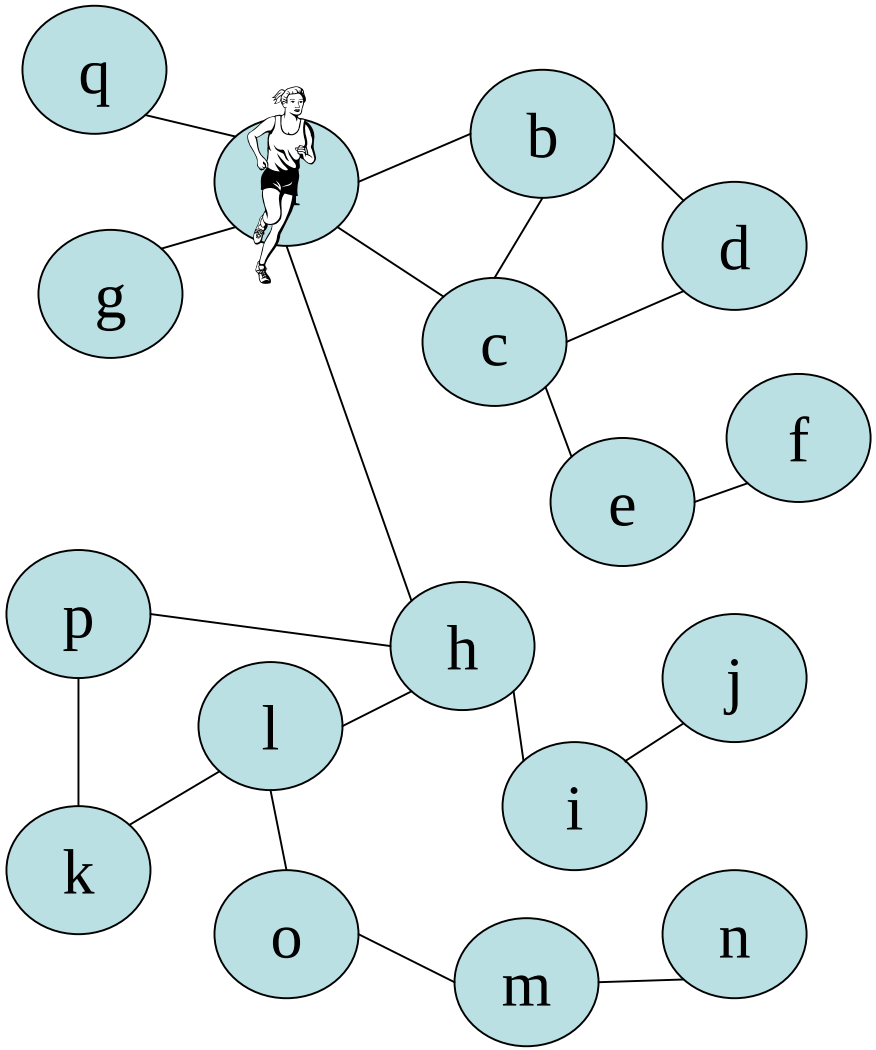
$$\boxed{\mathrm{TW}[k] = \mathsf{L}^{k+1}}$$

$$\mathsf{L}^{k+1} \ : \ \mathsf{L} \quad \text{with} \quad \text{at most} \quad k+1 \quad \text{vars.}$$

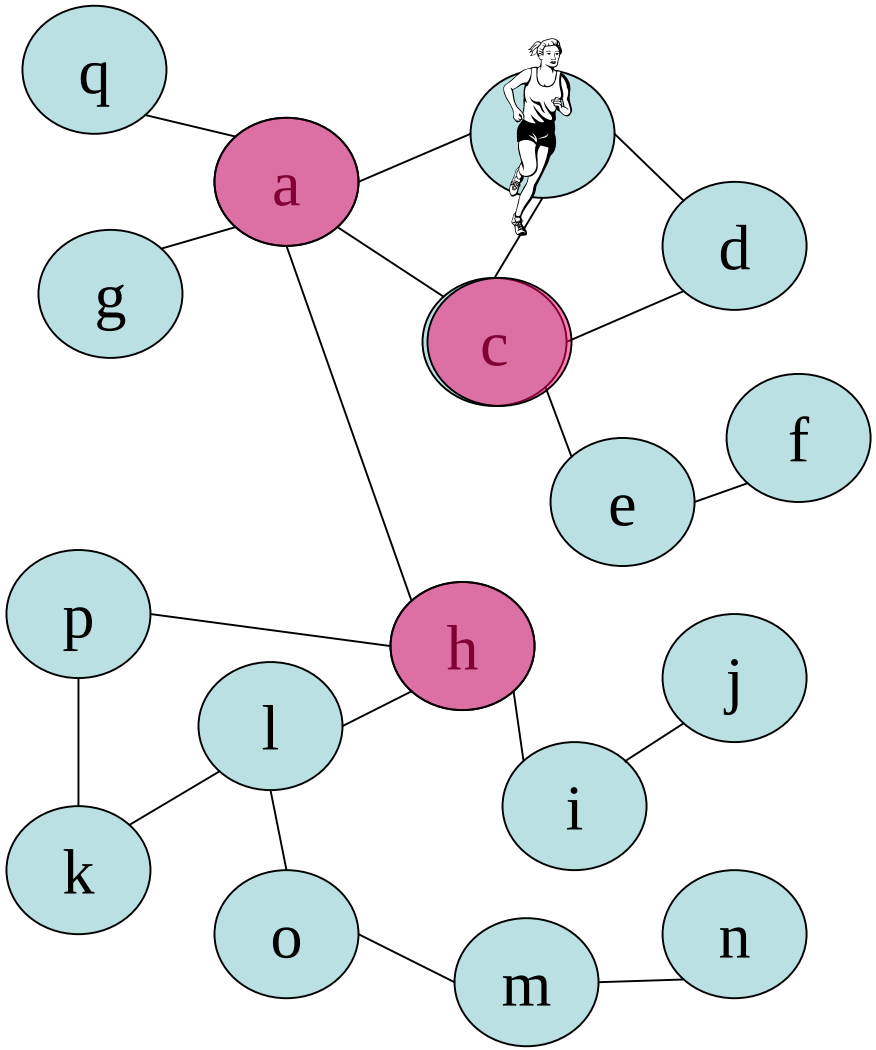
Game characterization of Treewidth

- A robber and *k* cops play the game on a graph
- The cops have to capture the robber
- Each cop controls a vertex of the graph
- Each cop, at any time, can fly to any vertex of the graph
- The robber tries to elude her capture, by running arbitrarily fast on the vertices of the graph,
  but on those vertices controlled by cops

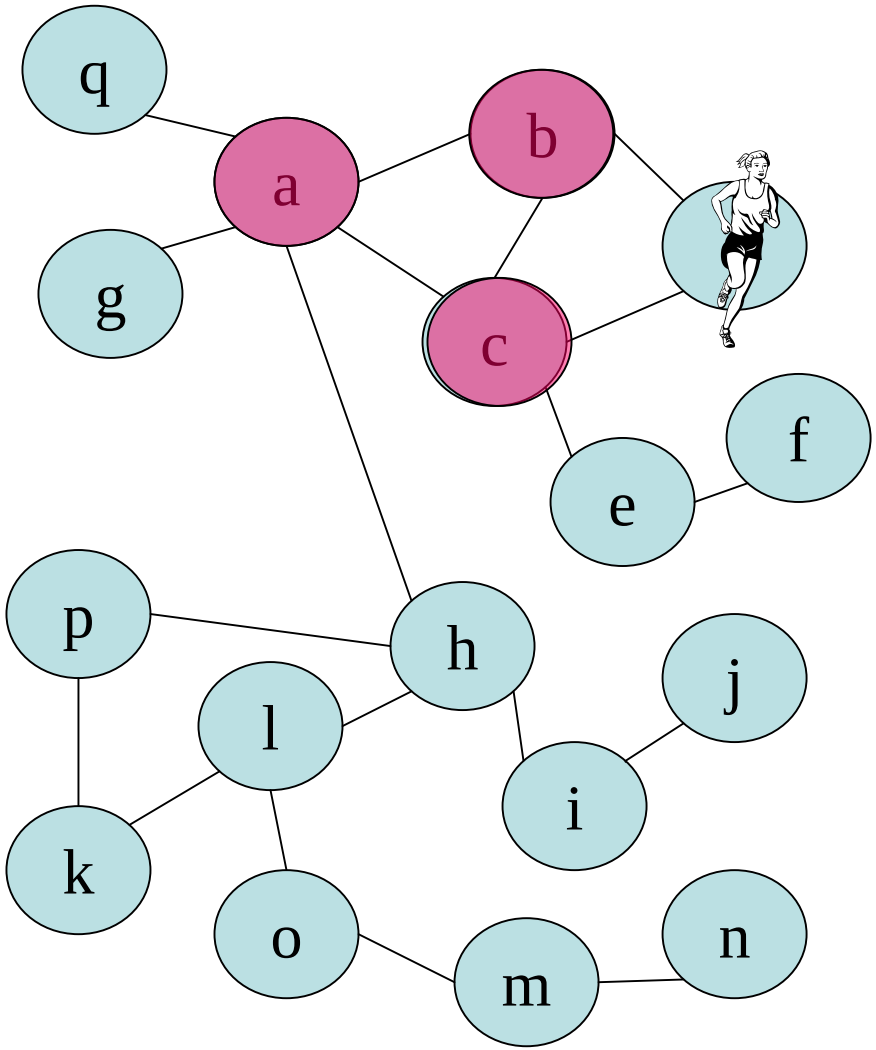
Playing the game



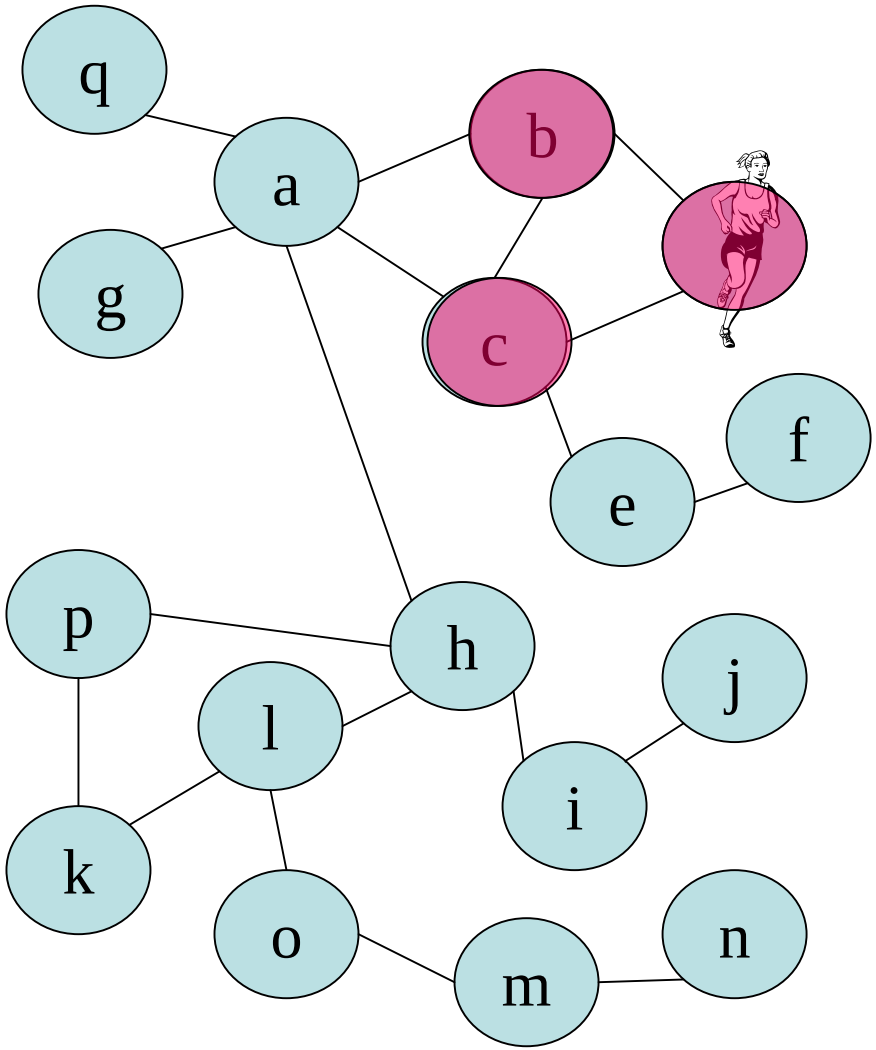
Playing the game



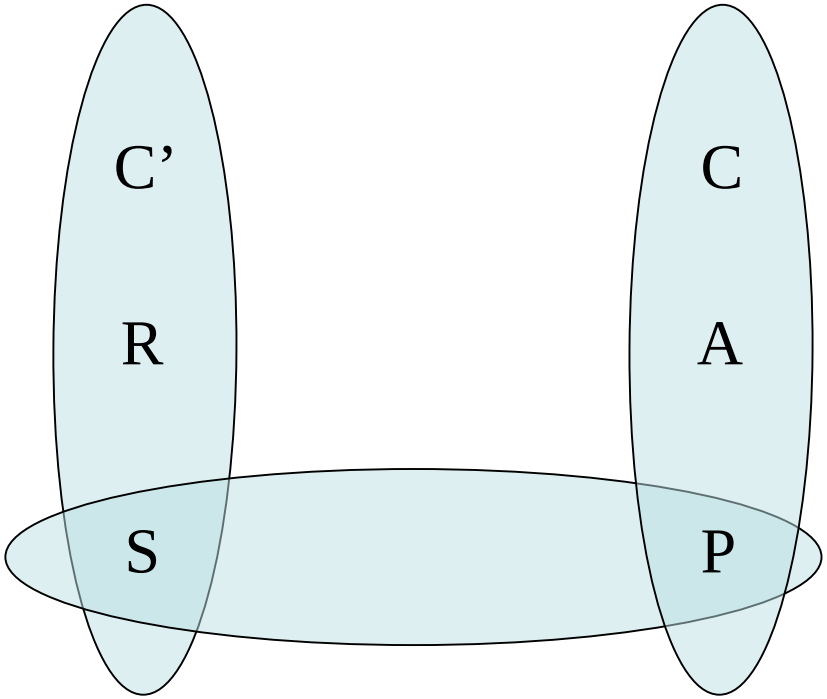
Playing the game



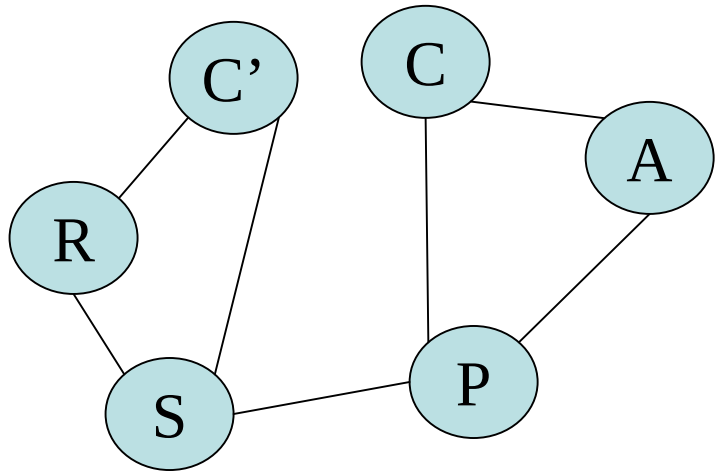
Playing the game



Hypergraphs vs Graphs (1)

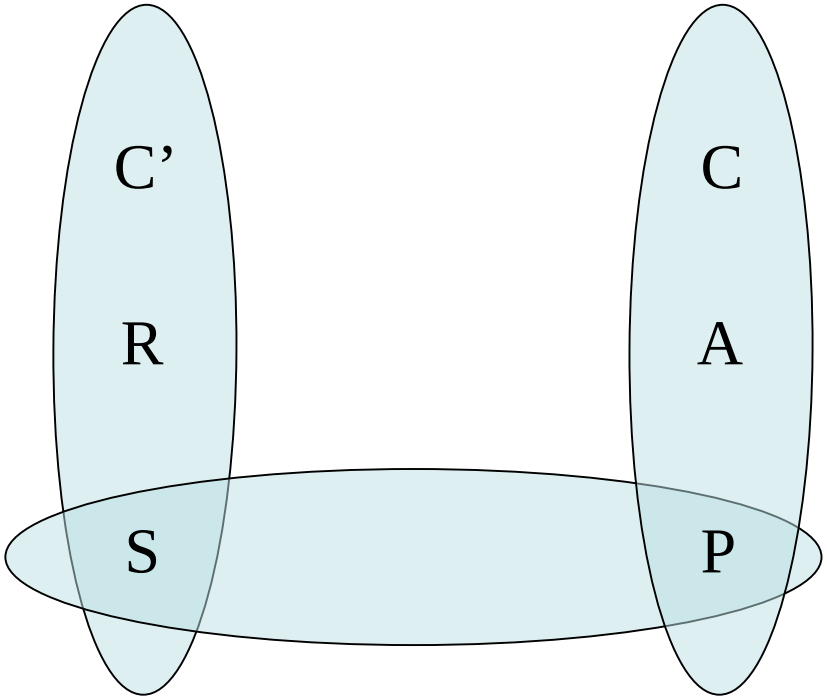


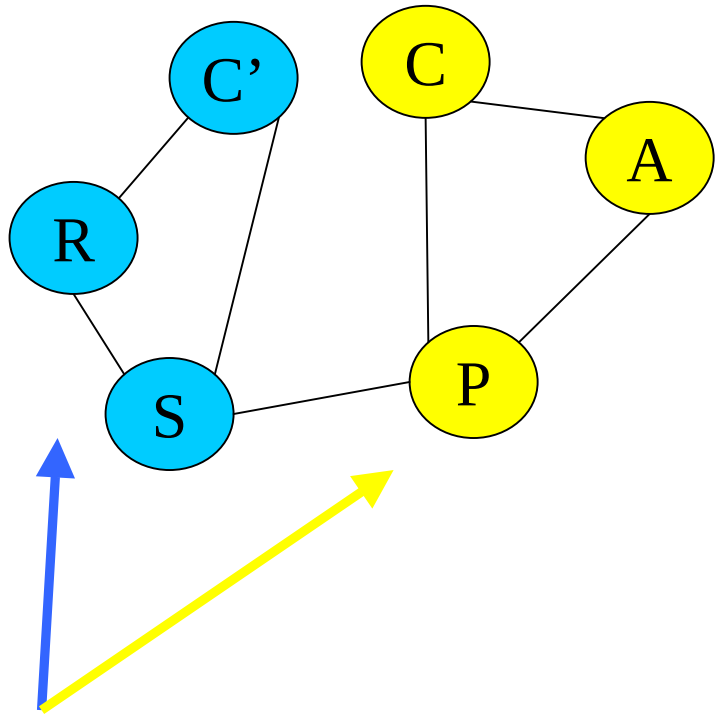
An acyclic hypergraph



Its cyclic primal graph

Hypergraphs vs Graphs (1)





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

Drawbacks of treewidth

Acyclic queries may have unbounded TW!

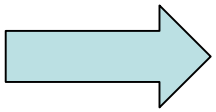
Example:

$q \leftarrow p_1(X_1, X_2, \ldots, X_n, Y_1) \wedge \ldots \wedge p_n(X_1, X_2, \ldots, X_n, Y_n)$

is acyclic, obviously polynomial,

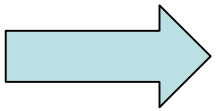
but has treewidth *n-1*

Beyond treewidth



Bounded Degree of Cyclicity (Hinges)

[Gyssens & Paredaens ’84, Gyssens, Jeavons, Cohen’94]
Does not generalize bounded treewidth.



Bounded Query width

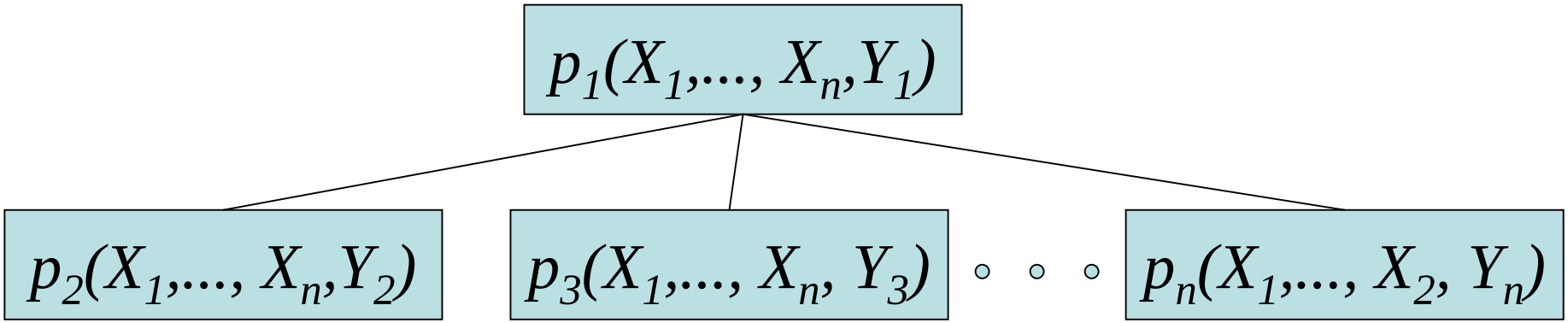
[Chekuri & Rajaraman ’97]

Group together query atoms (hyperedges) instead of variables

Query Decomposition

$$q \leftarrow p_1(X_1, X_2, \ldots, Y_1) \wedge \ldots \wedge p_m(X_1, X_2, \ldots, Y_n)$$

Query width = 1 = acyclicity

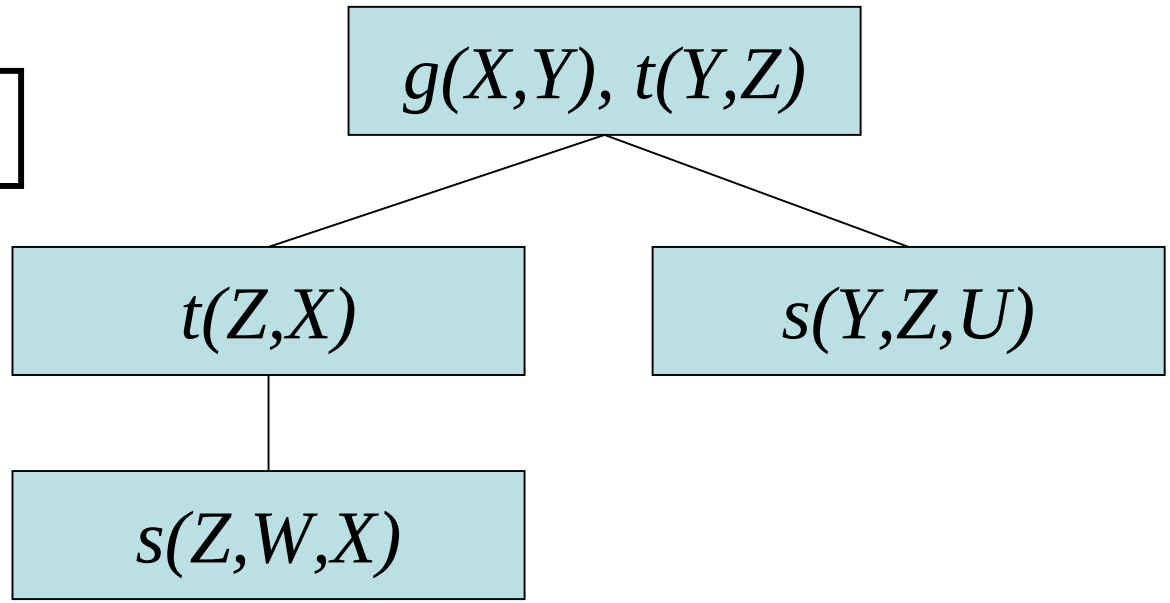


- Every atom/hyperarc appears in some node
- Connectedness conditions for variables and atoms

Decomposition of cyclic queries

$$q \leftarrow s(Y,Z,U) \wedge g(X,Y) \wedge t(Z,X) \wedge s(Z,W,X) \wedge t(Y,Z)$$

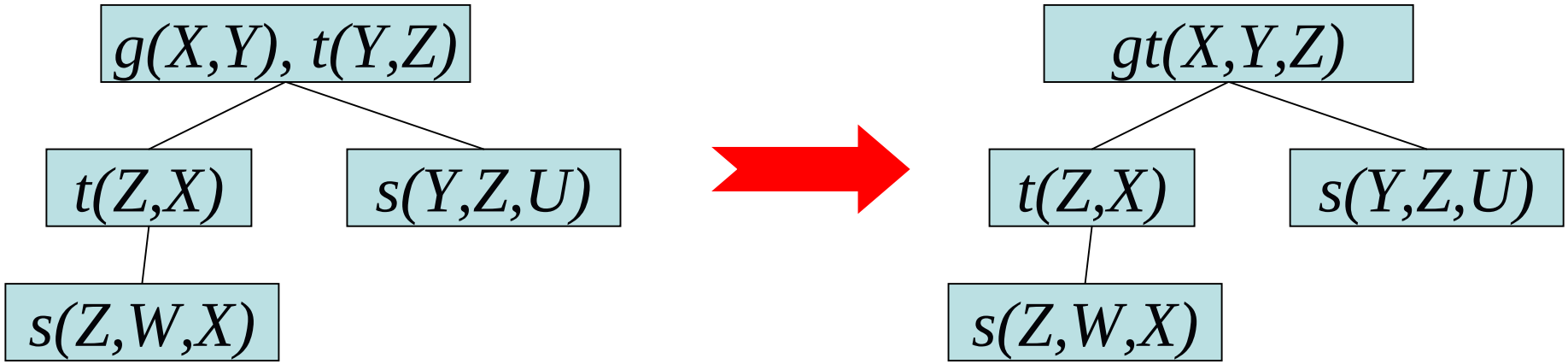
Query width = 2



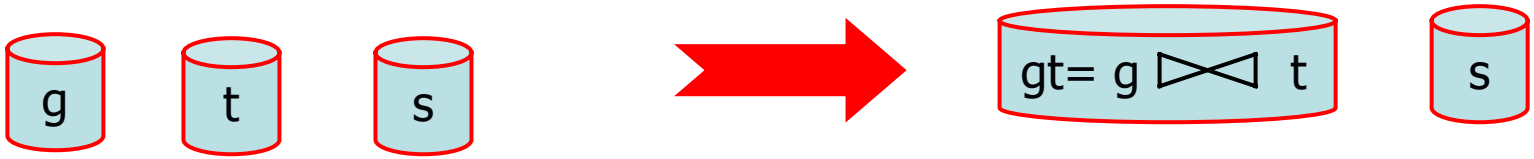
BCQ is polynomial for queries of bounded query width, **if** a query decomposition is given

Transform a query of bounded width into an acyclic query over a modified database

$$q \leftarrow s(Y,Z,U) \wedge g(X,Y) \wedge t(Z,X) \wedge s(Z,W,X) \wedge t(Y,Z)$$



Relations:



Relations:

Problems by Chekuri & Rajaraman '97

Are the following problems solvable in polynomial time for fixed *k* ?

- Decide whether *Q* has query width at most *k*
- Compute a query decomposition of *Q* of width *k*

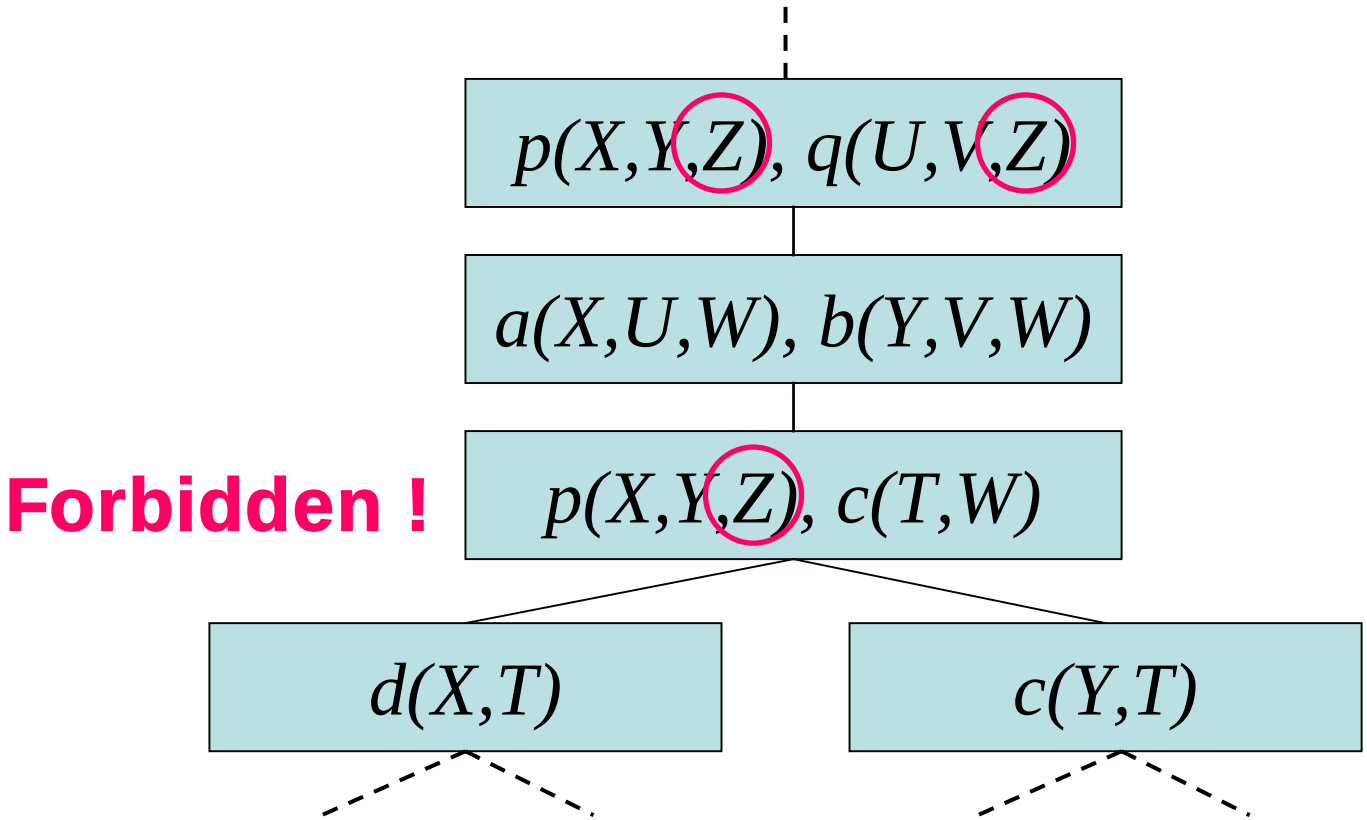
A negative answer (G.L.S. '99)

Theorem: Deciding whether a query has query width at most *k* is NP-complete

Proof: Very involved reduction from EXACT COVERING BY 3-SETS

Important Observation

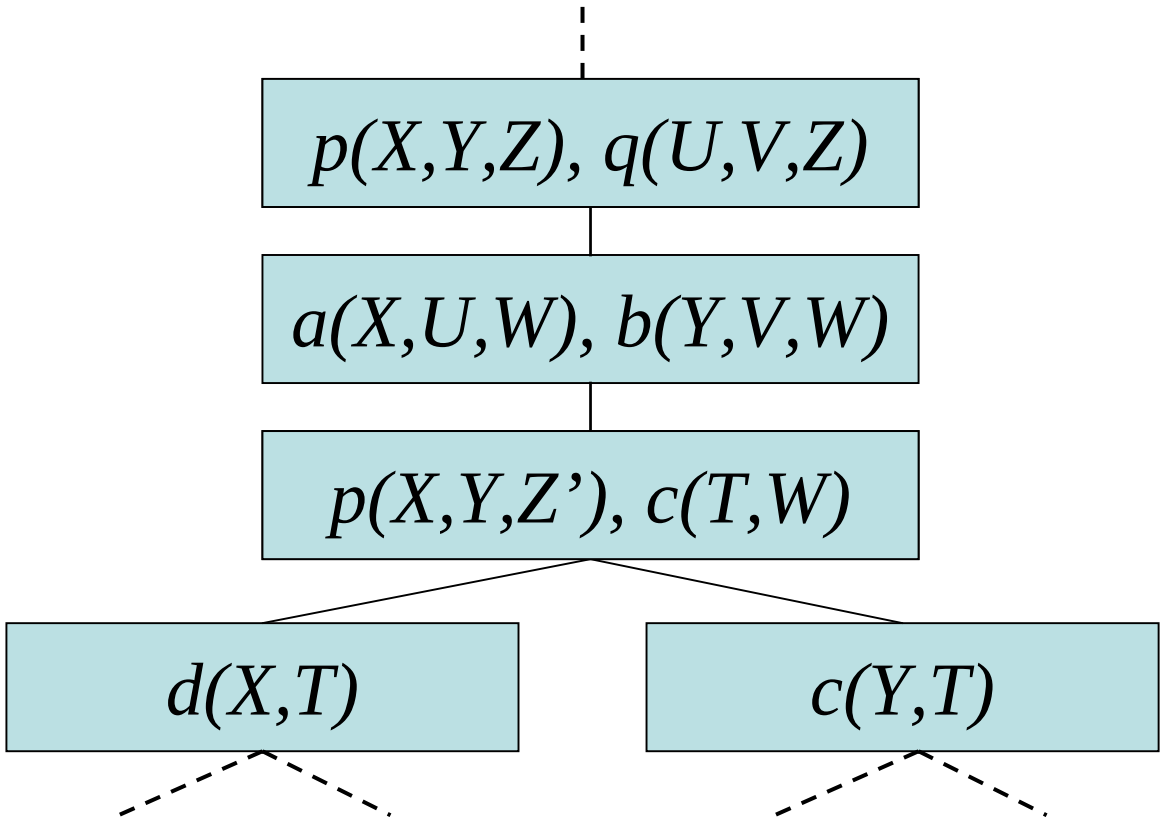
NP-hardness id due to an overly strong condition in the definition of query decomposition



Important Observation

But the reuse of *p(X,Y,Z)* is harmless here:

we could added an atom *p(X,Y,Z')* without changing the query



Hypertree Decompositions

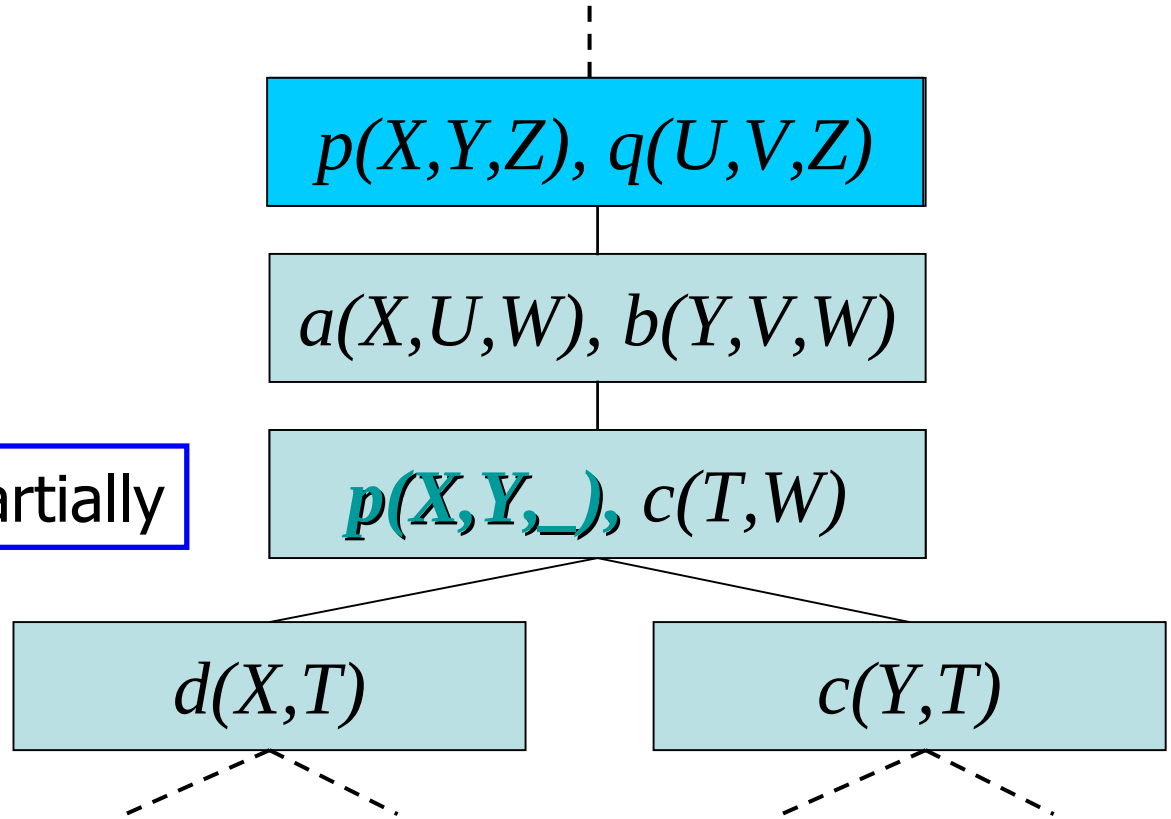
Query atoms can be used "partially" as long as the full atom appears somewhere else

More liberal than query decomposition

Grouping and Reusing Atoms

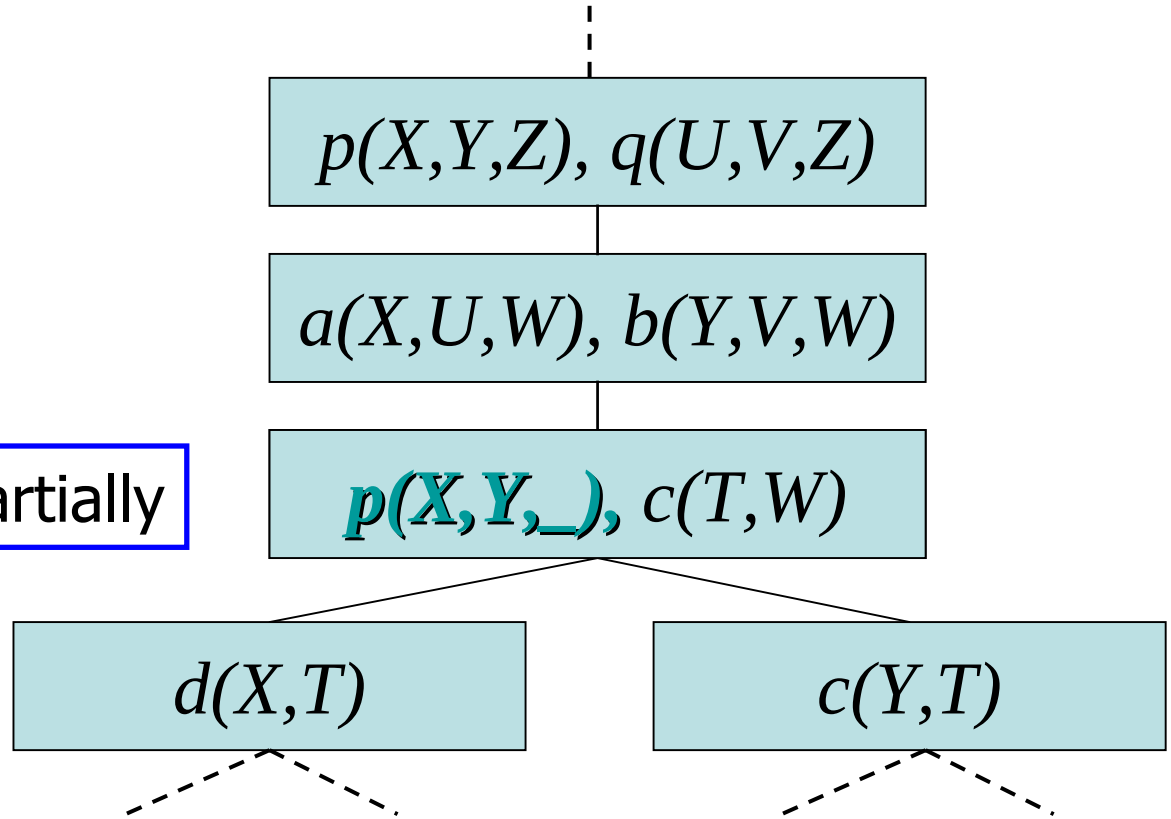
We group atoms

We use *p(X,Y,Z)* partially

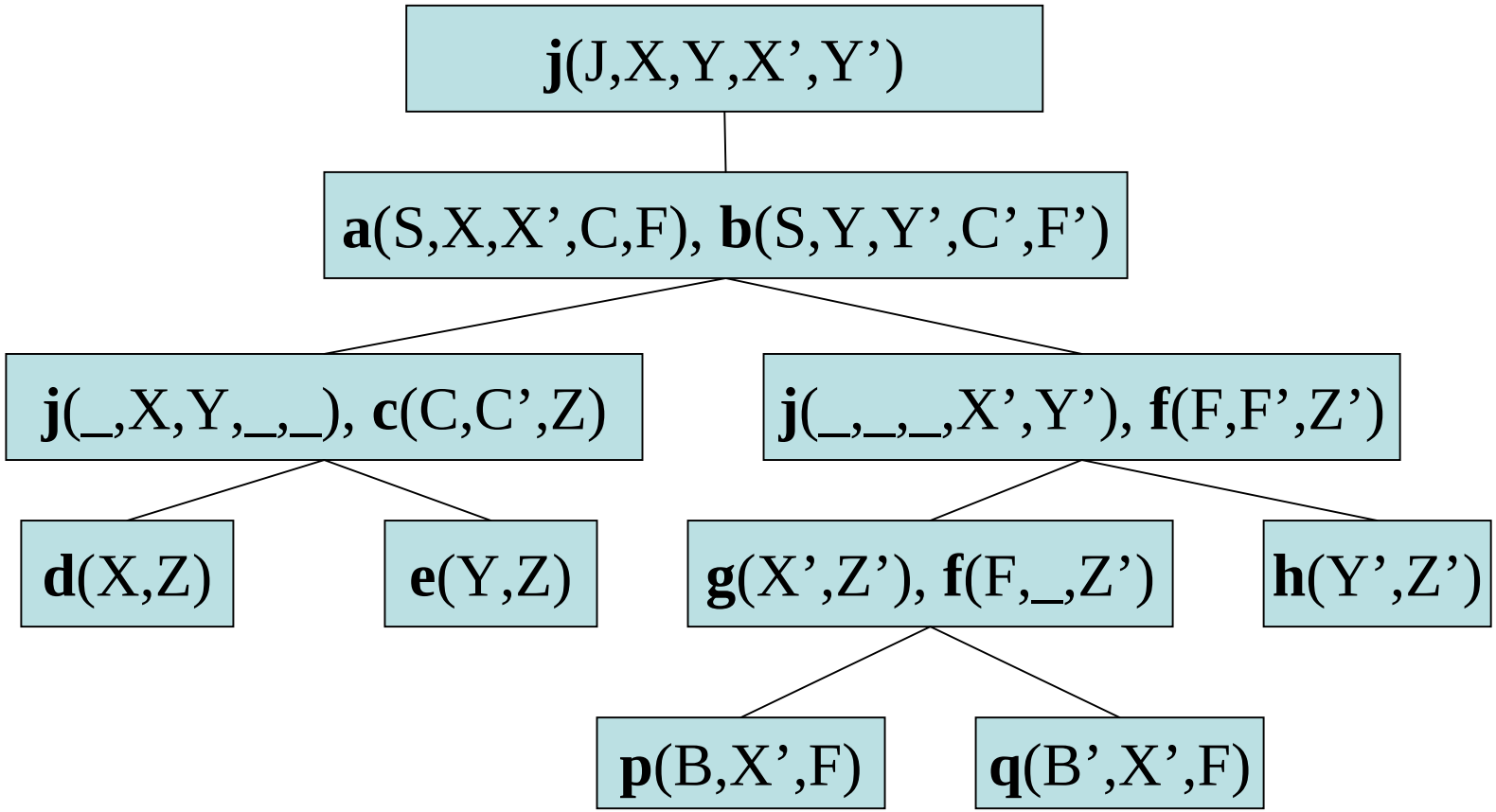


Reusing atoms

We use *p(X,Y,Z)* partially

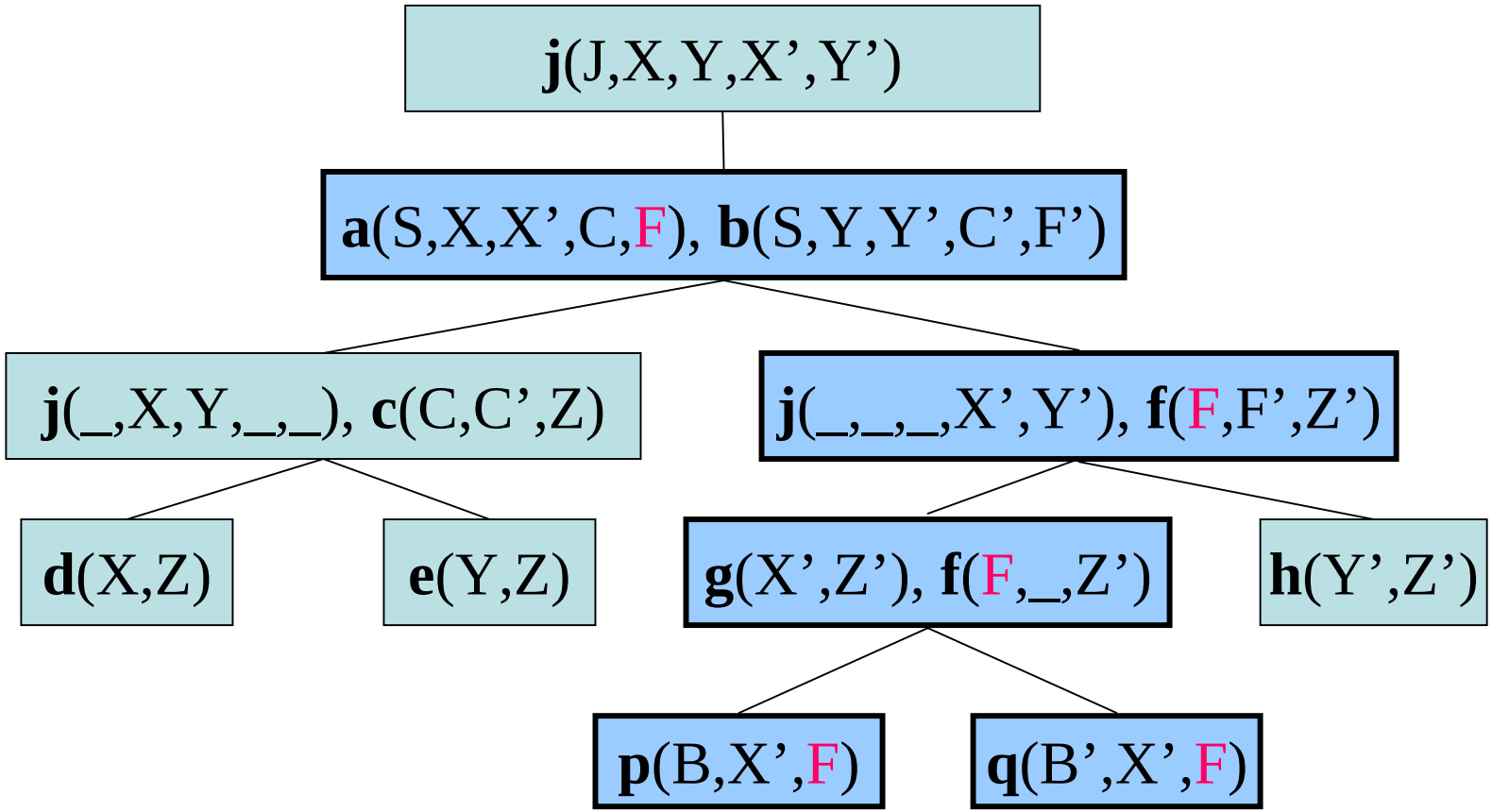


$$ans \leftarrow a(S,X,X',C,F) \wedge b(S,Y,Y',C',F') \wedge c(C,C',Z) \wedge d(X,Z) \wedge$$
$$e(Y,Z) \wedge f(F,F',Z') \wedge g(X',Z') \wedge h(Y',Z') \wedge$$
$$j(J,X,Y,X',Y') \wedge p(B,X',F) \wedge q(B',X',F)$$



Hypertree of width 2

Generalized Hypertree decomposition = Hypertree + Connectedness condition

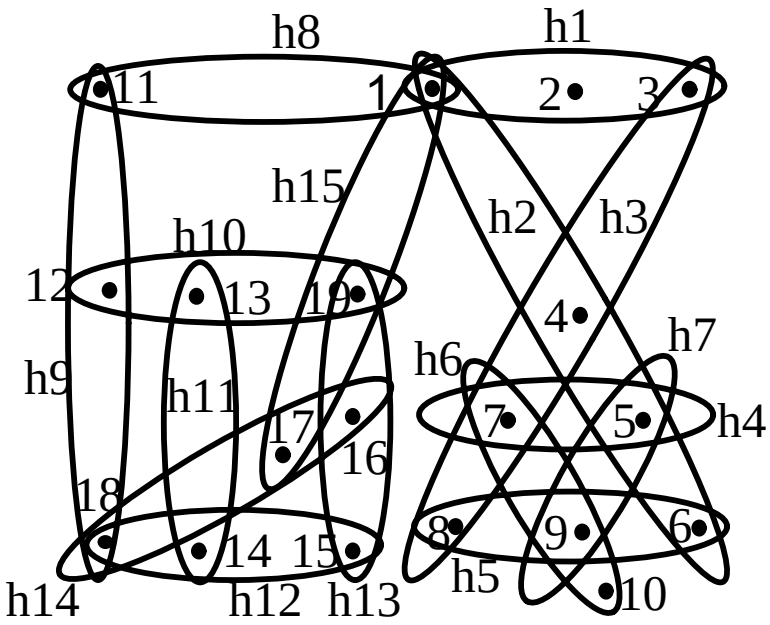


Alternative view and generalized HT-Decompositions:

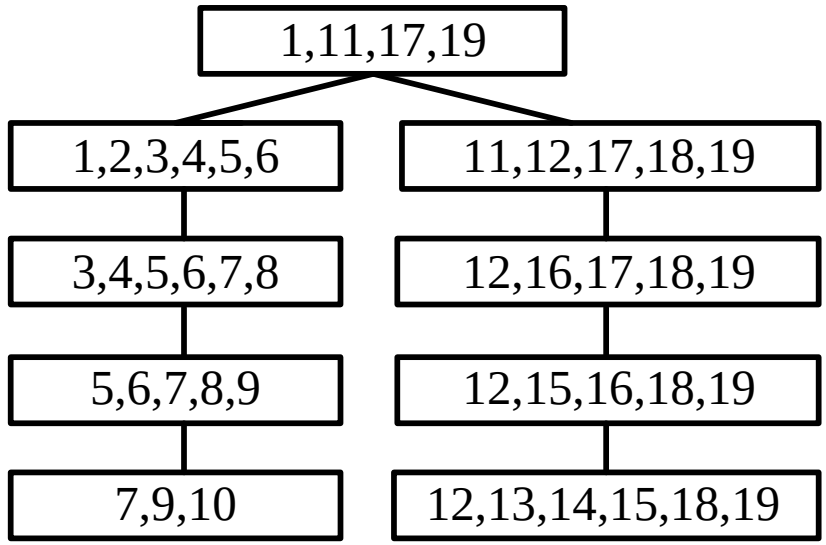
GHD = tree decomp + set covering

Tree decomposition of hypergraph

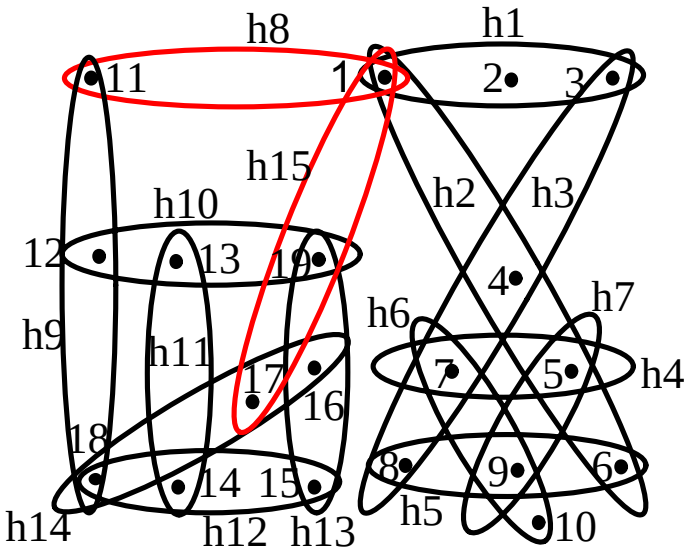
H

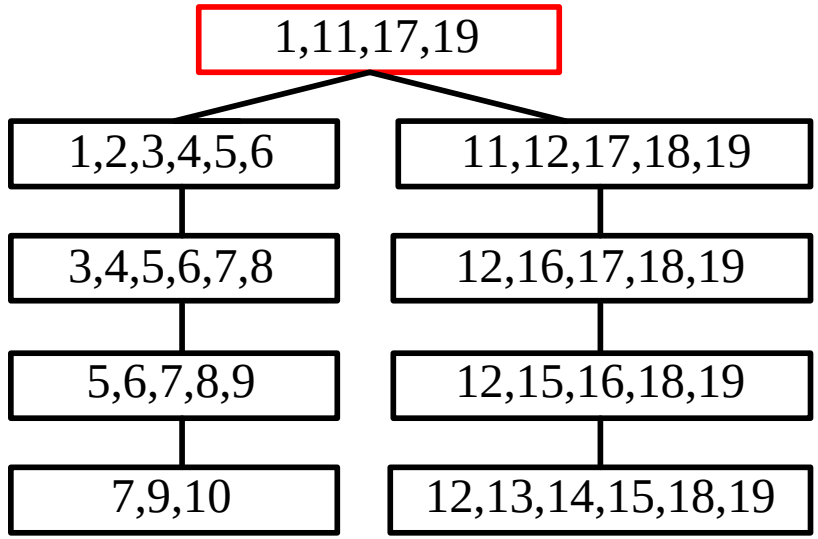


Tree decomp of G(H)

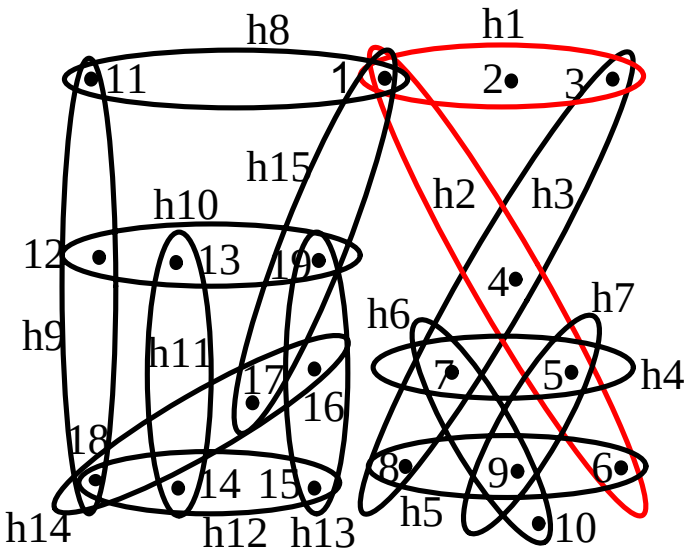


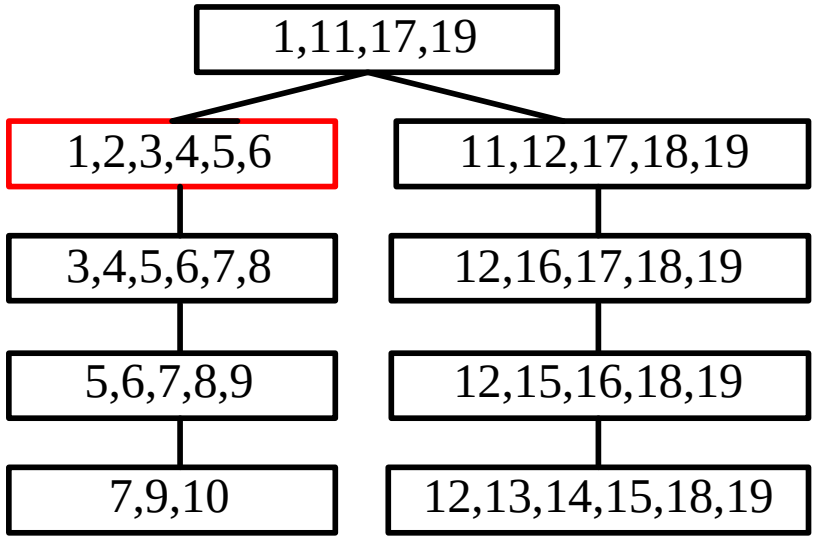
Hypergraph and the tree decomposition



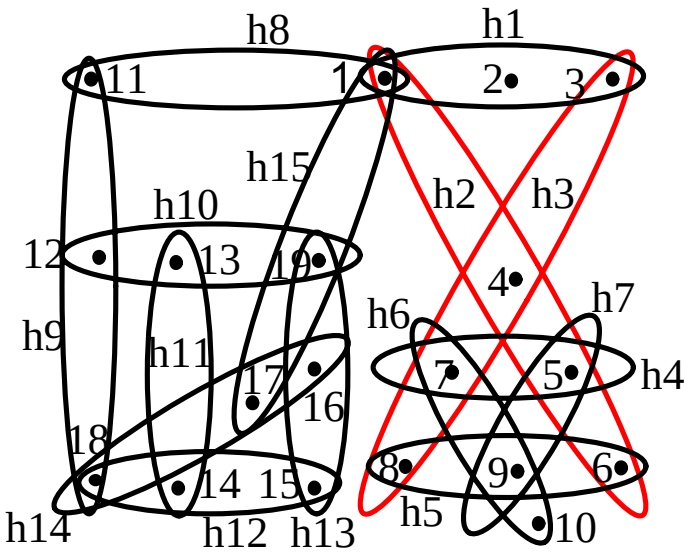


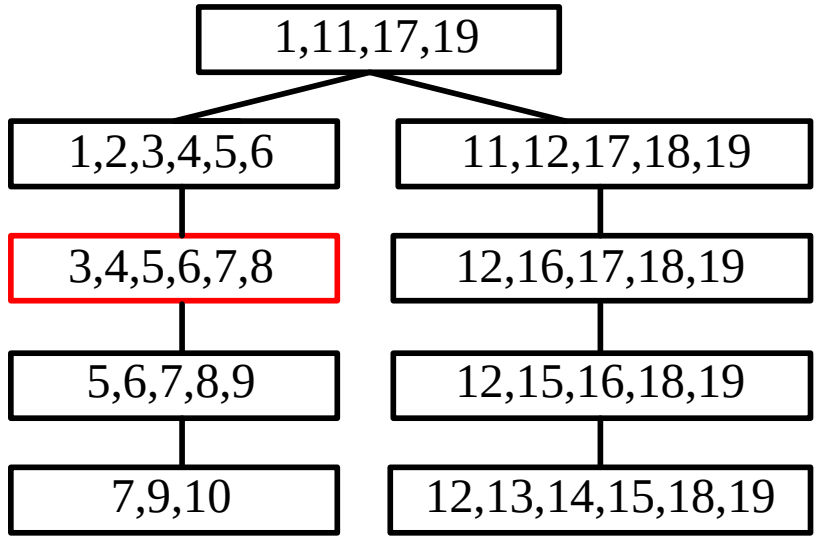
Hypergraph and the tree decomposition



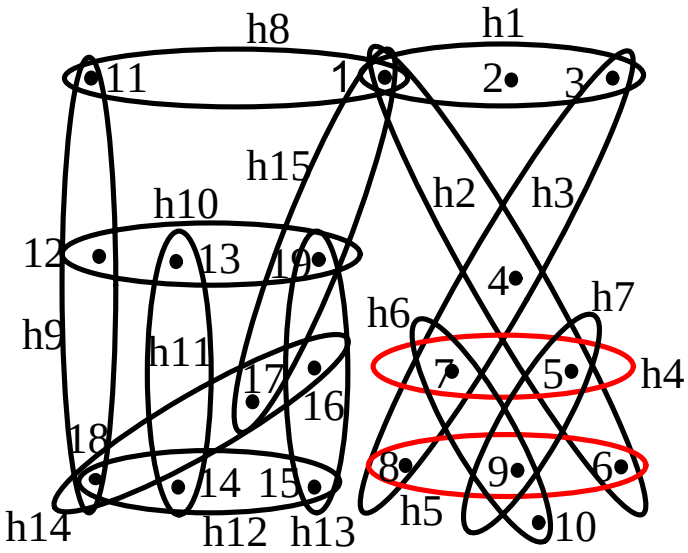


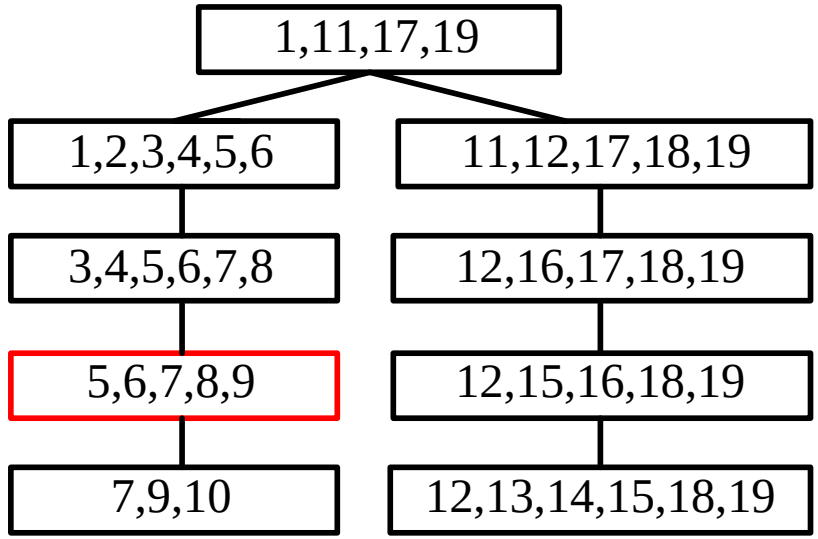
Hypergraph and the tree decomposition



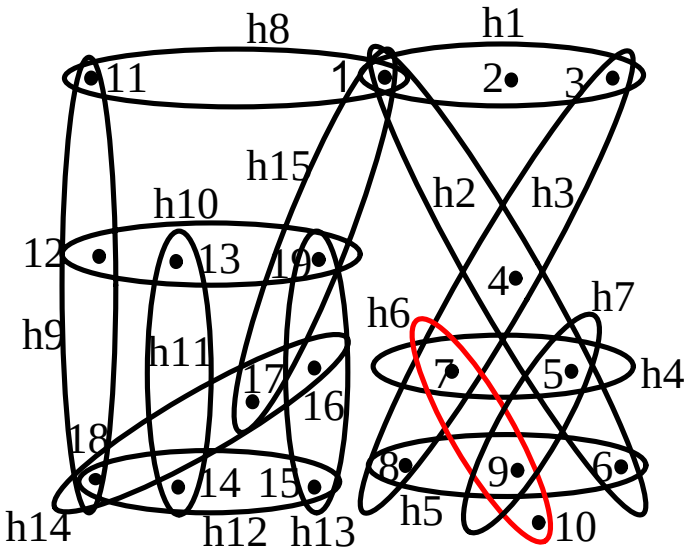


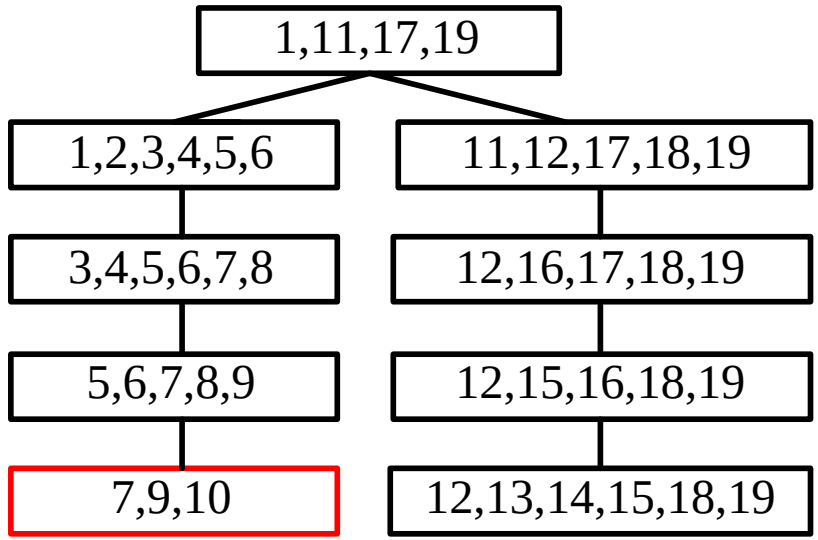
Hypergraph and the tree decomposition



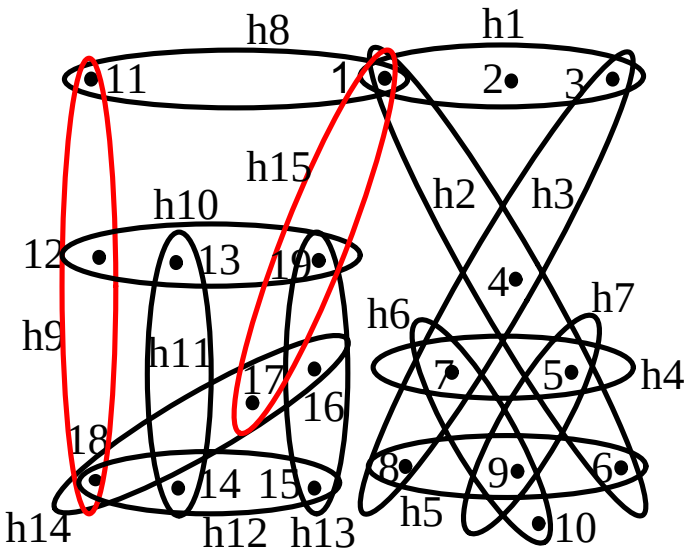


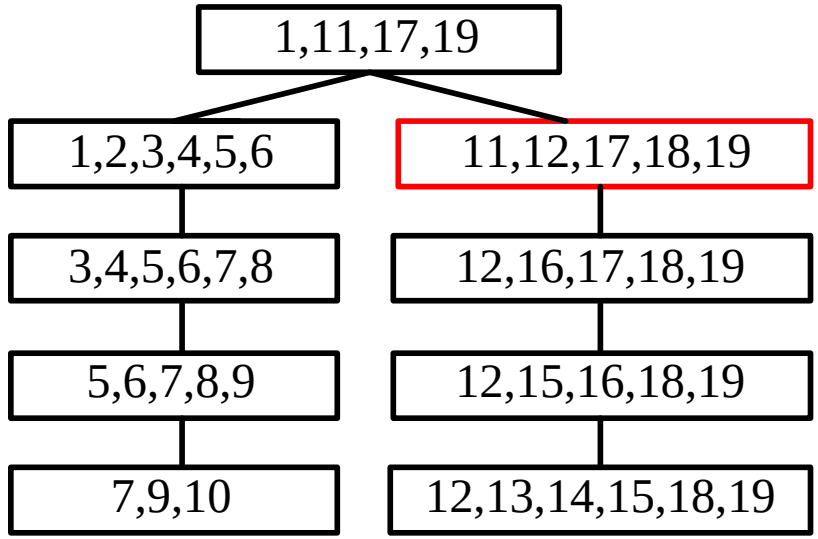
Hypergraph and the tree decomposition



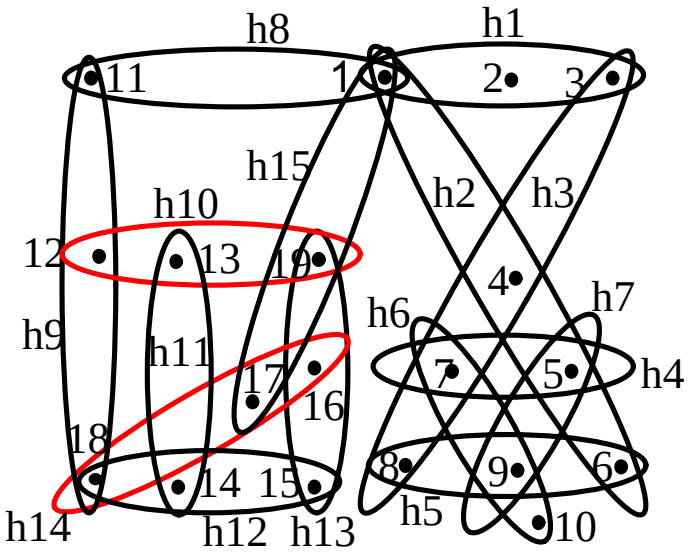


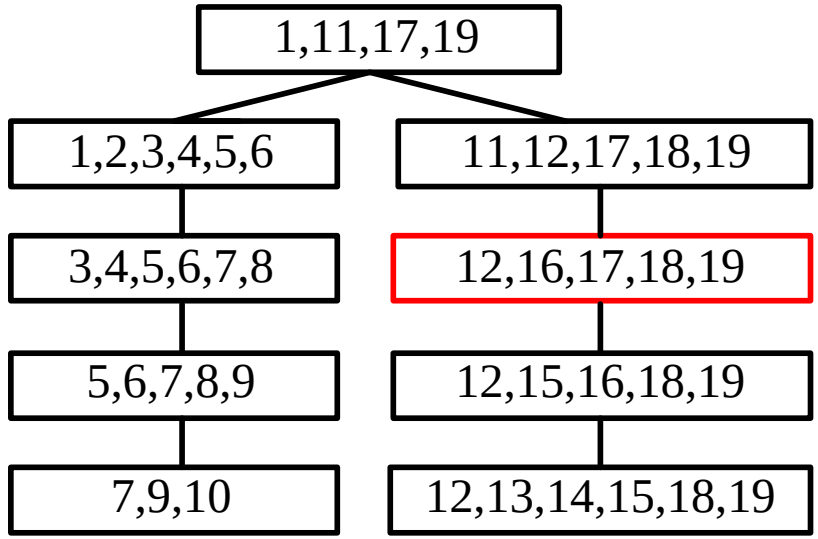
Hypergraph and the tree decomposition



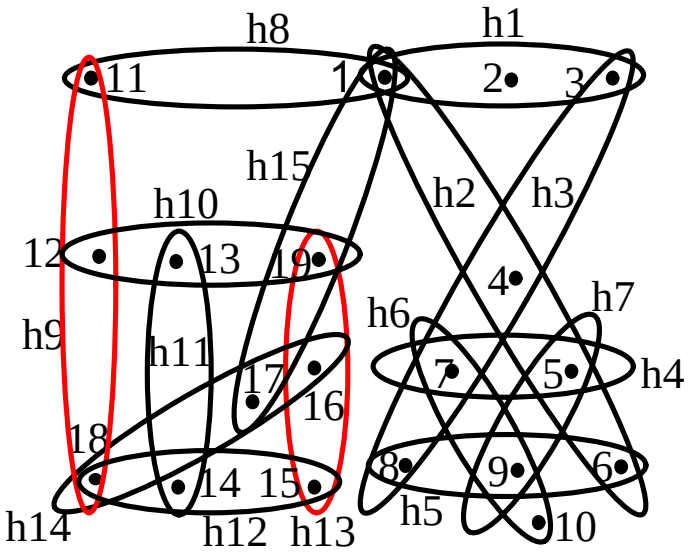


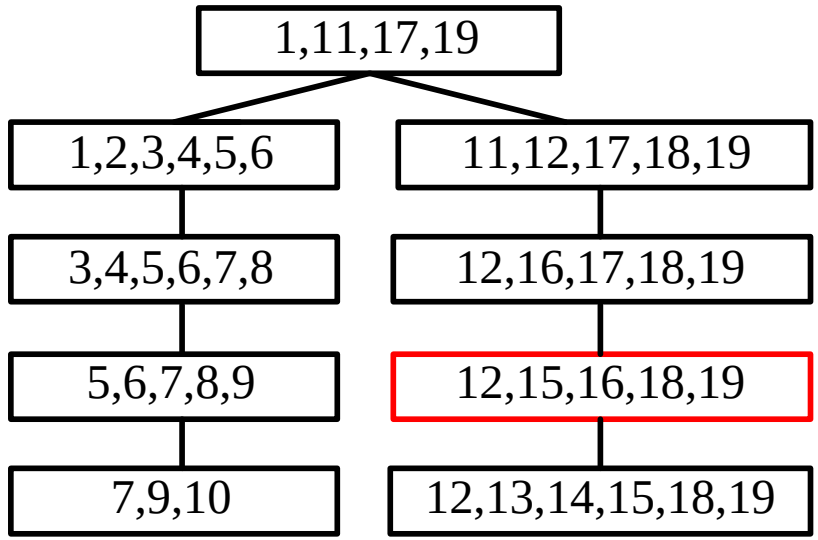
Hypergraph and the tree decomposition



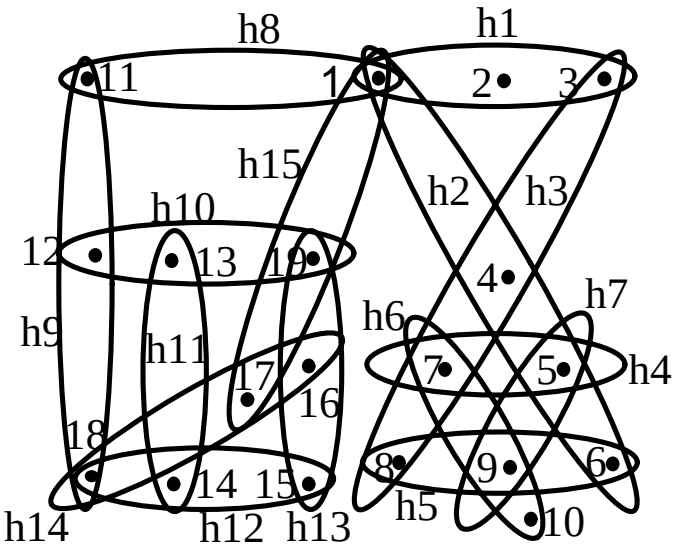


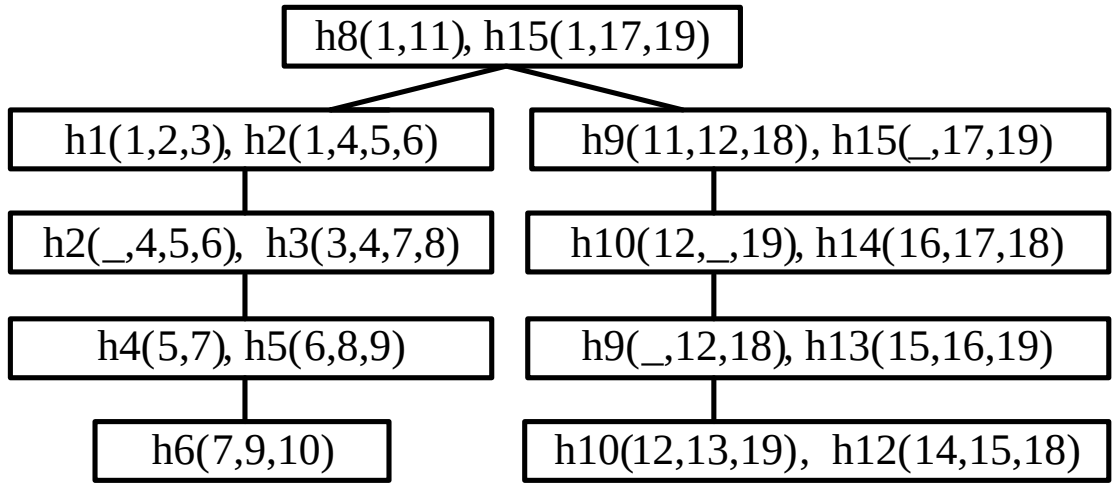
Hypergraph and the tree decomposition





Generalized hypertree decomposition





Generalized hypetree decomposition of width 2

QUESTION:

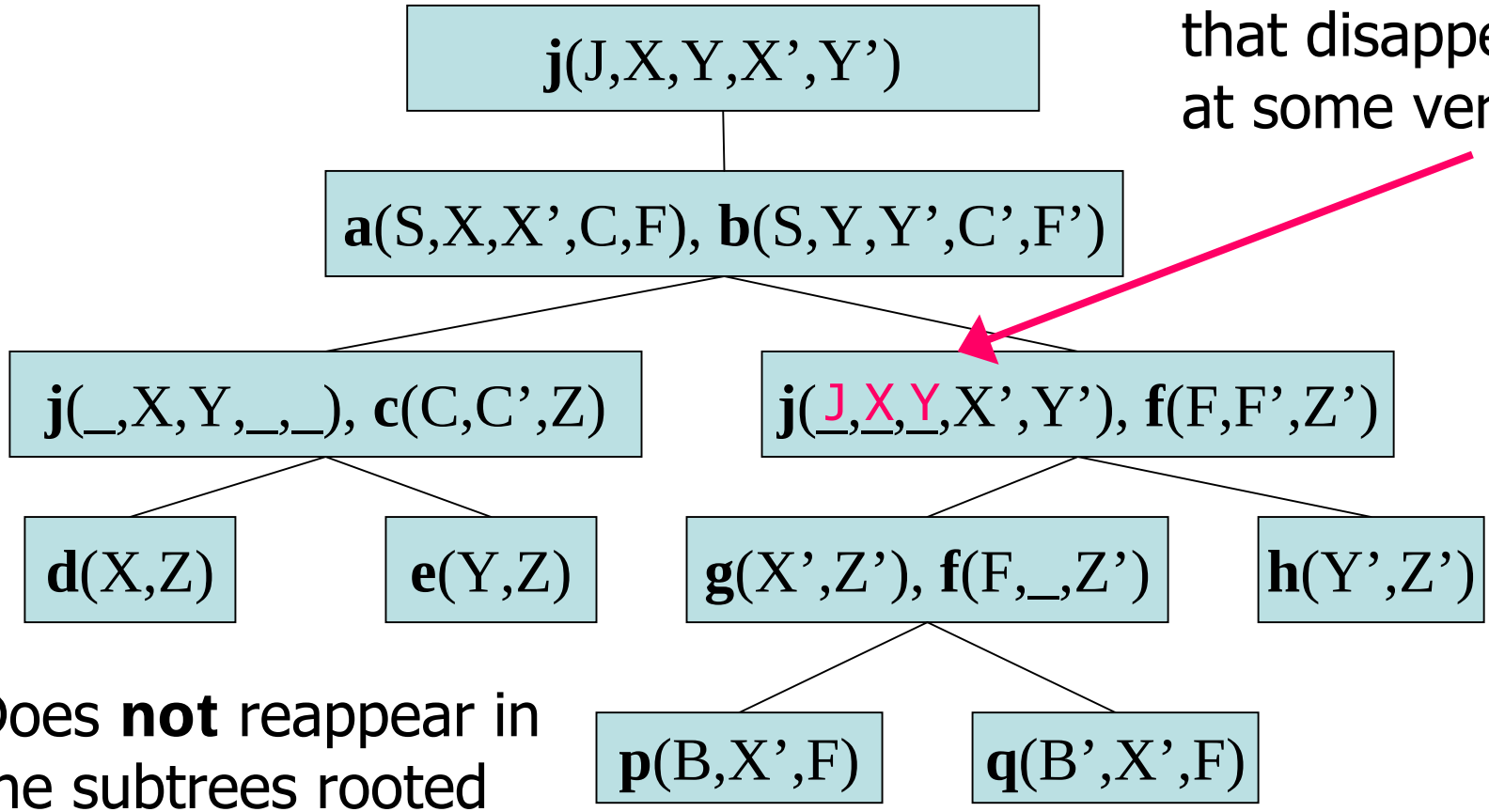
Can we determine in polynomial time

Whether ghw(H) < k for constant k

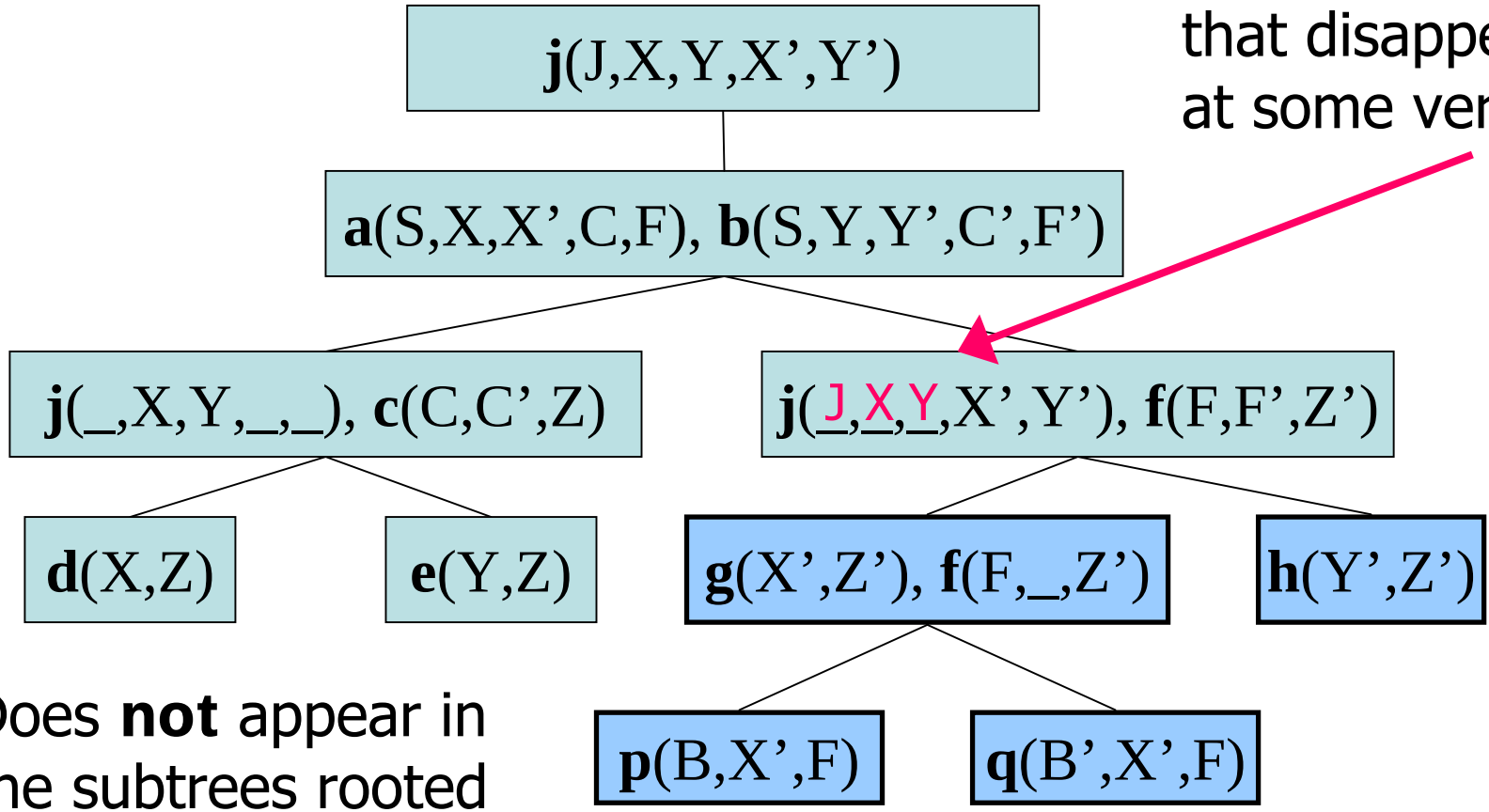
Announcement: ghw(H) < 4? NP-complete

[Schwentick et. al. 06]

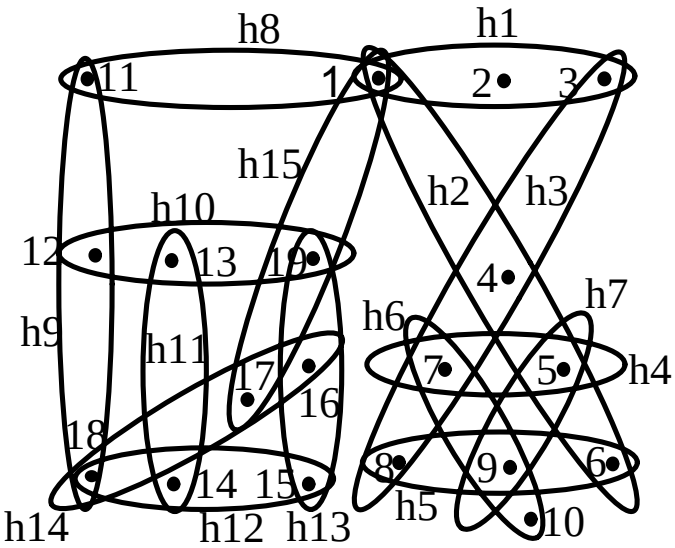
Hypertree Decomposition
= Generalized HTD +Special Condition



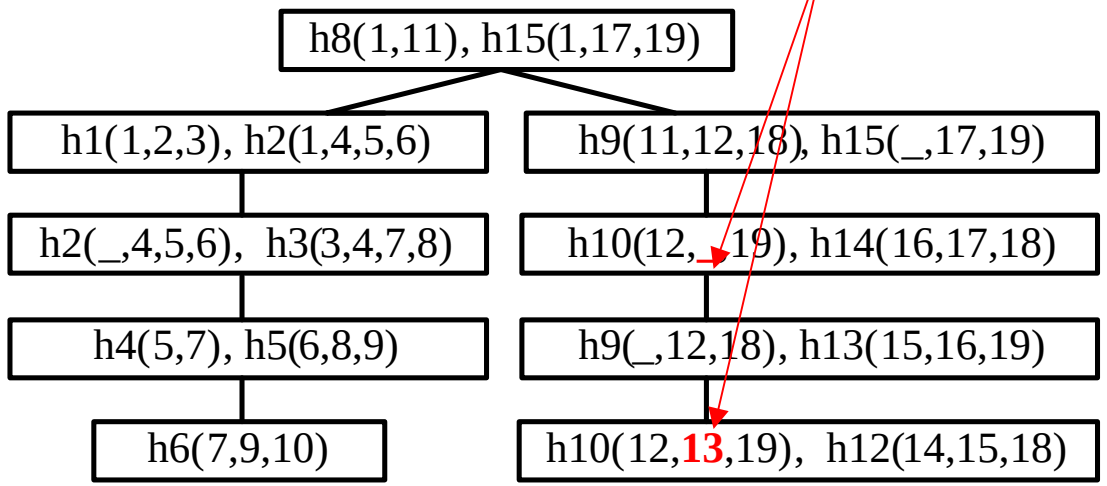
Special Condition



Generalized hypertree decomposition

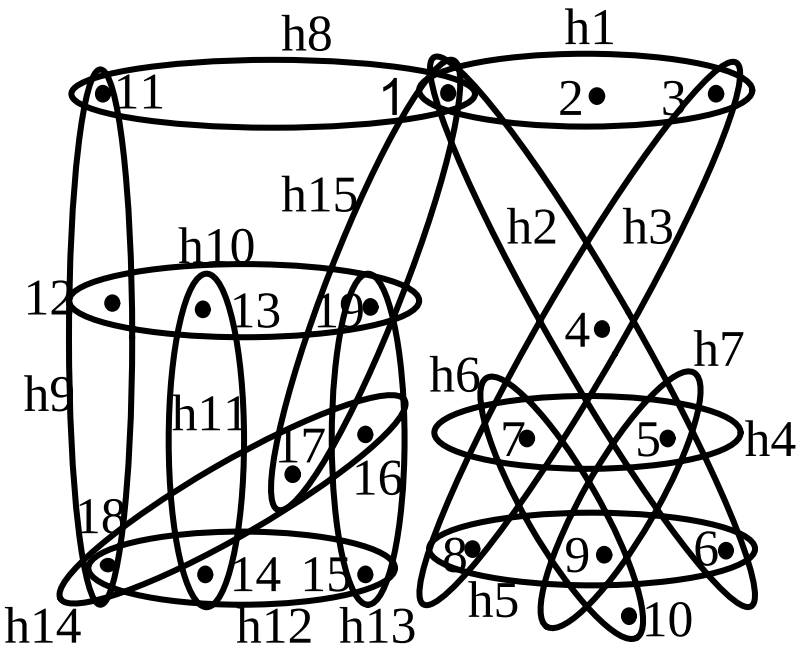


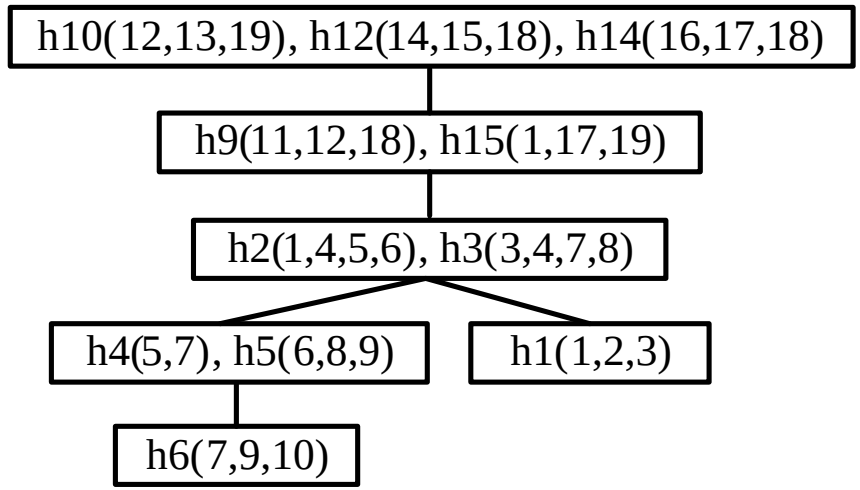
Special condition violated



Generalized hypetree decomposition of width 2

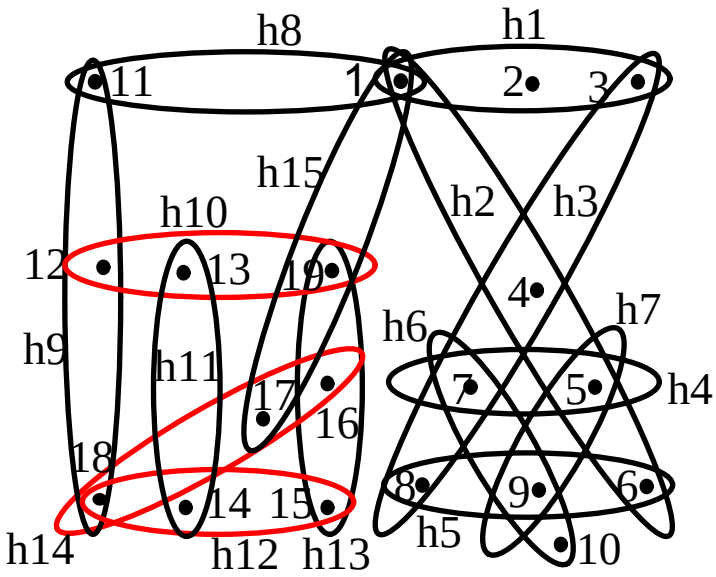
Hypertree decomposition

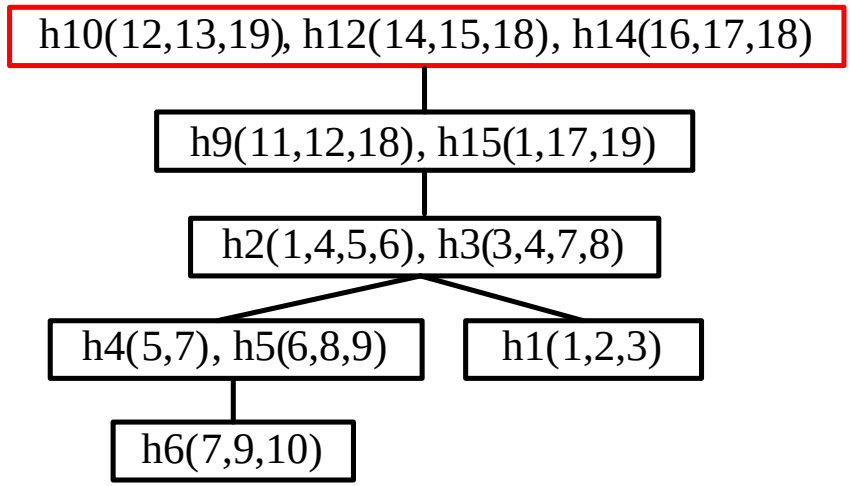




Hypertee decomposition of width 3

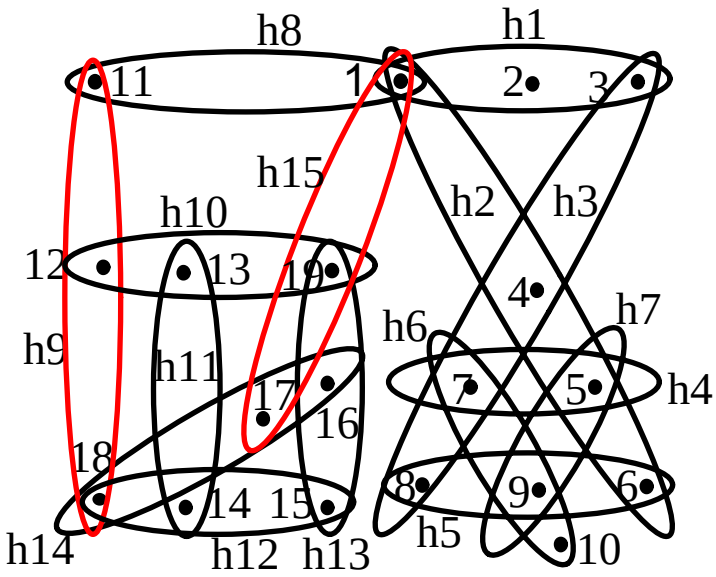
Hypertree decomposition

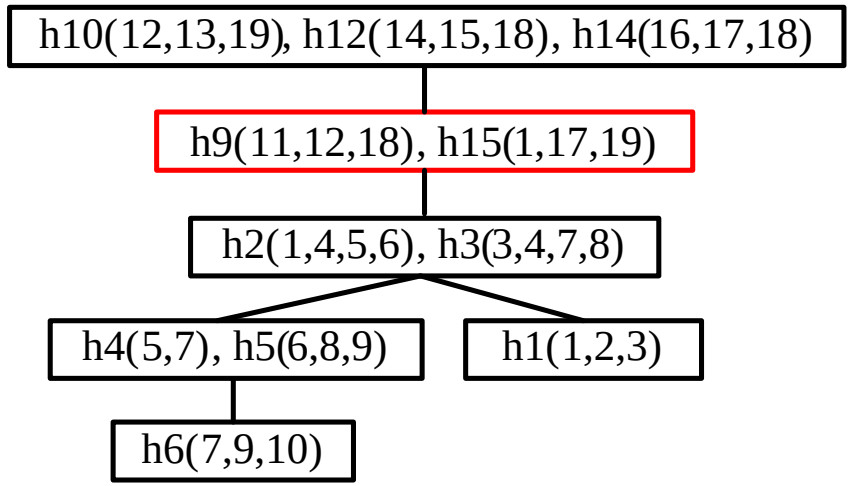




Hypertee decomposition of width 3

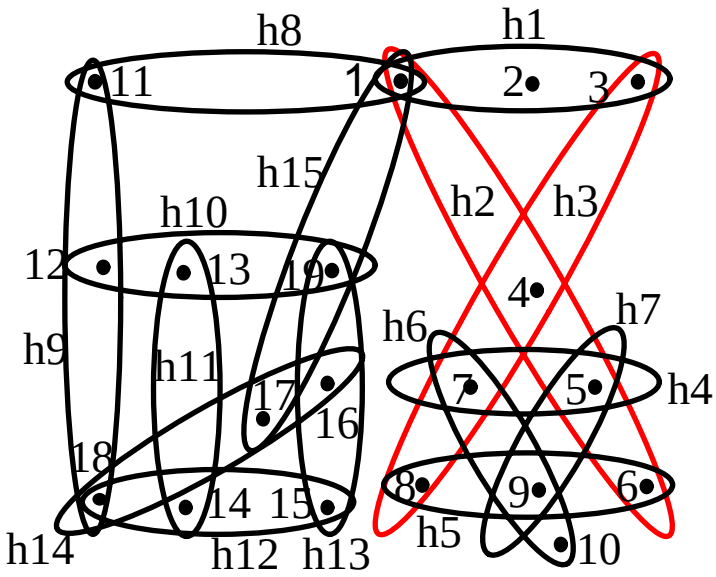
Hypertree decomposition

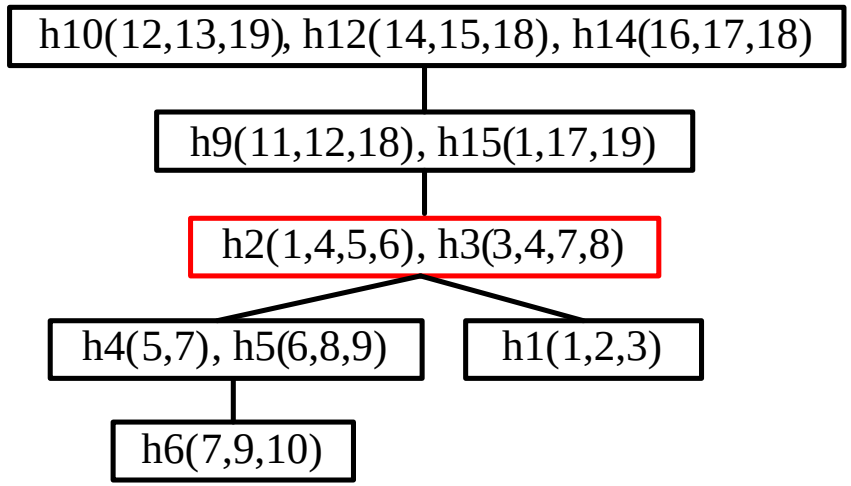




Hypertee decomposition of width 3

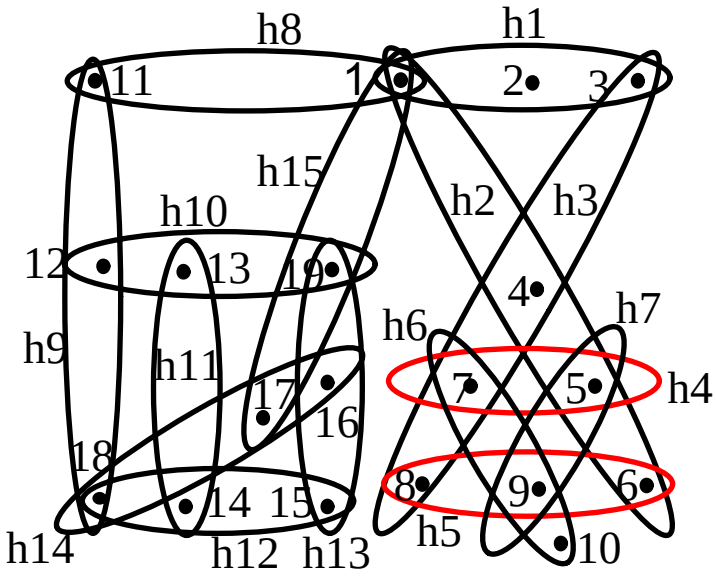
Hypertree decomposition

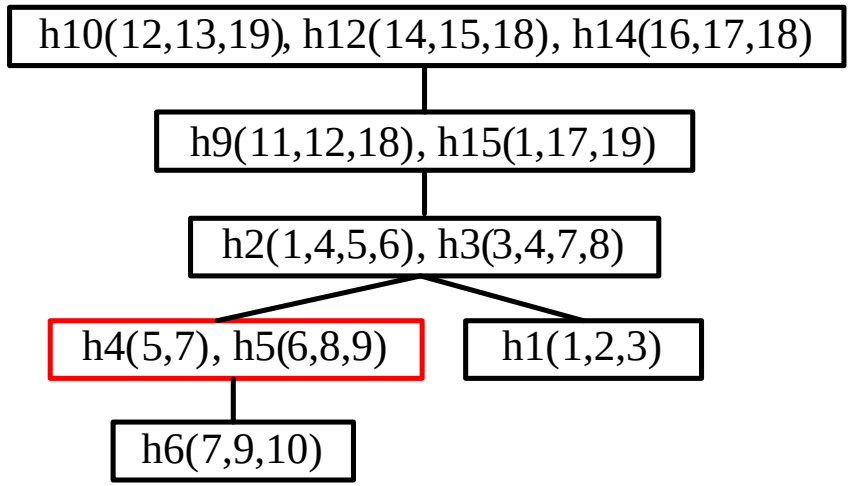




Hypertee decomposition of width 3

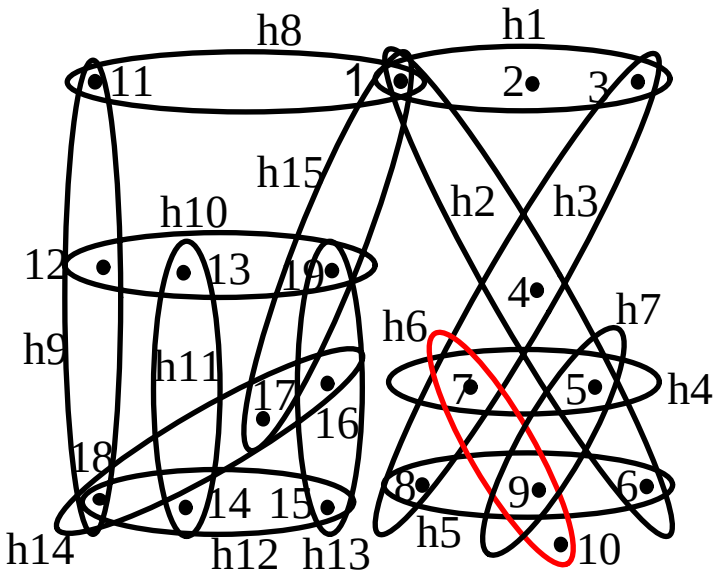
Hypertree decomposition

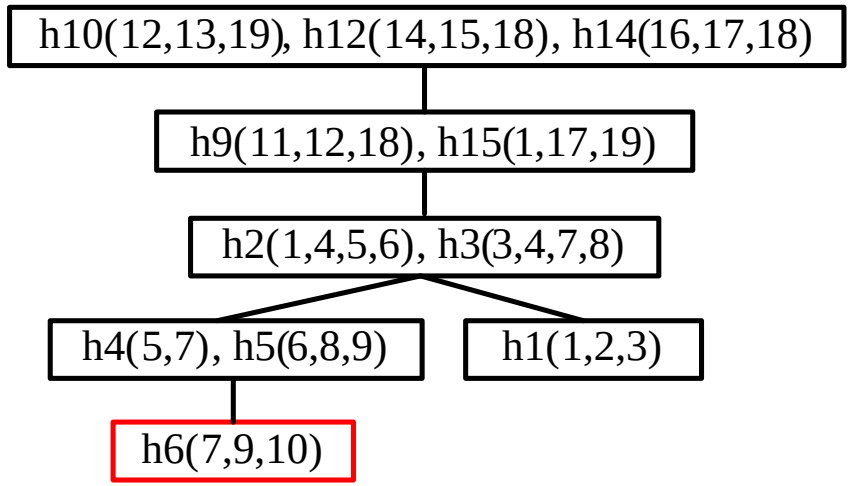




Hypertee decomposition of width 3

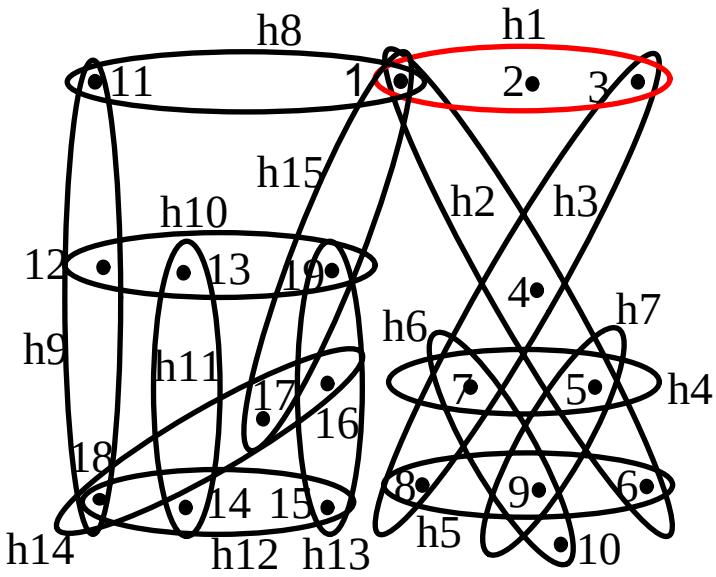
Hypertree decomposition

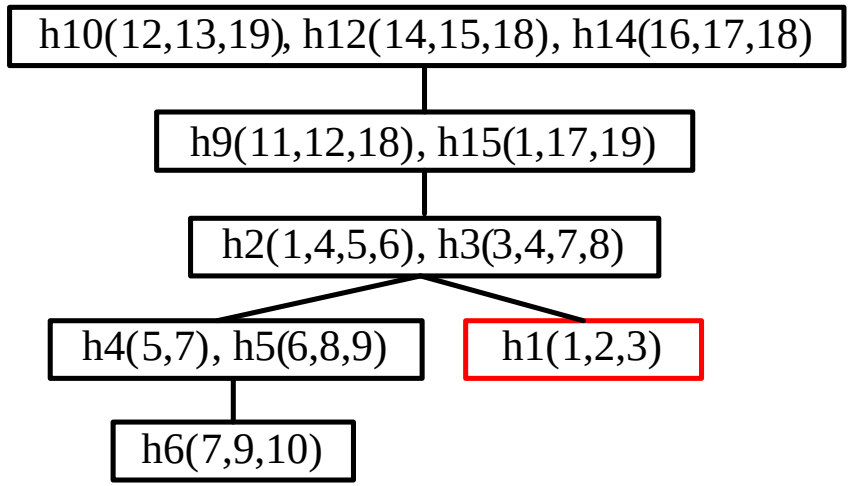




Hypertee decomposition of width 3

Hypertree decomposition





Hypertee decomposition of width 3

Positive Results on Hypertree Decompositions

- For each query Q, $hw(Q) \leq qw(Q)$
- In some cases, $hw(Q) < qw(Q)$
- For fixed k, deciding whether $hw(Q) \leq k$ is in polynomial time (LOGCFL)
- Computing hypertree decompositions is feasible in polynomial time (for fixed k).

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

Evaluating CSP's having bounded (gen.) hypertree width

k fixed

Given:

a database db of relations

a CSP *Q* over db such that $hw(Q) \leq k$ *or* $ghw \leq k$

a width *k* hypertree decomposition of *Q*

- Deciding whether *(Q,db) solvable* is in $O(n^{k+1} \log n)$ and complete for LOGCFL

- Computing *Q(db)* is feasible in output-polynomial time

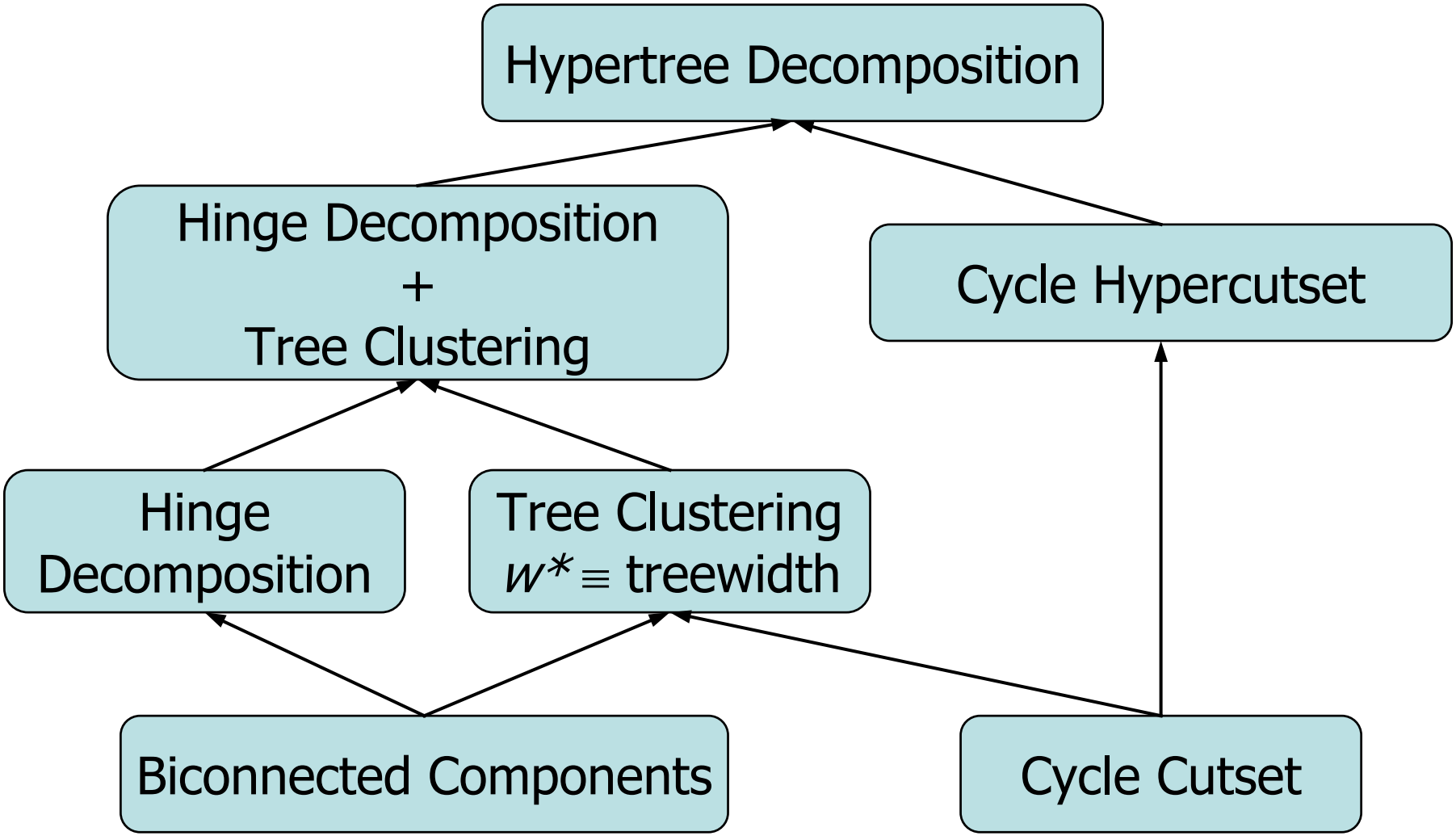
Observation: If H has n vertices, then

$$HW(H) \leq n/2+1$$

Does not hold for TW: $TW(K_n)=n-1$

Often HW < TW. H-Decomps are interesting
In case of bounded arity, too.

Comparison results



Relationship GHW vs. HW:

Observation:

ghw(H) = hw(H*)

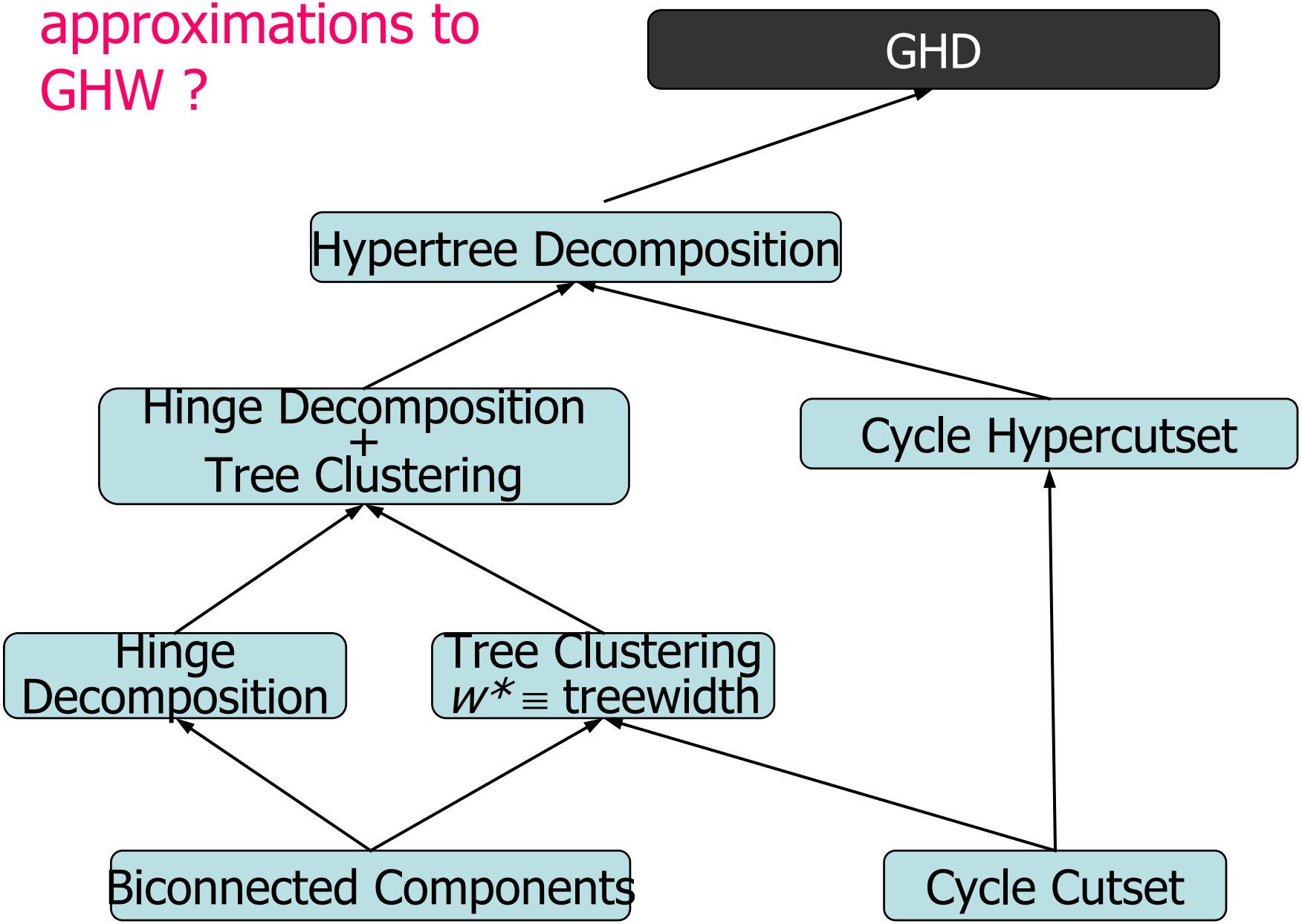
where $H^* = H \cup \{E' \mid \exists E \text{ in edges}(H): E' \subseteq E\}$

Exponential!

Approximation Theorem [Adler,G.,Grohe 05] :

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

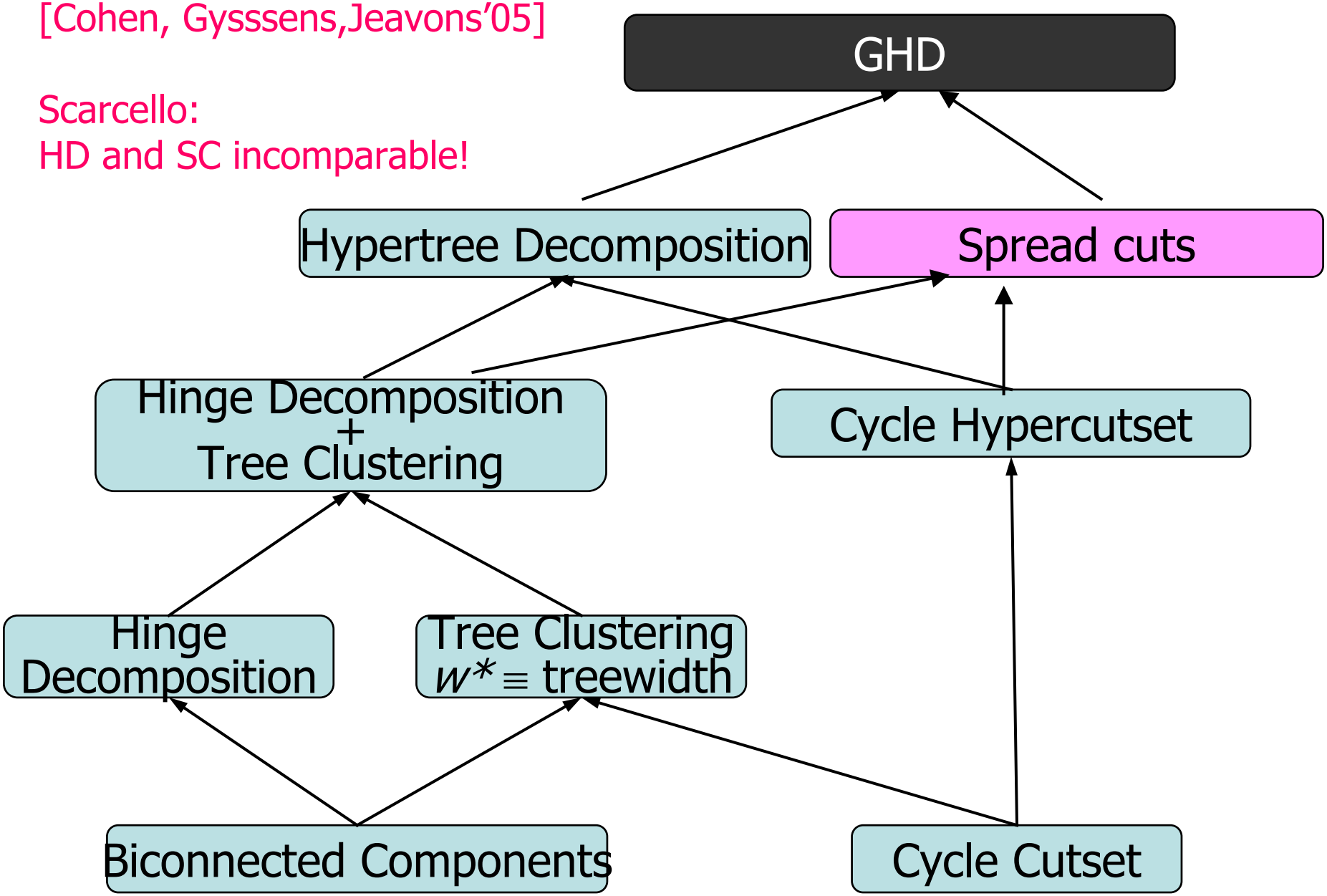
Q: Are there other approximations to GHW ?



YES: E.g. Spread Cuts

[Cohen, Gysssens,Jeavons'05]

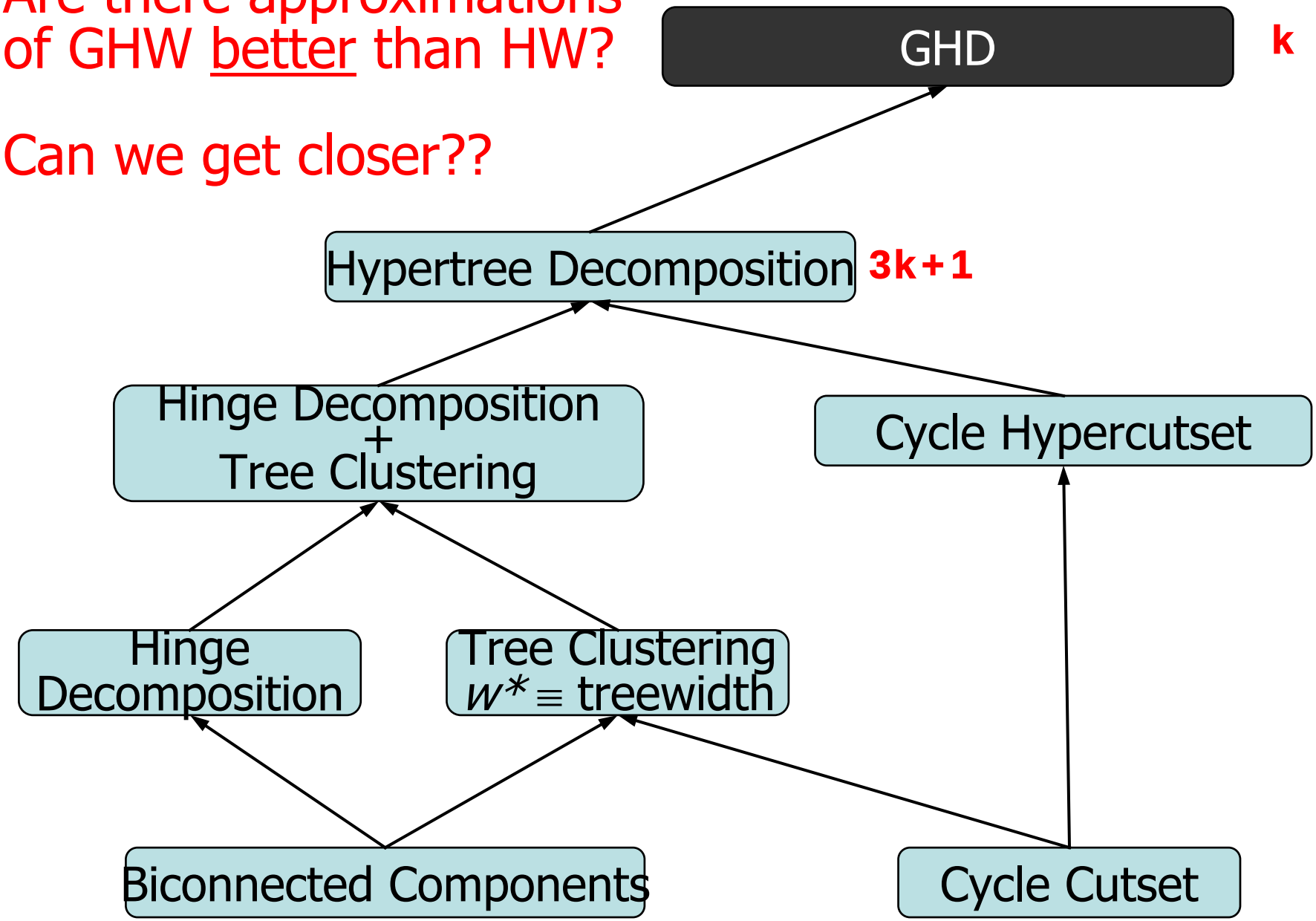
Scarcello:
HD and SC incomparable!



Open Question:

Are there approximations of GHW better than HW?

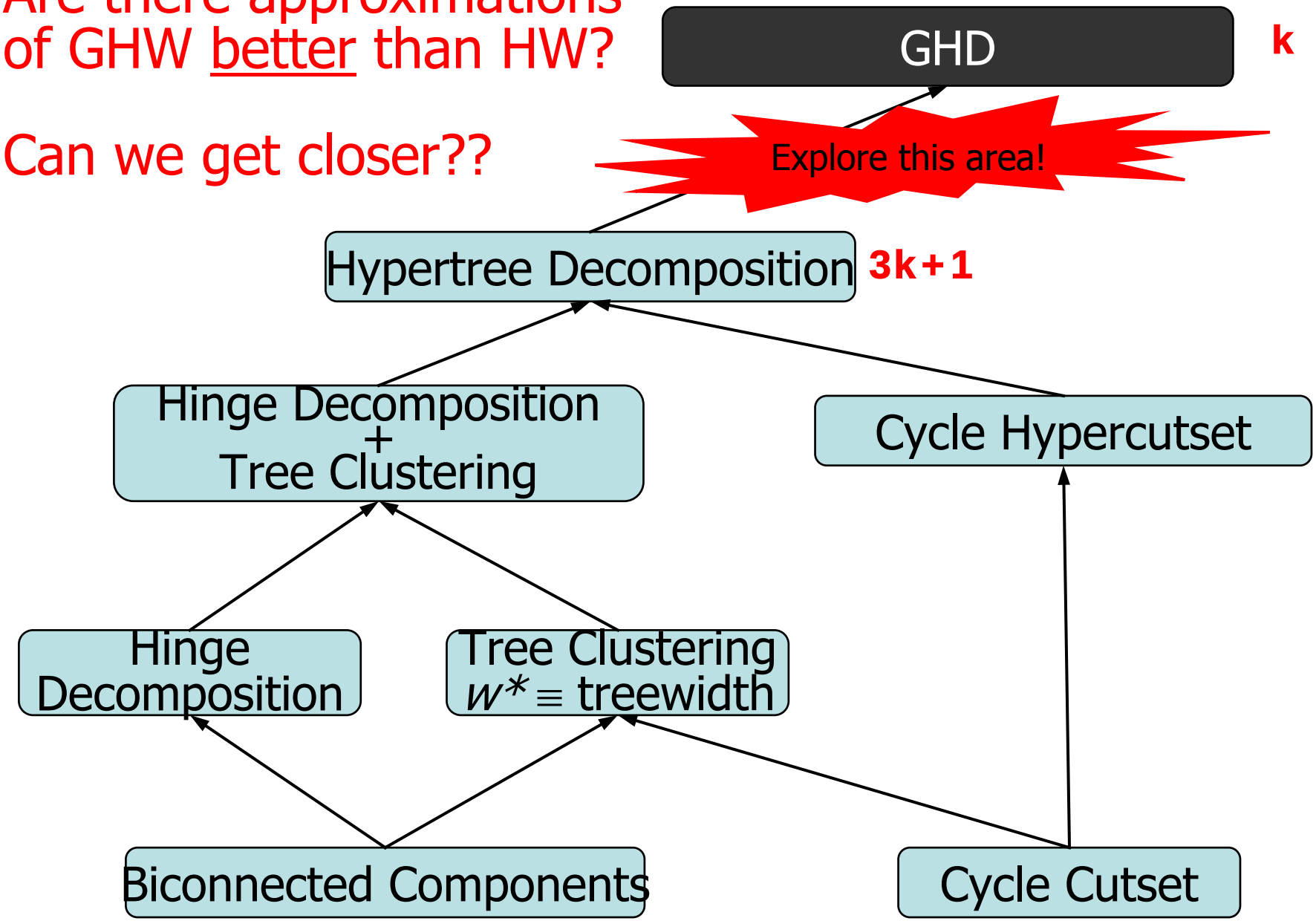
Can we get closer??



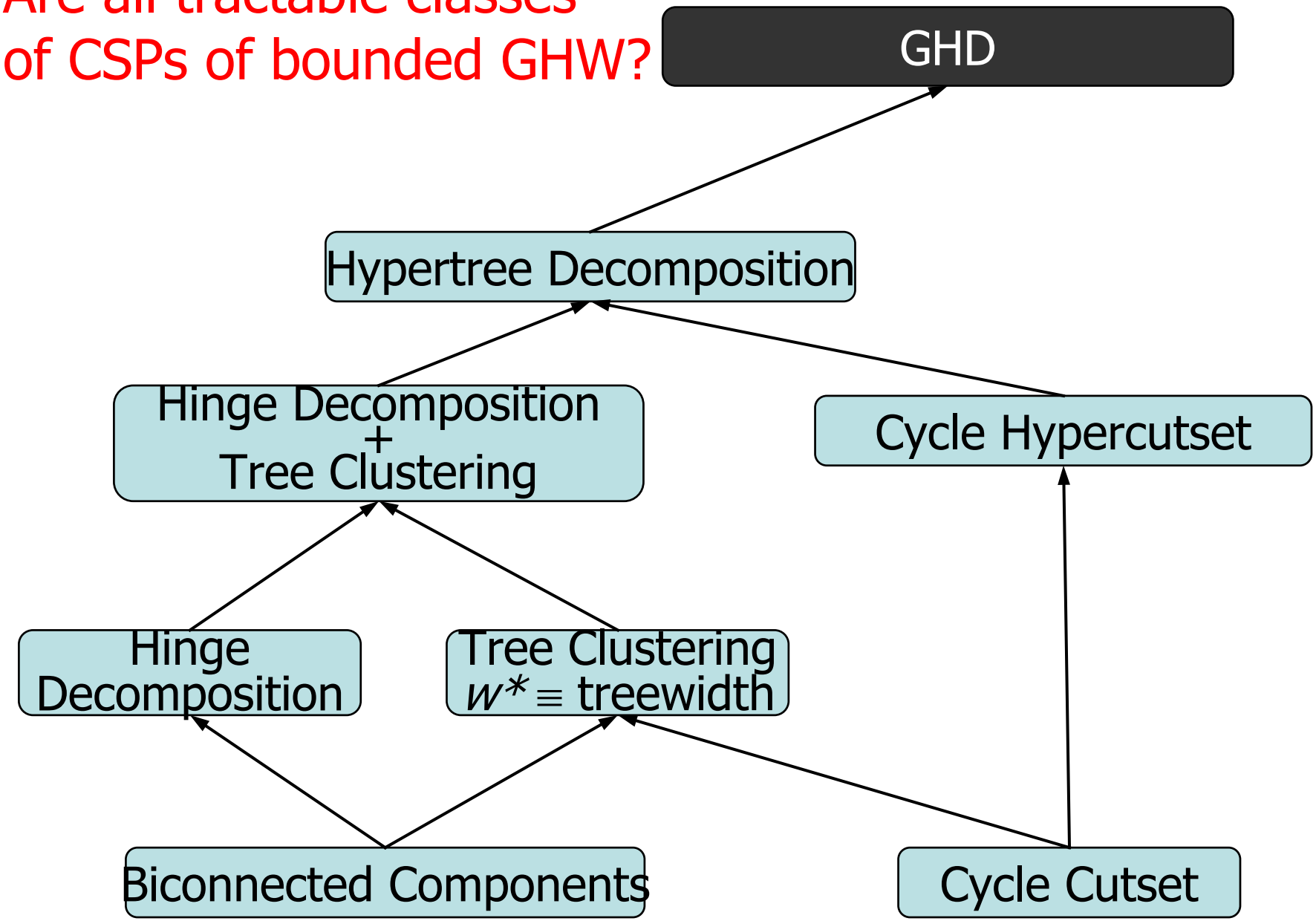
Open Question:

Are there approximations of GHW better than HW?

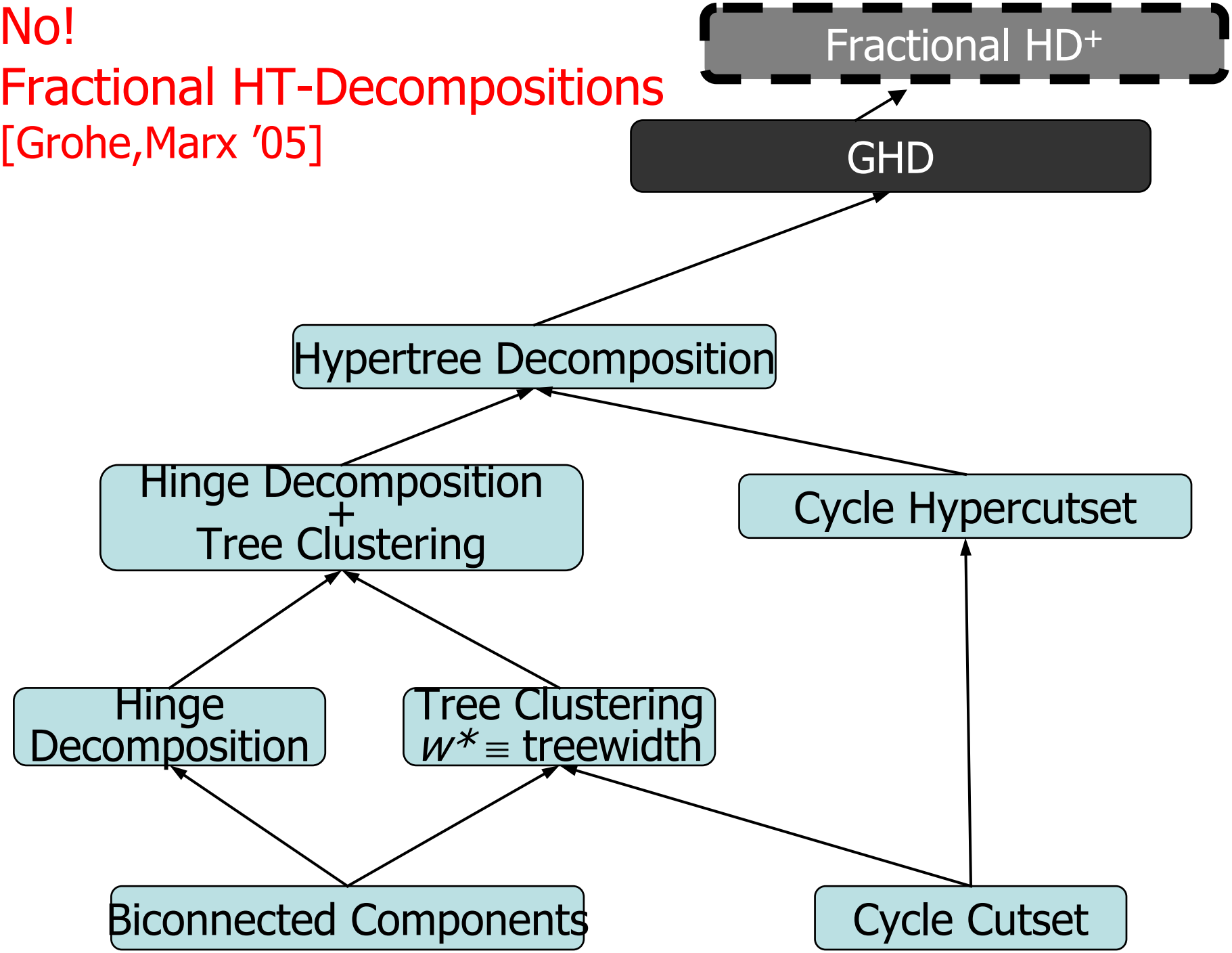
Can we get closer??



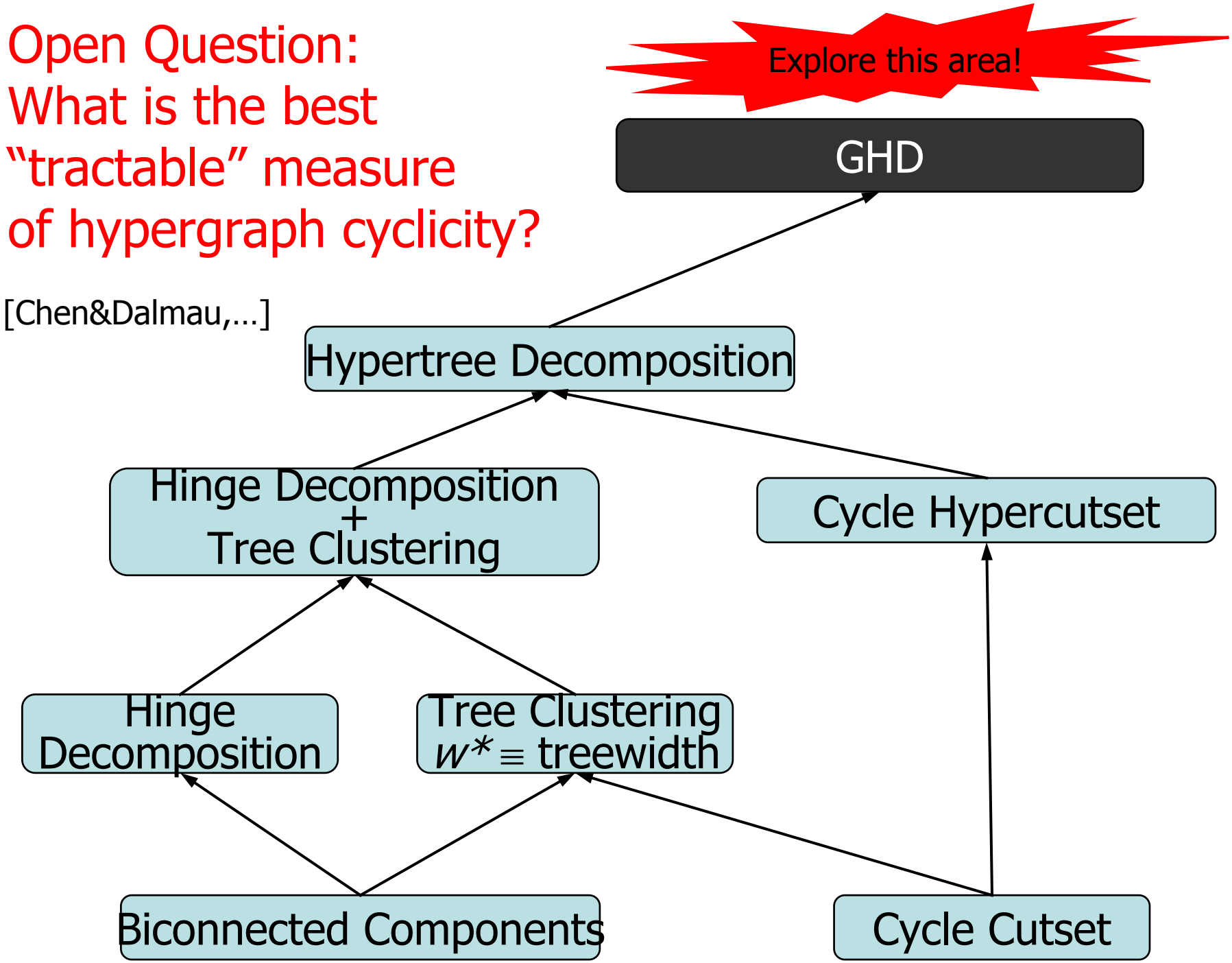
Question:
Are all tractable classes
of CSPs of bounded GHW?



No!
Fractional HT-Decompositions
[Grohe,Marx '05]



Open Question:
What is the best
"tractable" measure
of hypergraph cyclicity?



Characterizations of Hypertree width

- Logical characterization:

  Loosely guarded logic

- Game characterization:

  The robber and marshals game

Guarded Formulas

$$\ldots\ \exists \overline{X}\ (g \wedge \varphi) \ldots$$

Guard atom: $free(\varphi) \subseteq var(g)$

k-guarded Formulas (loosely guarded):

$$\ldots\ \exists \overline{X}\ (\underbrace{g_1 \wedge g_2 \wedge \cdots \wedge g_k}_{k\text{-guard}} \wedge \varphi) \ldots$$

GF(FO), GF_k(FO) are well-studied fragments of FO (Van Benthem'97, Gradel'99)

Logical Characterization of HW

Theorem: $\mathrm{HW}_k = \mathrm{GF}_k(\mathrm{L})$

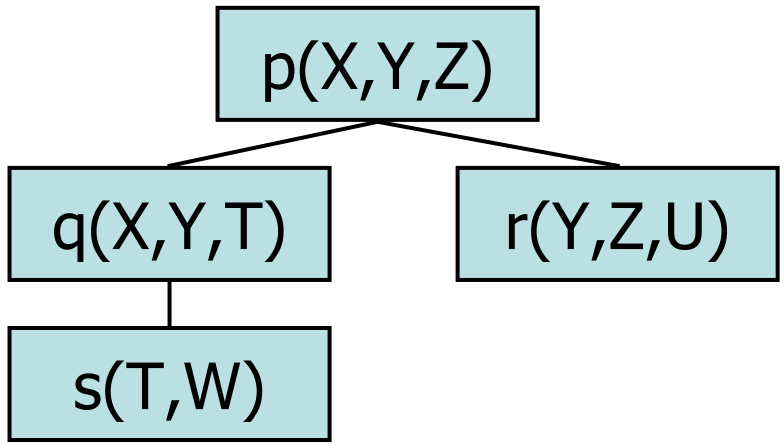
From this general result, we also get a nice logical characterization of acyclic queries:

Corollary: $\mathrm{HW}_1 = \mathrm{ACYCLIC} = \mathrm{GF}(\mathrm{L})$

An Example

$$\exists X, Y, Z, T, U, W . (p(X, Y, Z) \wedge q(X, Y, T) \wedge r(Y, Z, U) \wedge s(T, W))$$

Is acyclic:



Indeed, there exists an equivalent guarded formula:

$$\exists X, Y, Z . (p(X, Y, Z) \wedge \exists T . (\mathrm{q}(X, Y, T) \wedge \exists W . s(T, W)) \wedge \wedge \exists U . \mathrm{r}(Y, Z, U))$$

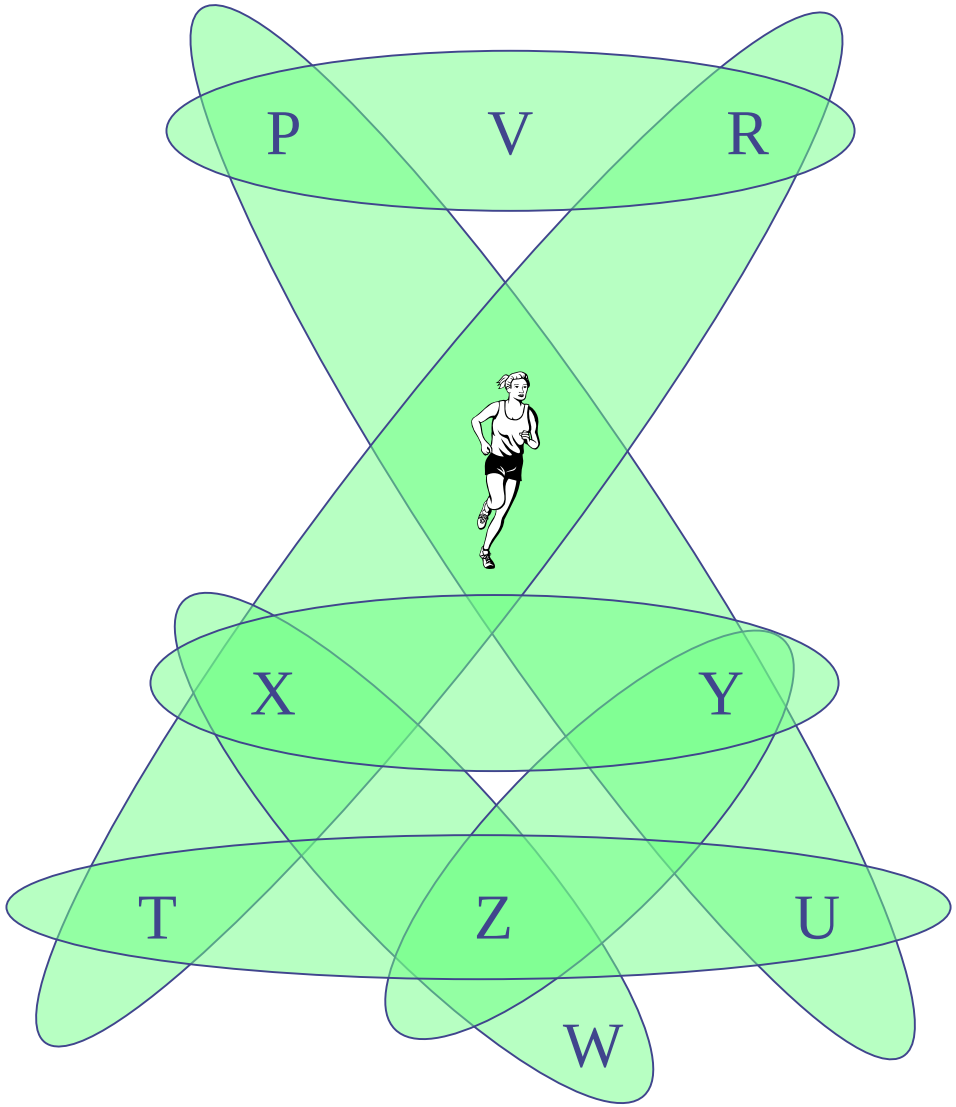
Guard

Guarded subformula

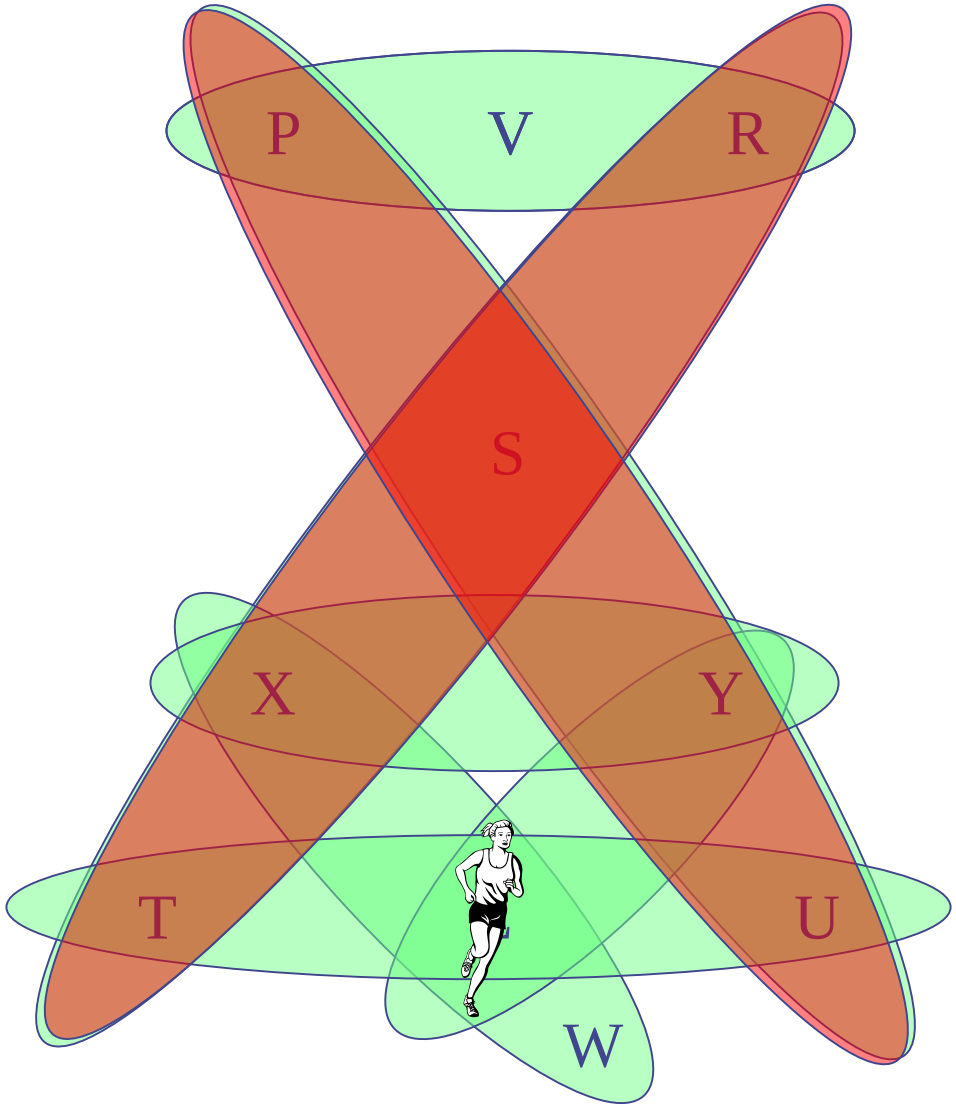
Game characterization: Robber and Marshals

- A robber and *k* marshals play the game on a hypergraph
- The marshals have to capture the robber
- The robber tries to elude her capture, by running arbitrarily fast on the vertices of the hypergraph

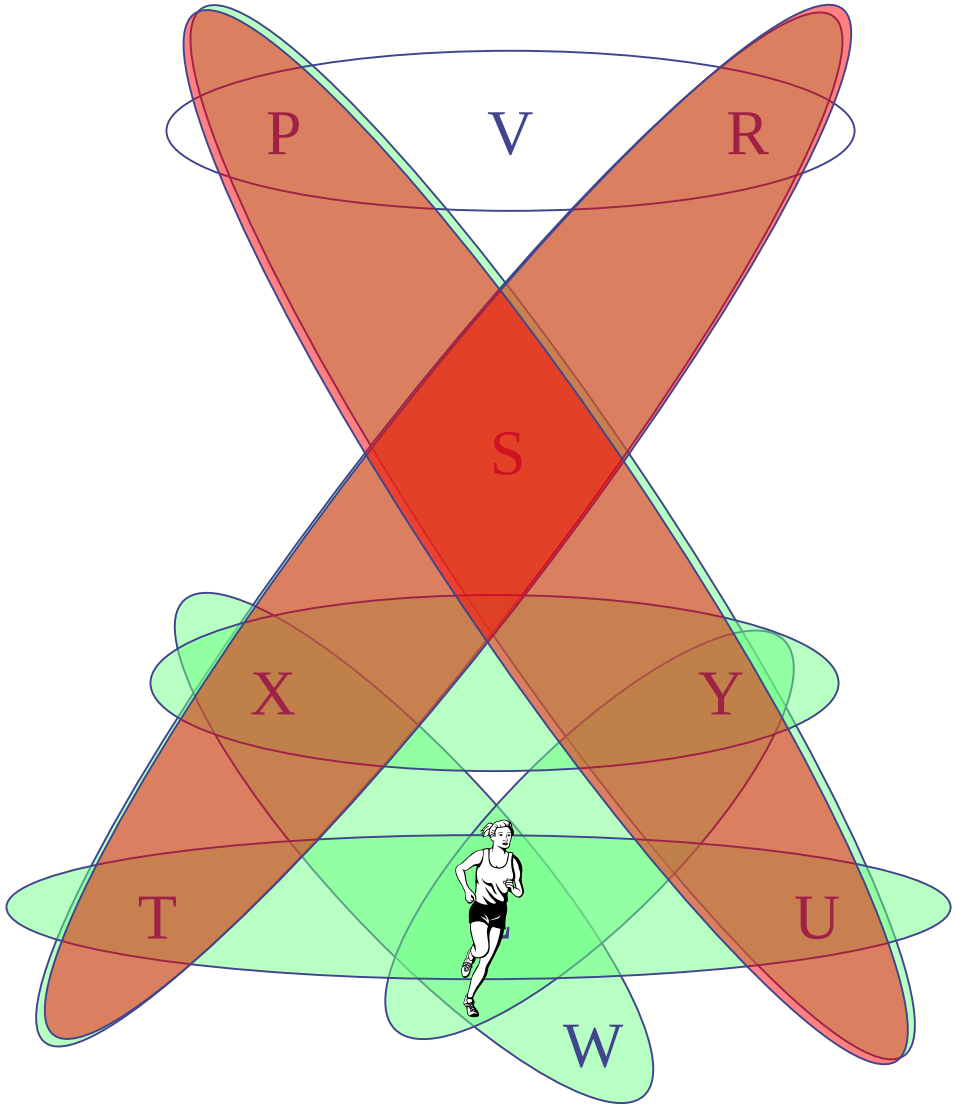
Step 0: the empty hypergraph



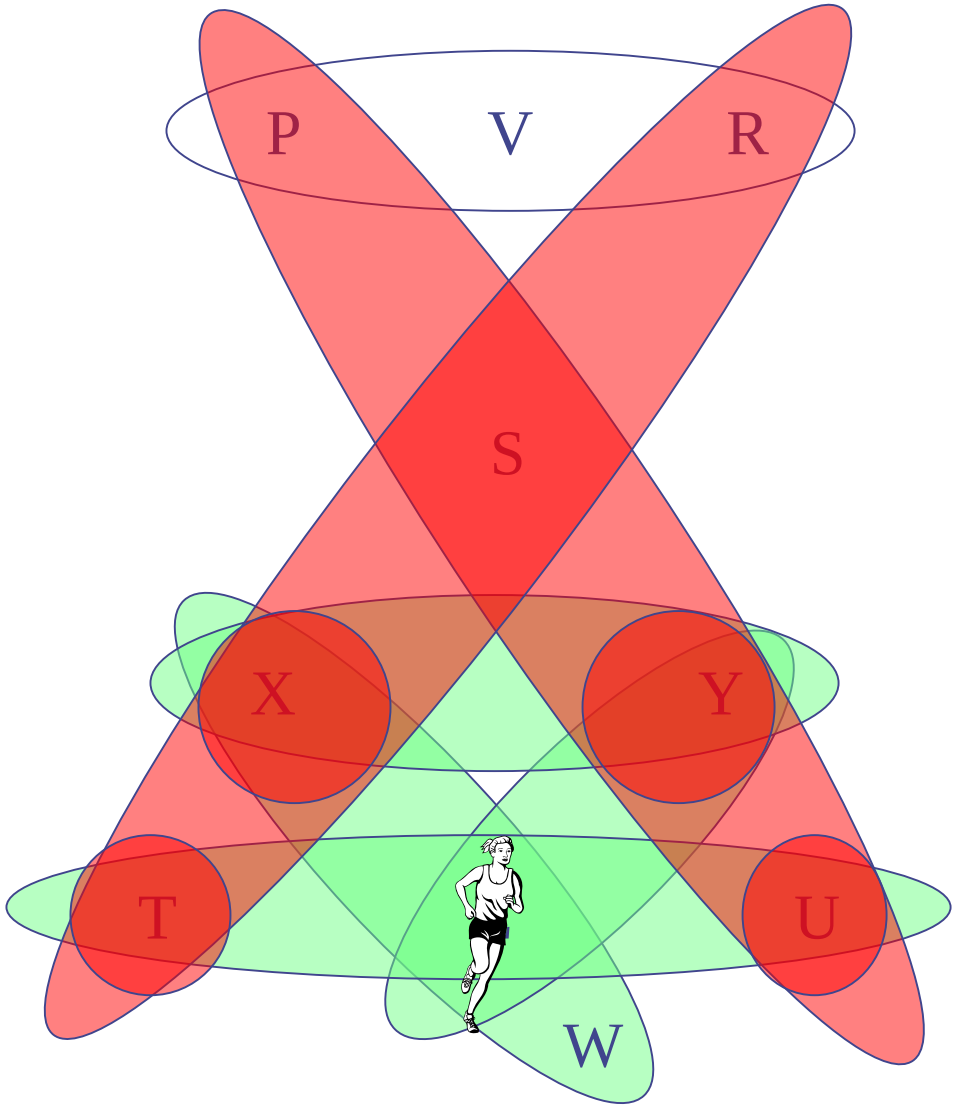
Step 1: first move of the marshals



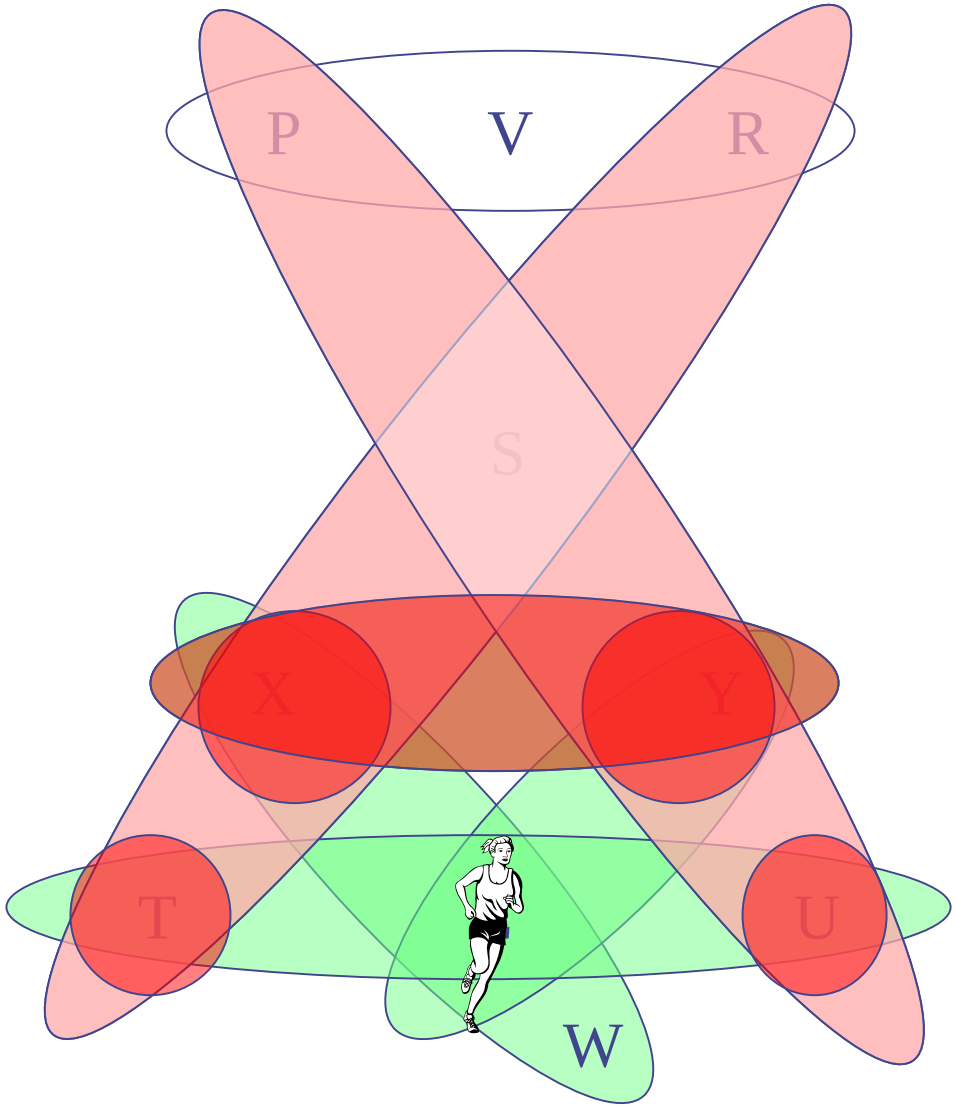
Step 1: first move of the marshals



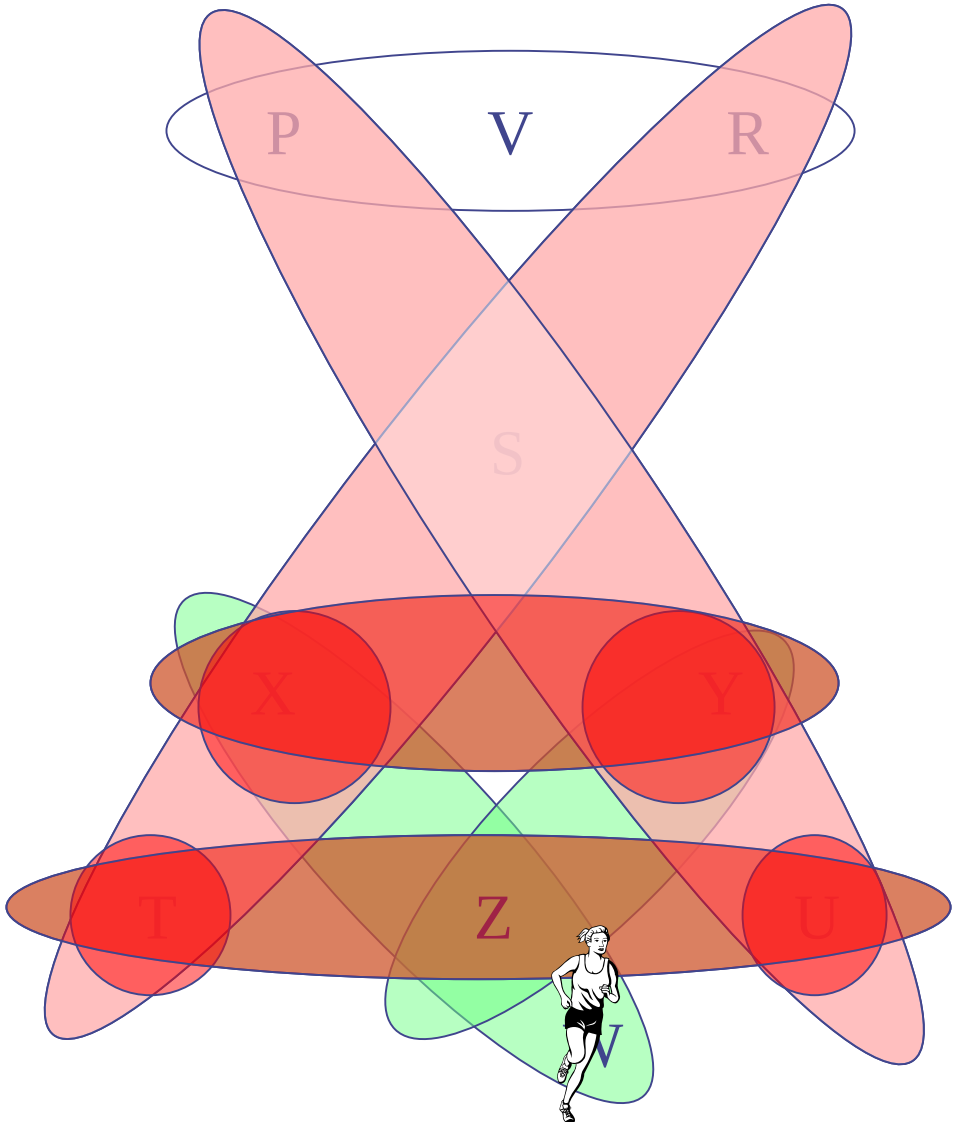
Step 2a: shrinking the space



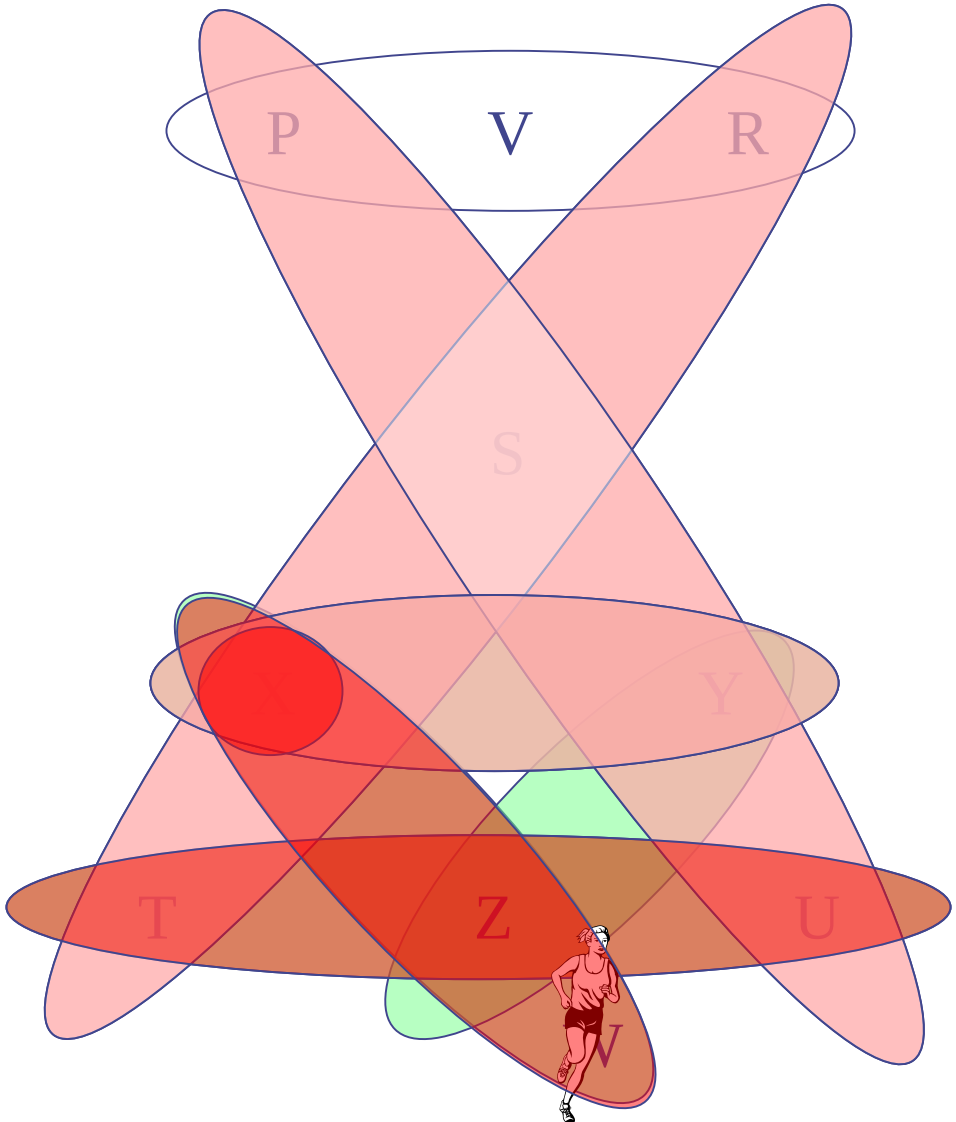
Step 2a: shrinking the space



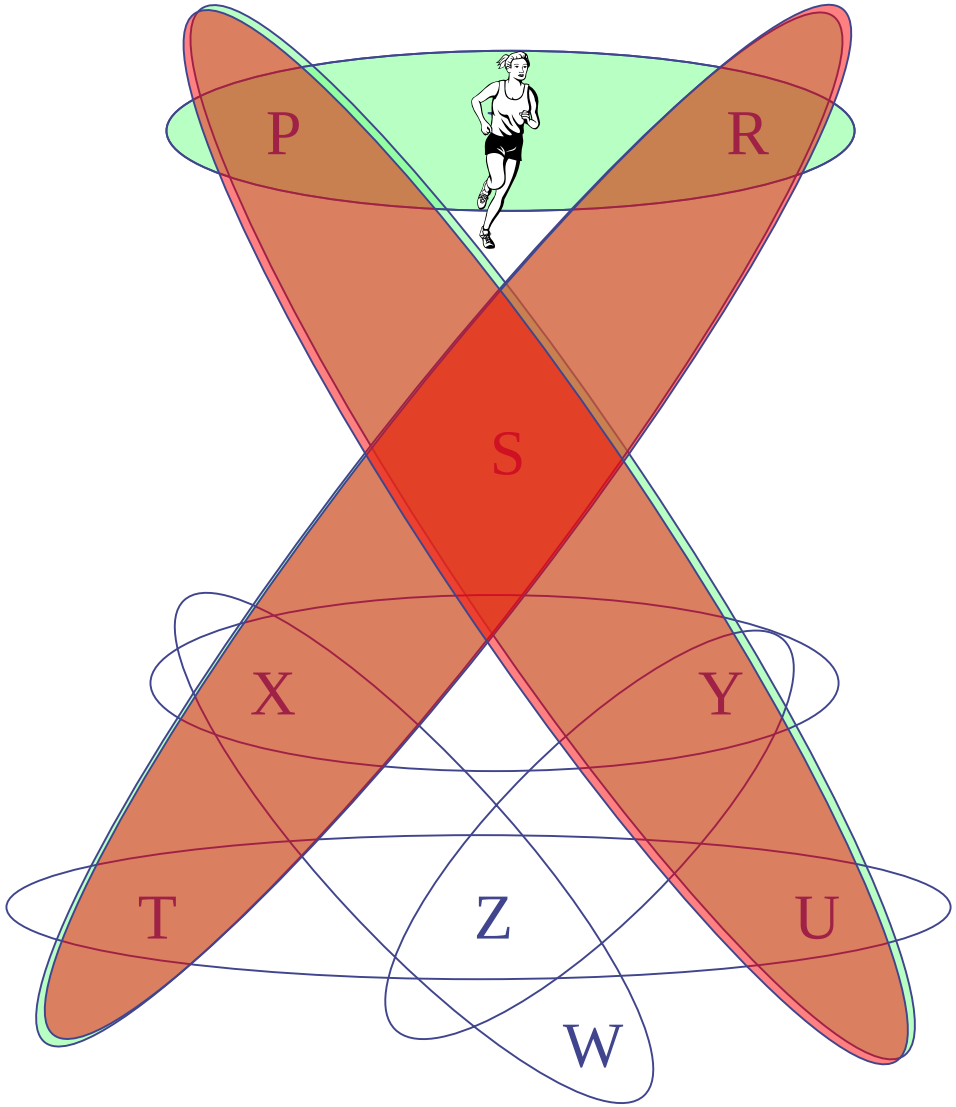
Step 2a: shrinking the space



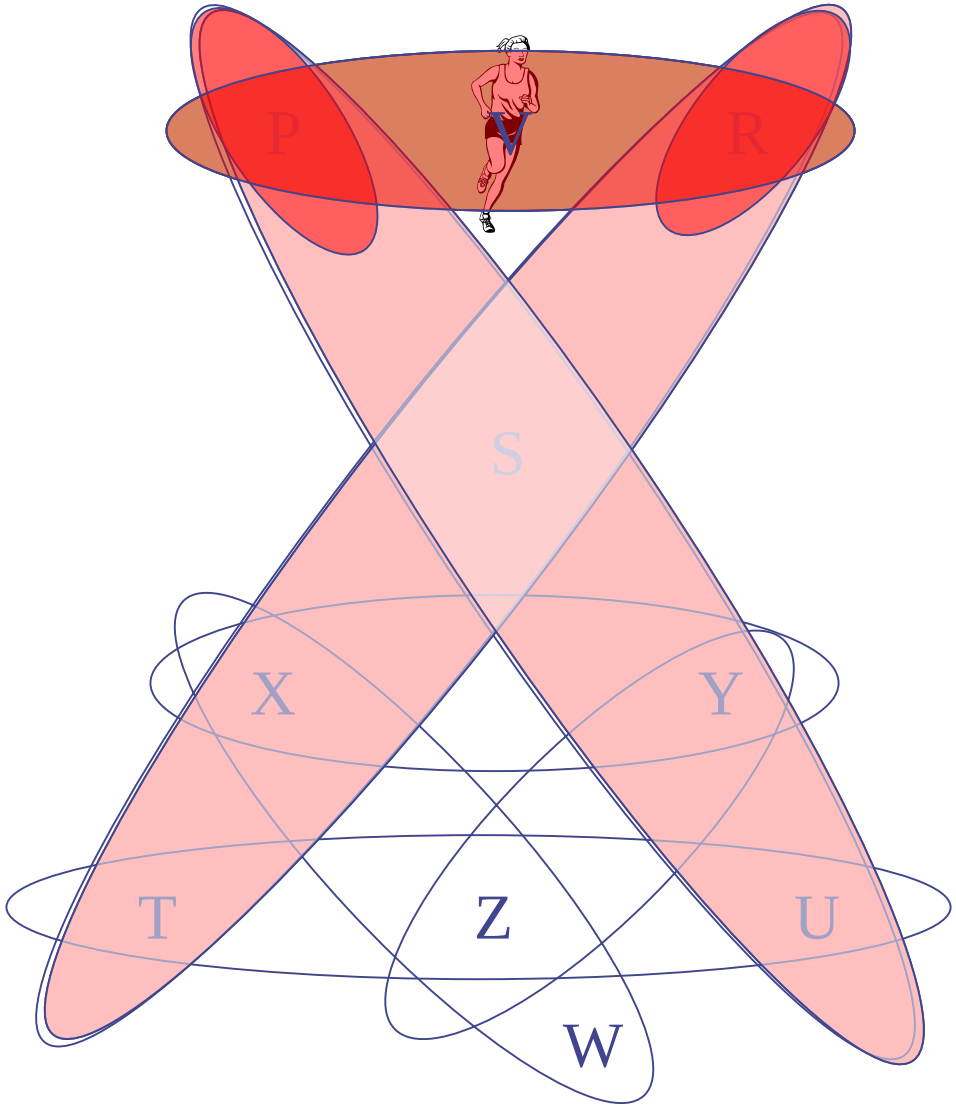
The capture



A different robber's choice



Step 2b: the capture



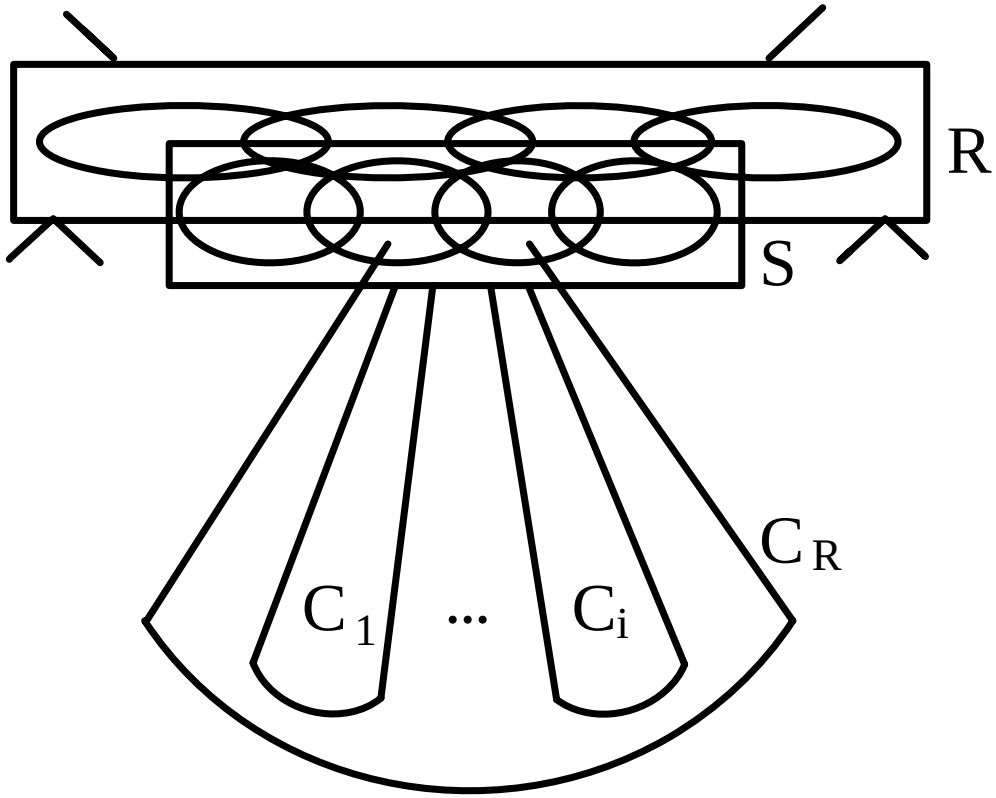
R&M Game and Hypertree Width

Let *H* be a hypergraph.

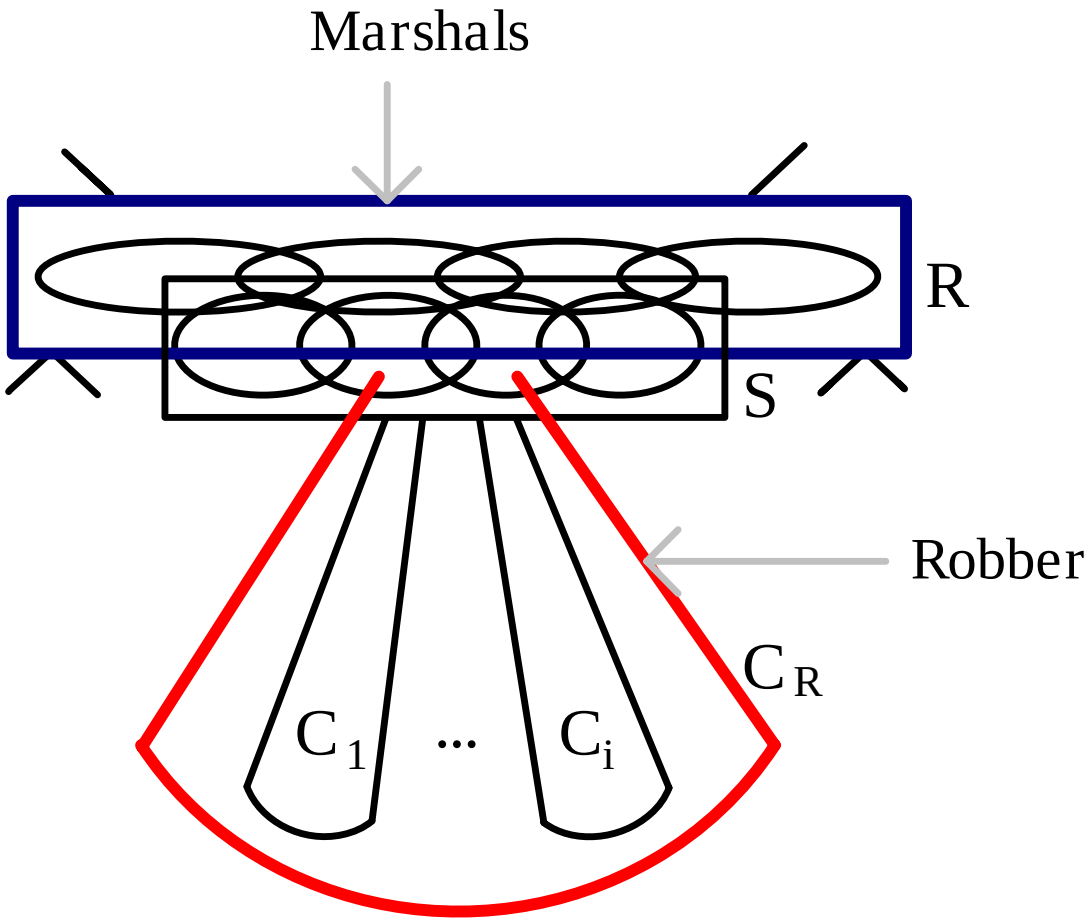
- **Theorem**: *H* has hypertree width $\leq k$ if and only if *k* marshals have a winning strategy on *H*.
- **Corollary**: *H* is acyclic if and only if one marshal has a winning strategy on *H*.

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

Marshals…

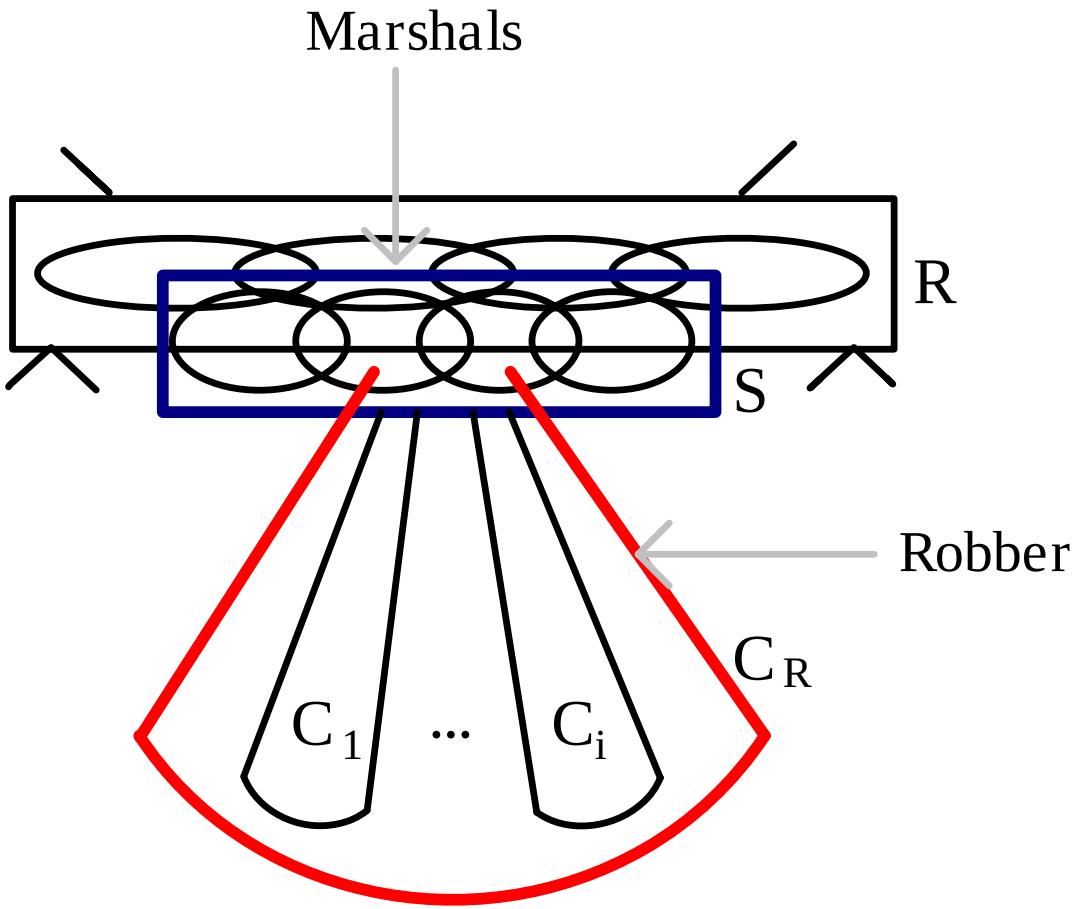


Marshals…



Polynomial algorithm: Alternating LOGSPACE

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



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

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

LOGSPACE CHECKABLE

Nasa problem

Part of relations for the Nasa problem

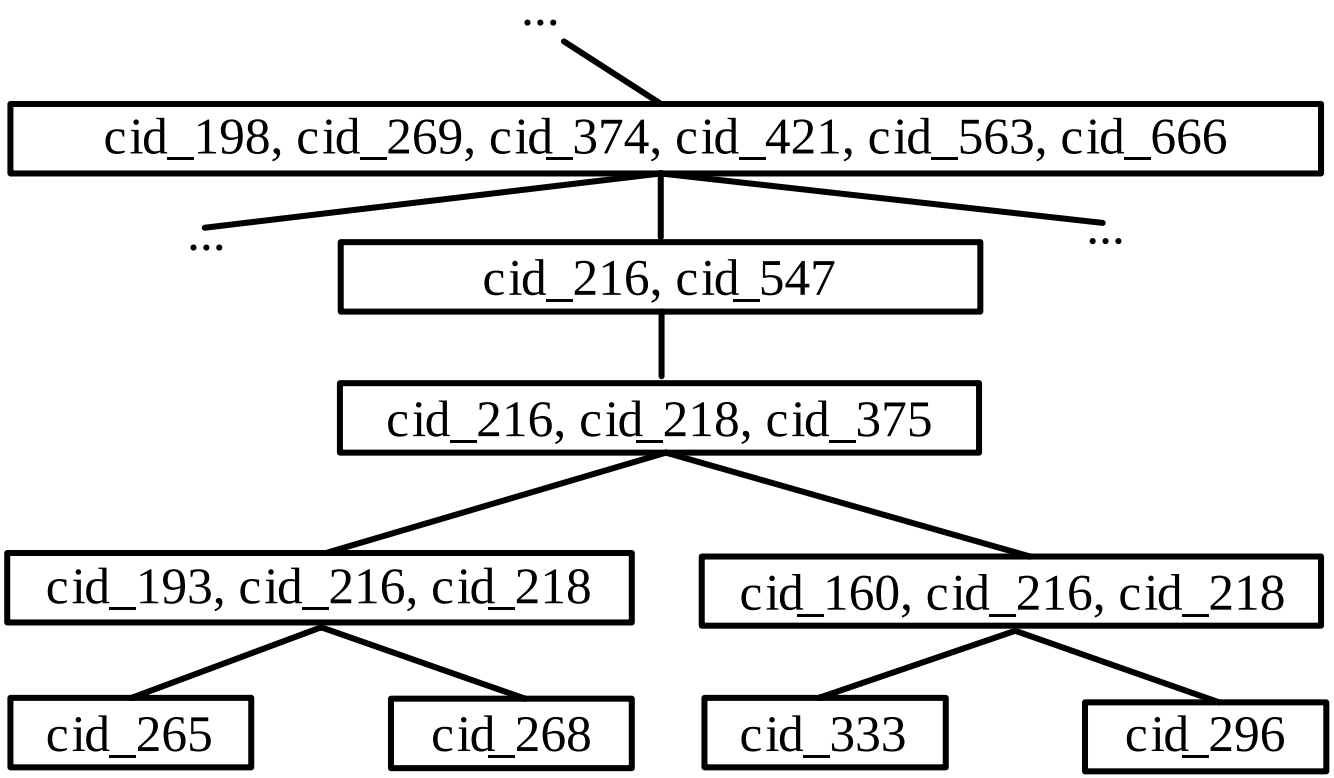
...

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

...

- 680 relations
- 579 variables

Nasa problem: hypertree



Part of hypertree for the Nasa problem

Best known hypertree-width for the Nasa problem is 22

Conclusions

Hypertree decomposition is a very natural notion.

Several hot open problems remain to be solved.

For papers and further material see:
http://ulisse.deis.unical.it/~frank/Hypertrees/