Hypertree Decompositions:
Stucture, Algorithms, and Applications

G. Gottlob
Oxford Univ. and TU Wien

This talk reports about joint work with
N. Leone, F.Scarcello

and: I. Adler, M. Grohe, Miklos, Schwentick

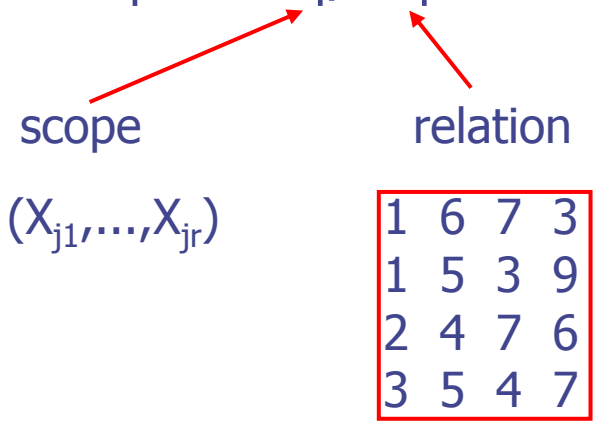
For papers and further material see:
http://ulisse.deis.unical.it/~frank/Hypertrees/
http://www.dbai.tuwien.ac.at/staff/proj/hypertree/

CSP

Set of variables $V=\{X_1,\ldots,X_n\}$, domain D

Set of constraints $\{C_1,\ldots,C_m\}$

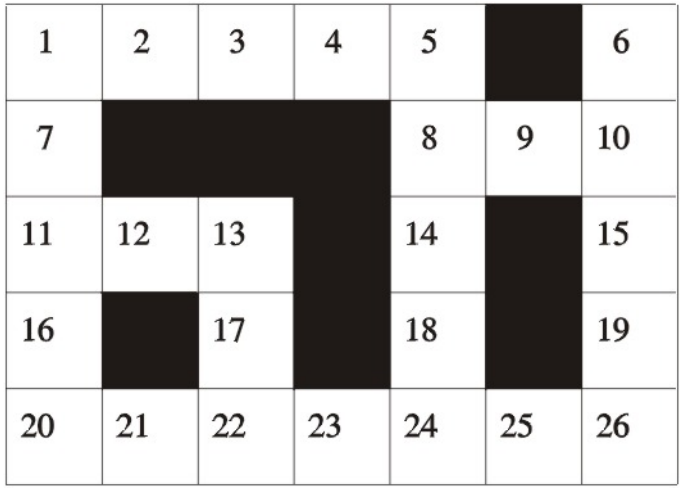
where: $C_i = \langle S_i, R_i \rangle$

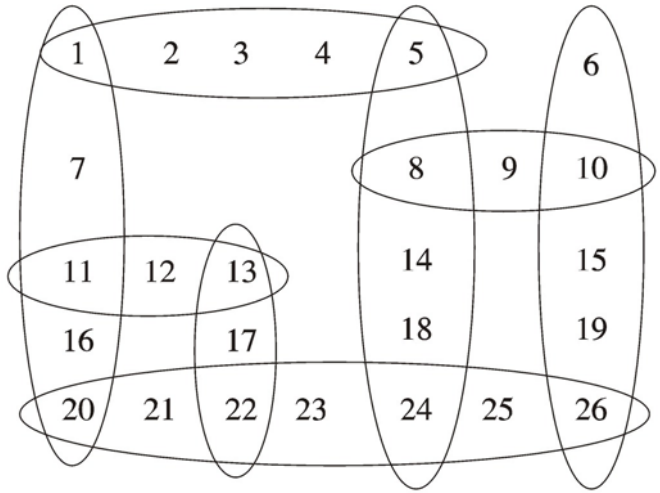


Solution to this CSP: A substitution
h: V→D such that $\forall i$: $h(S_i \in R_i)$

Associated hypergraph: $\{var(S_i) \mid 1 \leq i \leq m\}$

Example of CSP: Crossword Puzzle





1h:

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

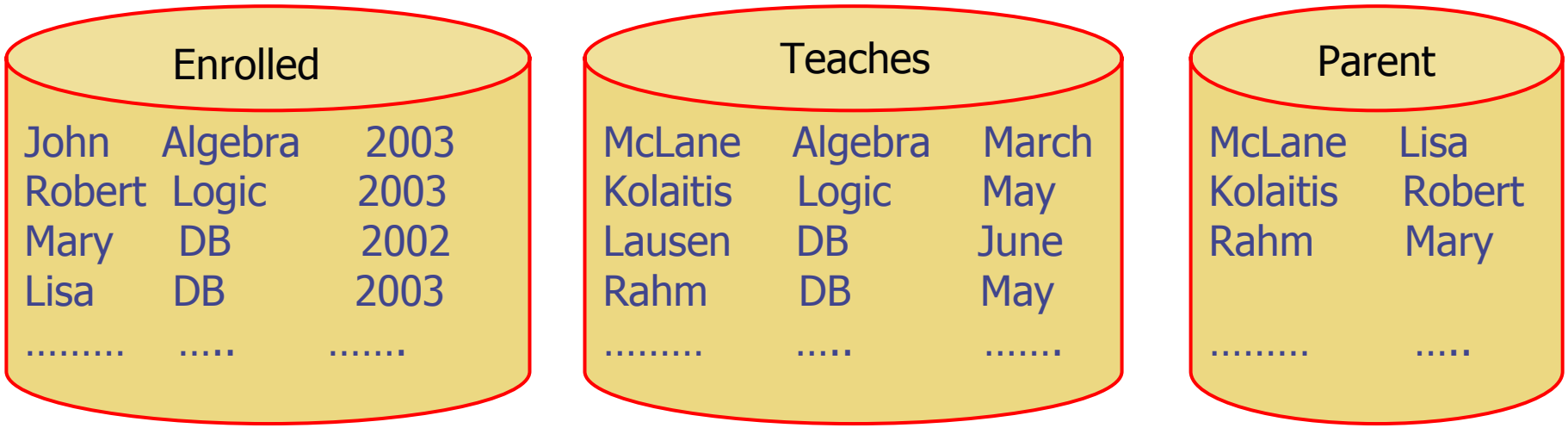
1v:

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

Conjunctive Database Queries are CSPs !

DATABASE:

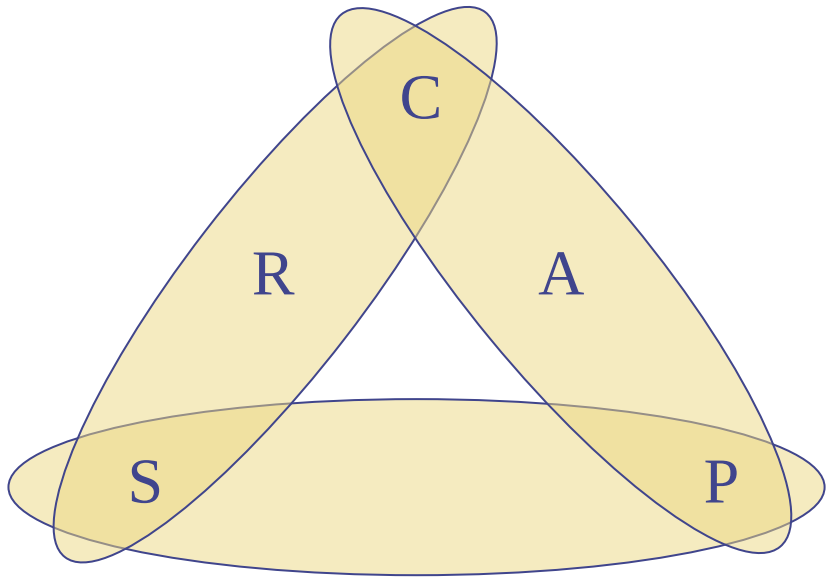


QUERY: Is there any teacher having a child enrolled in her course?

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

Queries, CSPs, and Hypergraphs

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



Conjunctive Queries, CSPs (2)

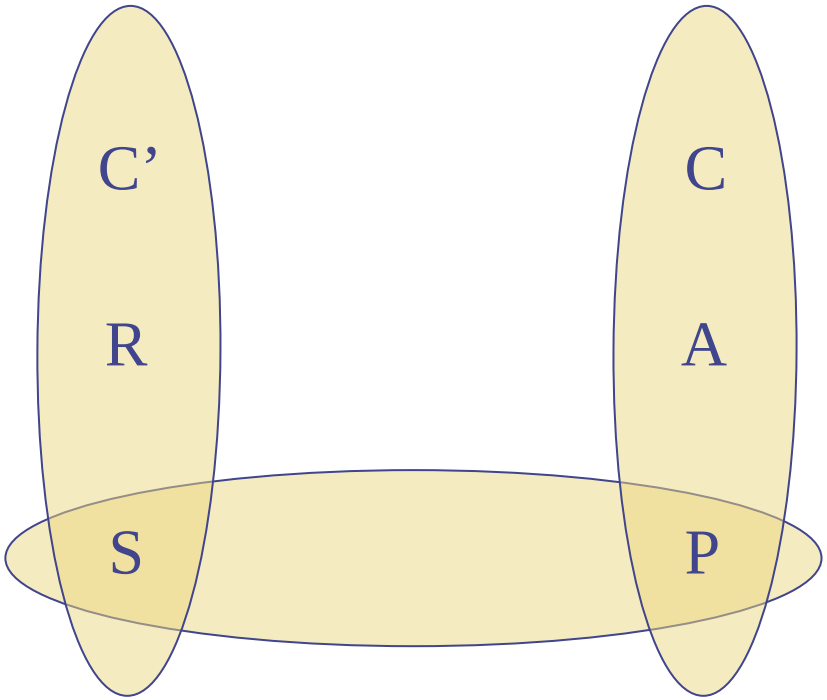
- Database schema (scopes):
  - *Enrolled (Pers#, Course, Reg-Date)*
  - *Teaches (Pers#, Course, Assigned)*
  - *Parent (Pers1, Pers2)*

- Is there any teacher whose child attend some course?

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

Acyclic CSP

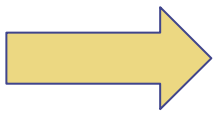
$ans \leftarrow Enrolled(S,C',R) \wedge Teaches(P,C,A) \wedge Parent(P,S)$



Complexity of CSPs

- NP-complete in the general case
  (Bibel, Chandra and Merlin '77, etc.)
  NP-hard even for fixed constraint relations

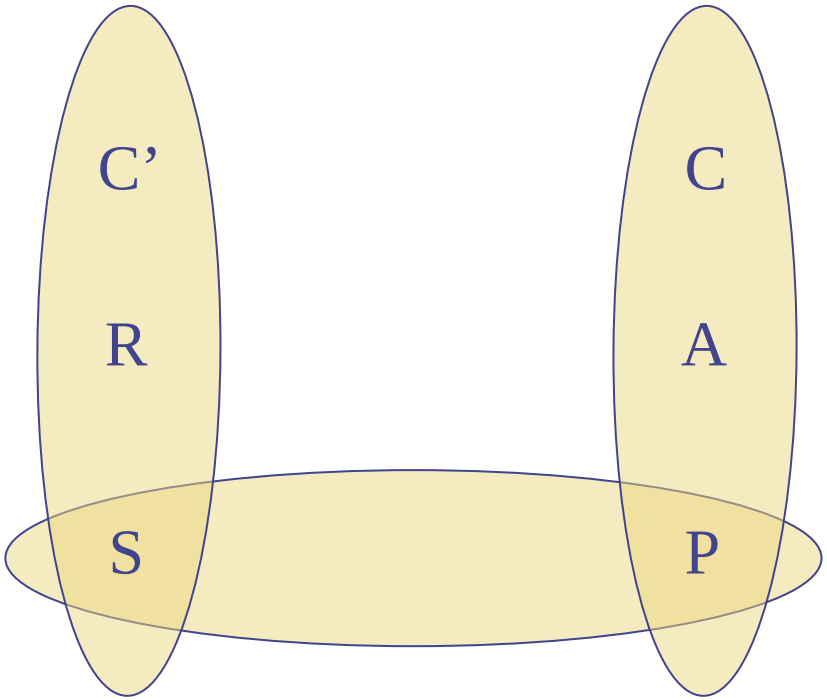
- Polynomial in case of acyclic hypergraphs
  (Yannakakis '81)
  LOGCFL-complete (in NC_2)
  (G.L.S. '98)

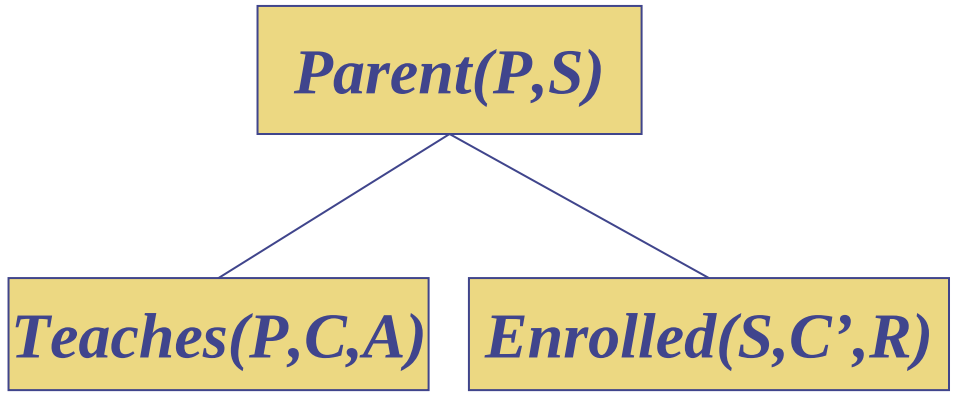


Interest in larger tractable classes of CQS

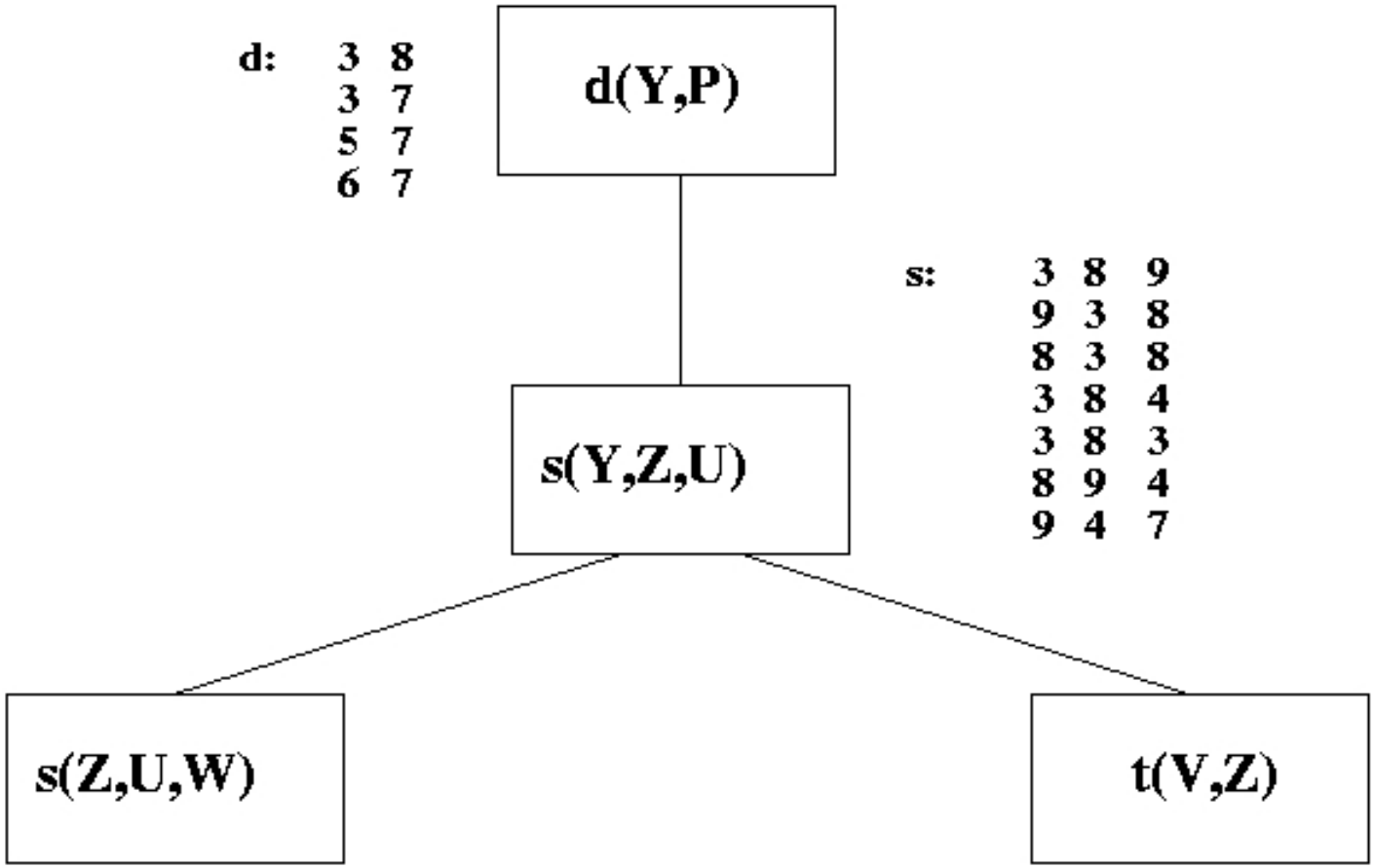
Acyclic queries or CSPs

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

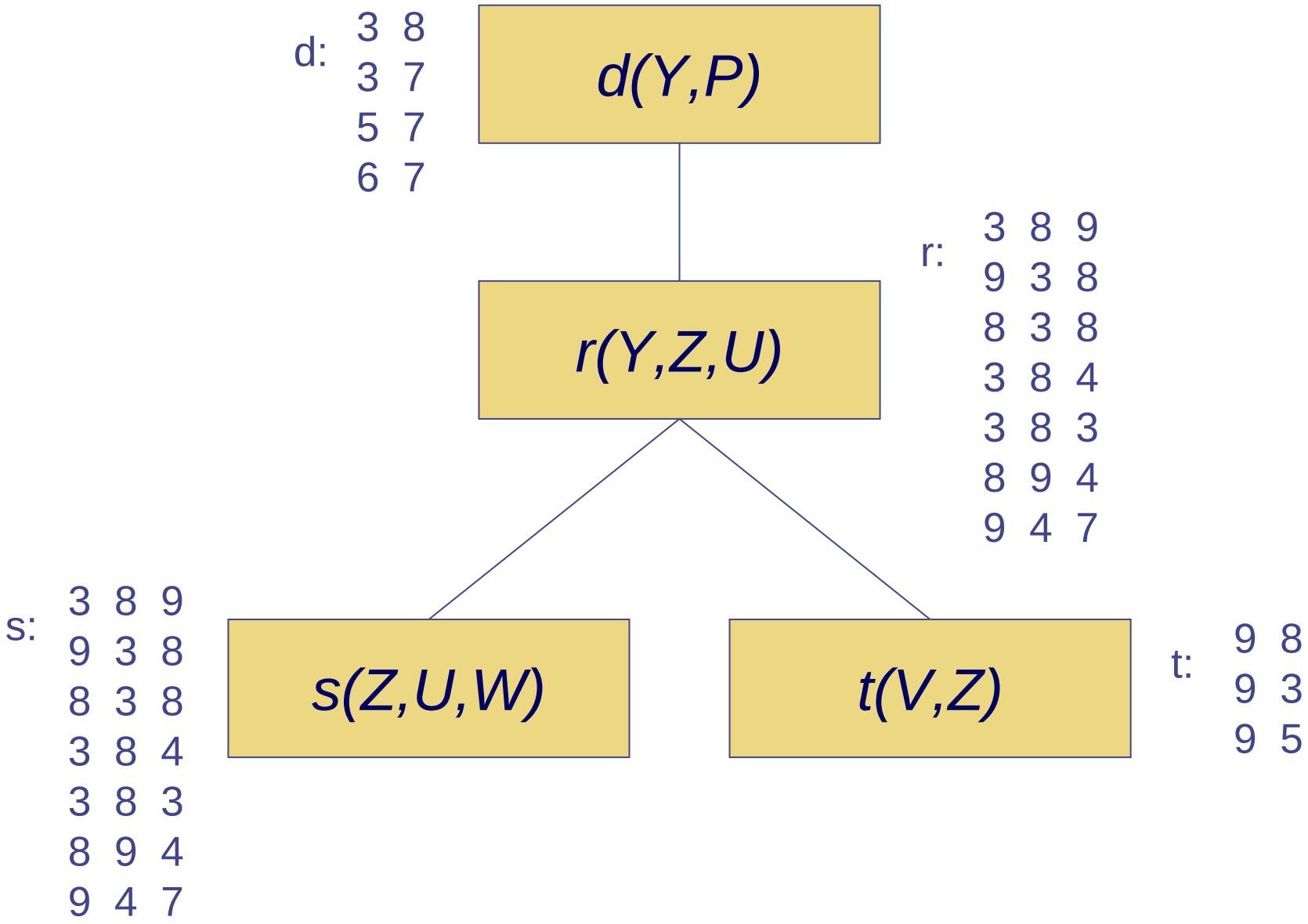


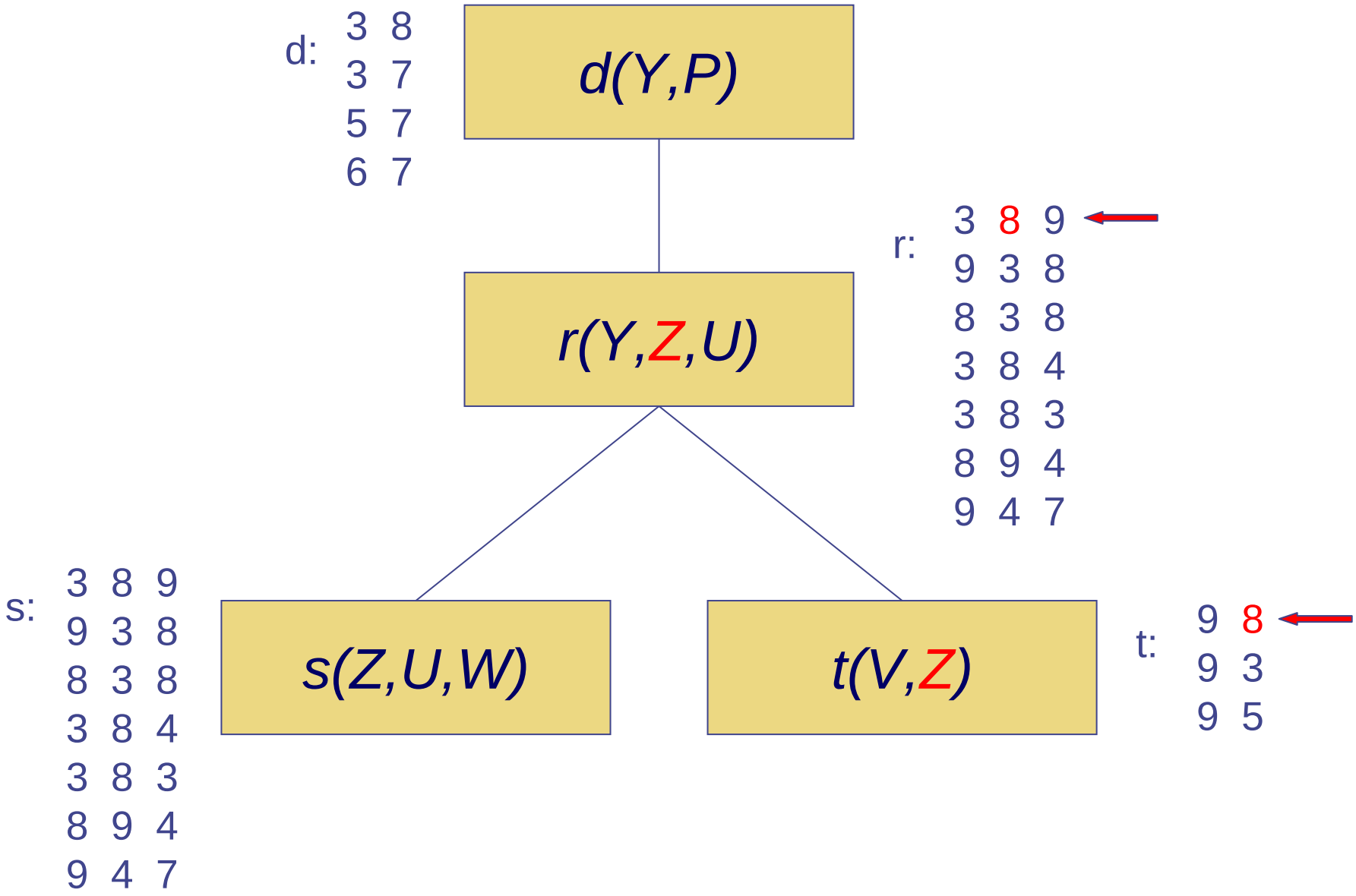


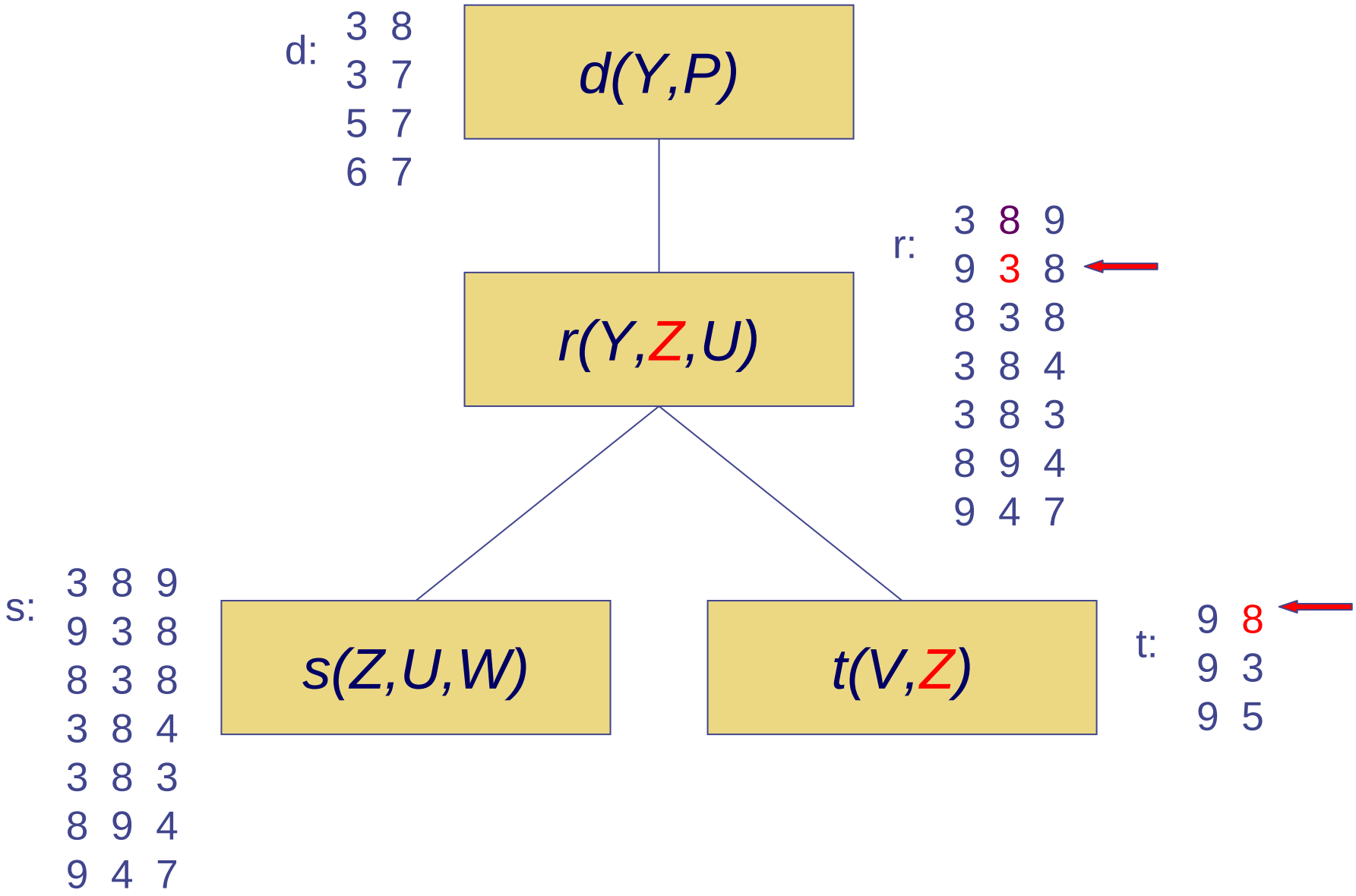
Join Tree

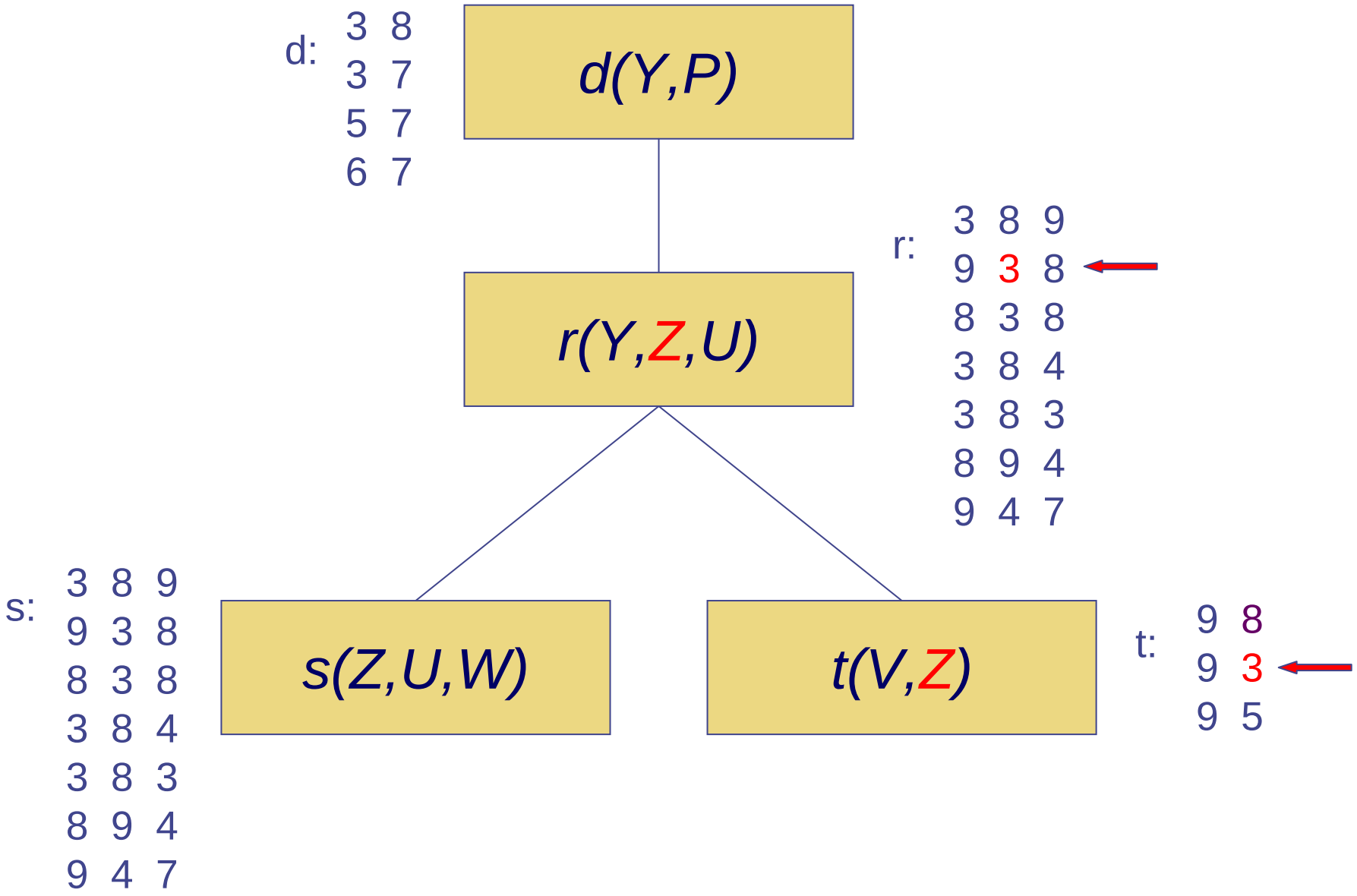

d:
3 8
3 7
5 7
6 7
d(Y,P)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Y,Z,U)
s(Z,U,W)
t(V,Z)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r:
9 8
9 3
9 5

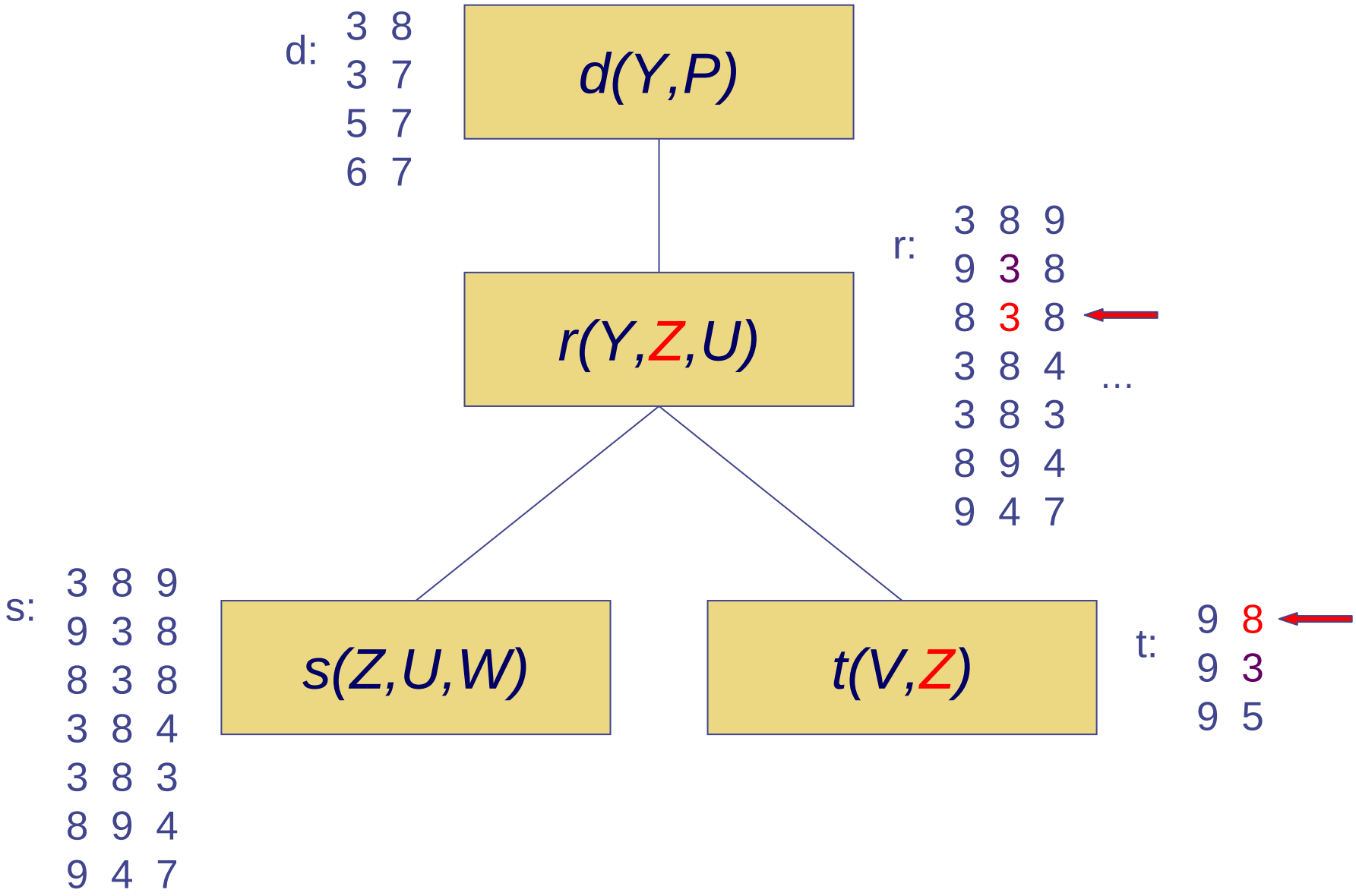
$n^2 \log n$

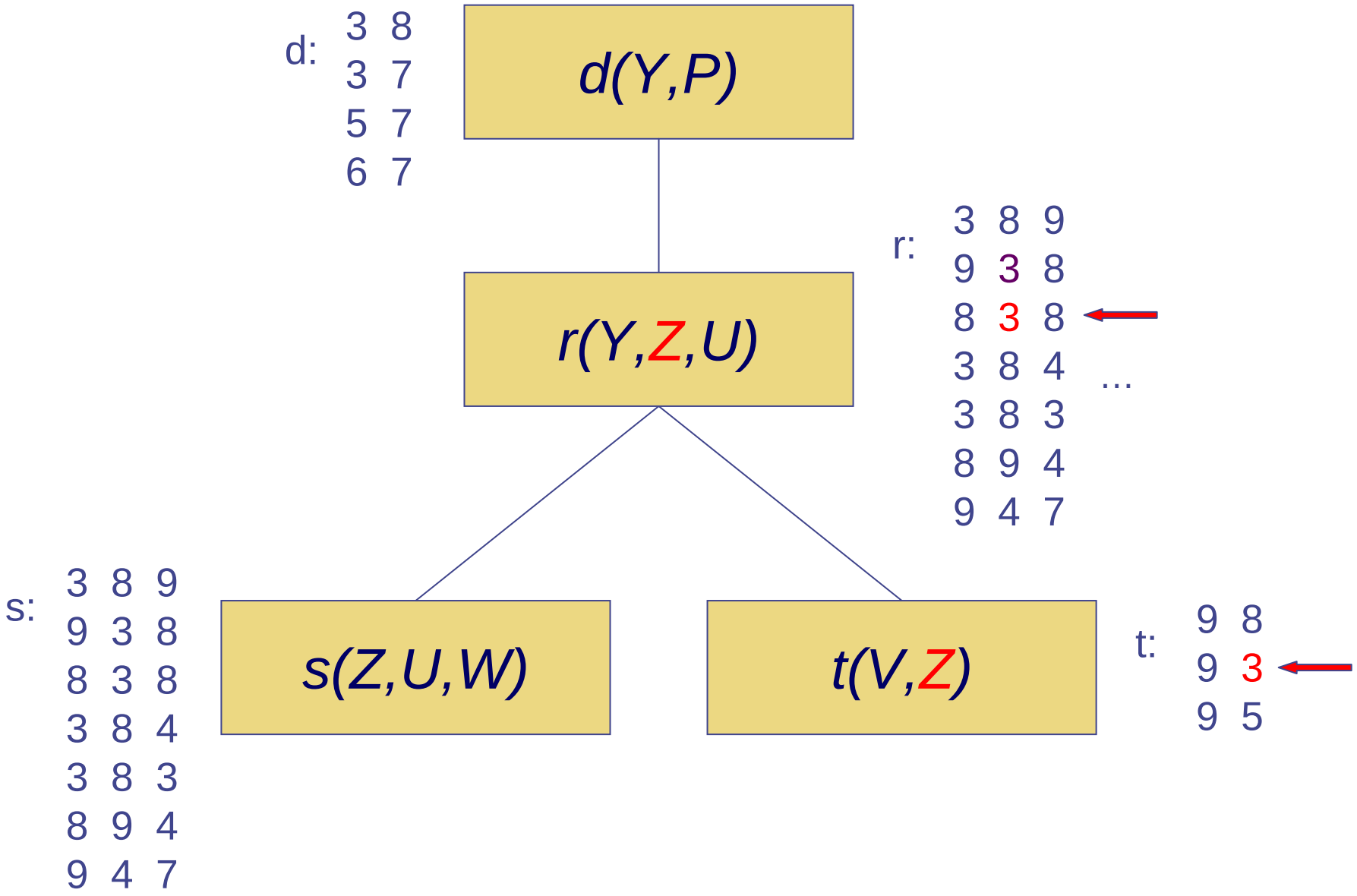

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

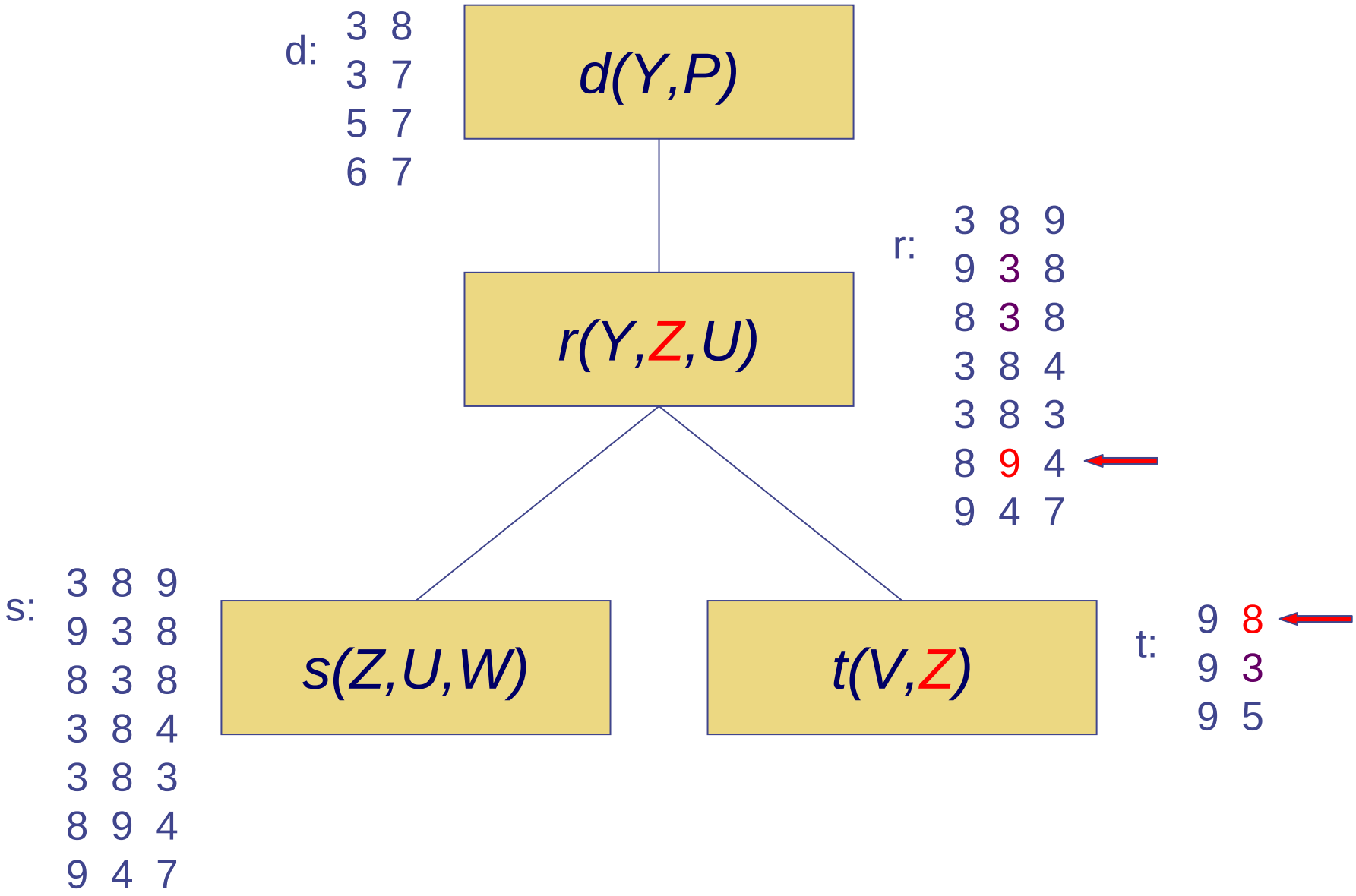

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

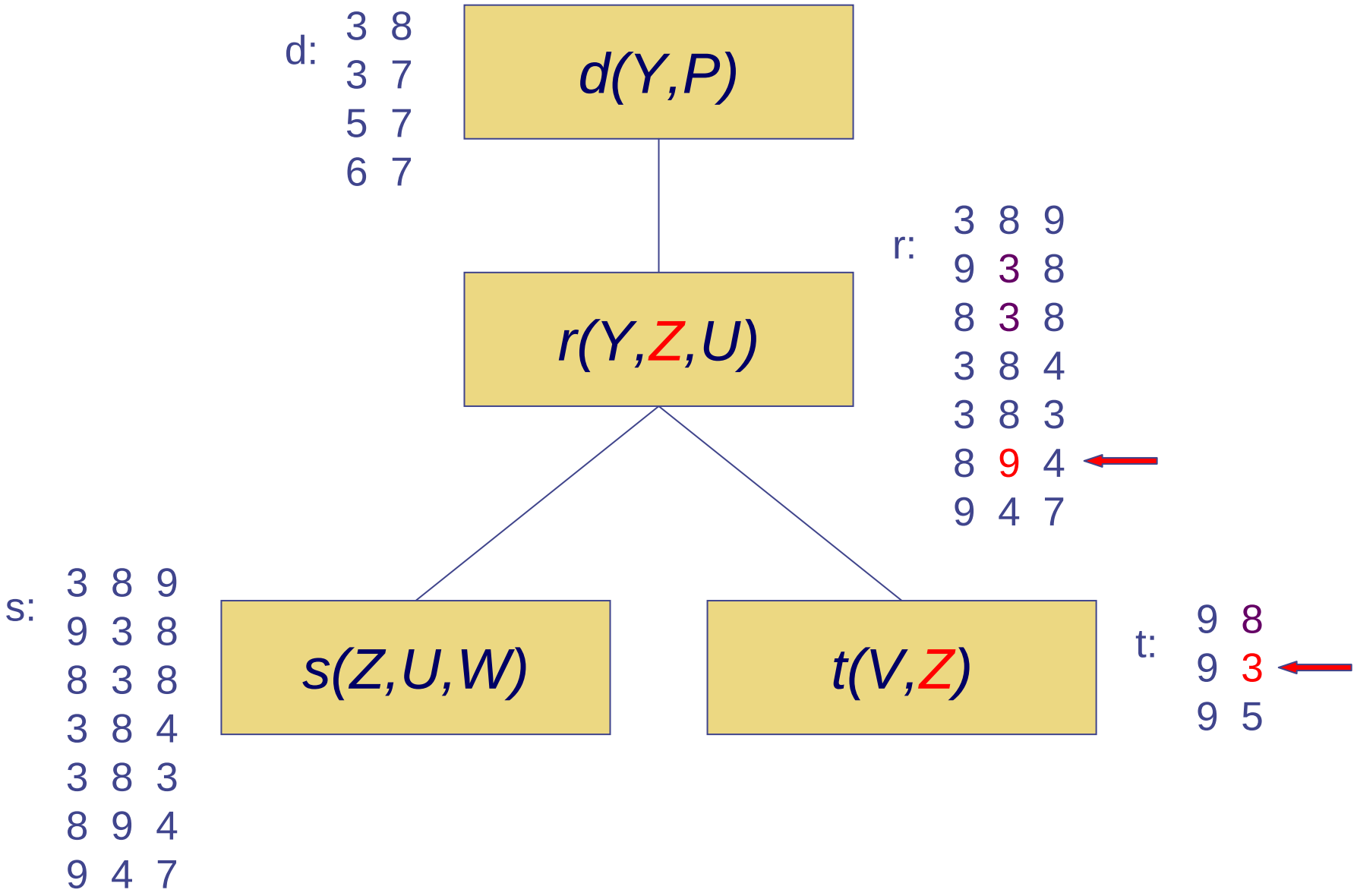

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

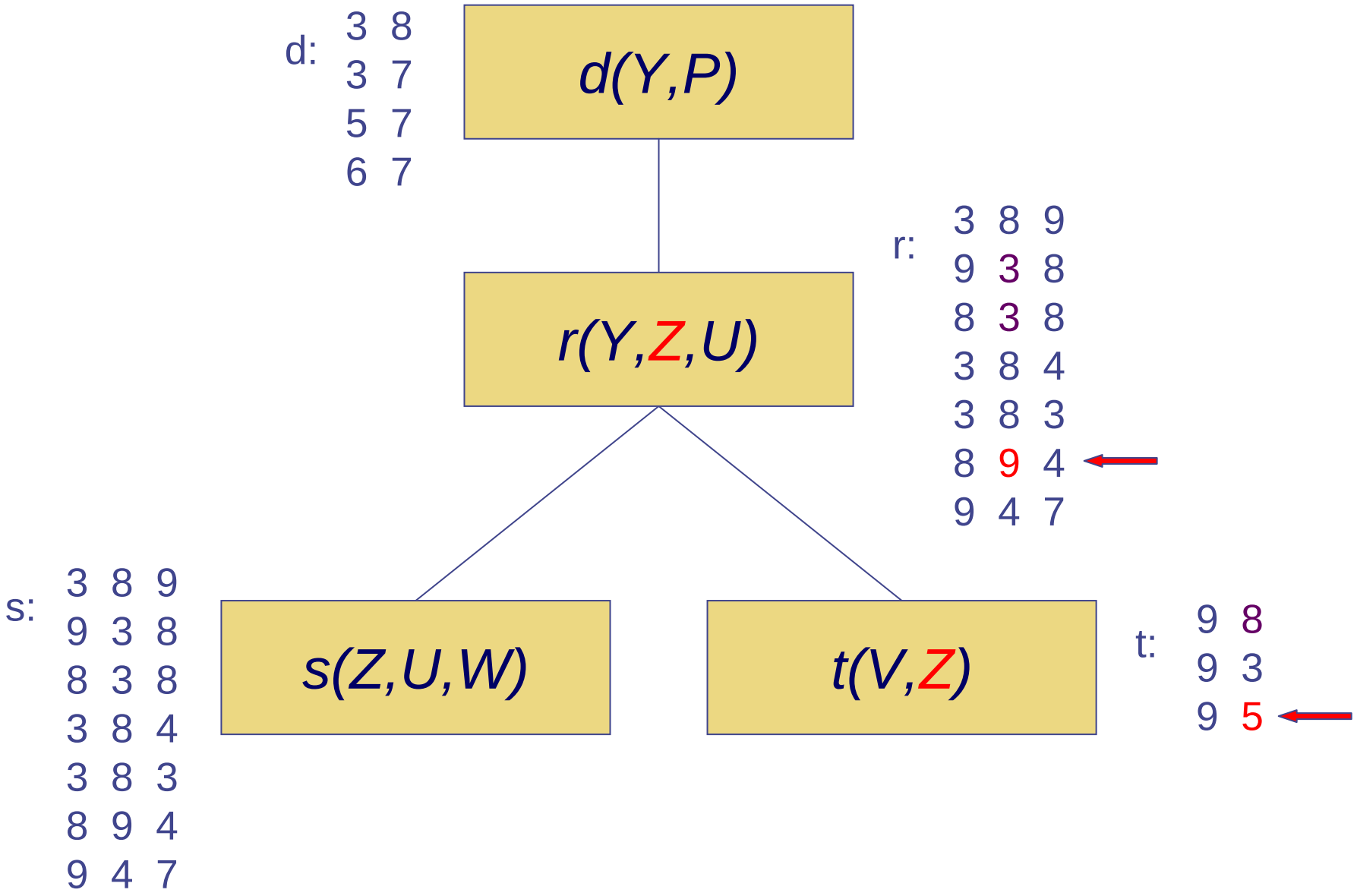

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

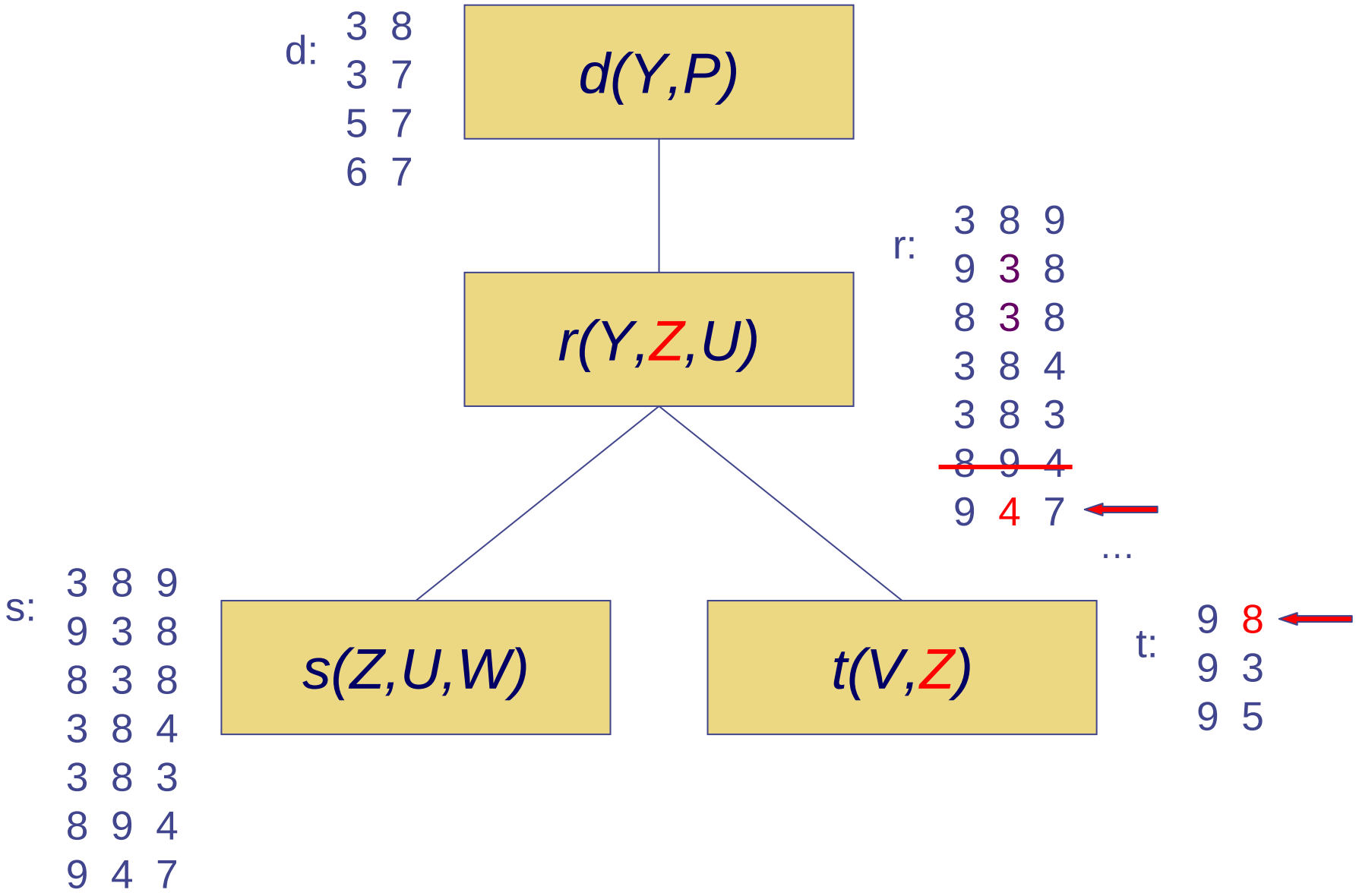

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
...
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

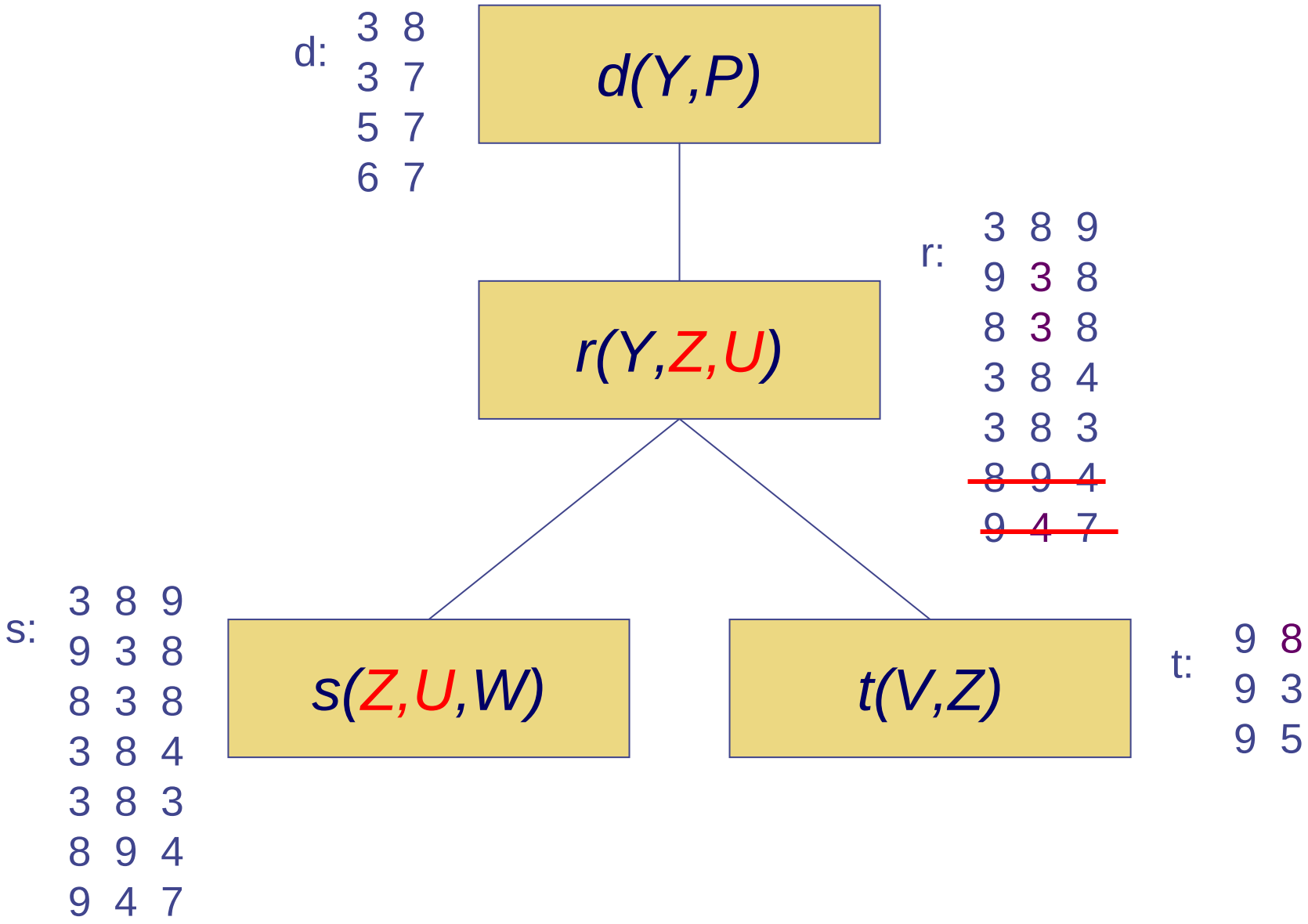

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
...
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

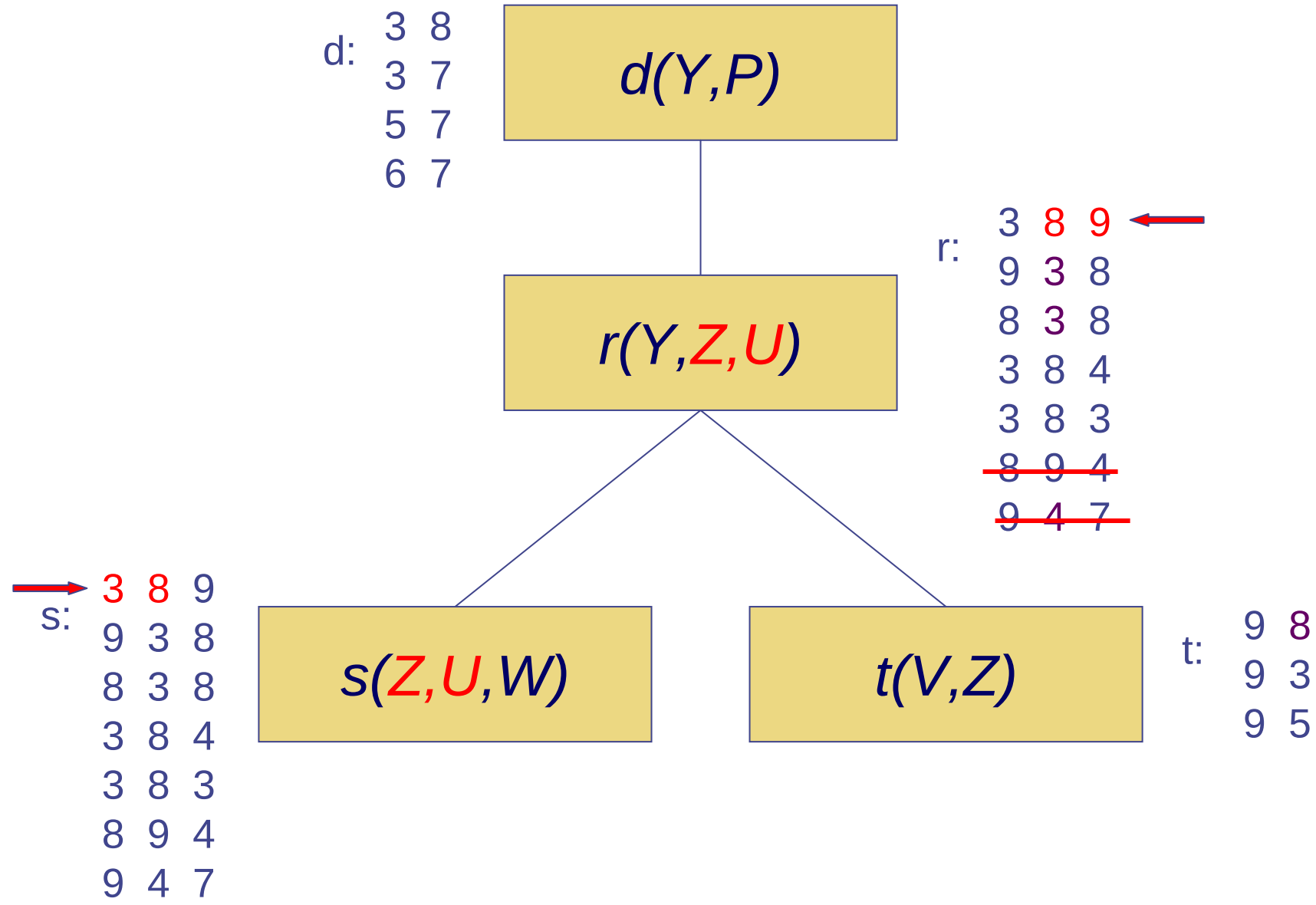

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

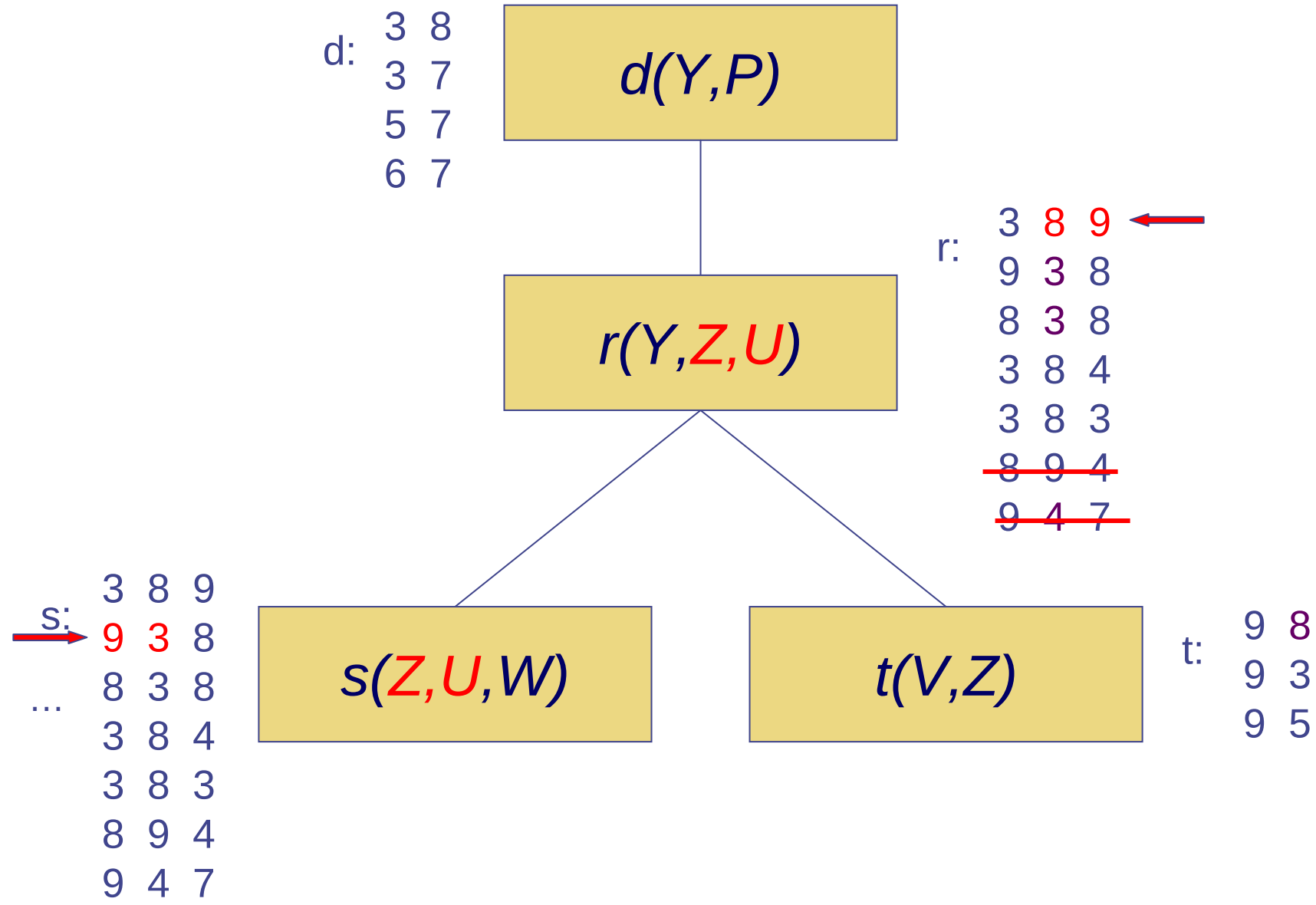

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

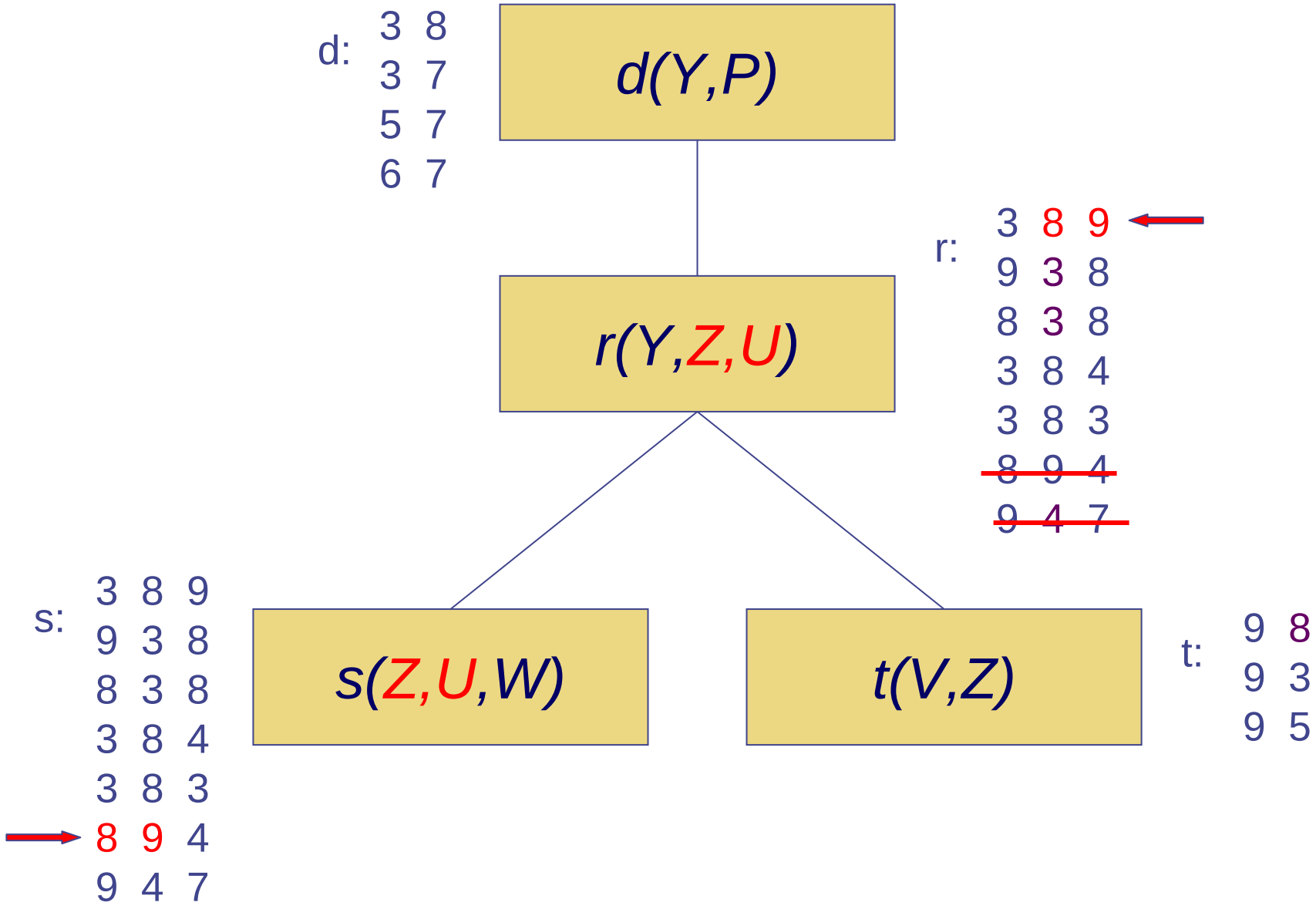

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

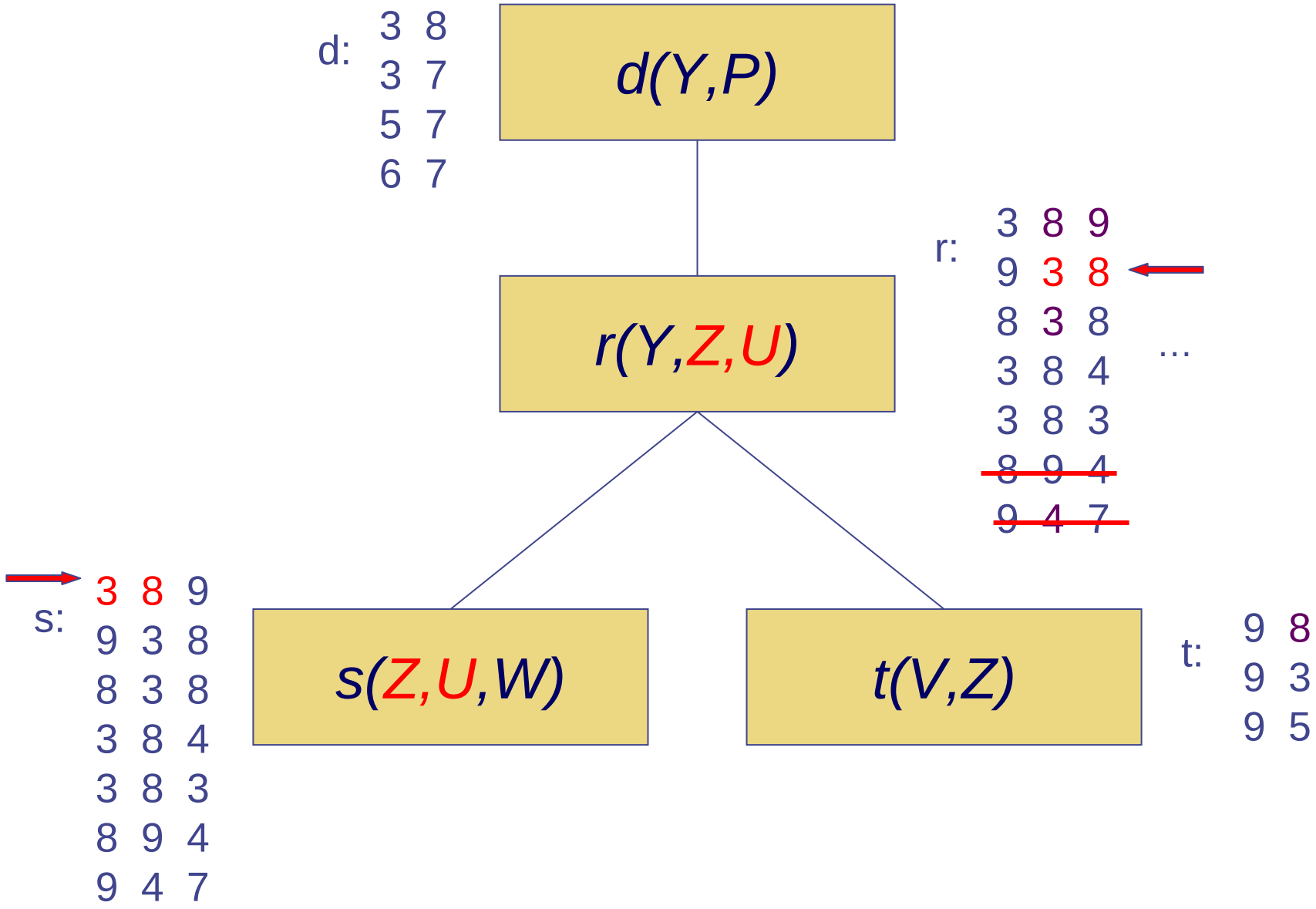

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
...
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

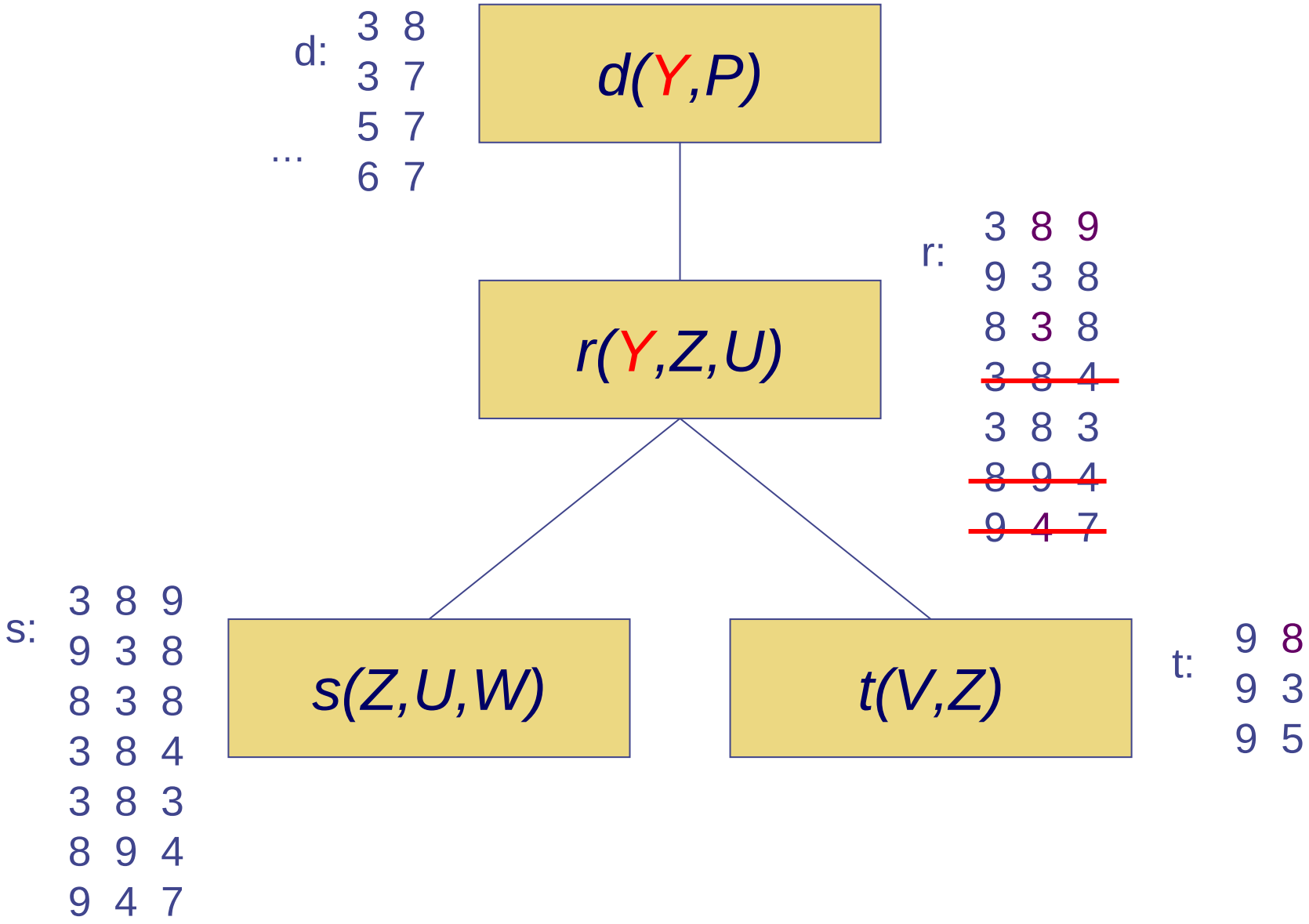

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

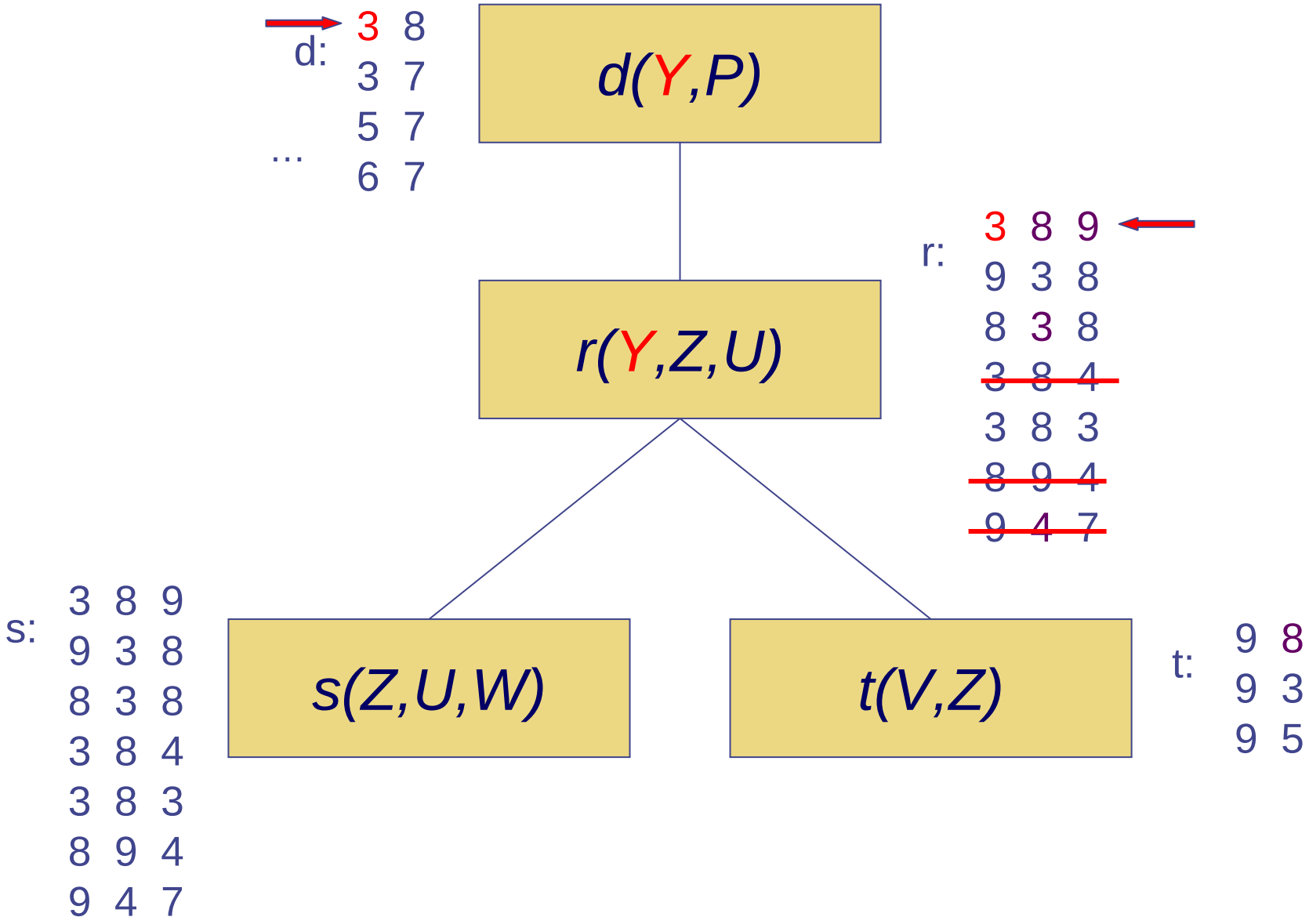

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

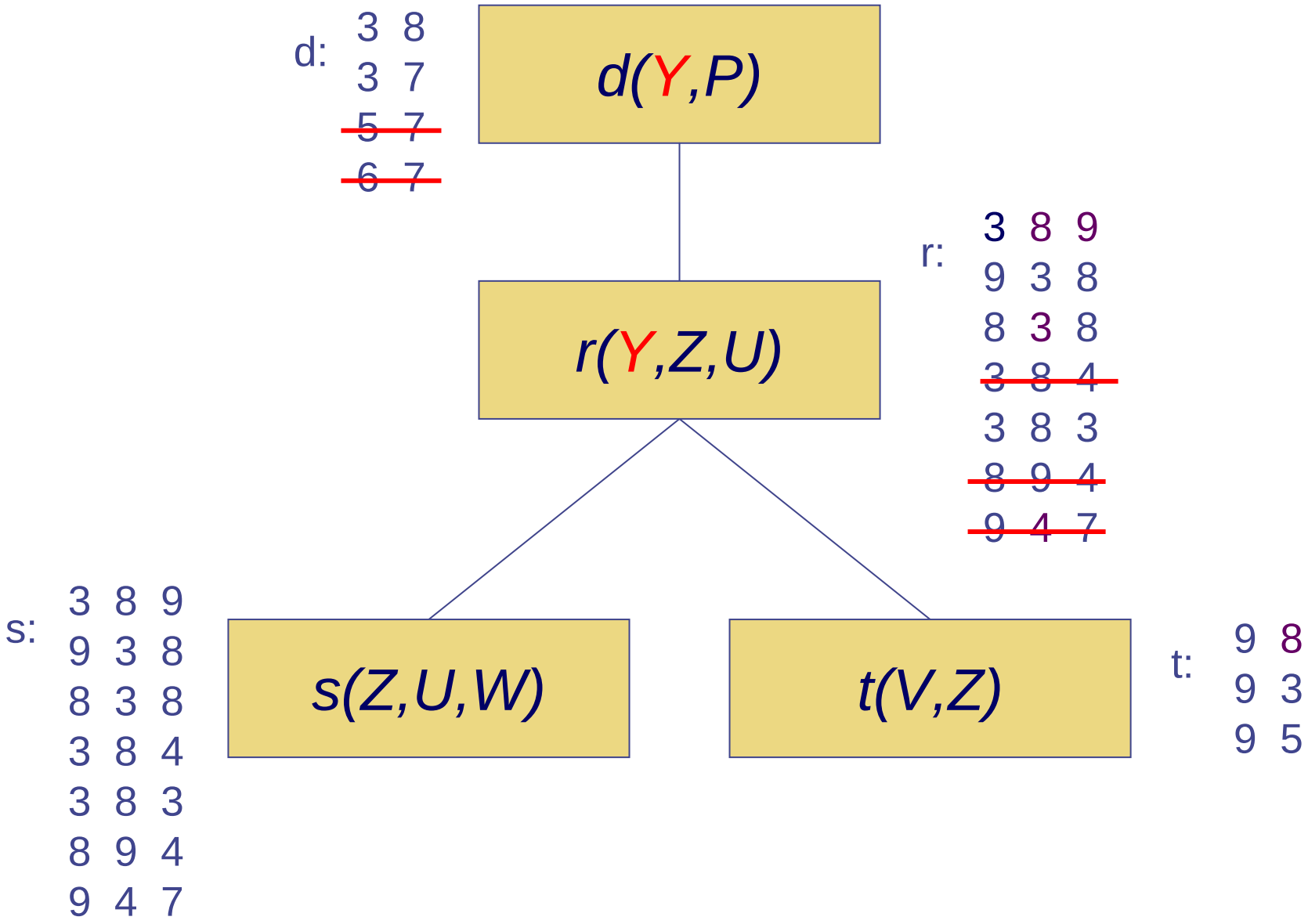

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
...
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

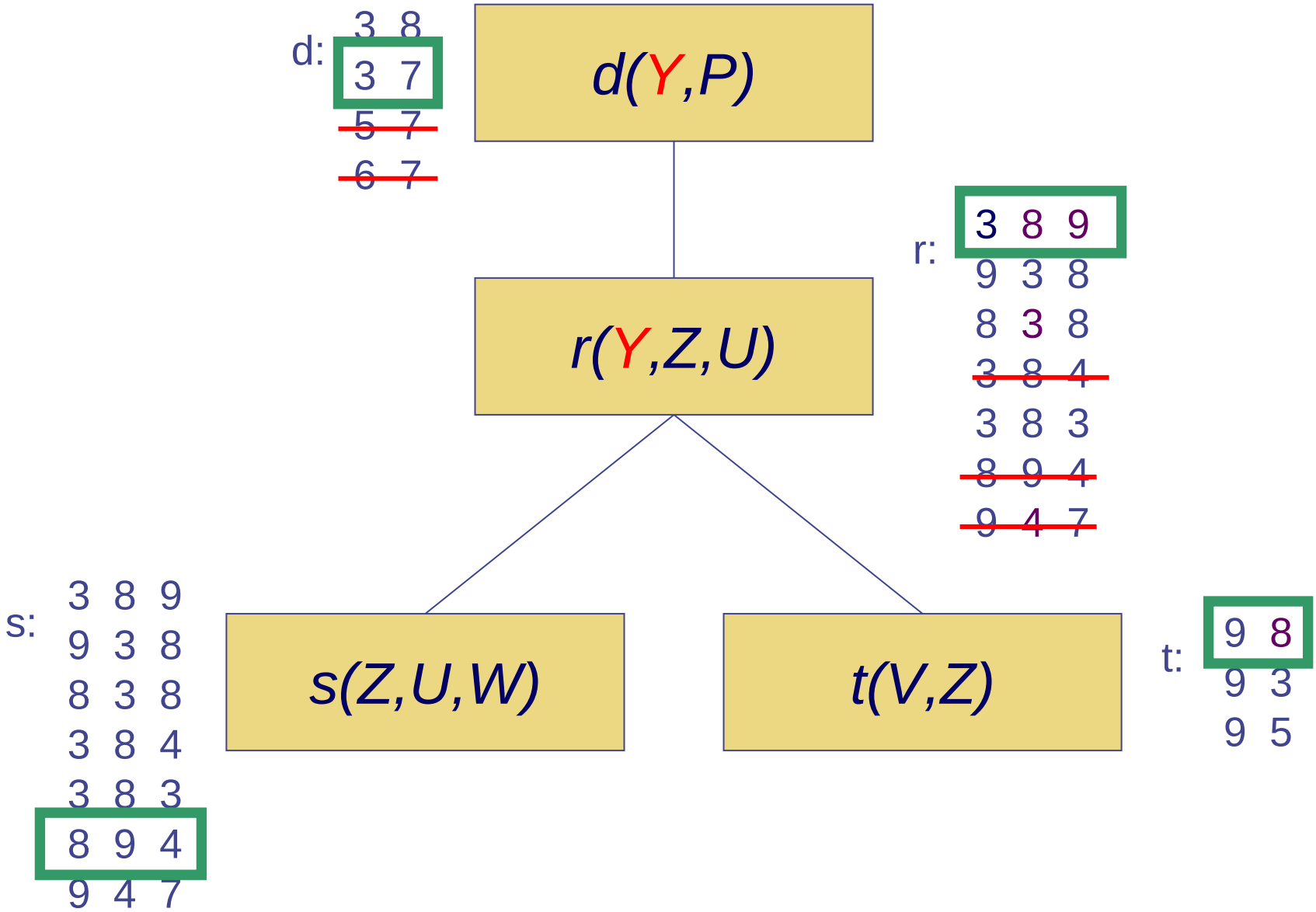

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5


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 CSPs or Boolean queries ?

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

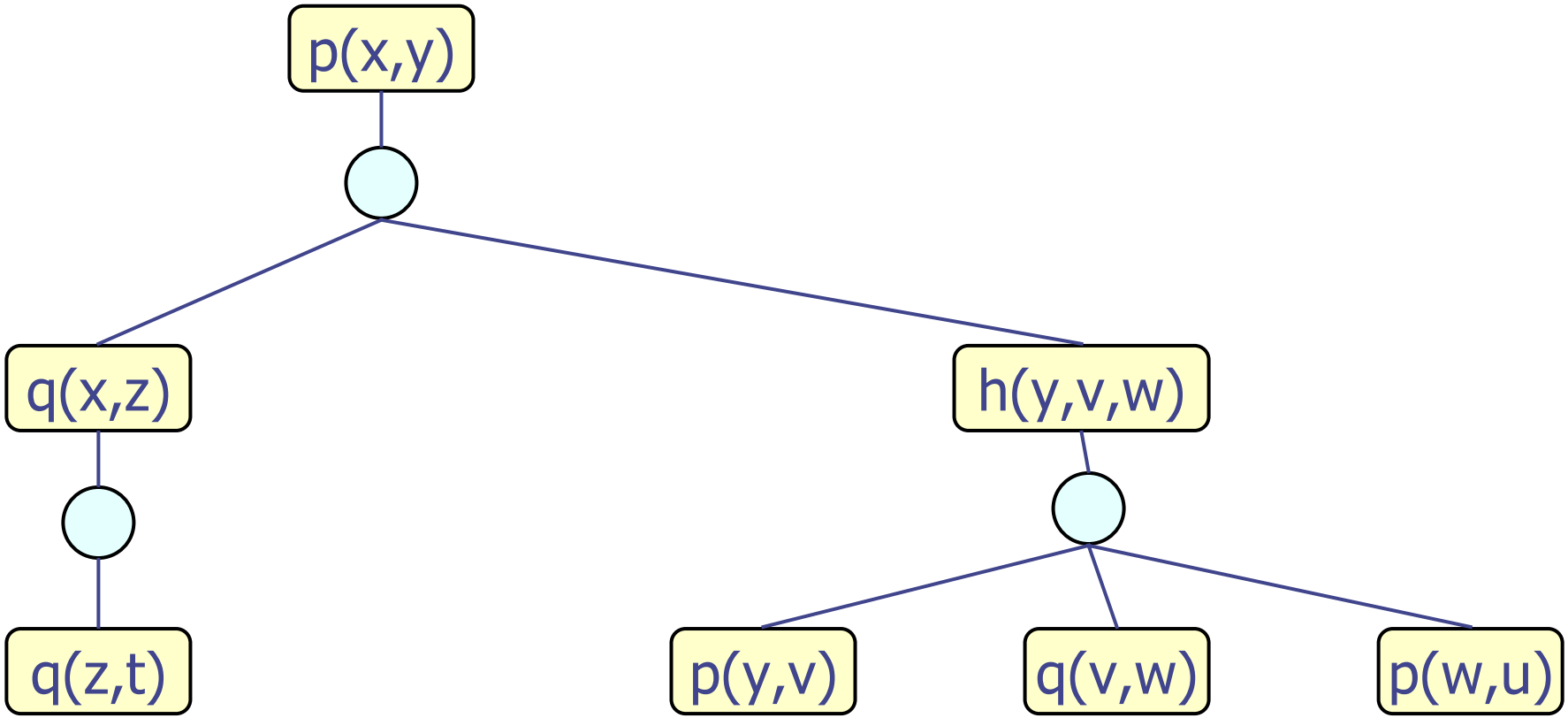
LOGCFL: class of problems/languages that are logspace-reducible to a CFL

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

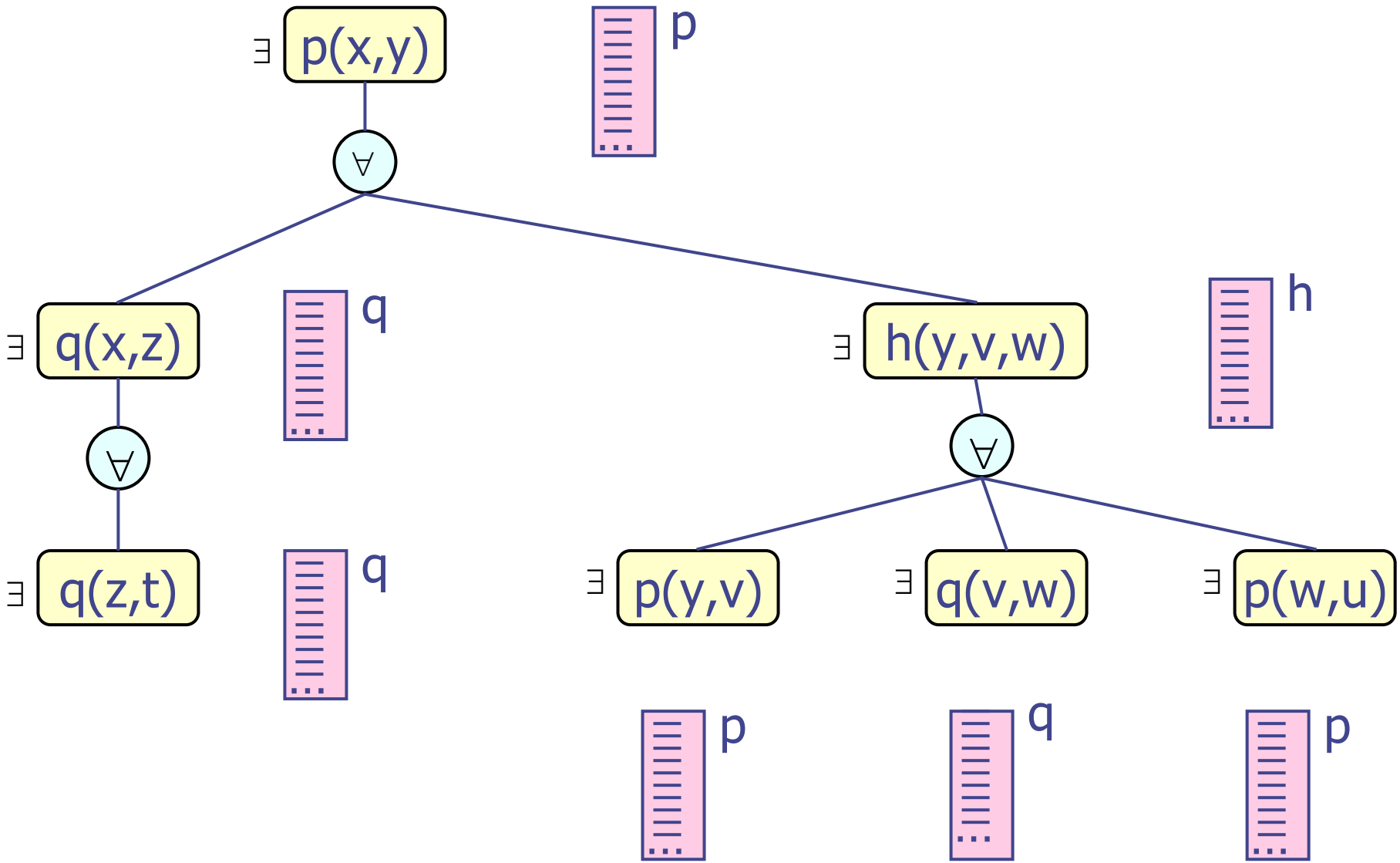
Characterization of LOGCFL [Ruzzo80]:

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

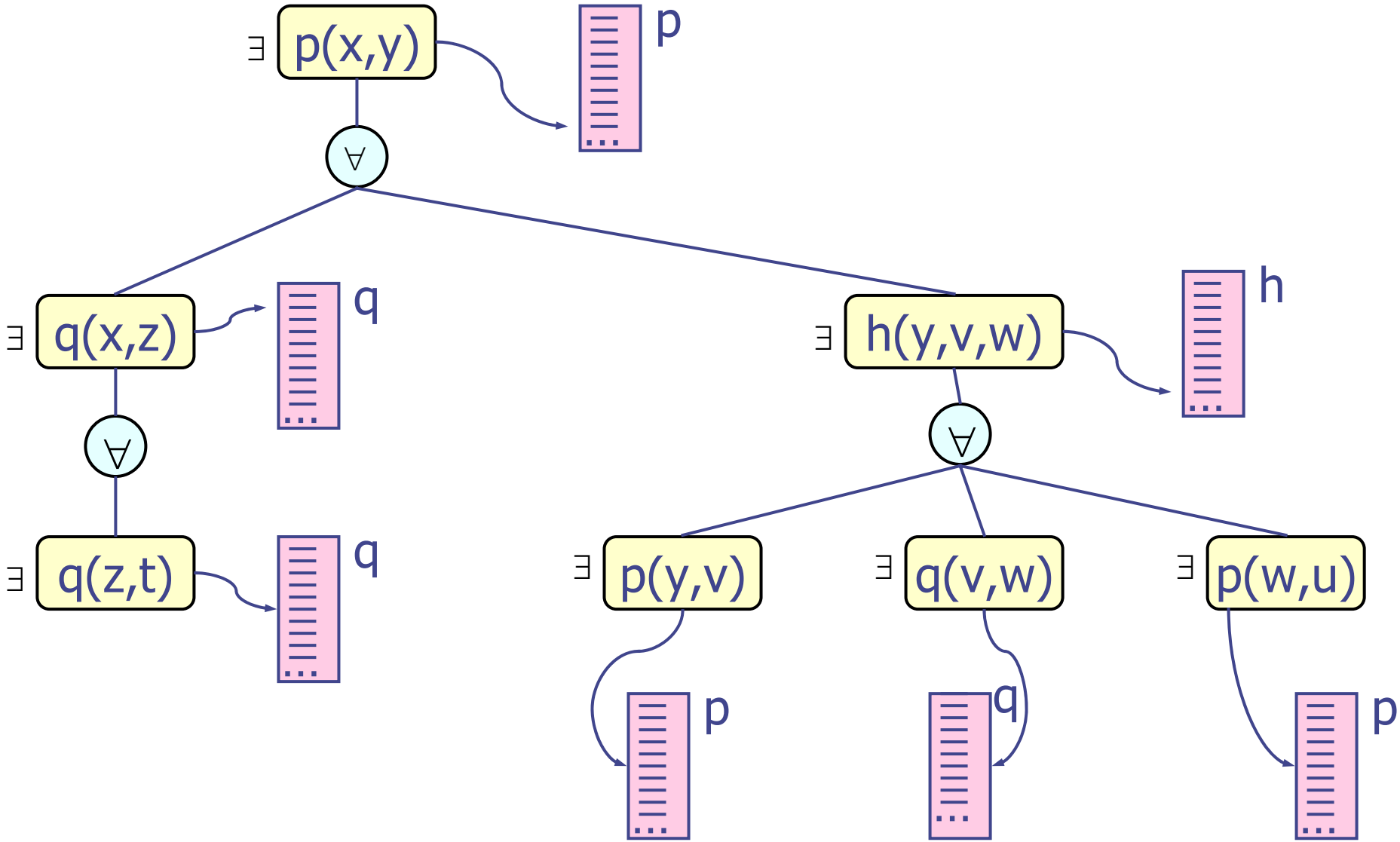
ABCQ is in LOGCFL



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

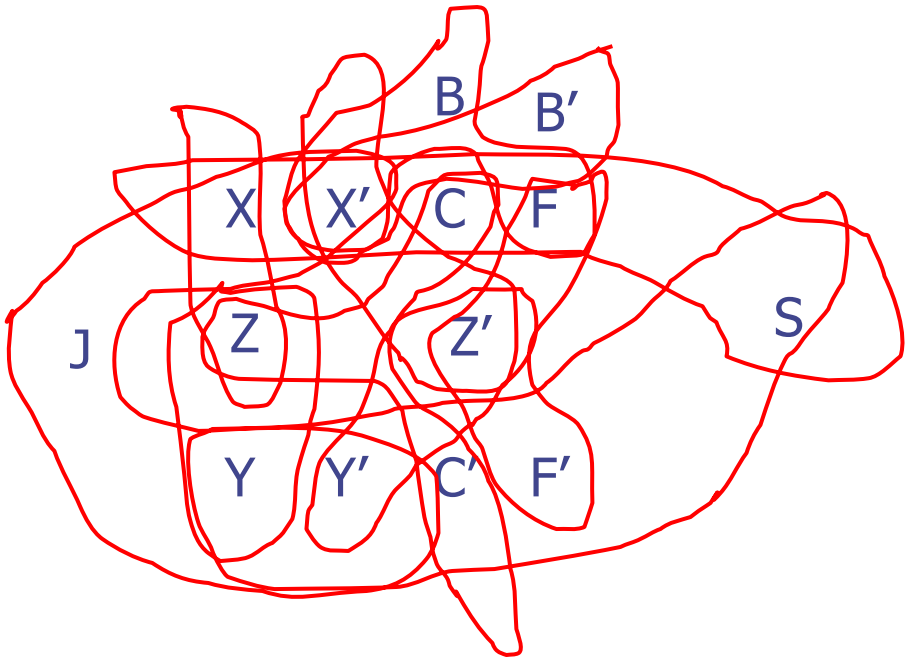
m number of atoms in the query

m = 11 !

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

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

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



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)* ≤ *k*
  - Computing a tree decomposition

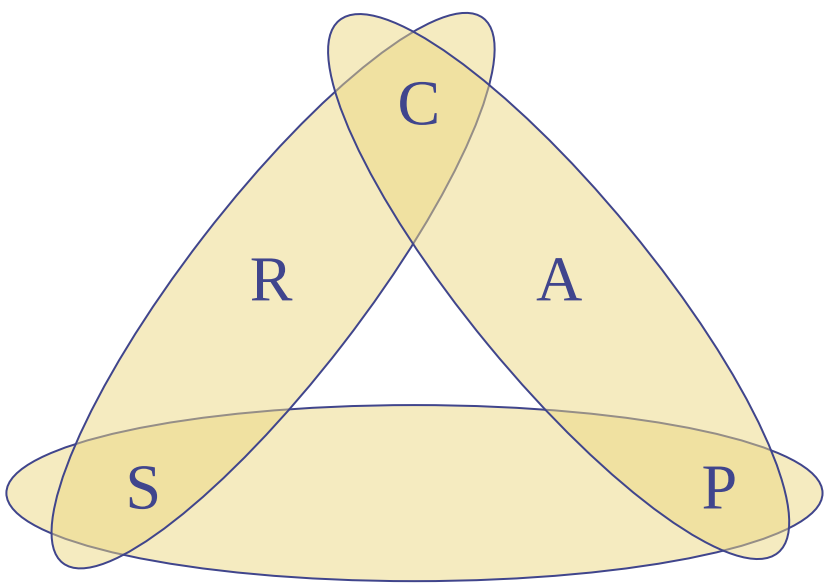
  } linear time (Bodlaender'96)
- Deciding CSP of treewidth *k*:

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

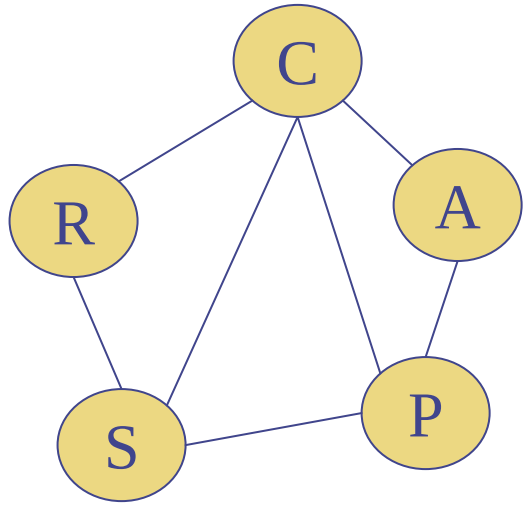
  LOGCFL-complete (G.L.S.'98)

Primal graphs of CSPs/Queries

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

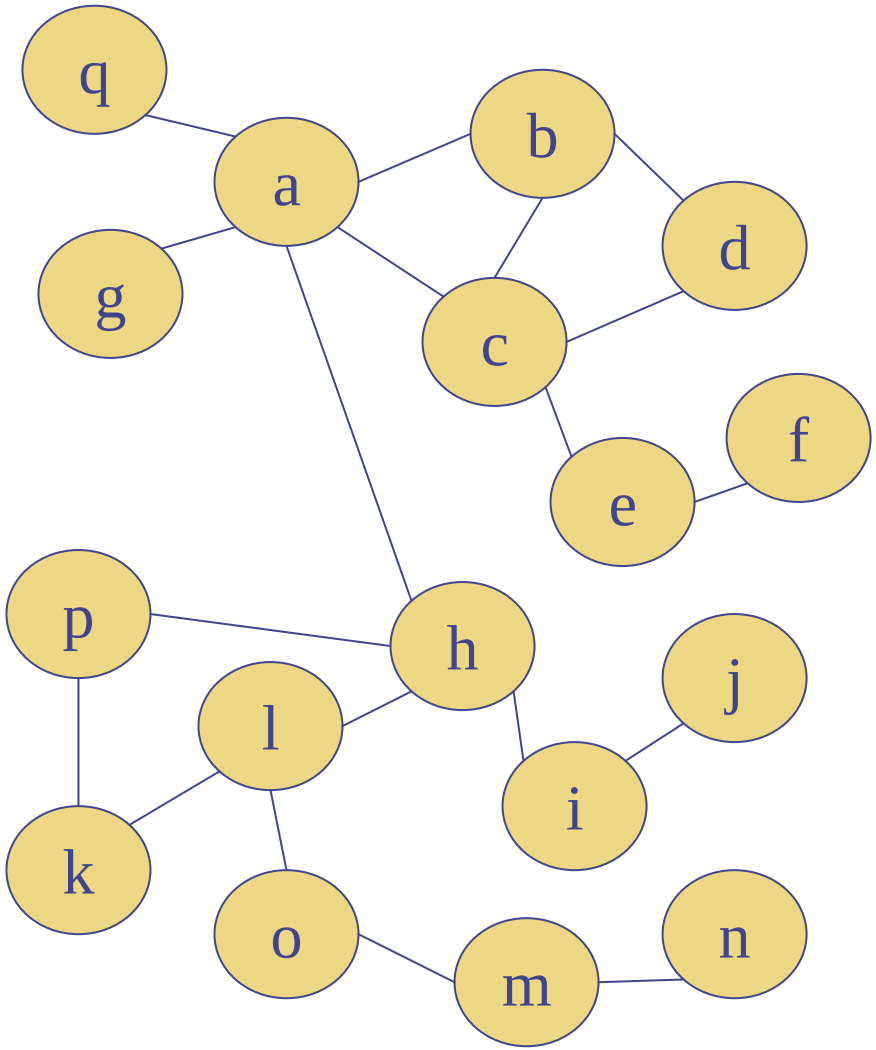


Hypergraph $H(Q)$

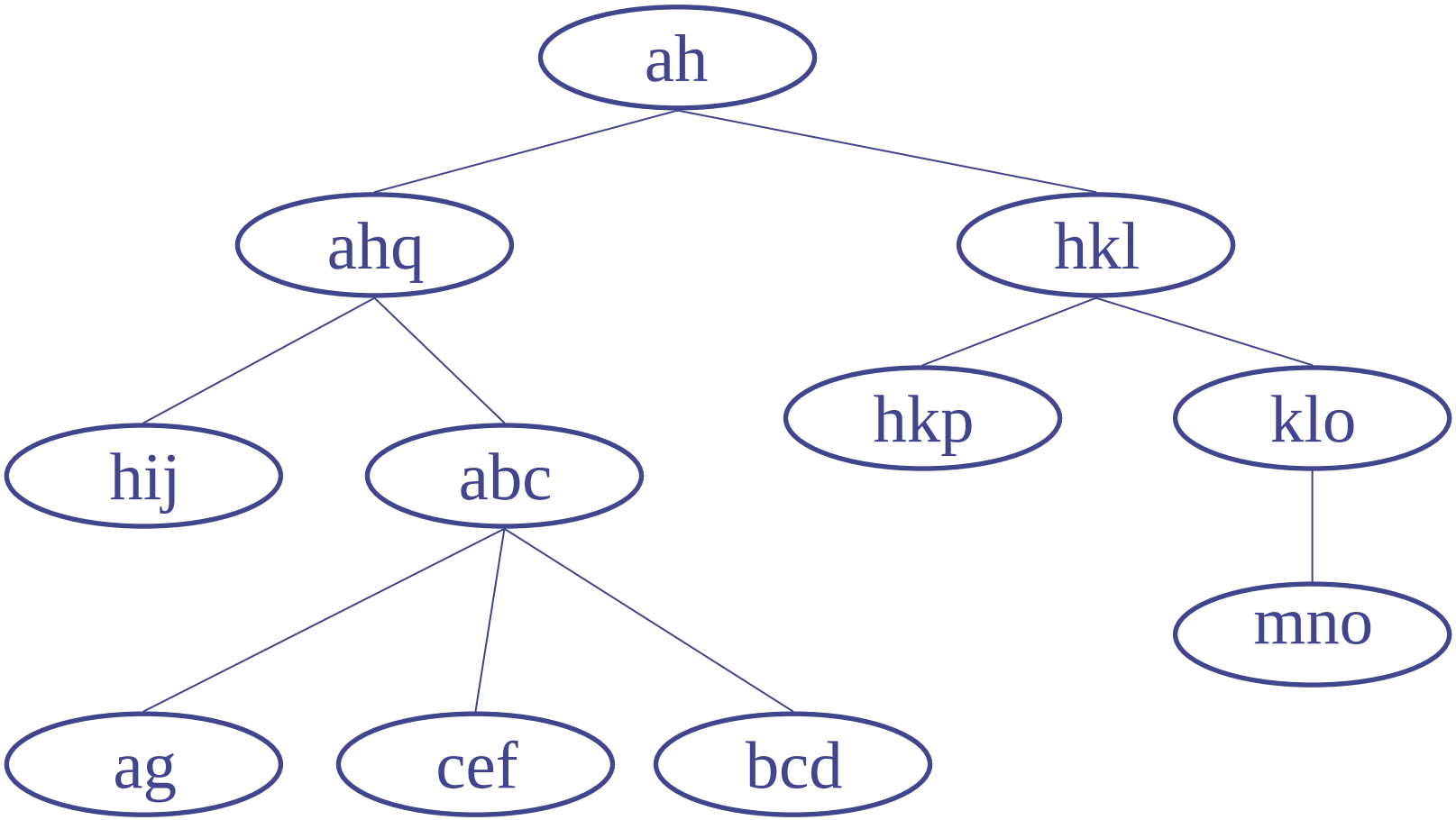


Primal graph $G(Q)$

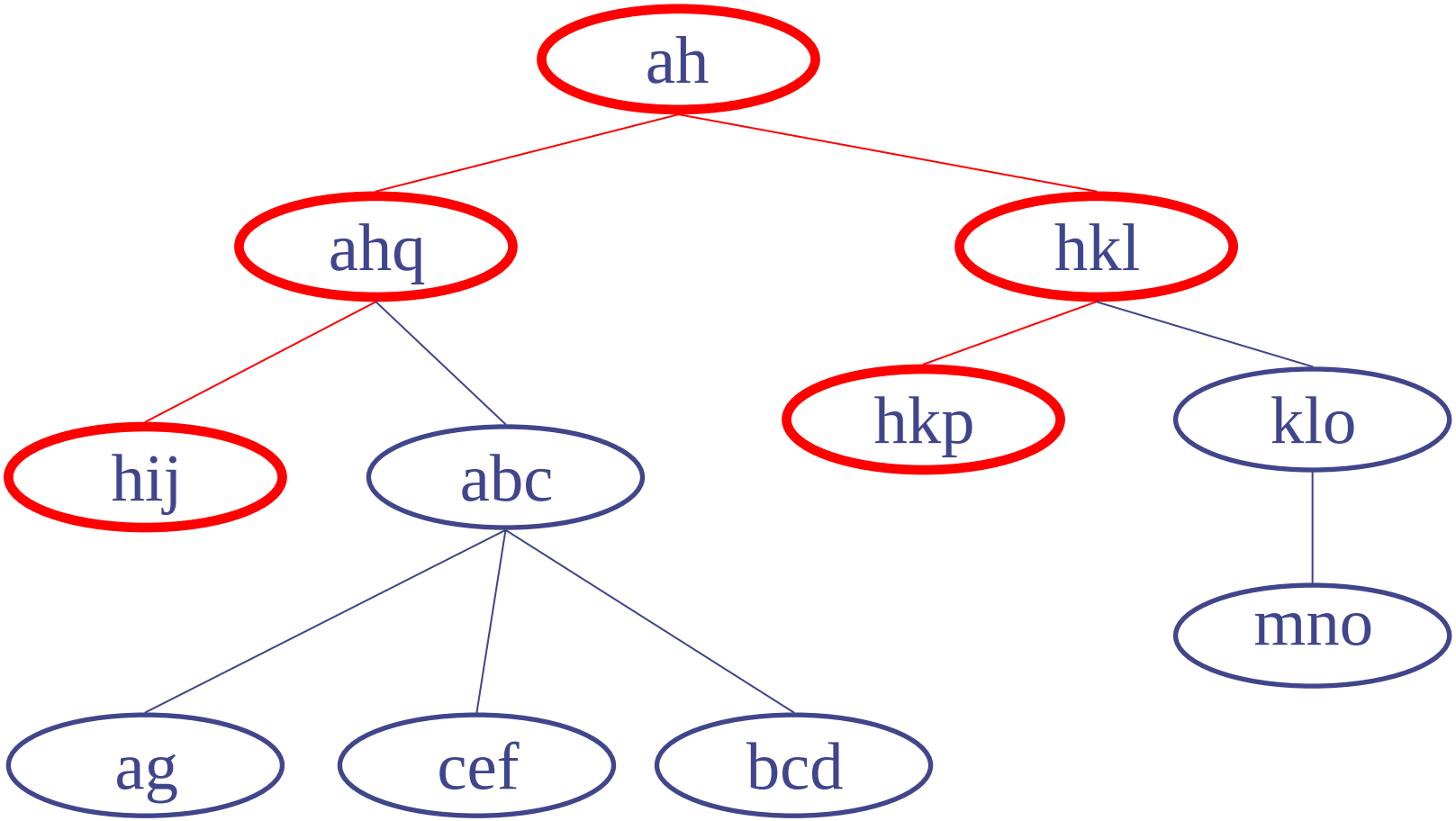
Example: a cyclic graph



A tree decomposition of width 2



Connectedness condition for *h*



Logical characterization of Treewidth

[Kolaitis & Vardi '98]

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

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

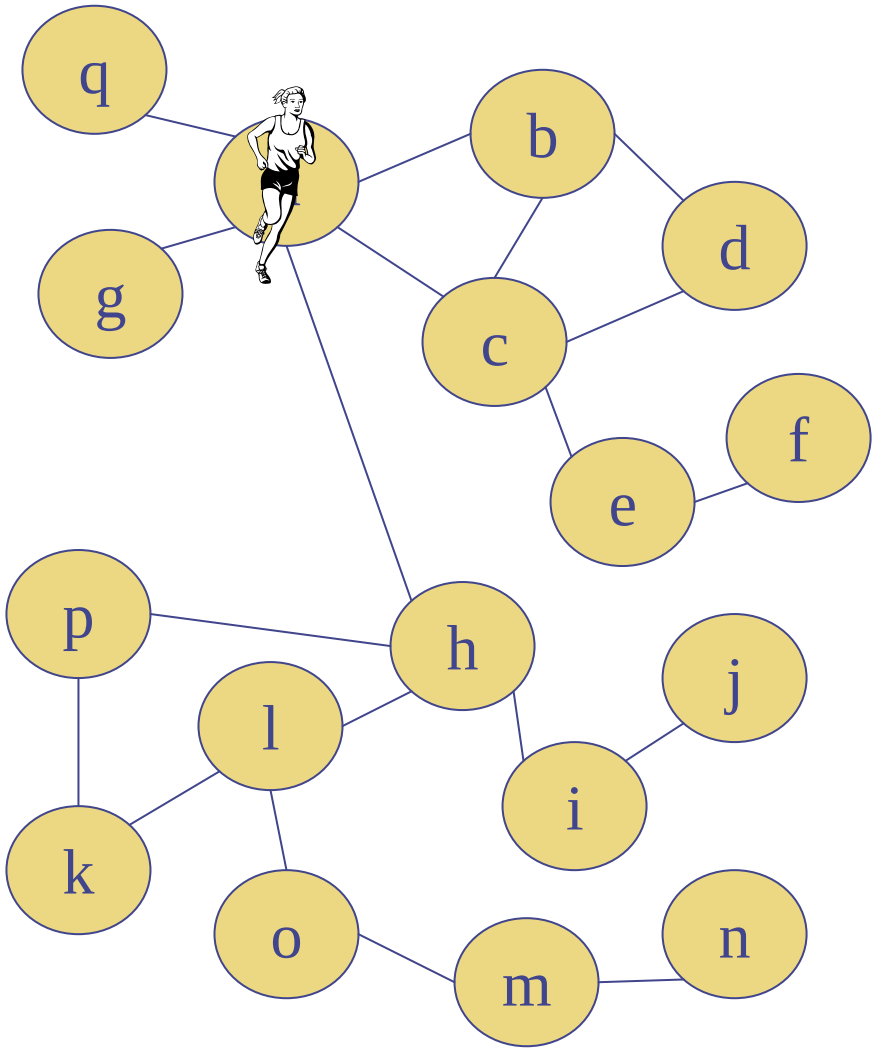
$$\boxed{\text{TW}[k] = \mathsf{L}^{k+1}}$$

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

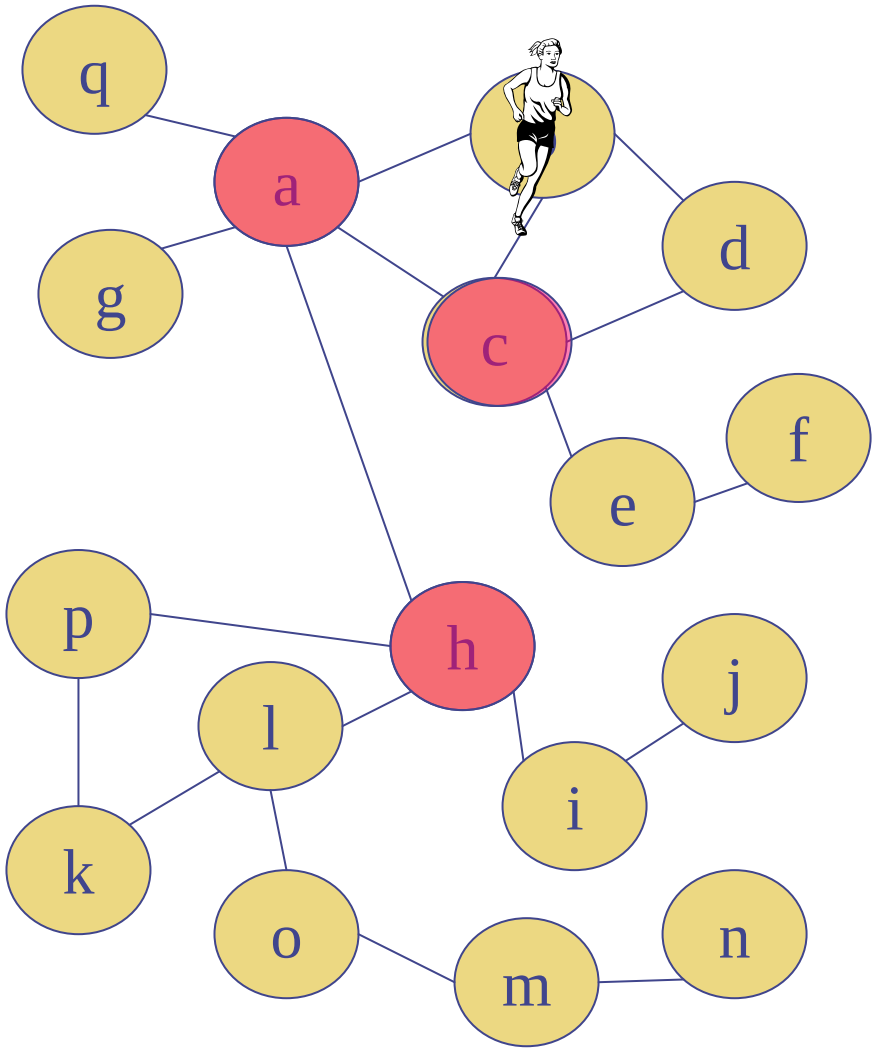
Game characterization of Treewidth

- 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

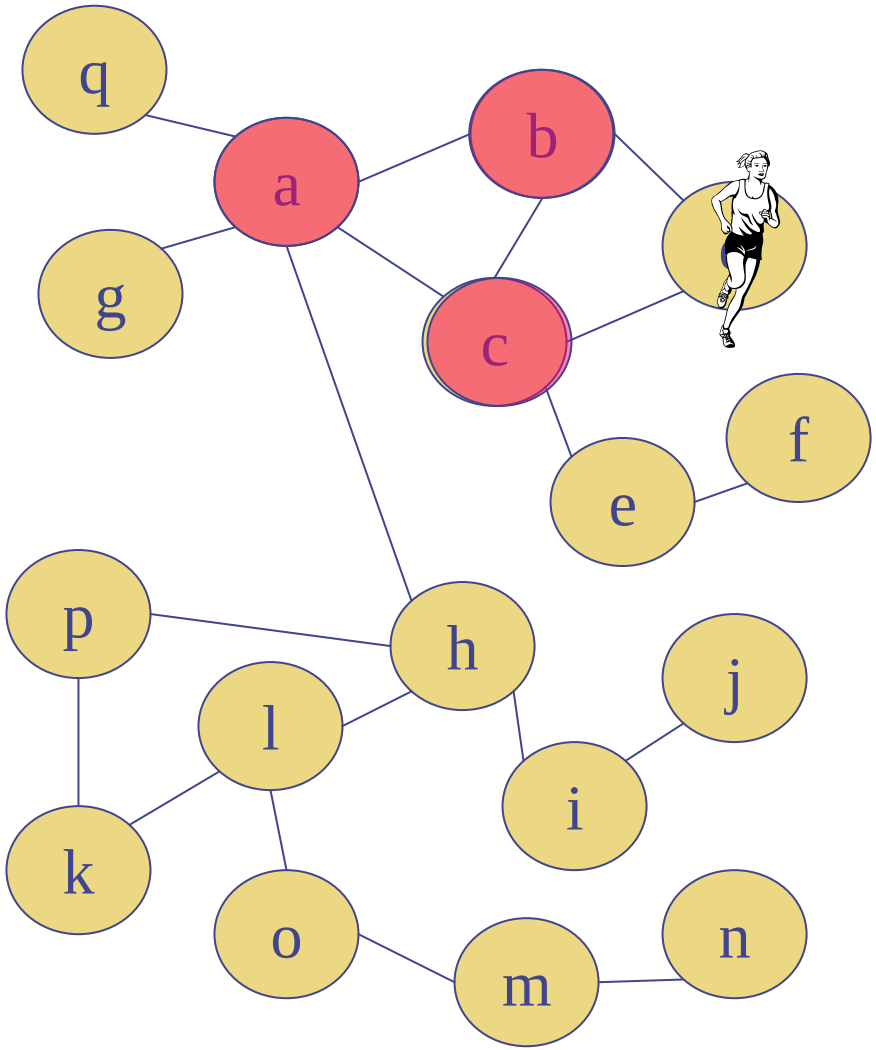
Playing the game



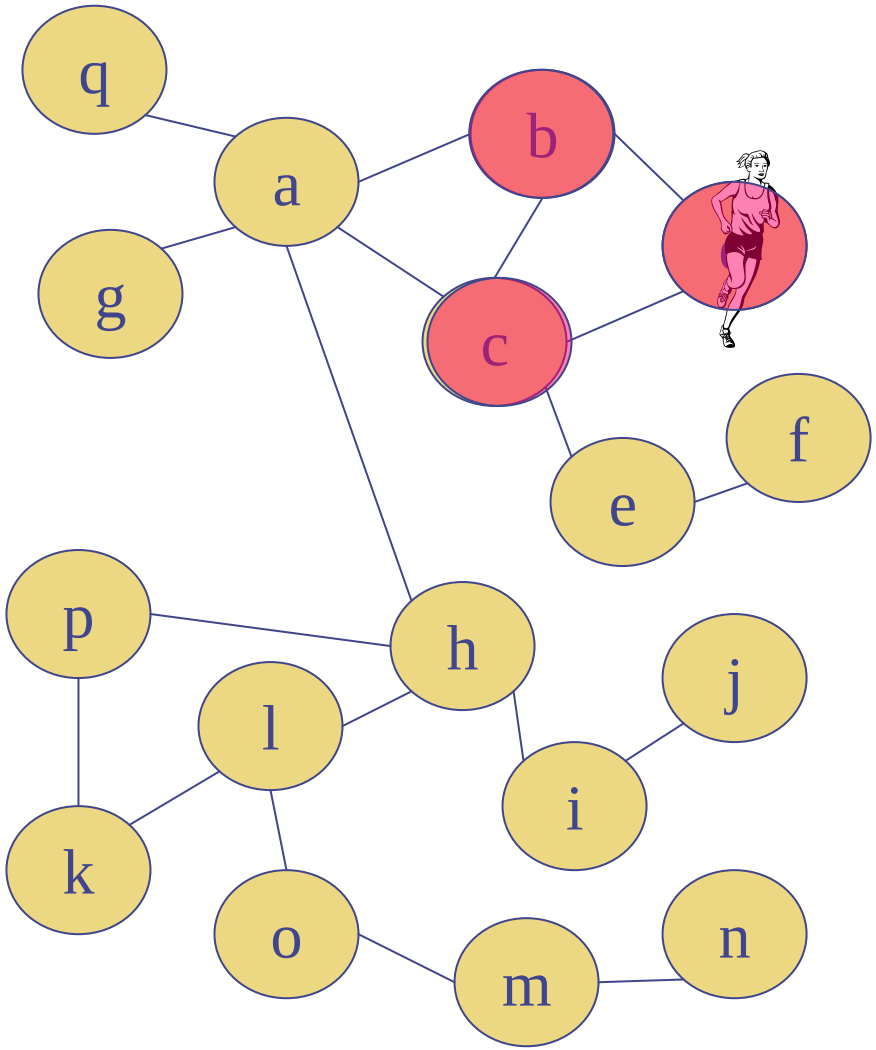
Playing the game



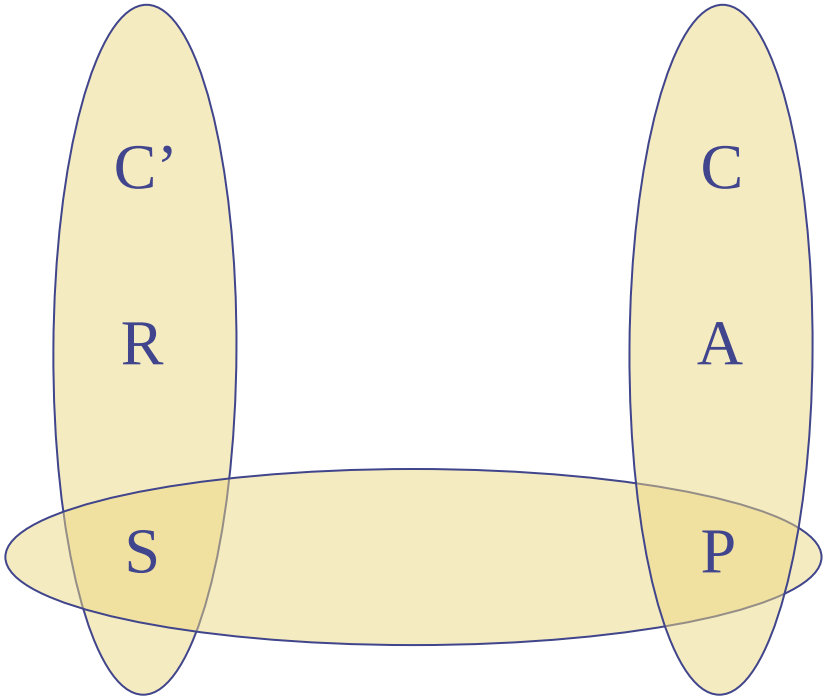
Playing the game



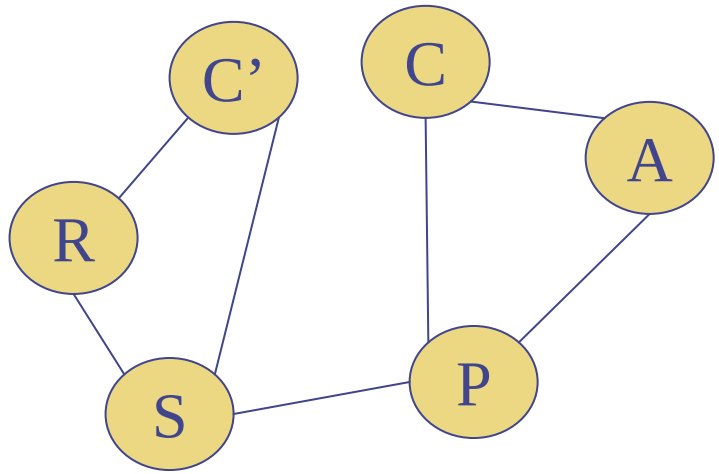
Playing the game



Hypergraphs vs Graphs (1)

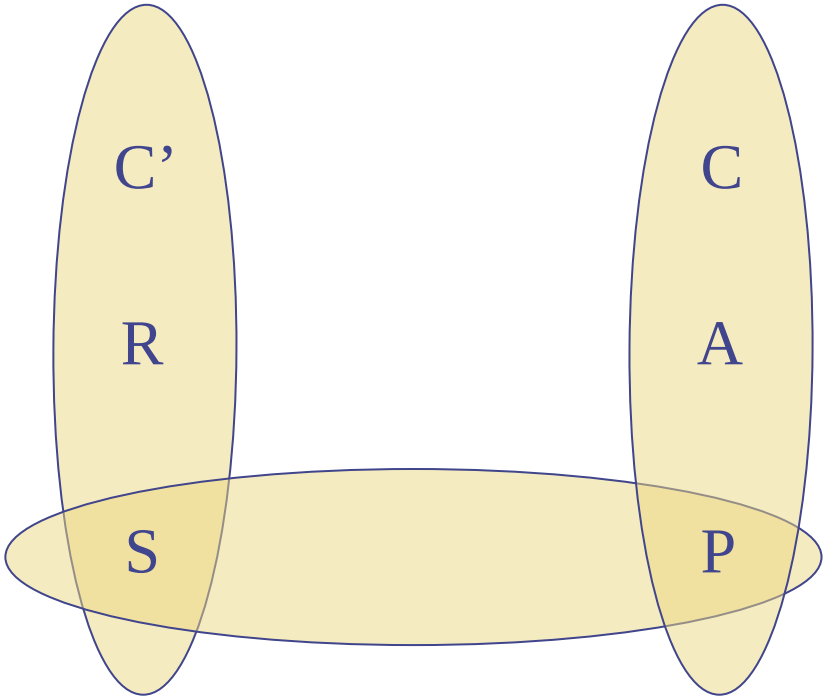


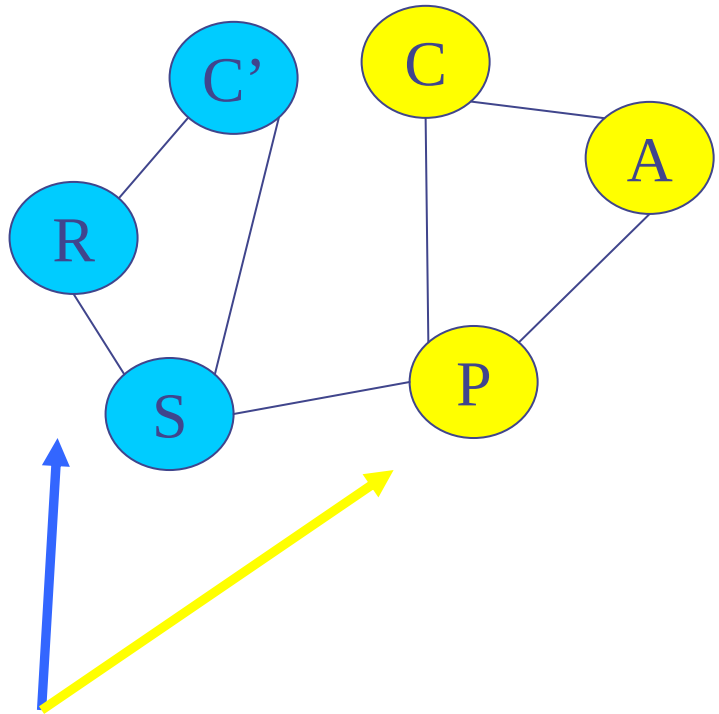
An acyclic hypergraph



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!

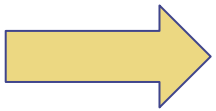
Example:

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

is acyclic, obviously polynomial,

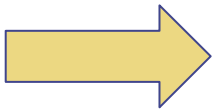
but has treewidth $n-1$

Beyond treewidth



Bounded Degree of Cyclicity (Hinges)

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



Bounded Query width

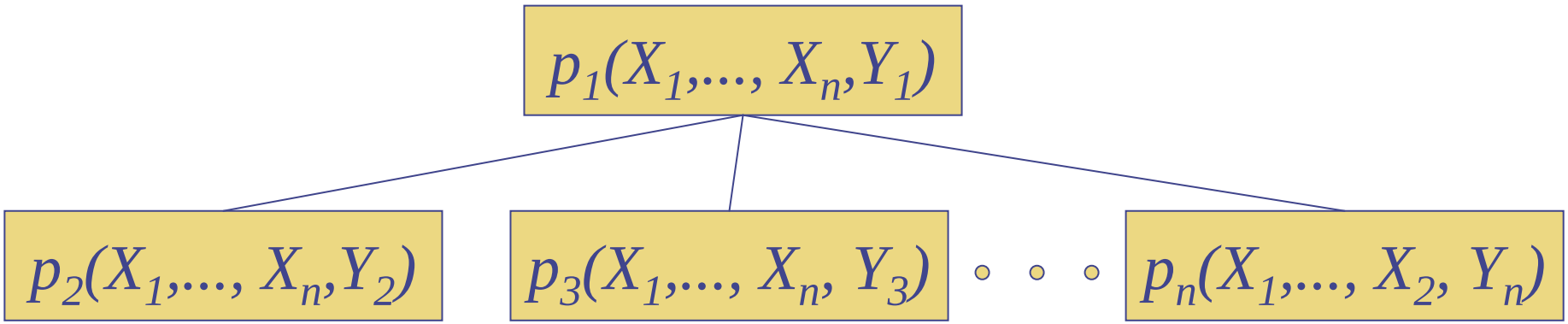
[Chekuri & Rajaraman ’97]

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

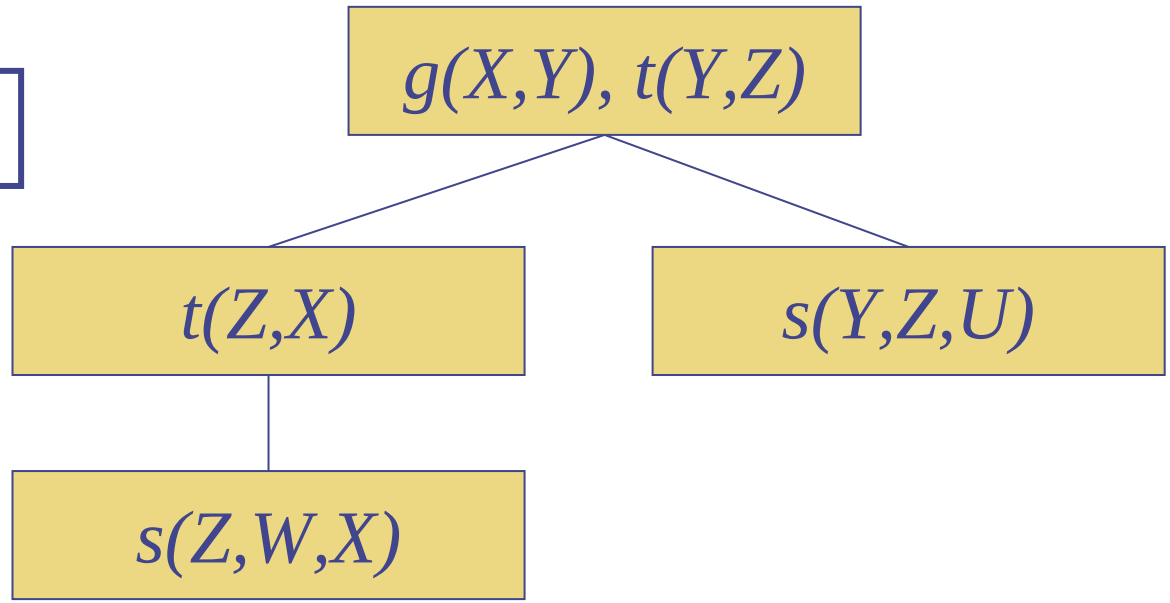


- Every atom/hyperarc appears in some node
- Connectedness conditions for variables and atoms

Decomposition of cyclic queries

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

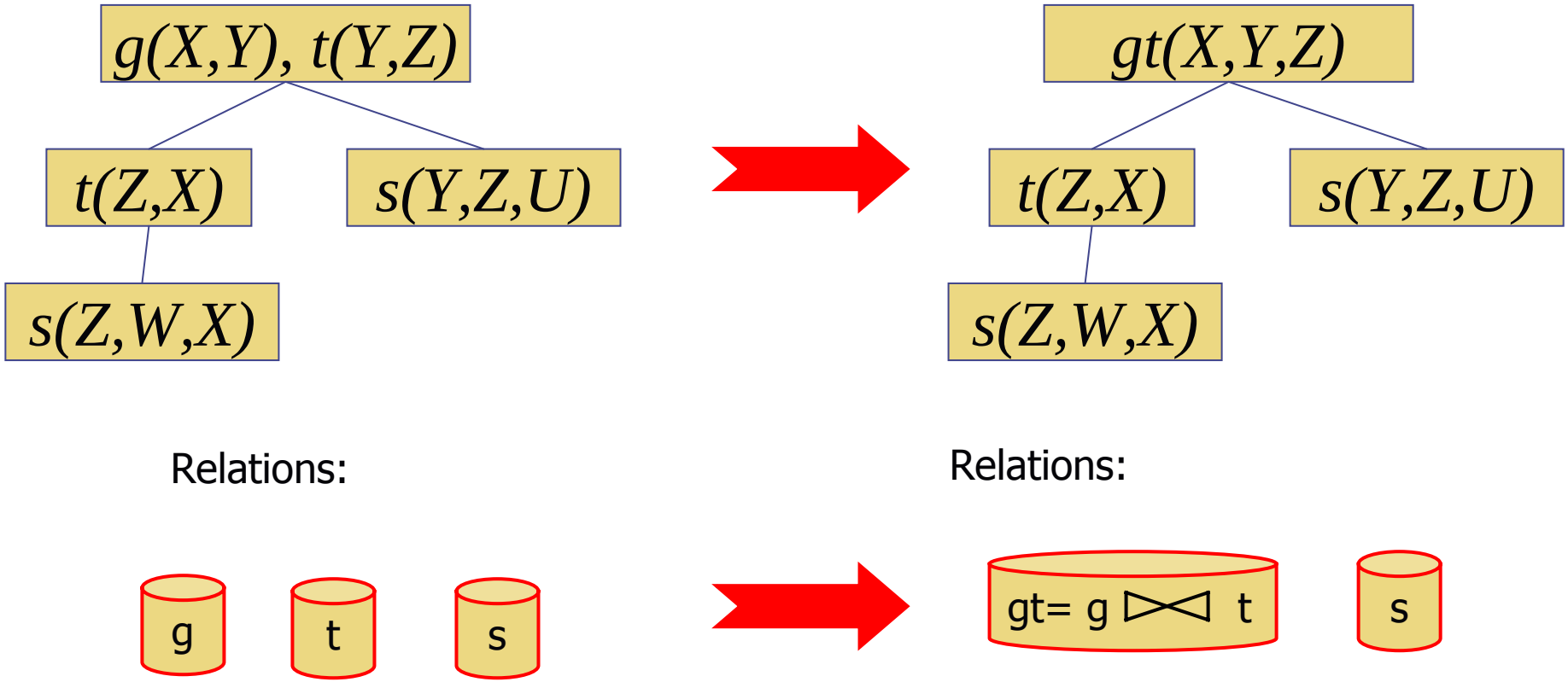
Query width = 2



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

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



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

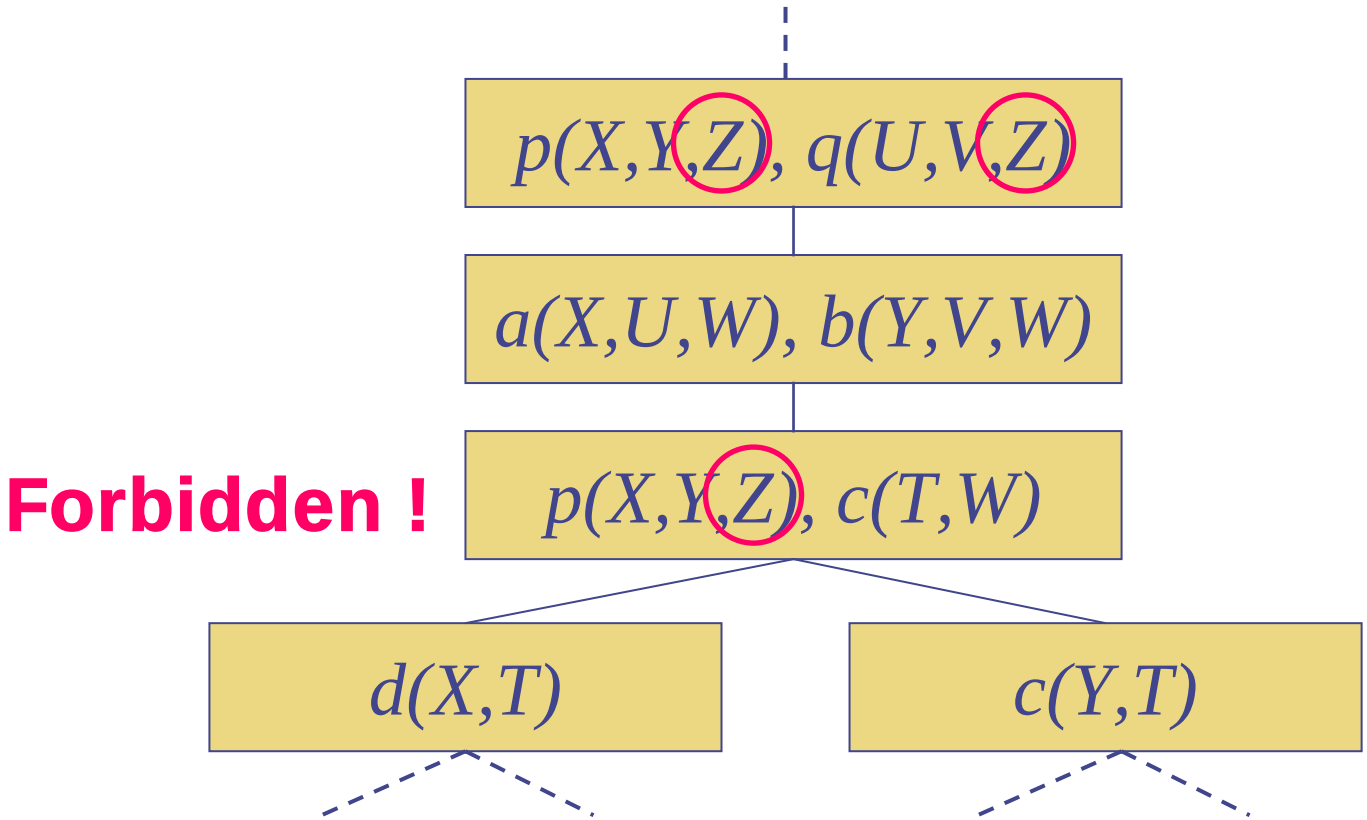
A negative answer (G.L.S. '99)

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

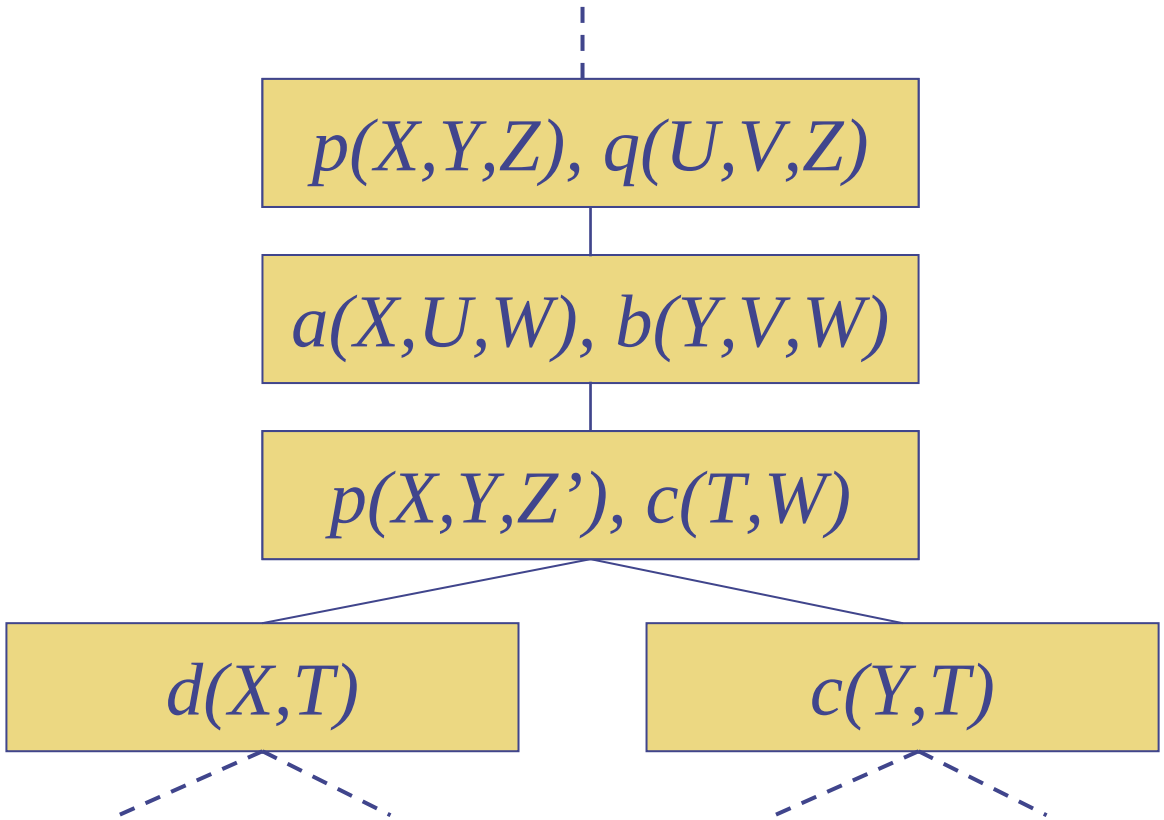
NP-hardness id due to an overly strong condition in the definition of query decomposition



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

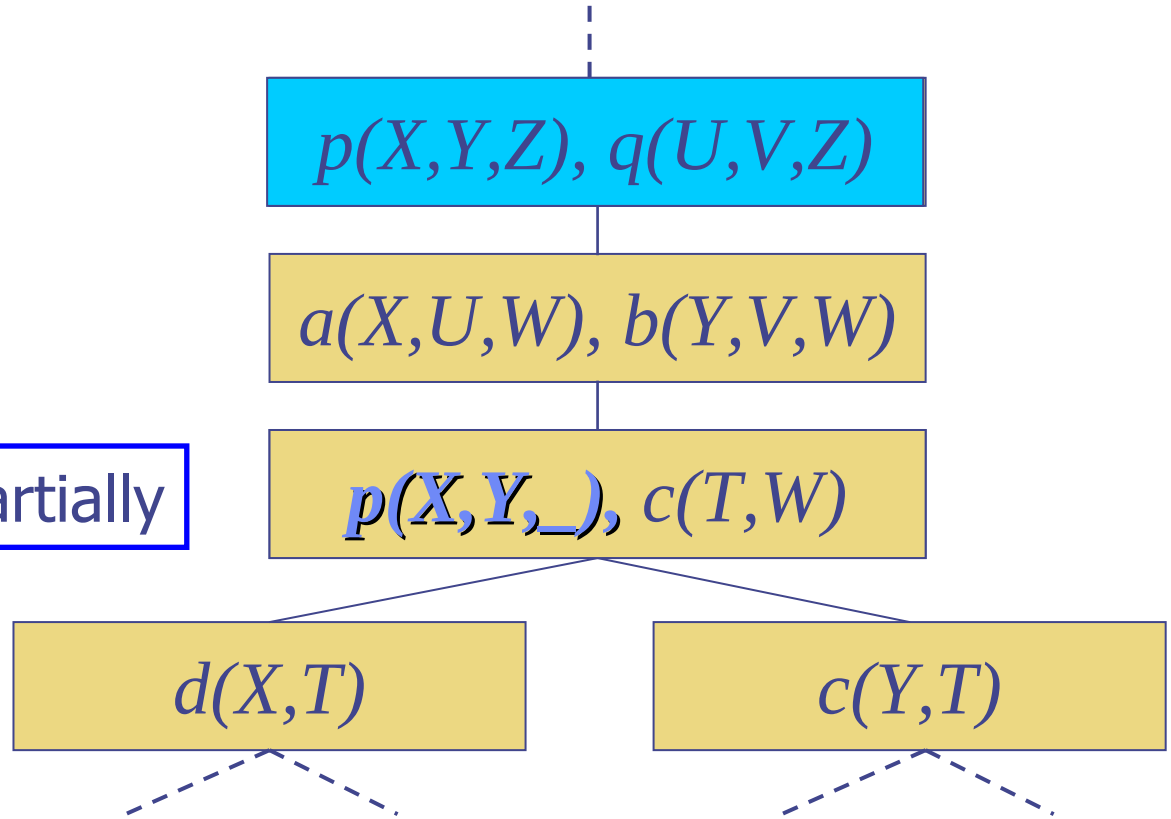
Query atoms can be used "partially" as long as the full atom appears somewhere else

More liberal than query decomposition

Grouping and Reusing Atoms

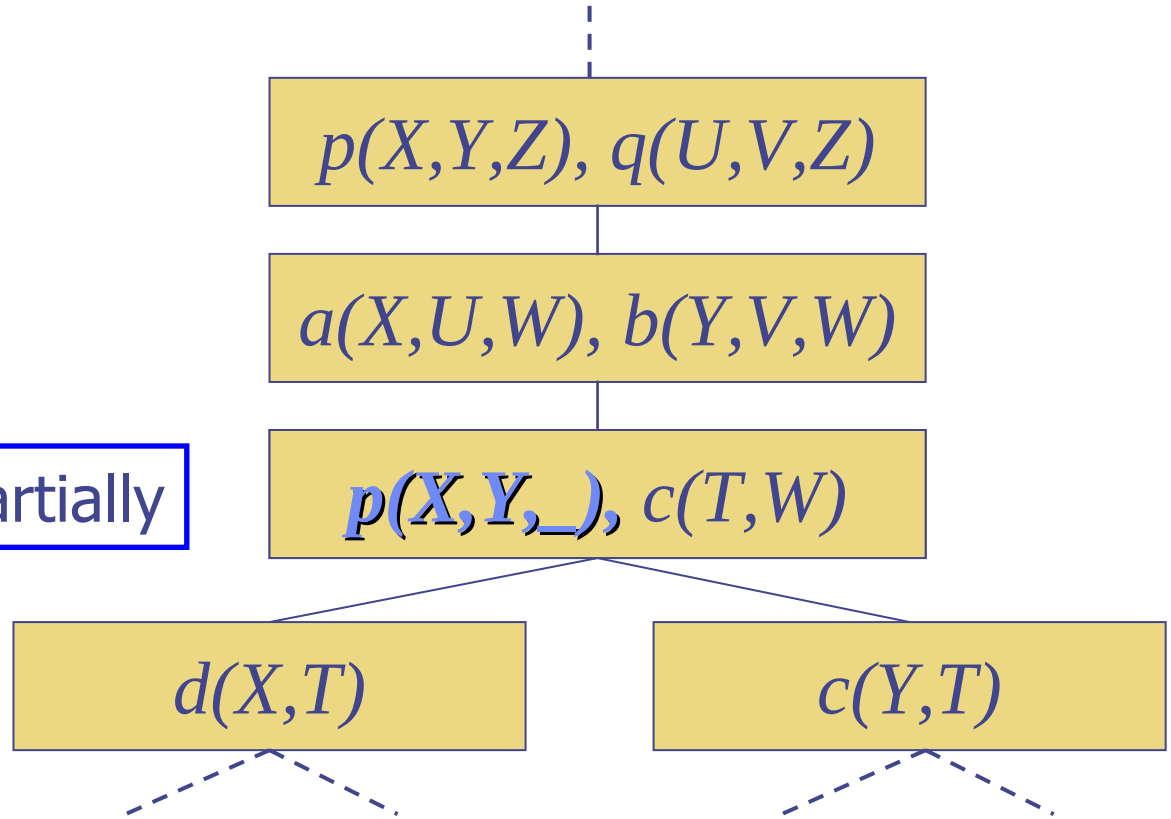
We group atoms

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

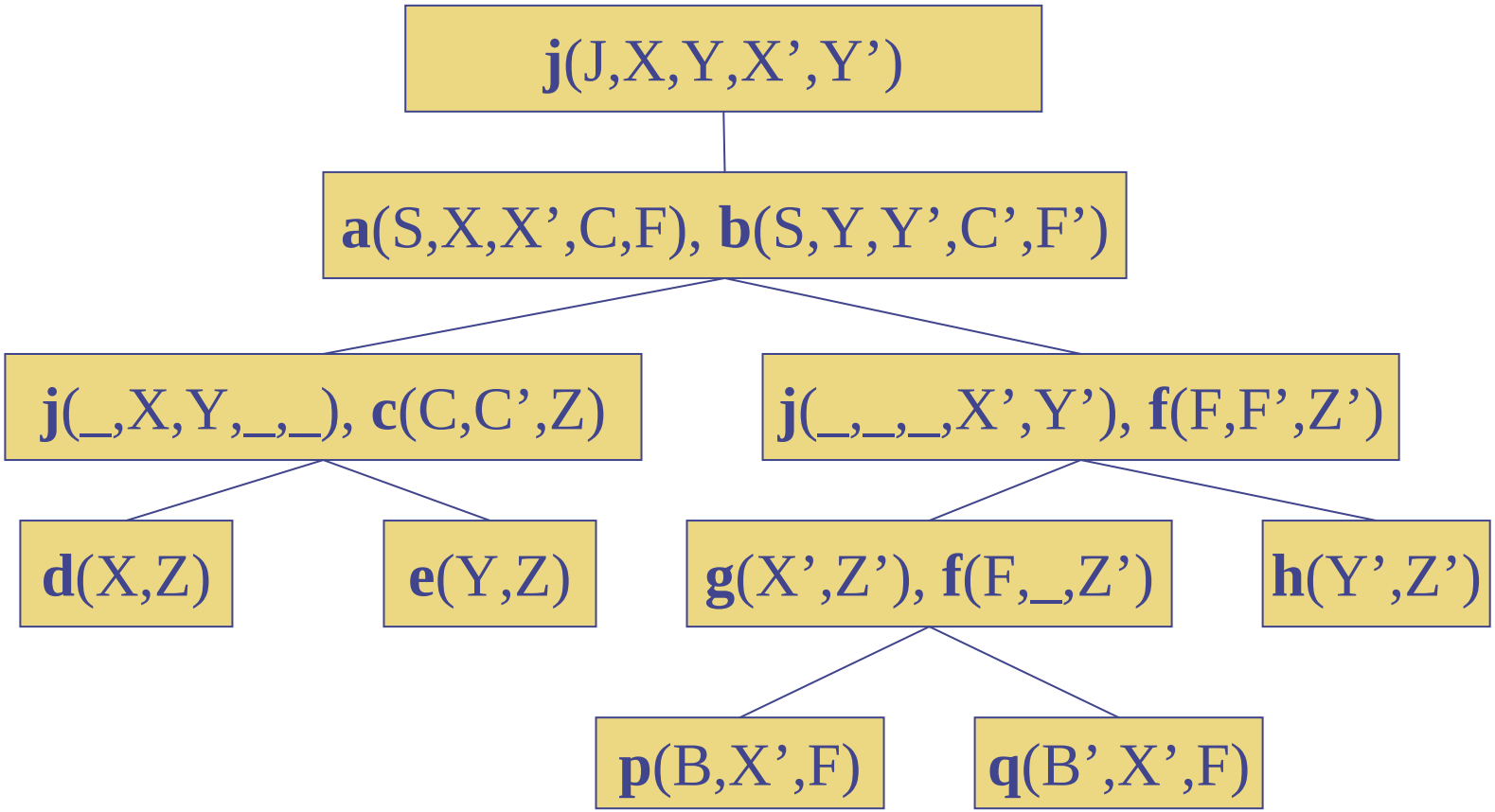


Reusing atoms

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

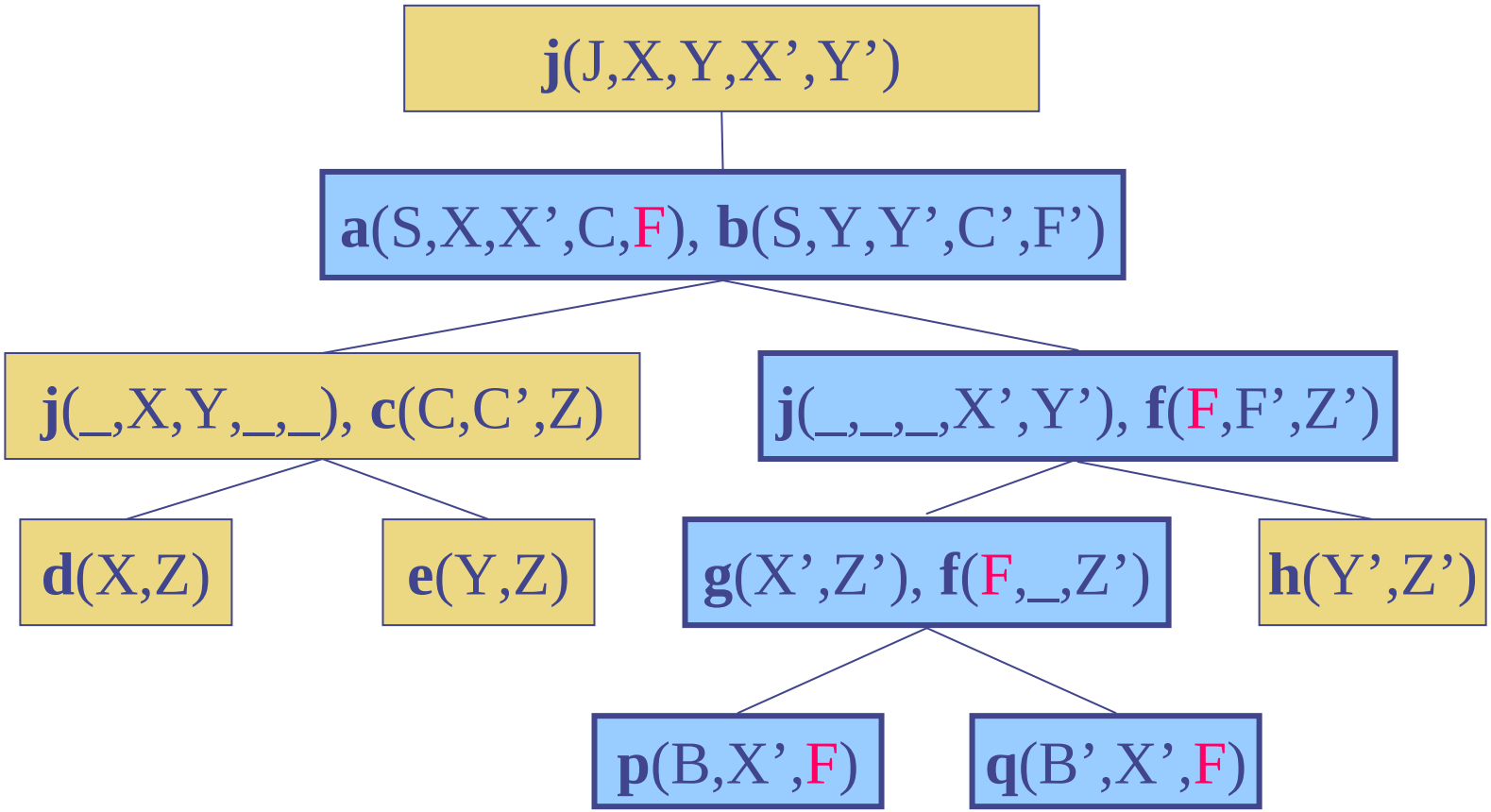


$$ans \leftarrow a(S,X,X',C,F) \wedge b(S,Y,Y',C',F') \wedge c(C,C',Z) \wedge d(X,Z) \wedge e(Y,Z) \wedge f(F,F',Z') \wedge g(X',Z') \wedge h(Y',Z') \wedge j(J,X,Y,X',Y') \wedge p(B,X',F) \wedge q(B',X',F)$$



Hypertree of width 2

Generalized Hypertree decomposition = Hypertree + Connectedness condition

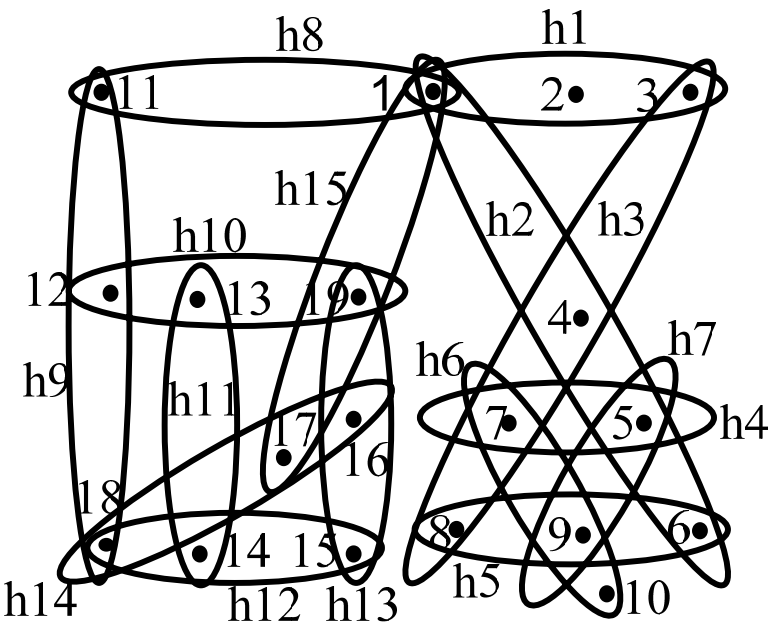


Alternative view and
generalized HT-Decompositions:

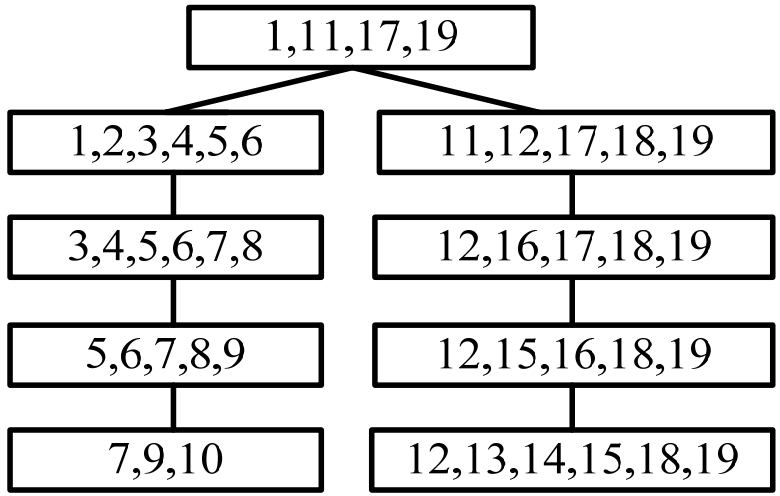
GHD = tree decomp + set covering

Tree decomposition of hypergraph

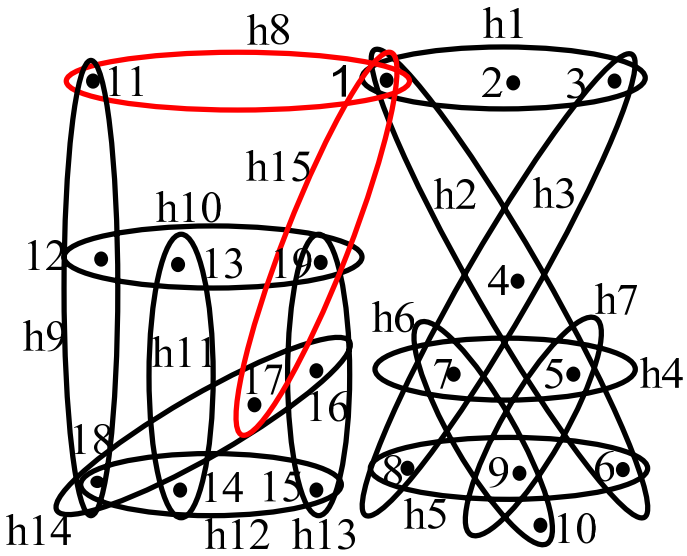
H

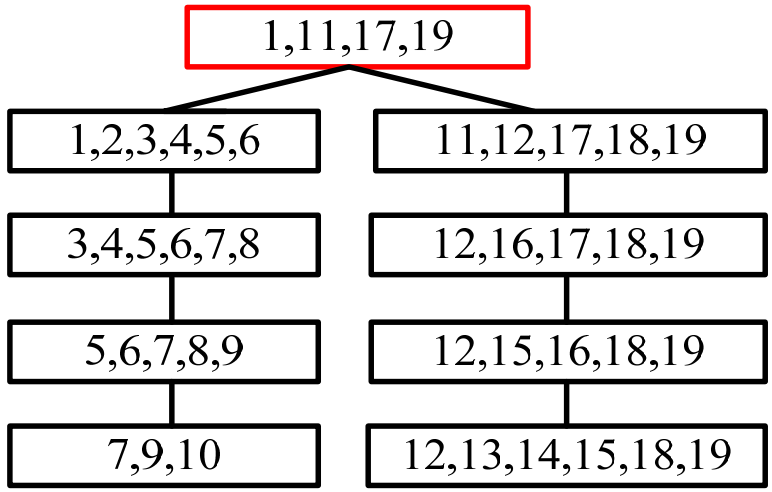


Tree decomp of G(H)

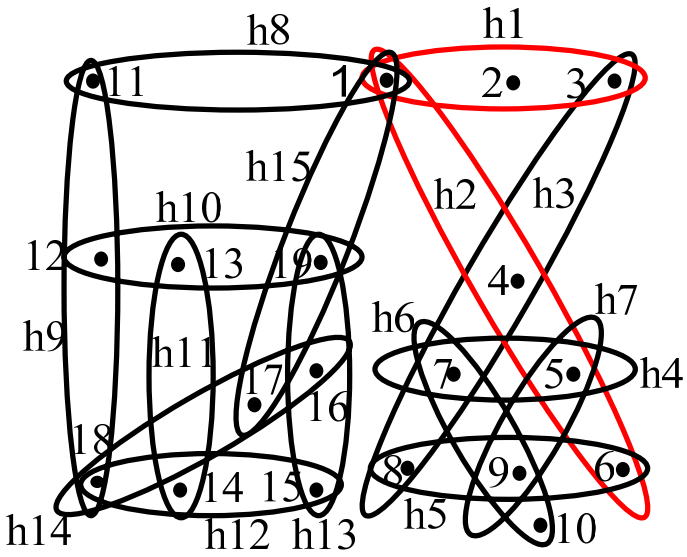


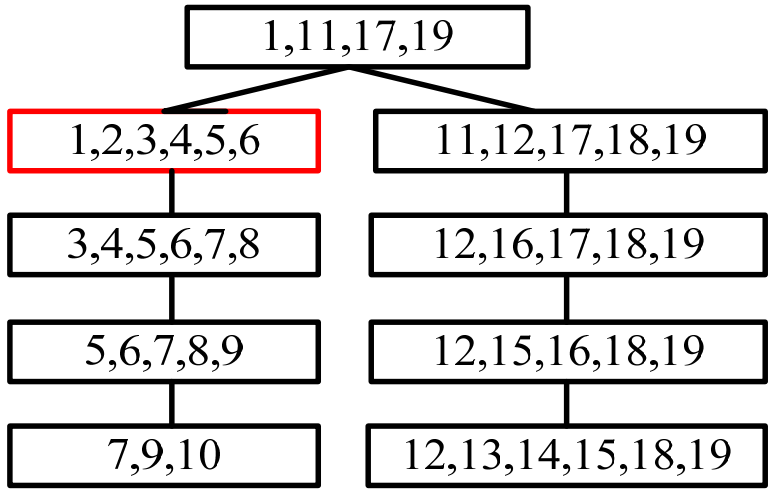
Hypergraph and the tree decomposition



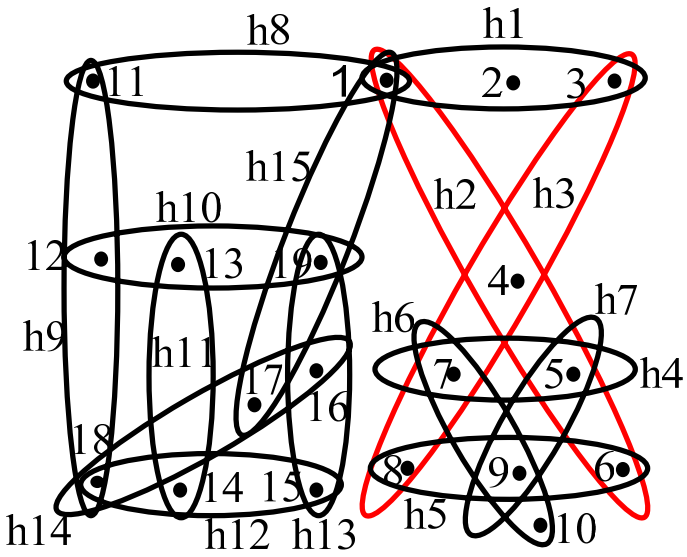


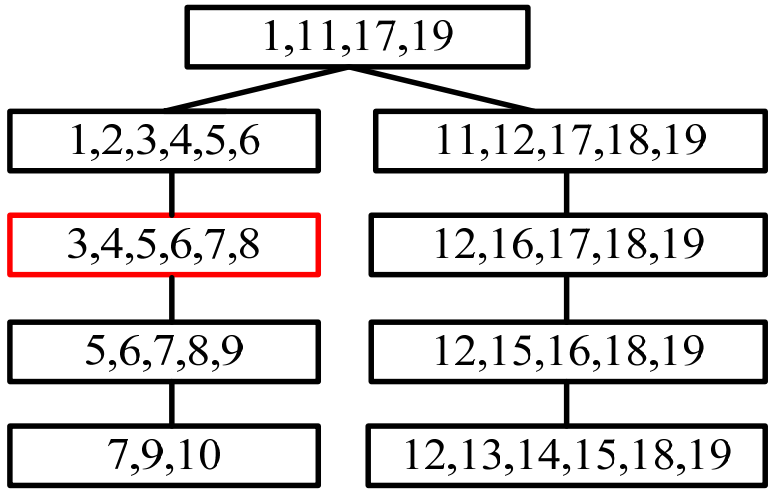
Hypergraph and the tree decomposition



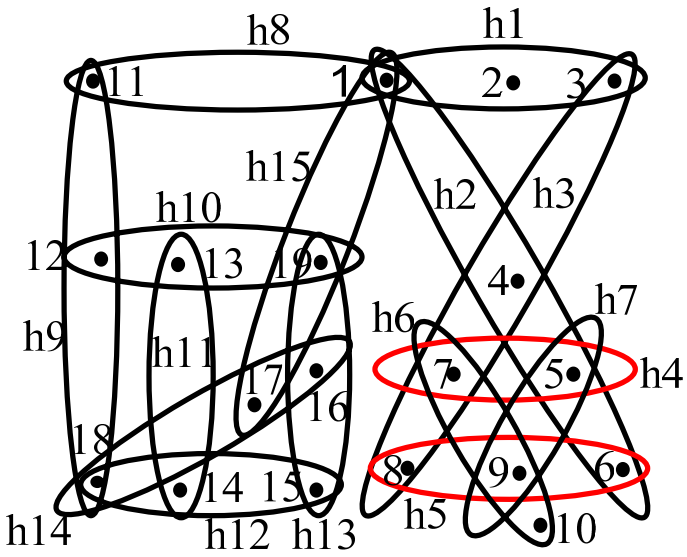


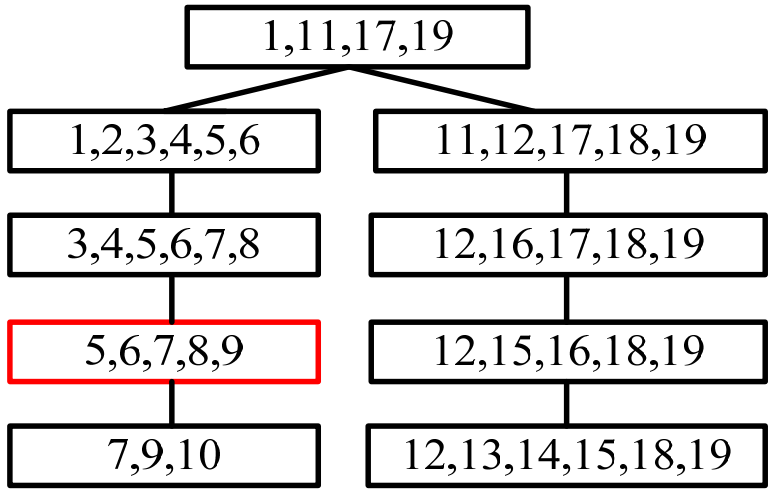
Hypergraph and the tree decomposition



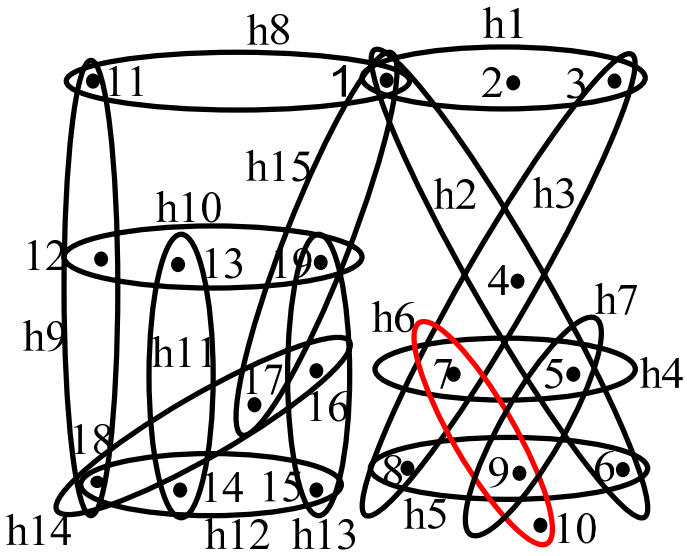


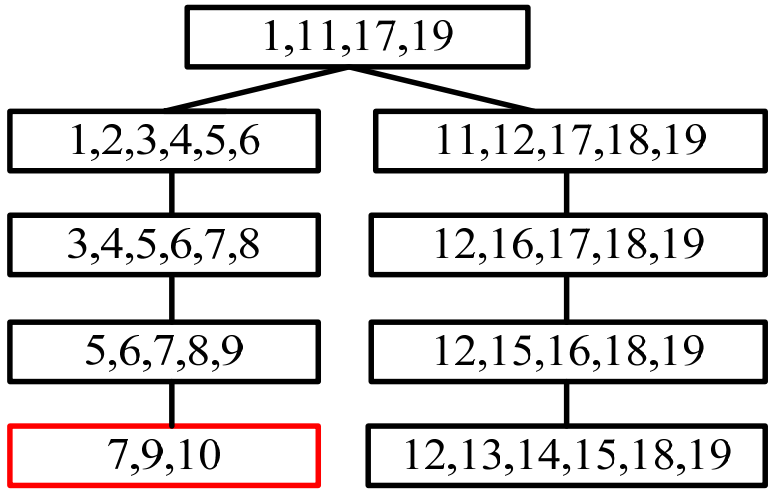
Hypergraph and the tree decomposition



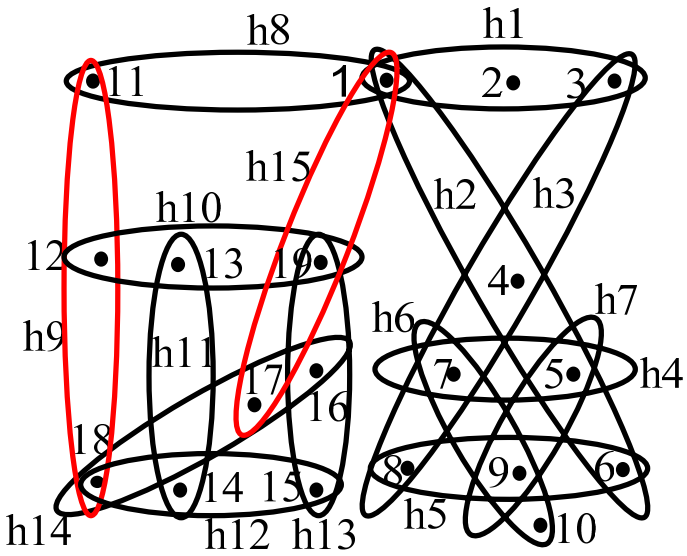


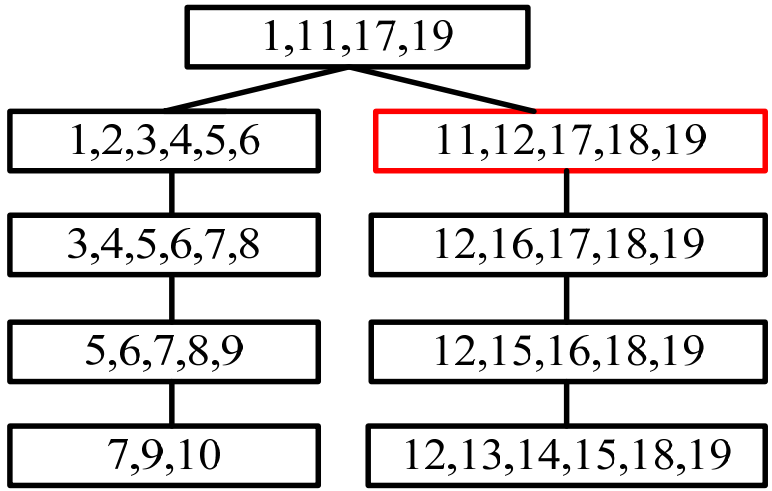
Hypergraph and the tree decomposition



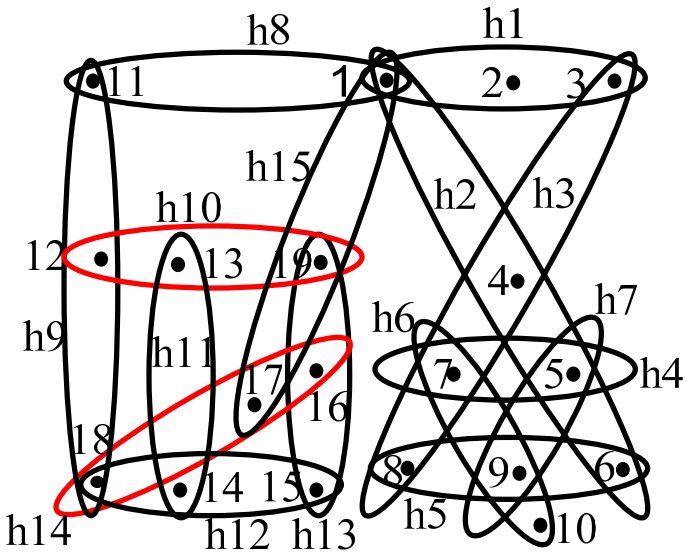


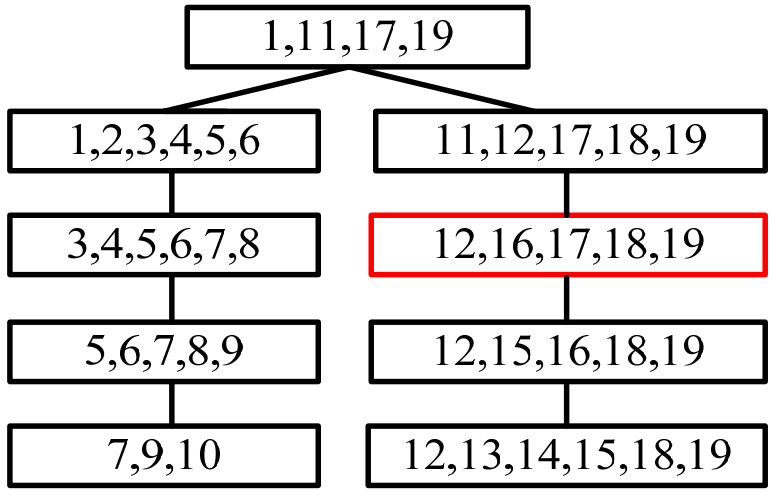
Hypergraph and the tree decomposition



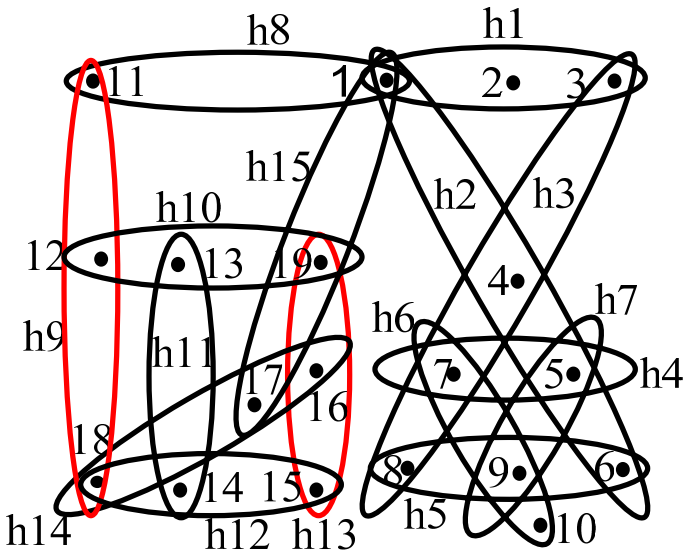


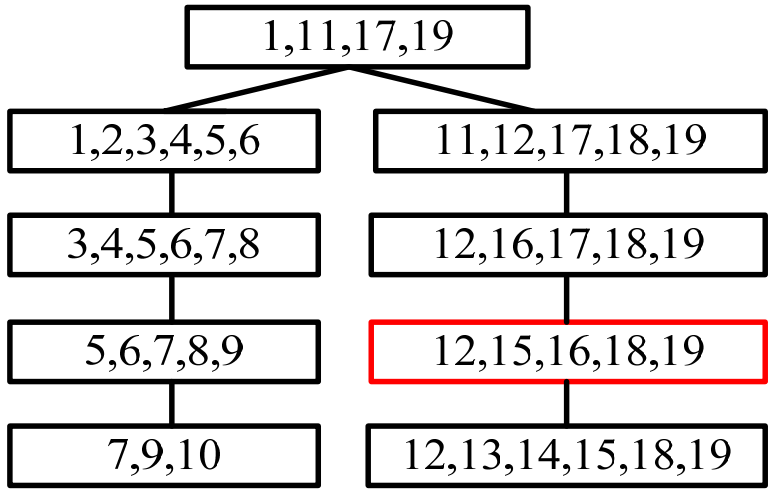
Hypergraph and the tree decomposition



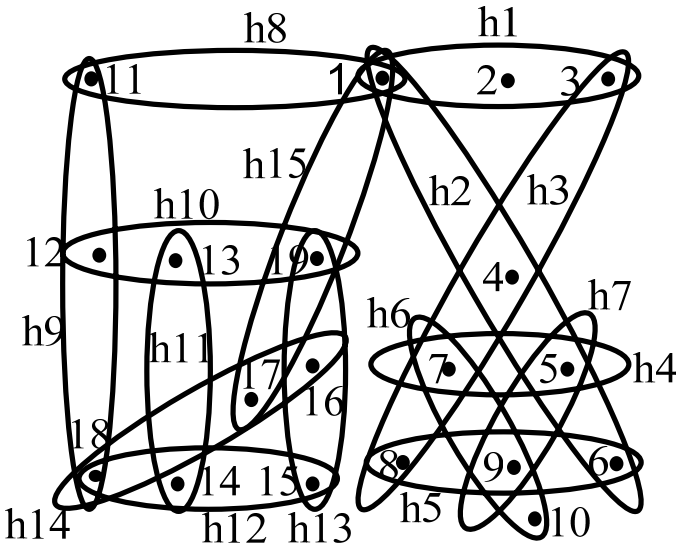


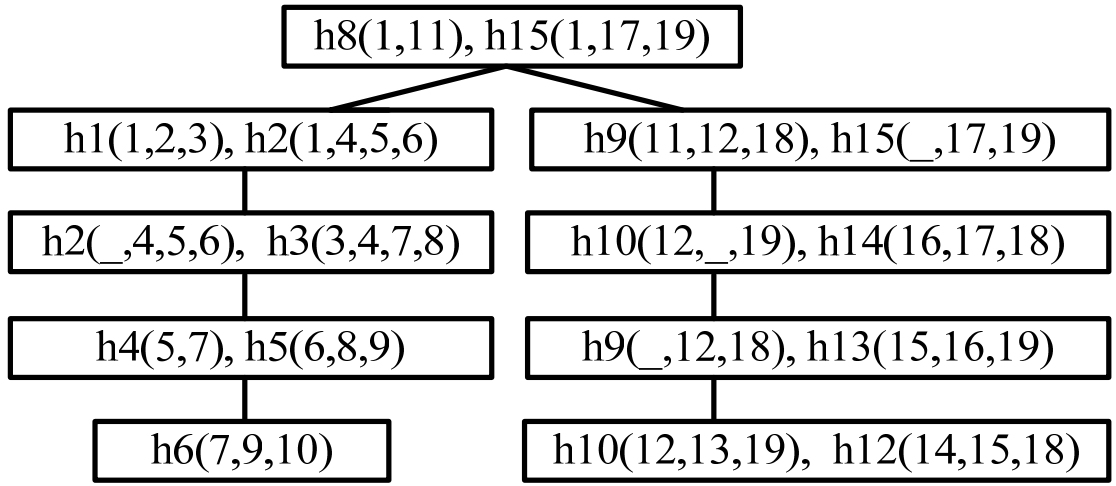
Hypergraph and the tree decomposition





Generalized hypertree decomposition





Generalized hypetree decomposition of width 2

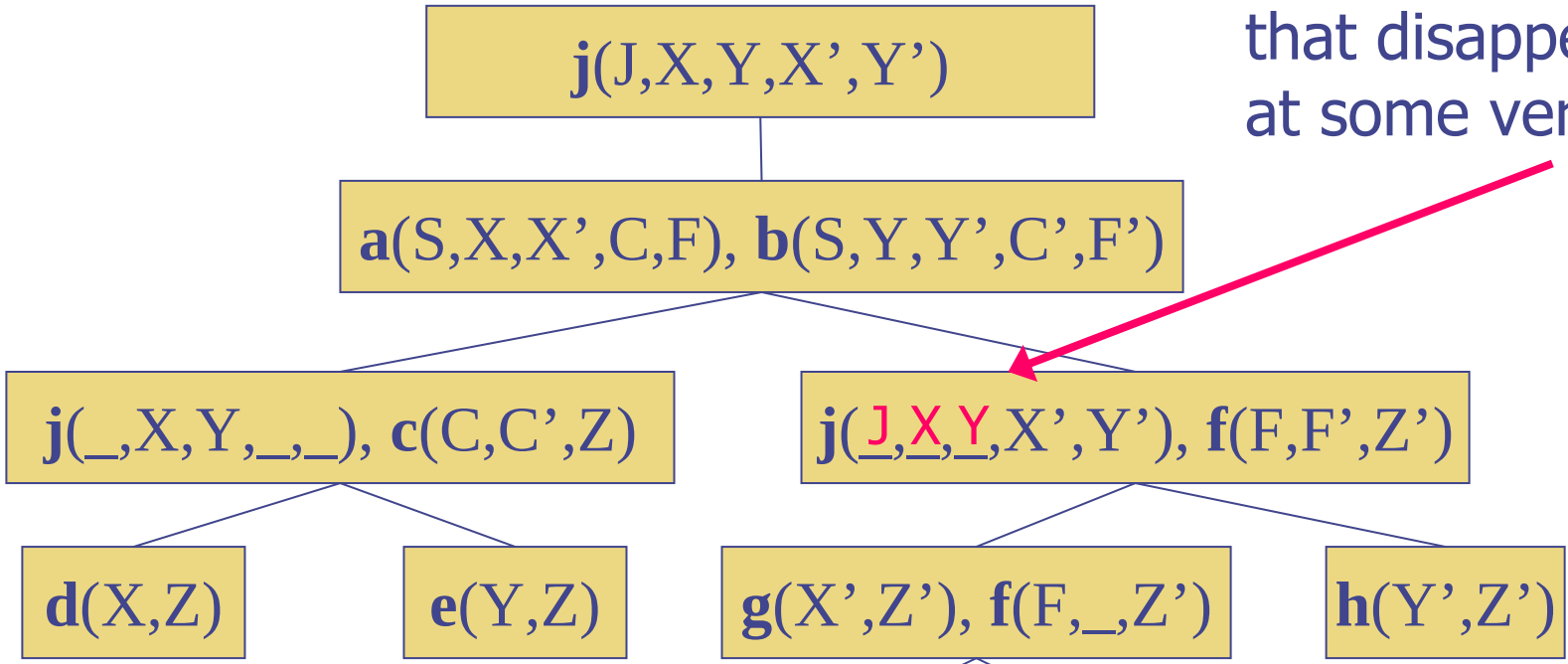
QUESTION:

Can we determine in polynomial time
Whether ghw(H) < k for constant k

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

[Schwentick et. al. 06]

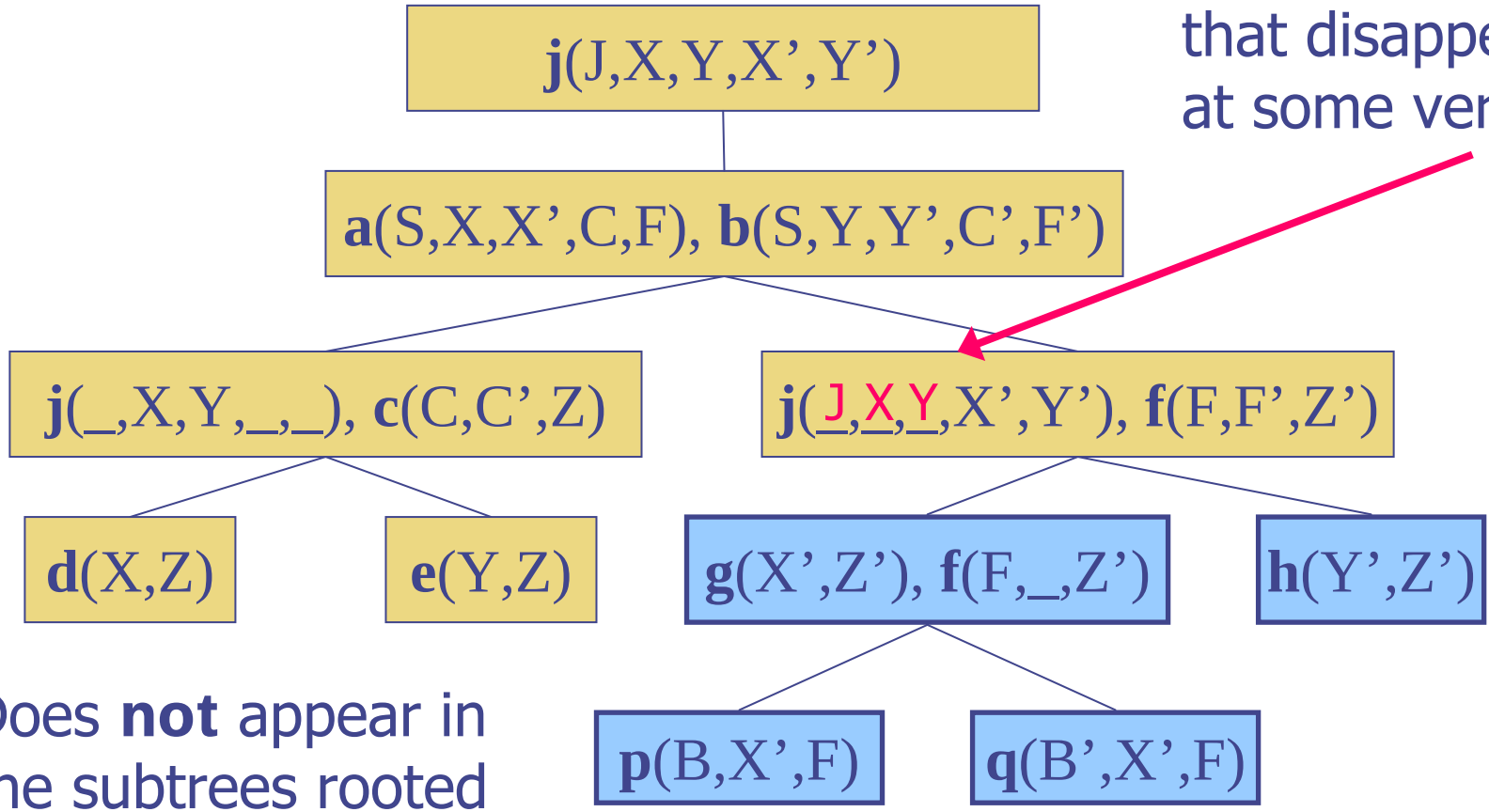
Hypertree Decomposition
= Generalized HTD +Special Condition



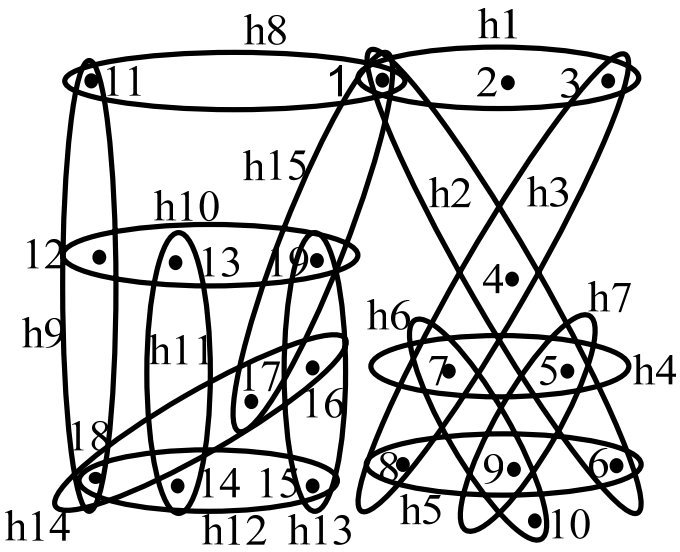
Each variable that disappeared at some vertex *v*

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

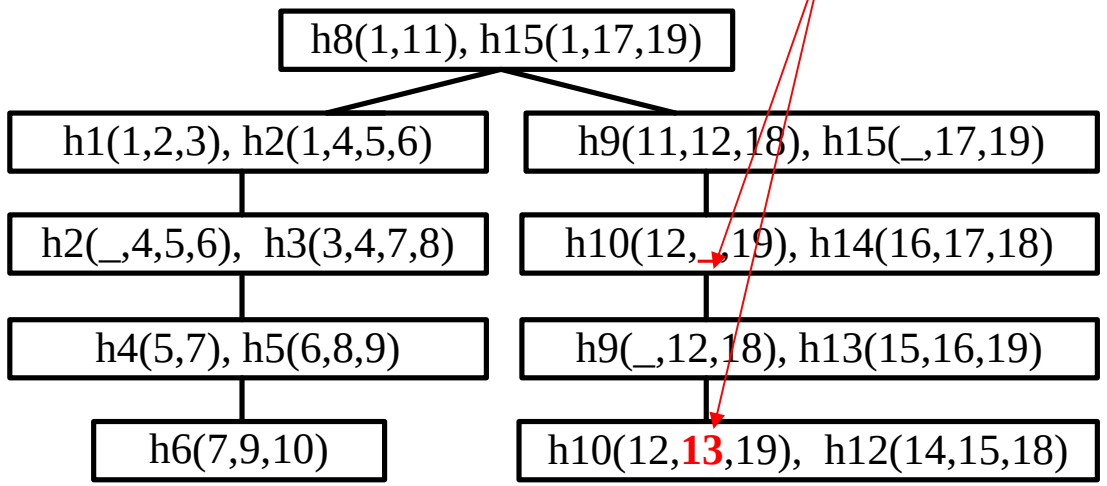
Special Condition



Generalized hypertree decomposition

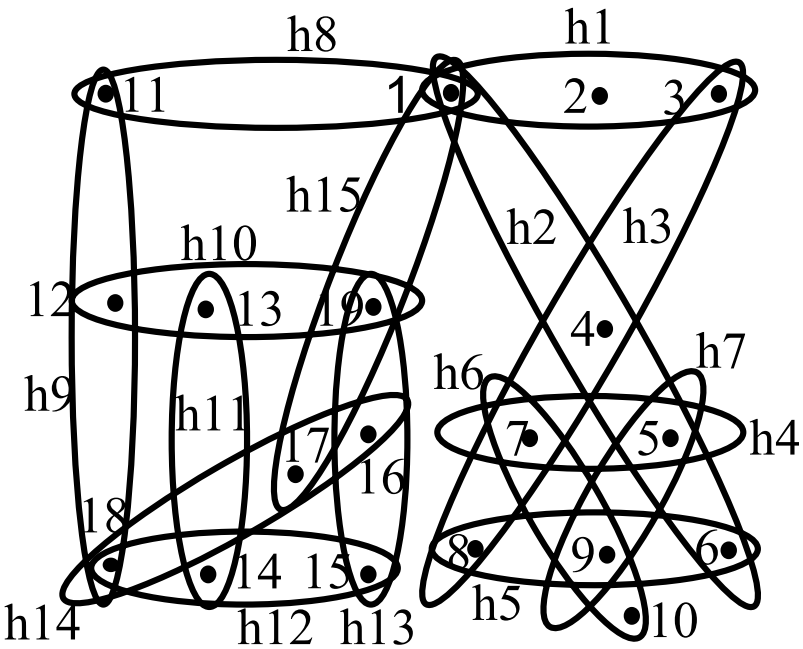


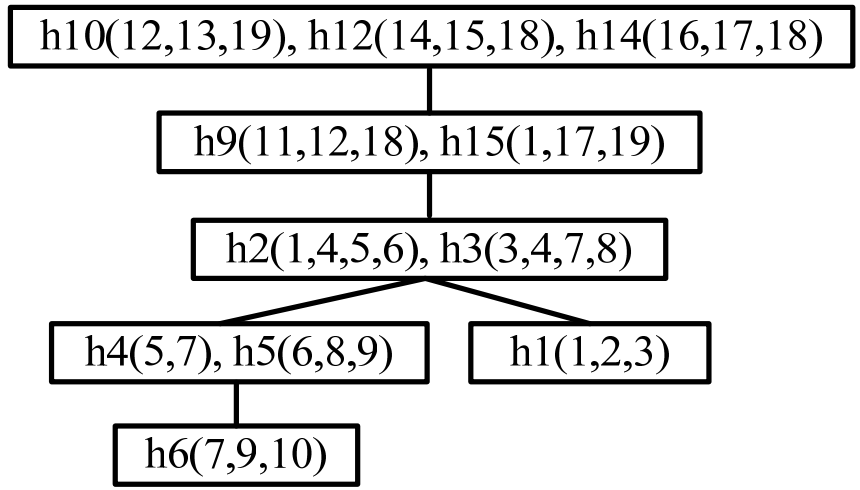
Special condition violated



Generalized hypetree decomposition of width 2

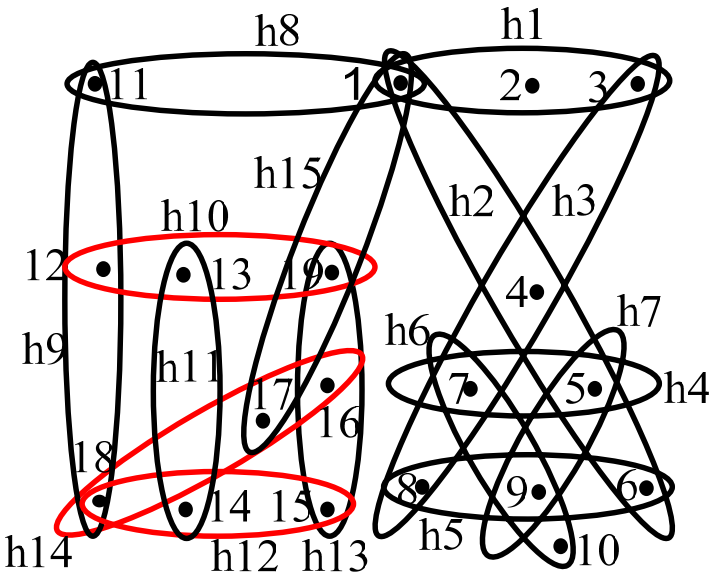
Hypertree decomposition

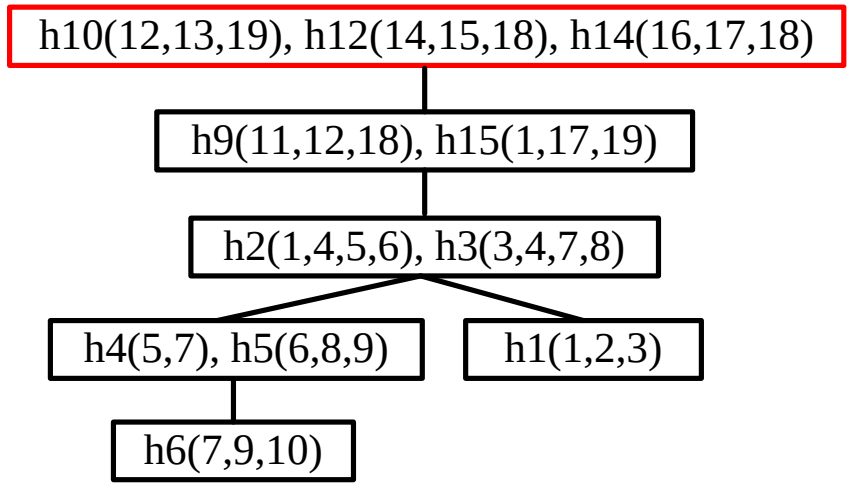




Hypertee decomposition of width 3

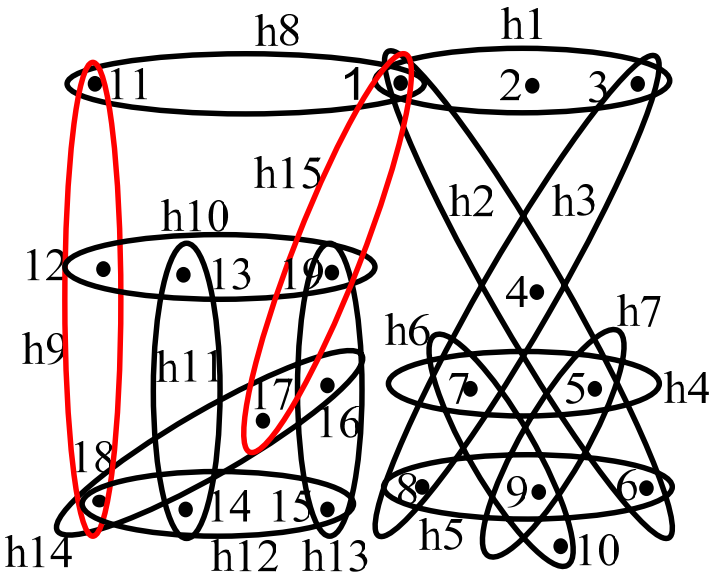
Hypertree decomposition

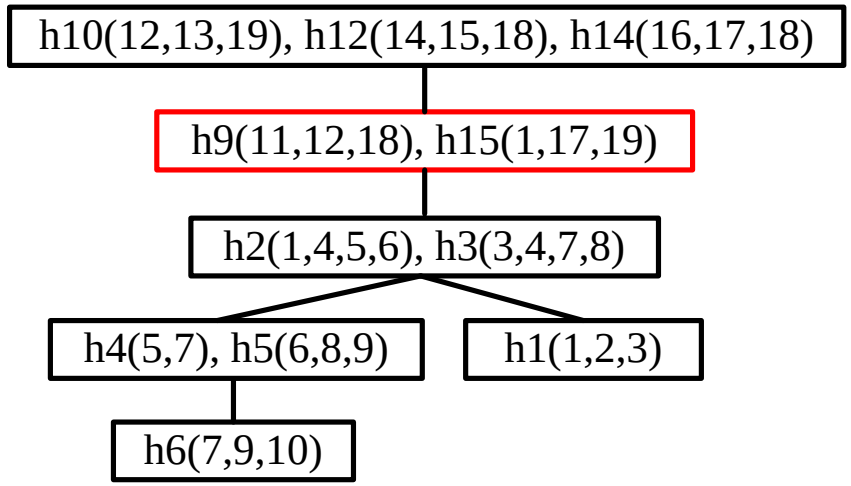




Hypertee decomposition of width 3

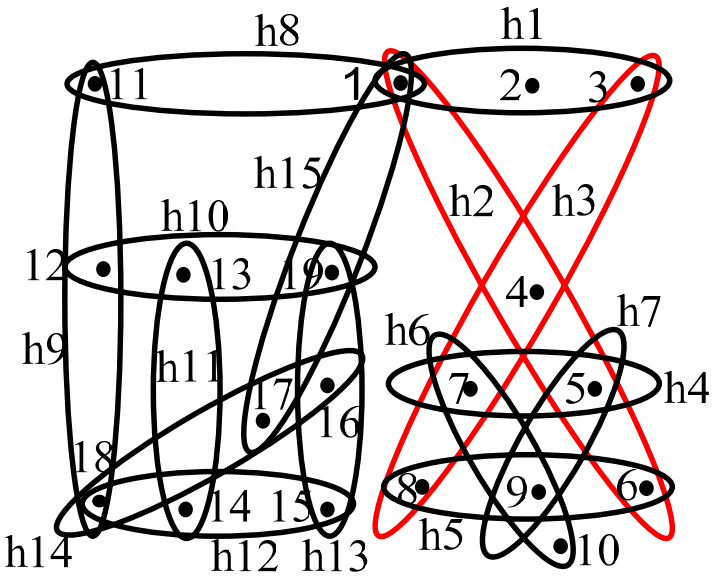
Hypertree decomposition

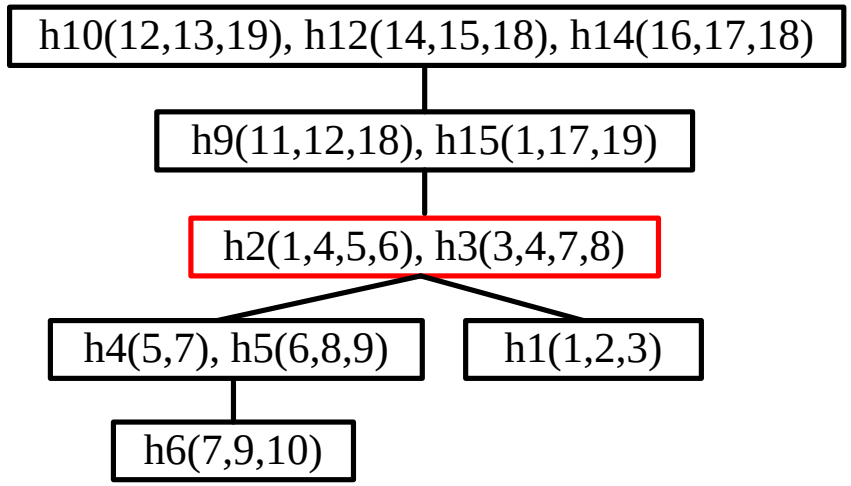




Hypertee decomposition of width 3

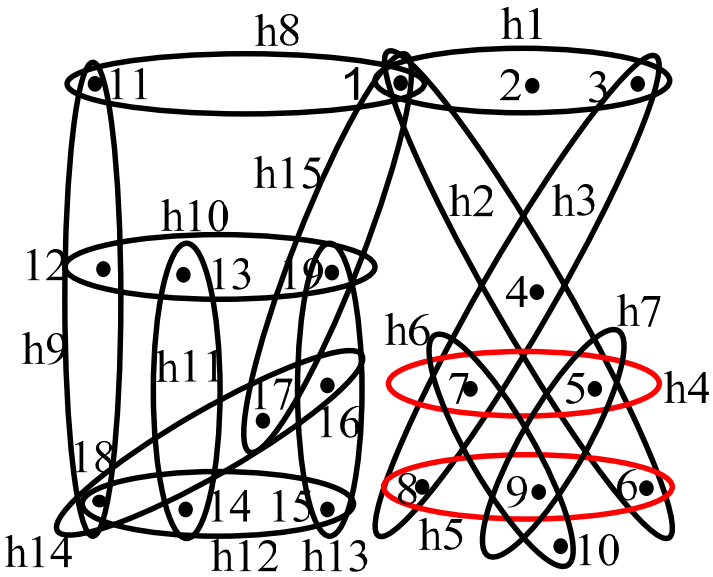
Hypertree decomposition

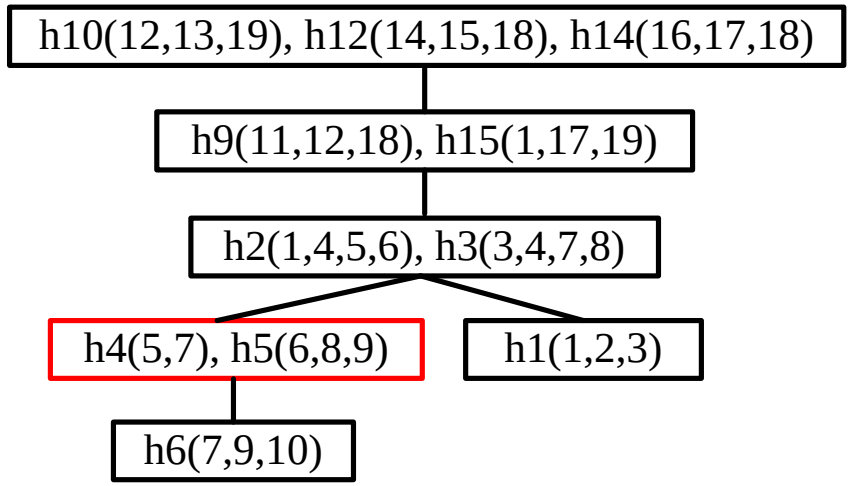




Hypertee decomposition of width 3

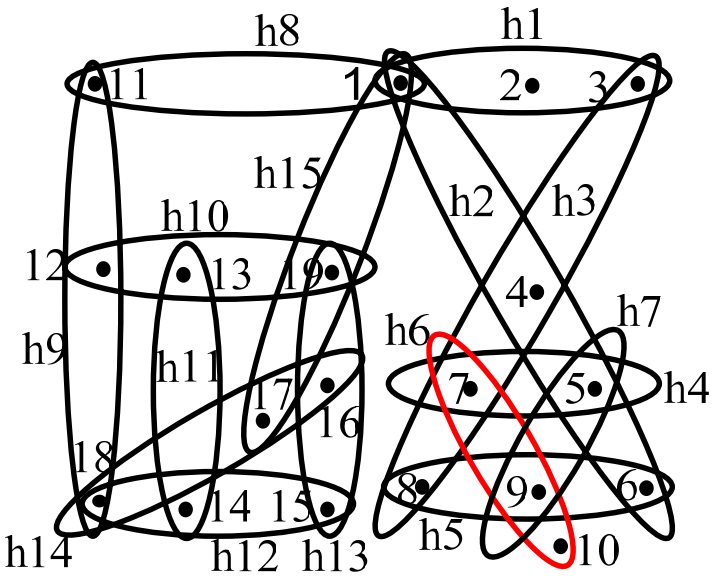
Hypertree decomposition

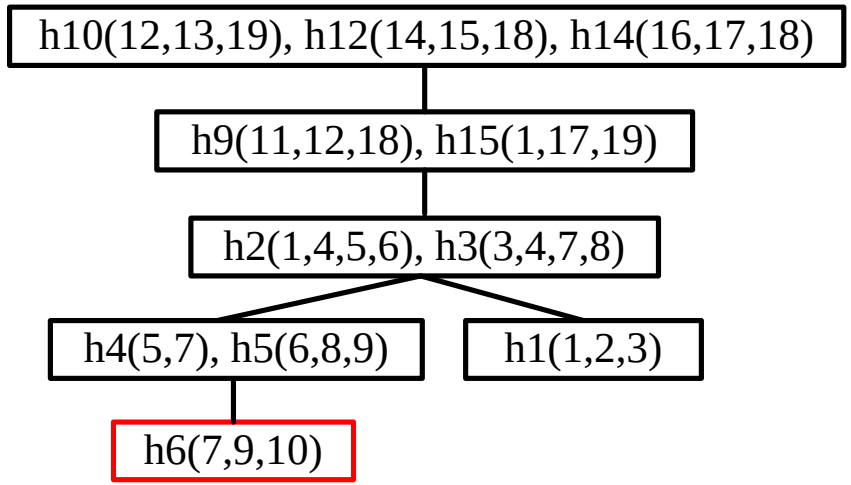




Hypertee decomposition of width 3

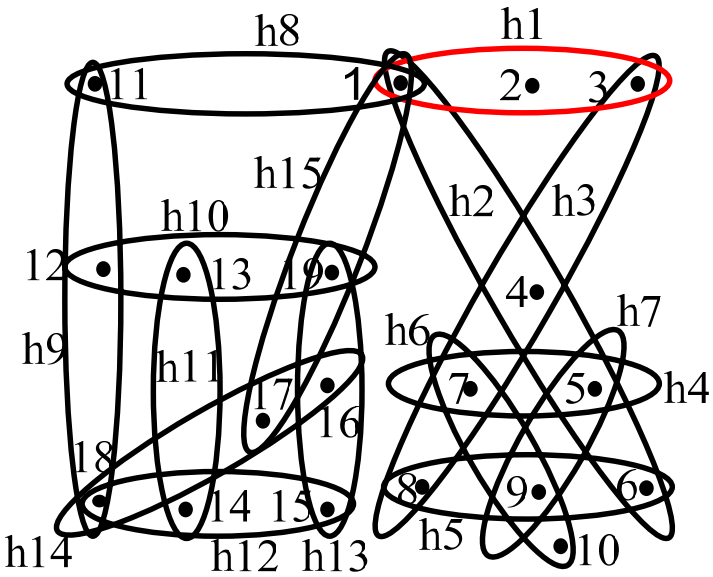
Hypertree decomposition

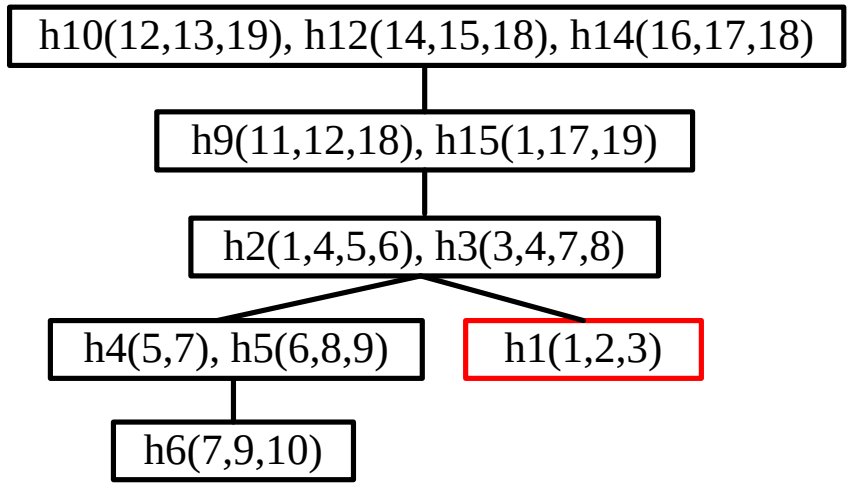




Hypertee decomposition of width 3

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

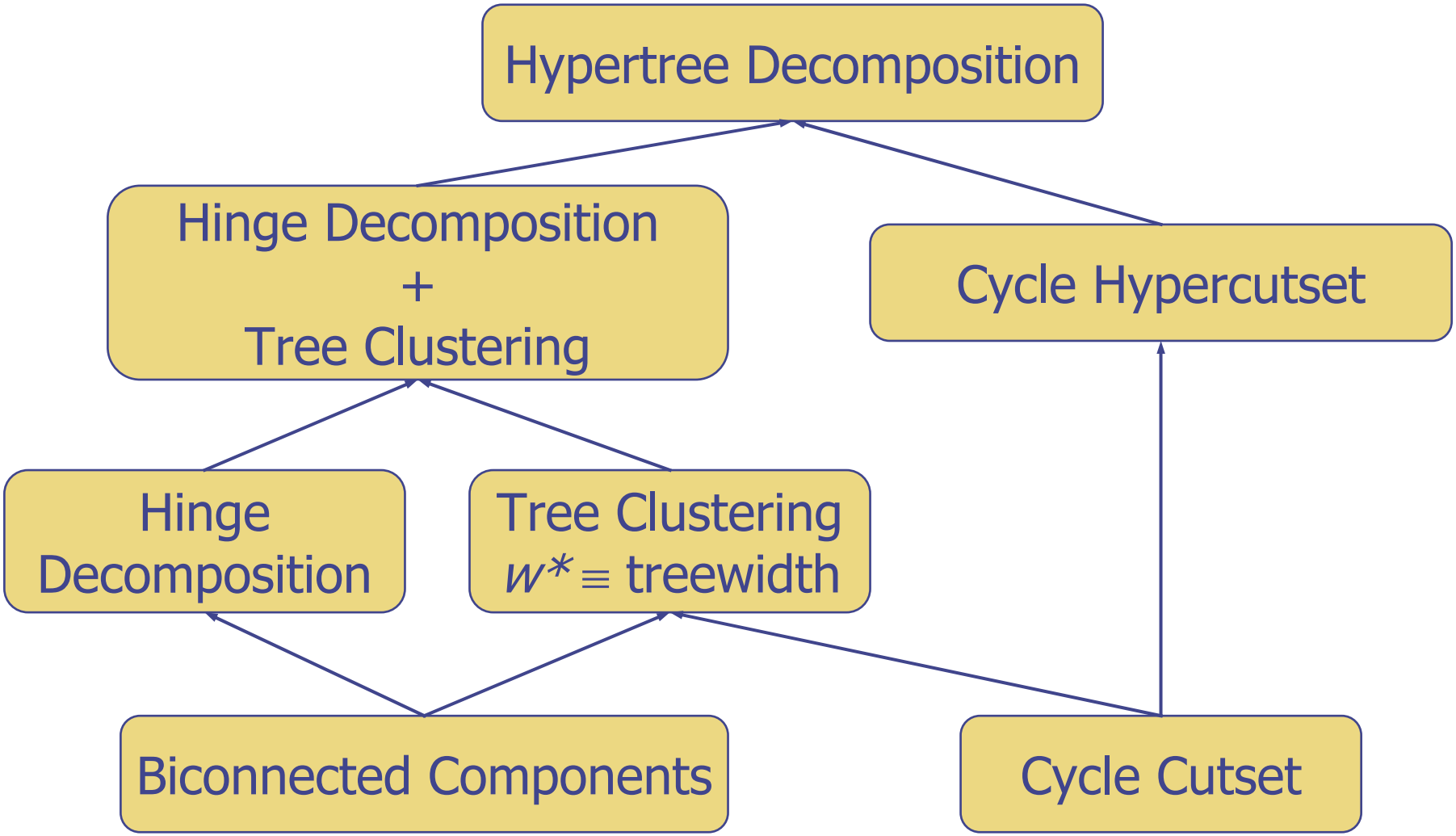
Observation: If H has n vertices, then

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

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*)

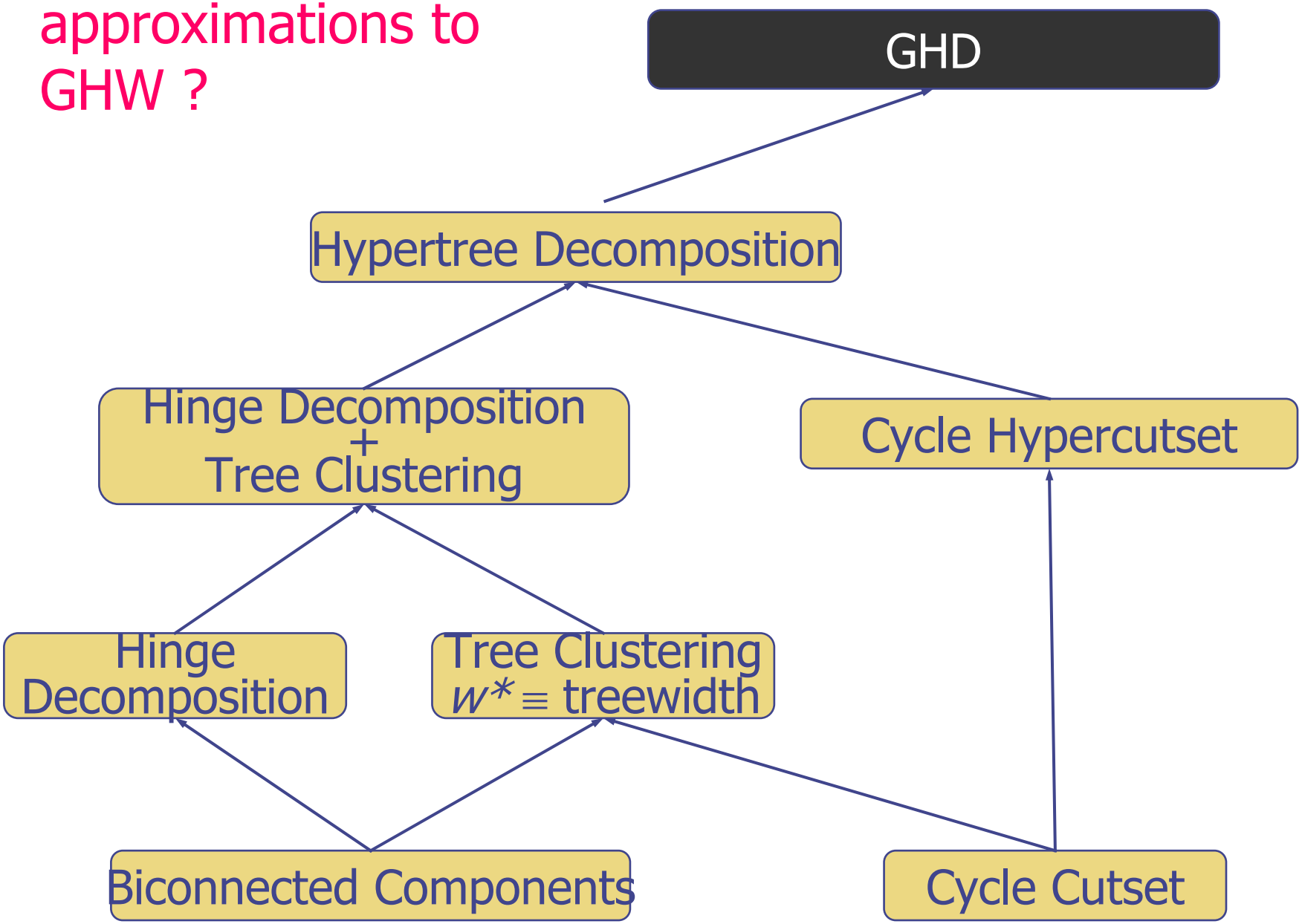
where $H^* = H \cup \{E' \mid \exists E \text{ in edges}(H): E' \subseteq E\}$

Exponential!

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

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

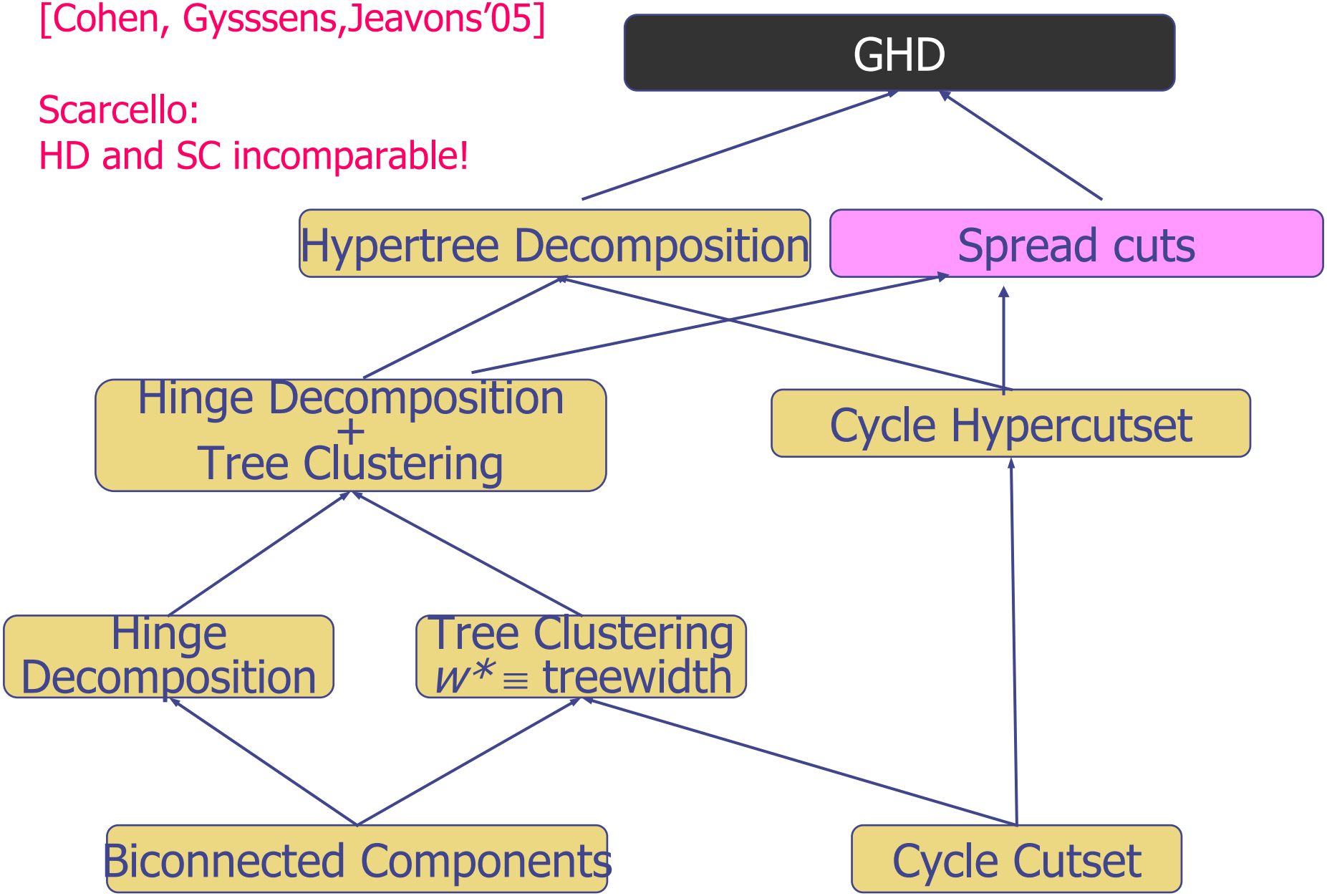
Q: Are there other approximations to GHW ?



YES: E.g. Spread Cuts

[Cohen, Gysssens,Jeavons'05]

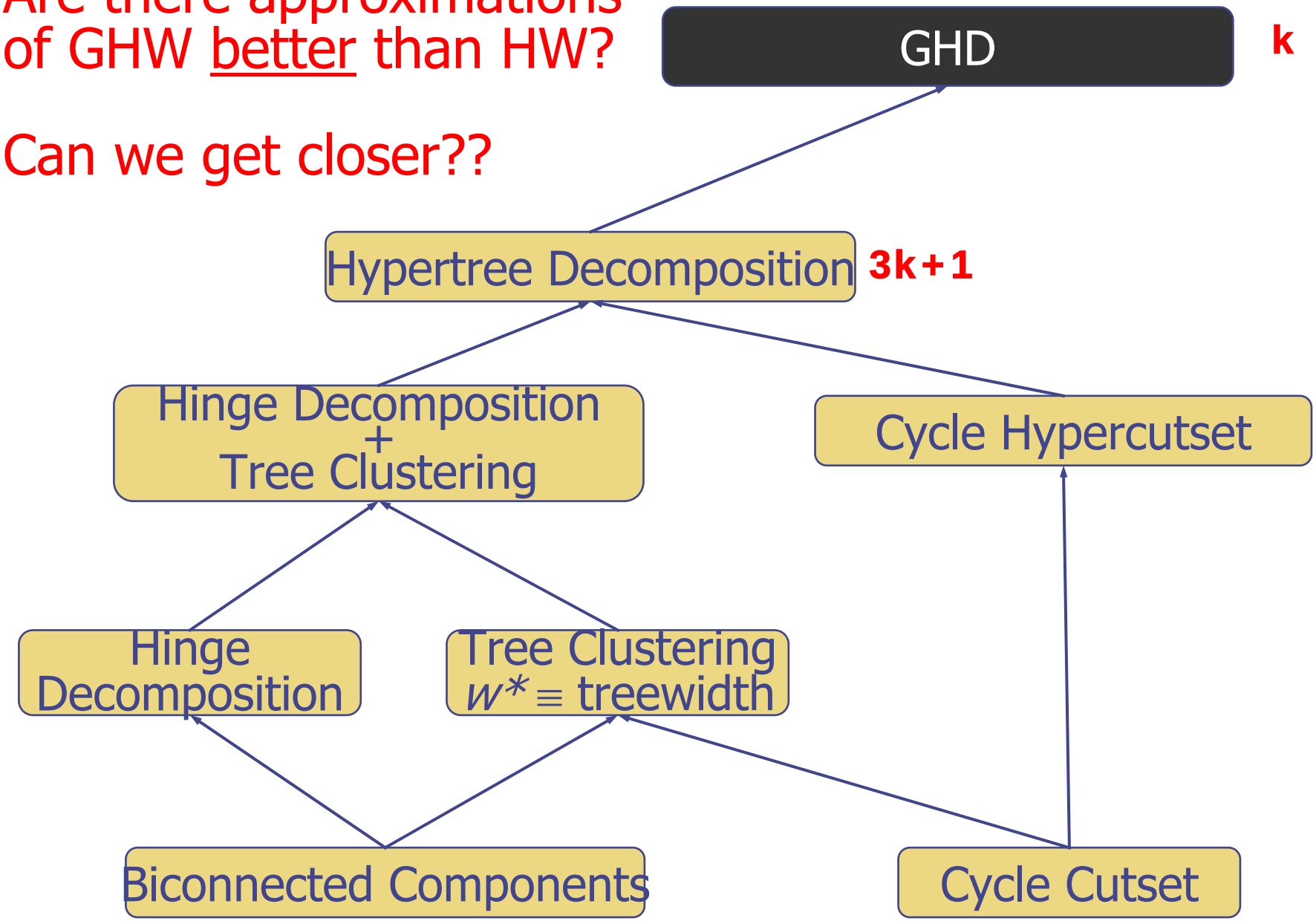
Scarcello:
HD and SC incomparable!



Open Question:

Are there approximations of GHW better than HW?

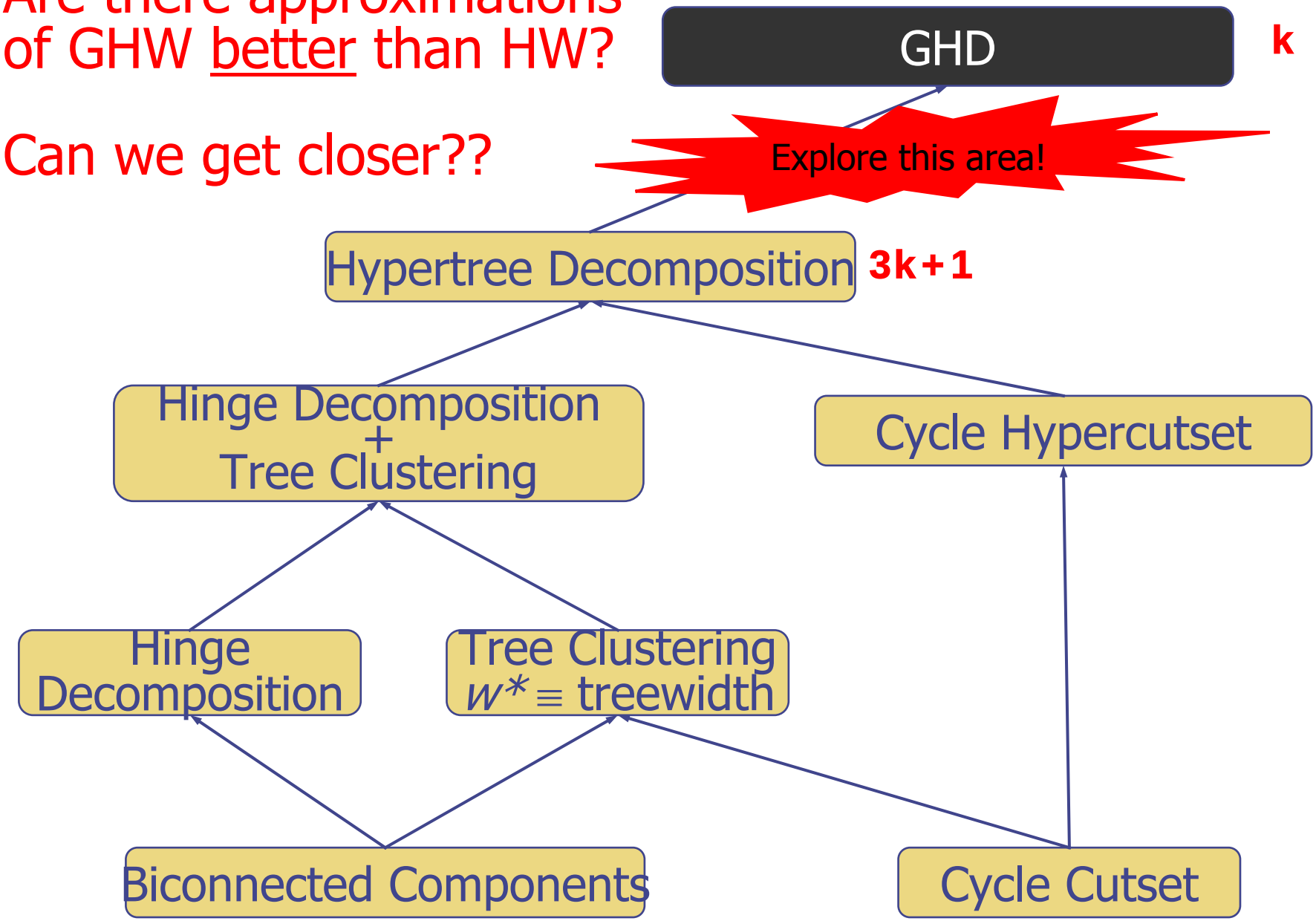
Can we get closer??



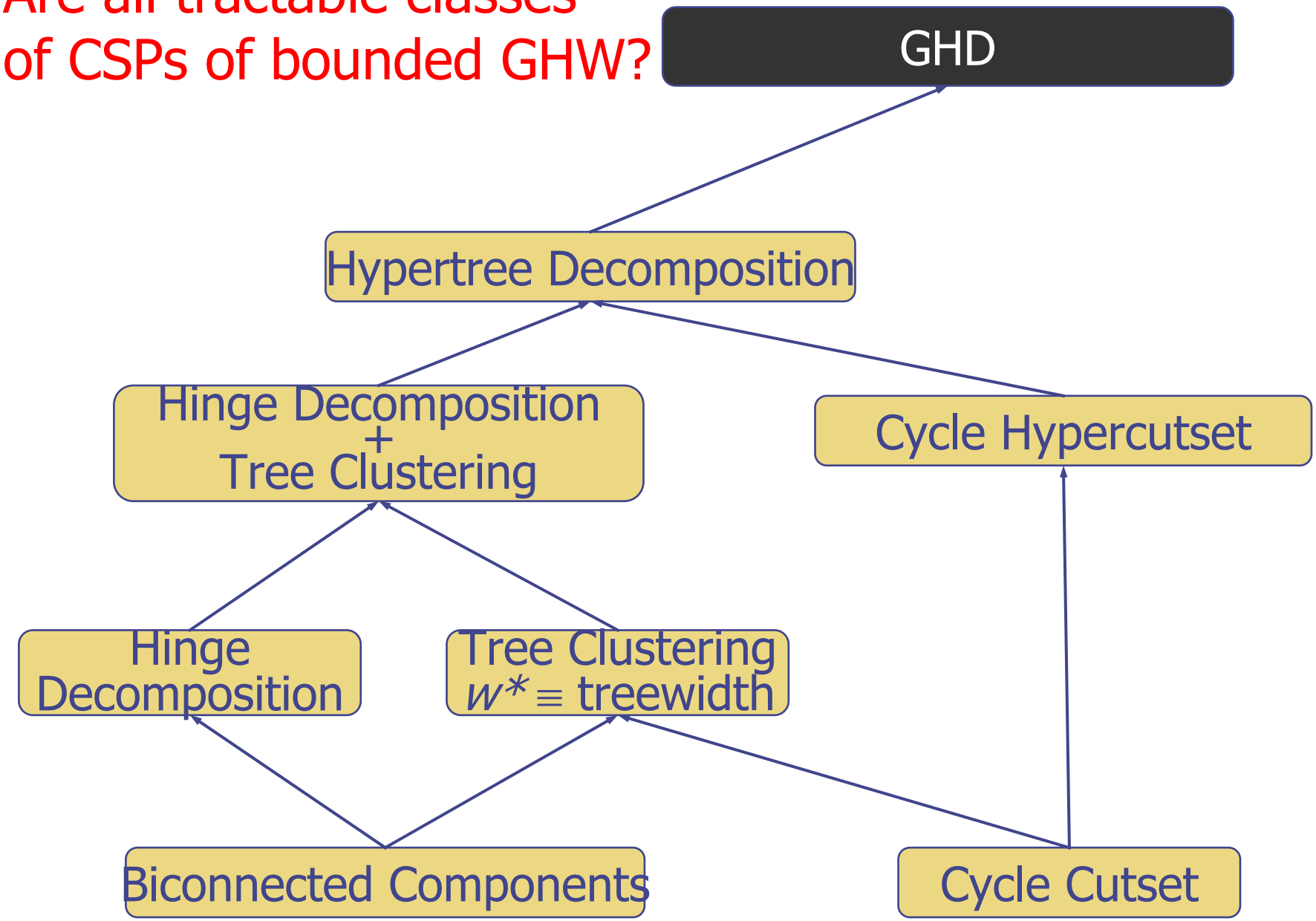
Open Question:

Are there approximations of GHW better than HW?

Can we get closer??



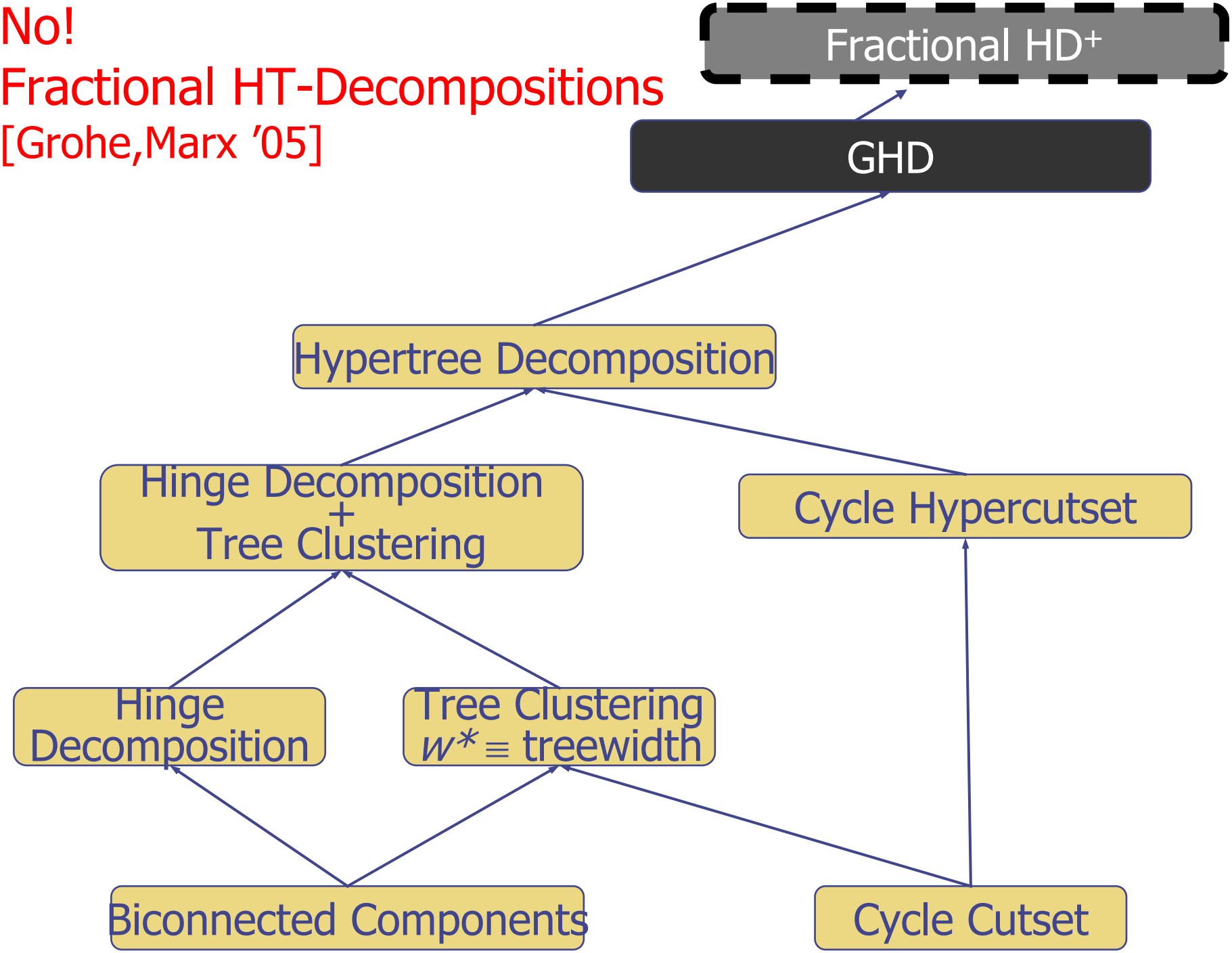
Question:
Are all tractable classes
of CSPs of bounded GHW?



No!

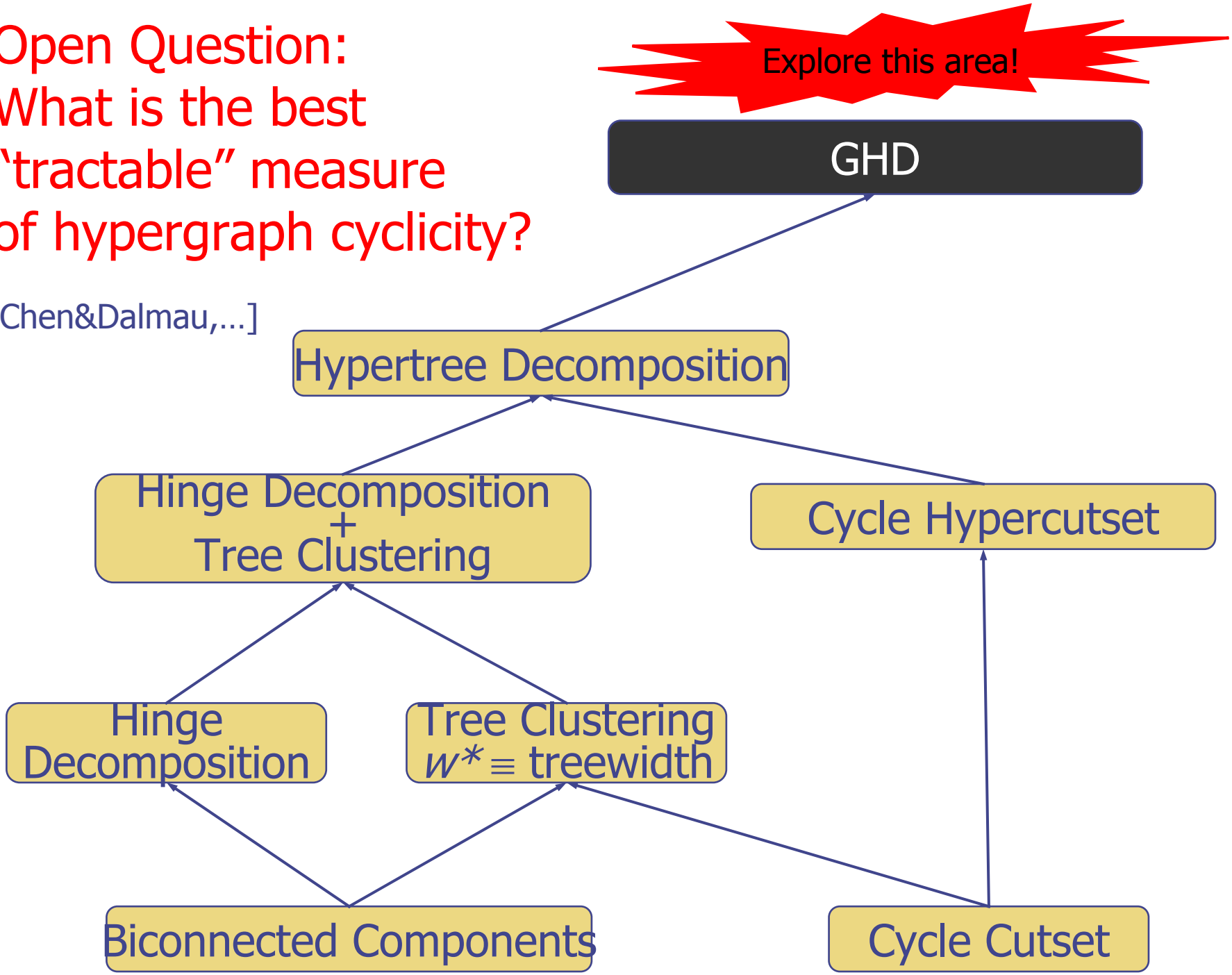
Fractional HT-Decompositions

[Grohe,Marx '05]



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

[Chen&Dalmau,...]



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: $HW_k = GF_k(\mathsf{L})$

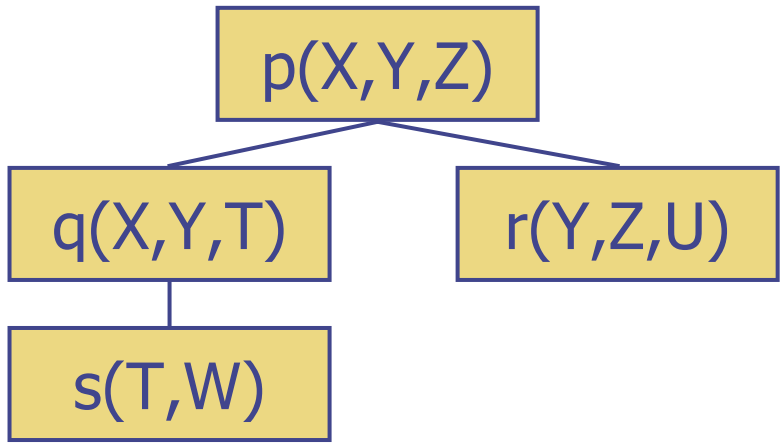
From this general result, we also get a nice logical characterization of acyclic queries:

Corollary: $HW_1 = ACYCLIC = GF(\mathsf{L})$

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:

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

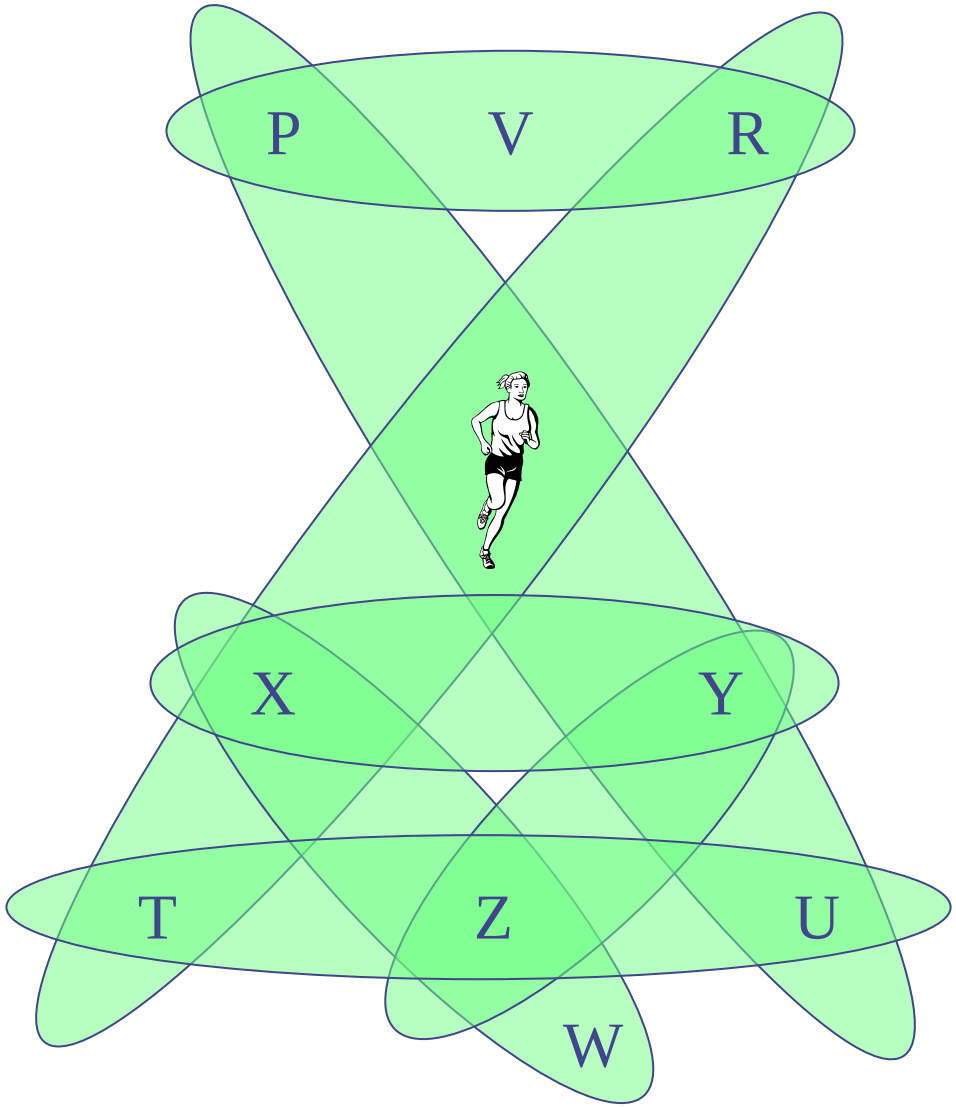
Guard

Guarded subformula

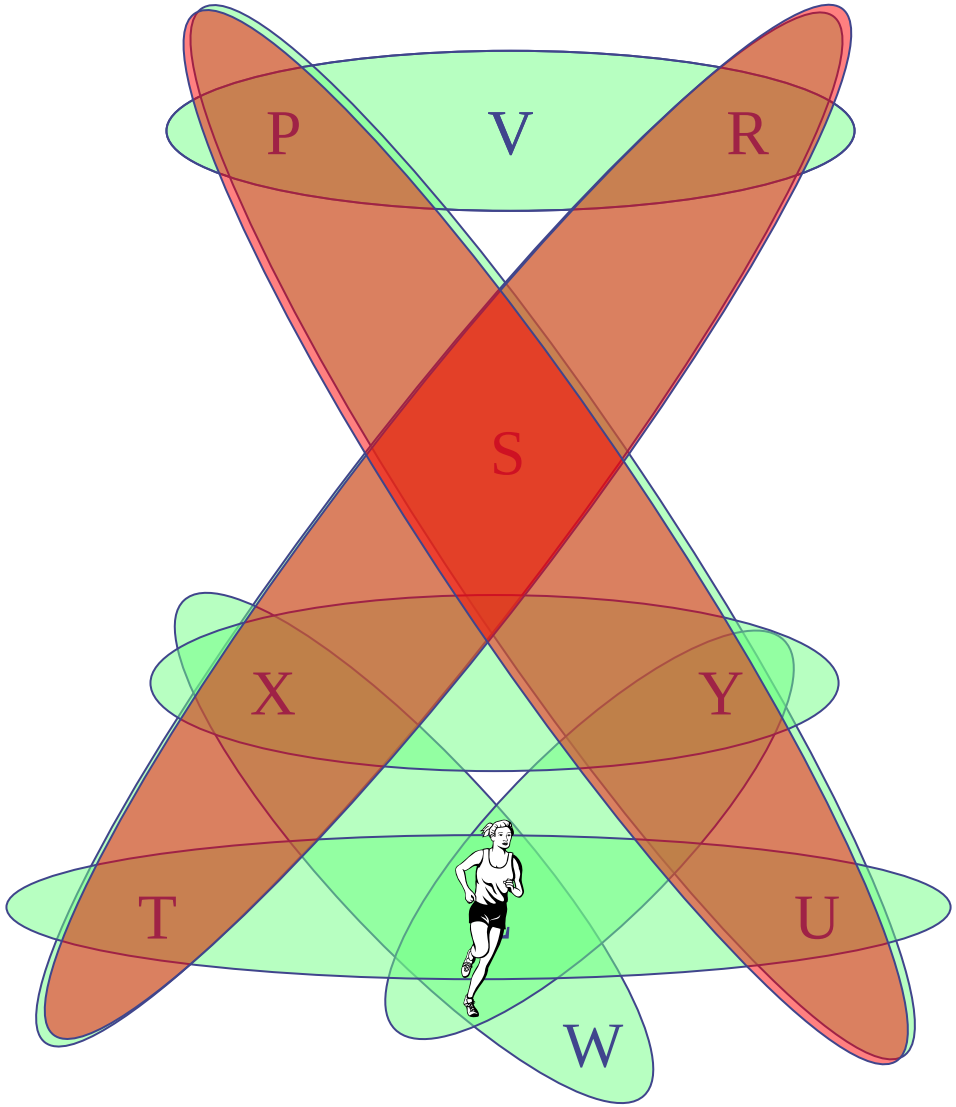
Game characterization: Robber and Marshals

- 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

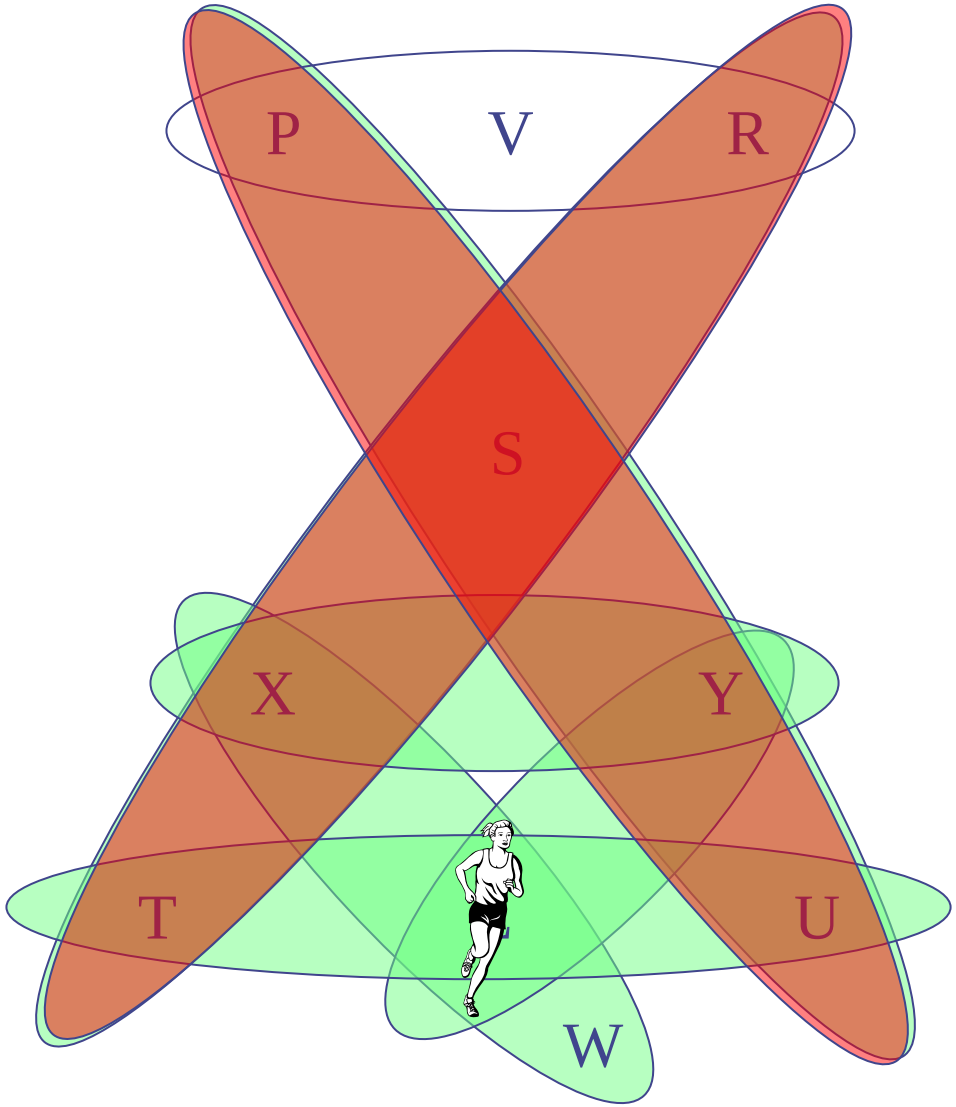
Step 0: the empty hypergraph



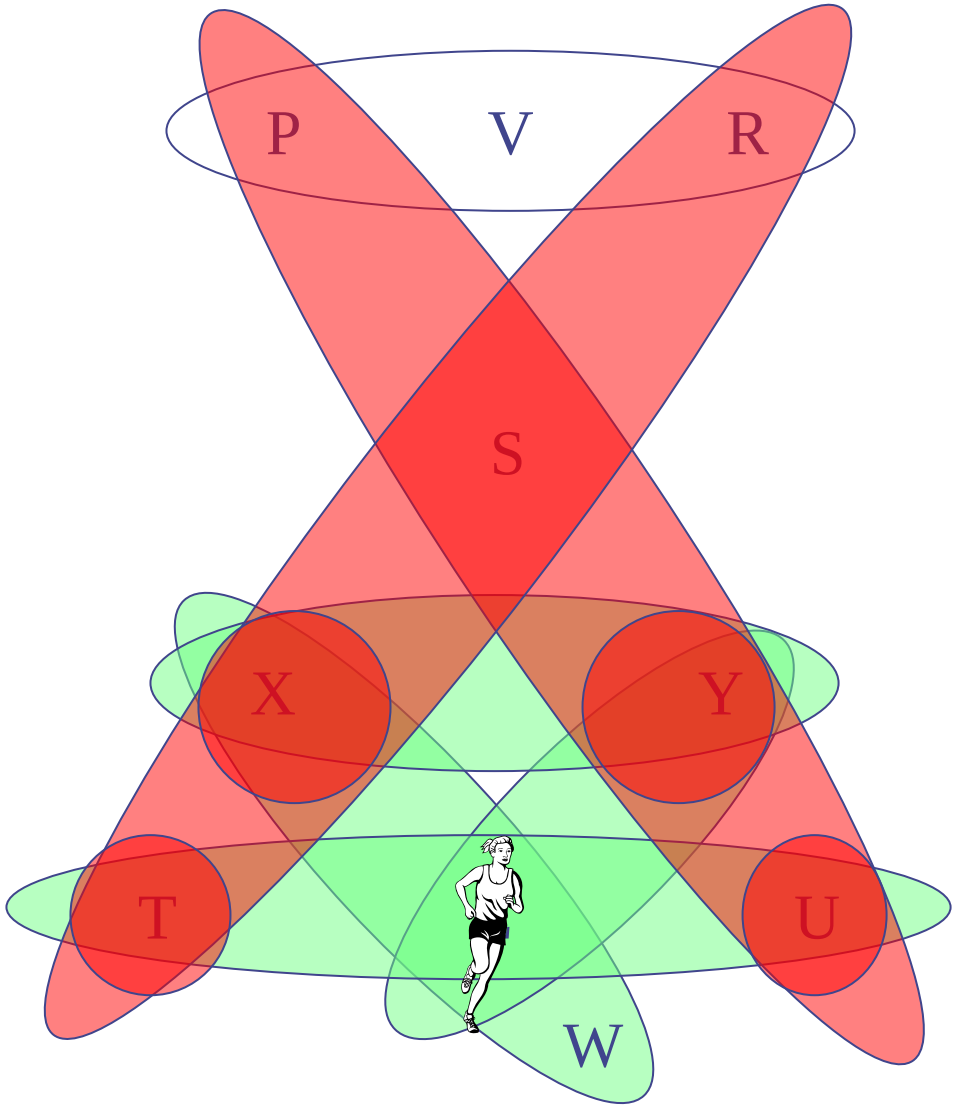
Step 1: first move of the marshals



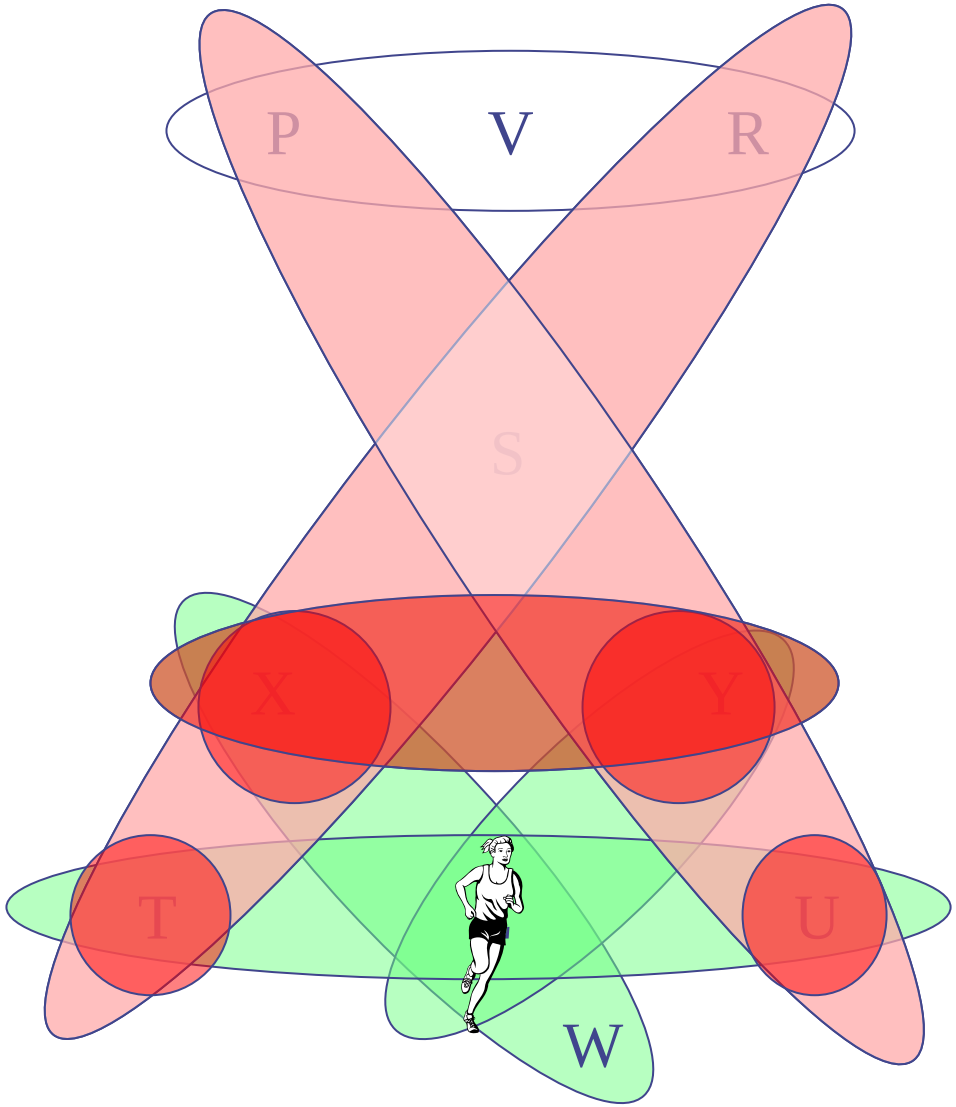
Step 1: first move of the marshals



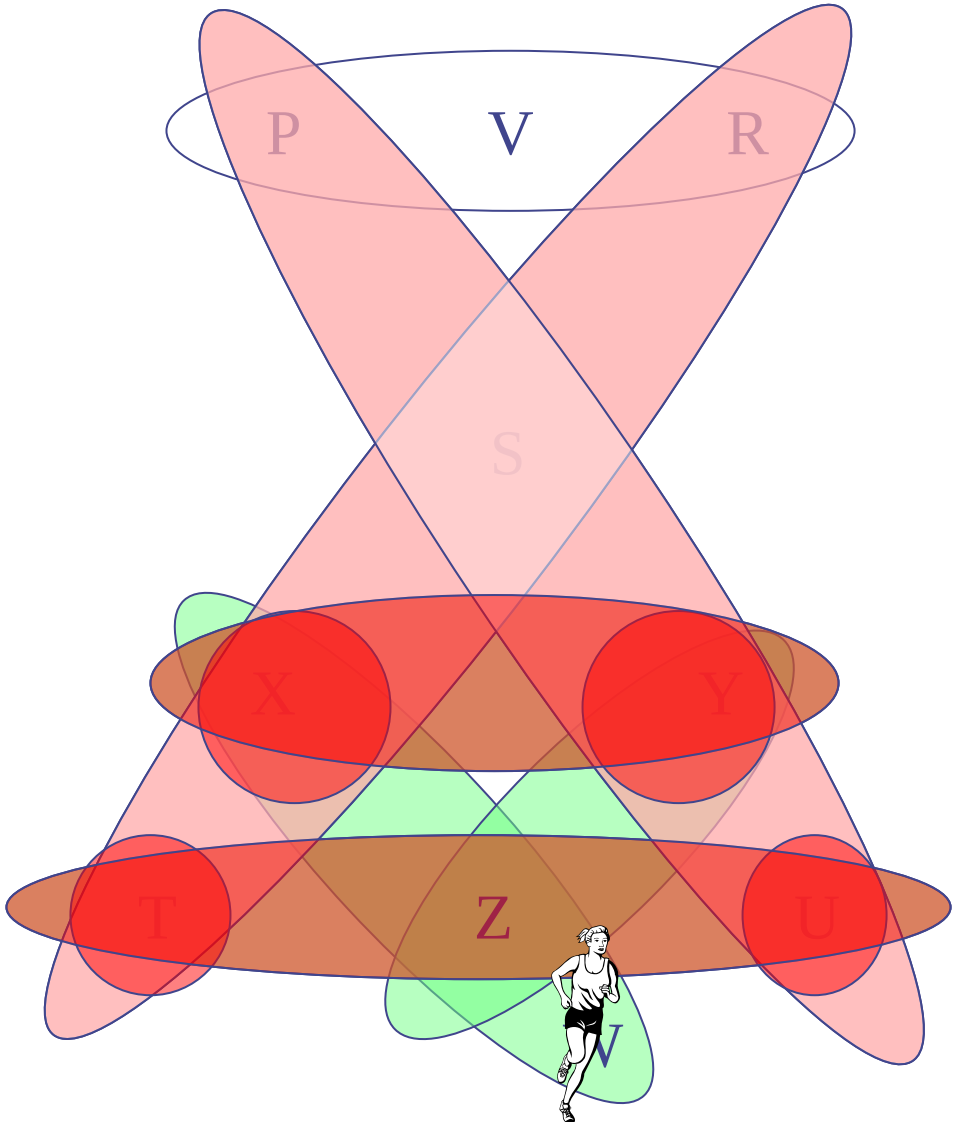
Step 2a: shrinking the space



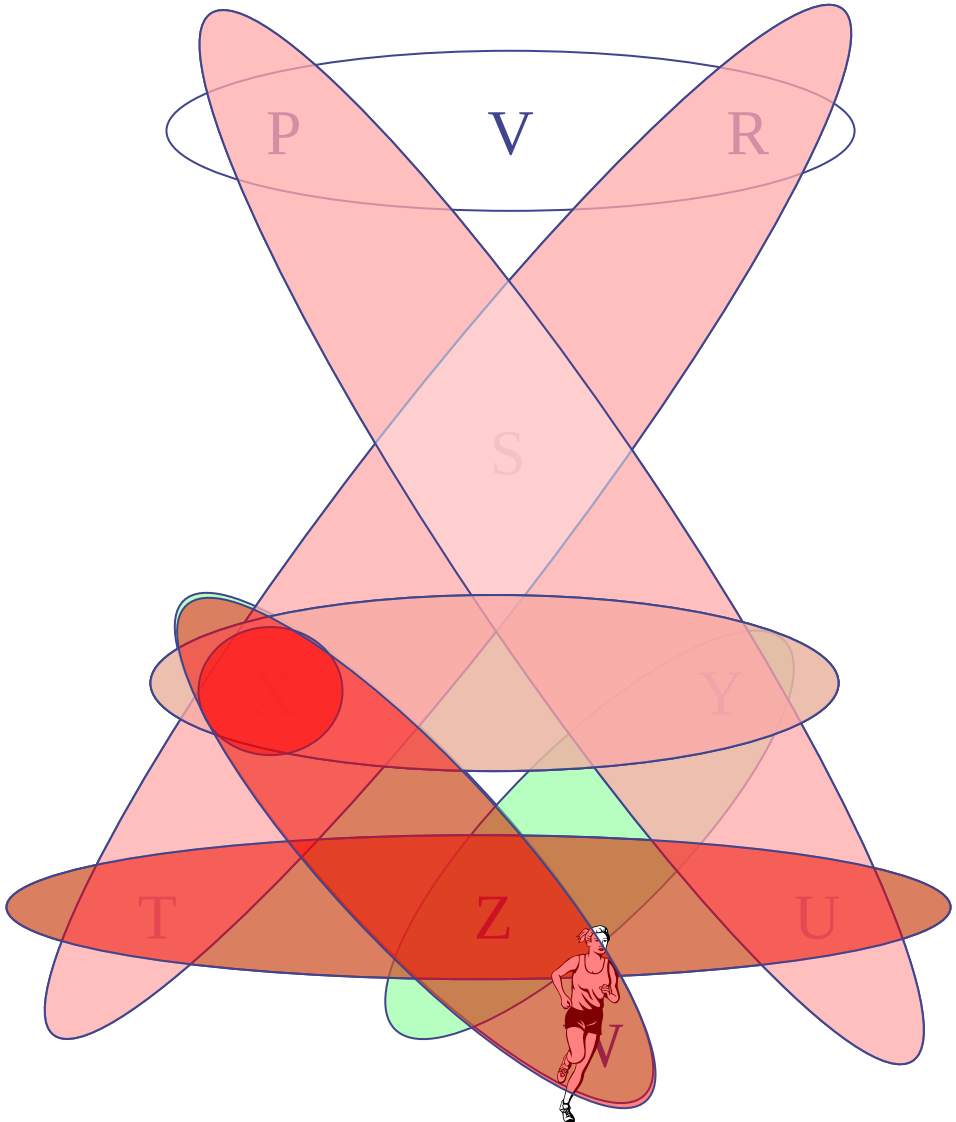
Step 2a: shrinking the space



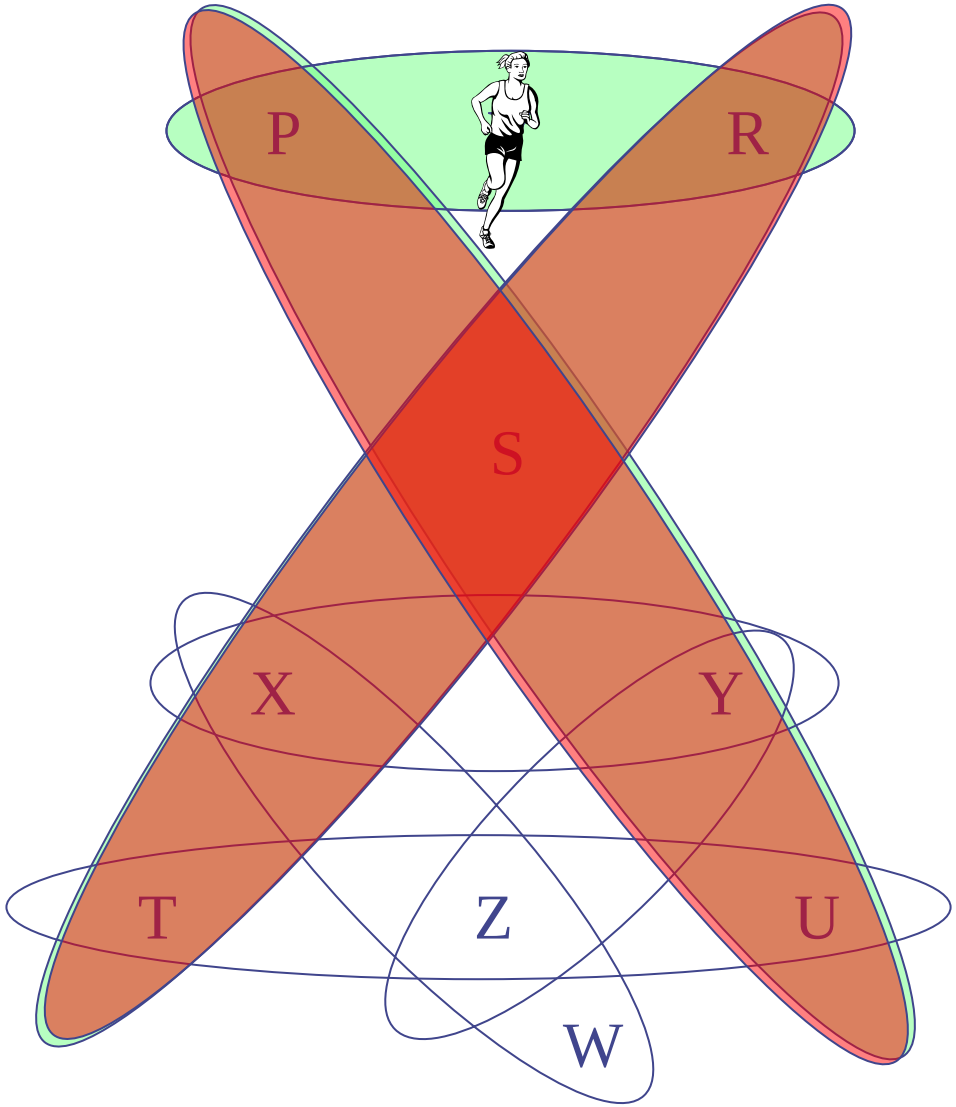
Step 2a: shrinking the space



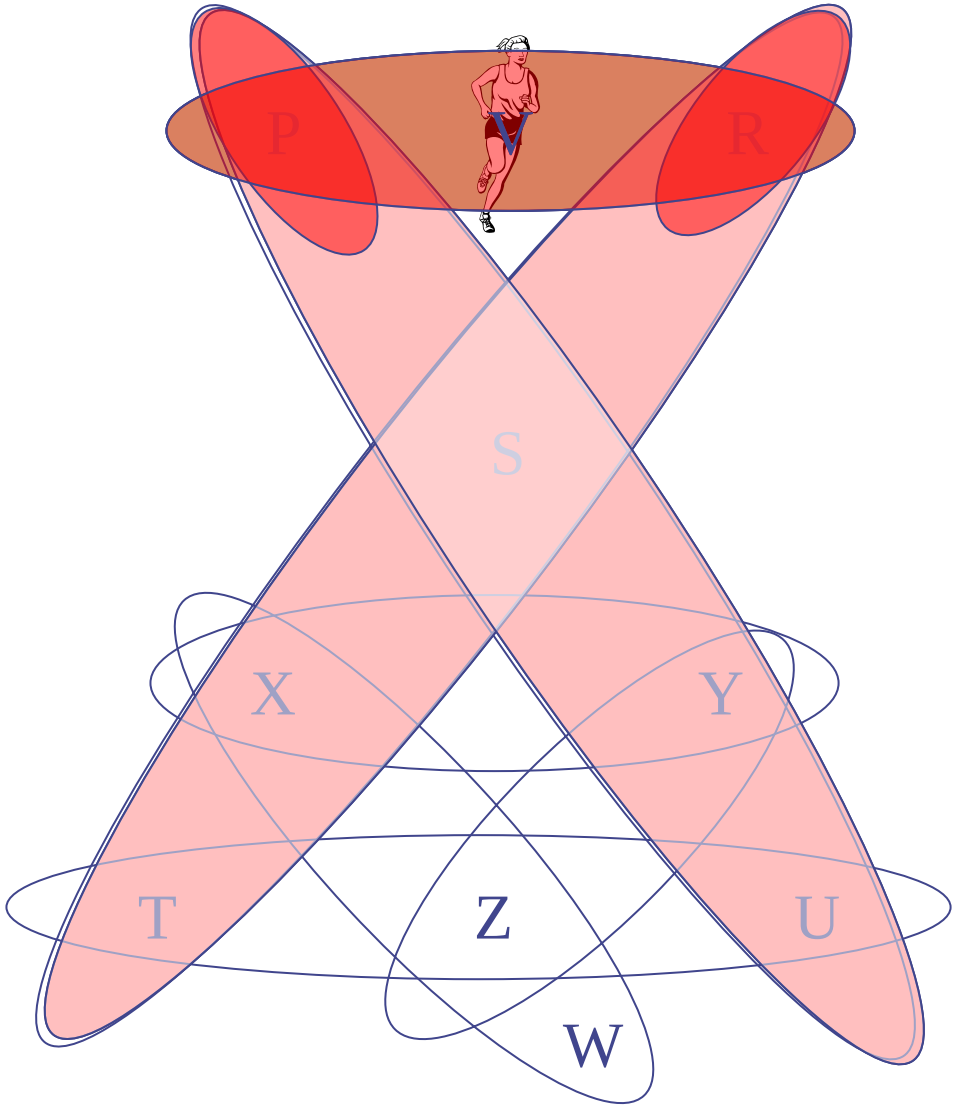
The capture



A different robber's choice



Step 2b: the capture



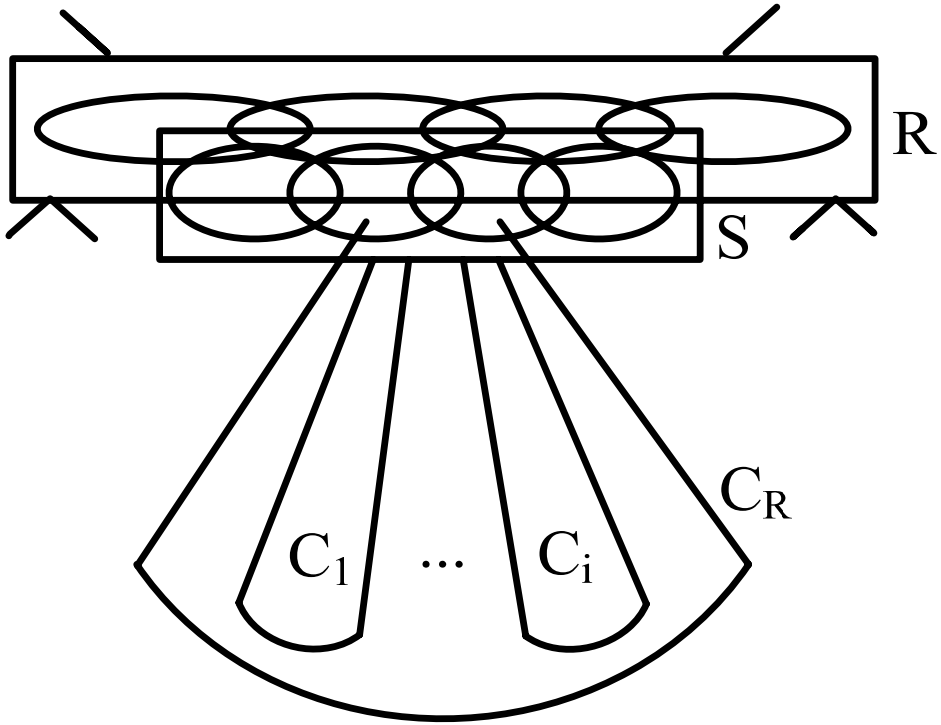
R&M Game and Hypertree Width

Let H be a hypergraph.

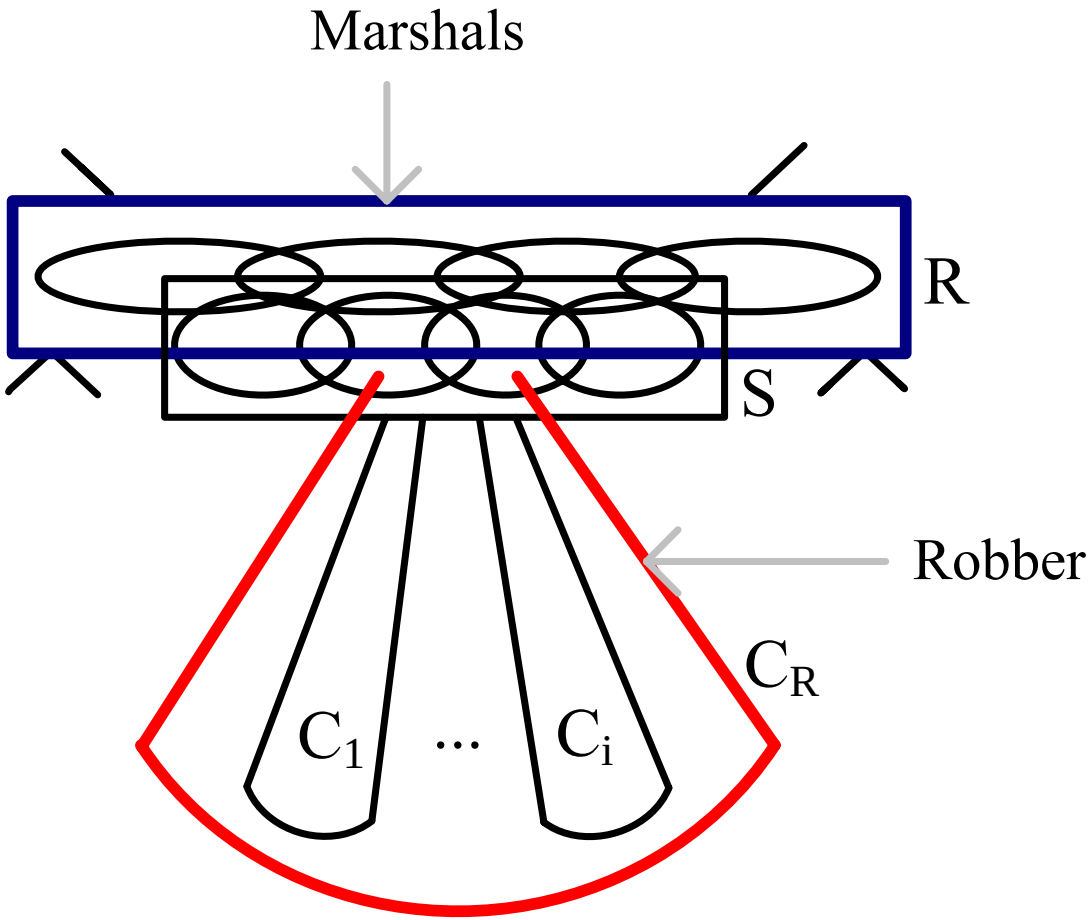
- **Theorem**: H has hypertree width $\leq k$ if and only if k marshals have a winning strategy on H.
- **Corollary**: H is acyclic if and only if one marshal has a winning strategy on H.

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

Marshals...

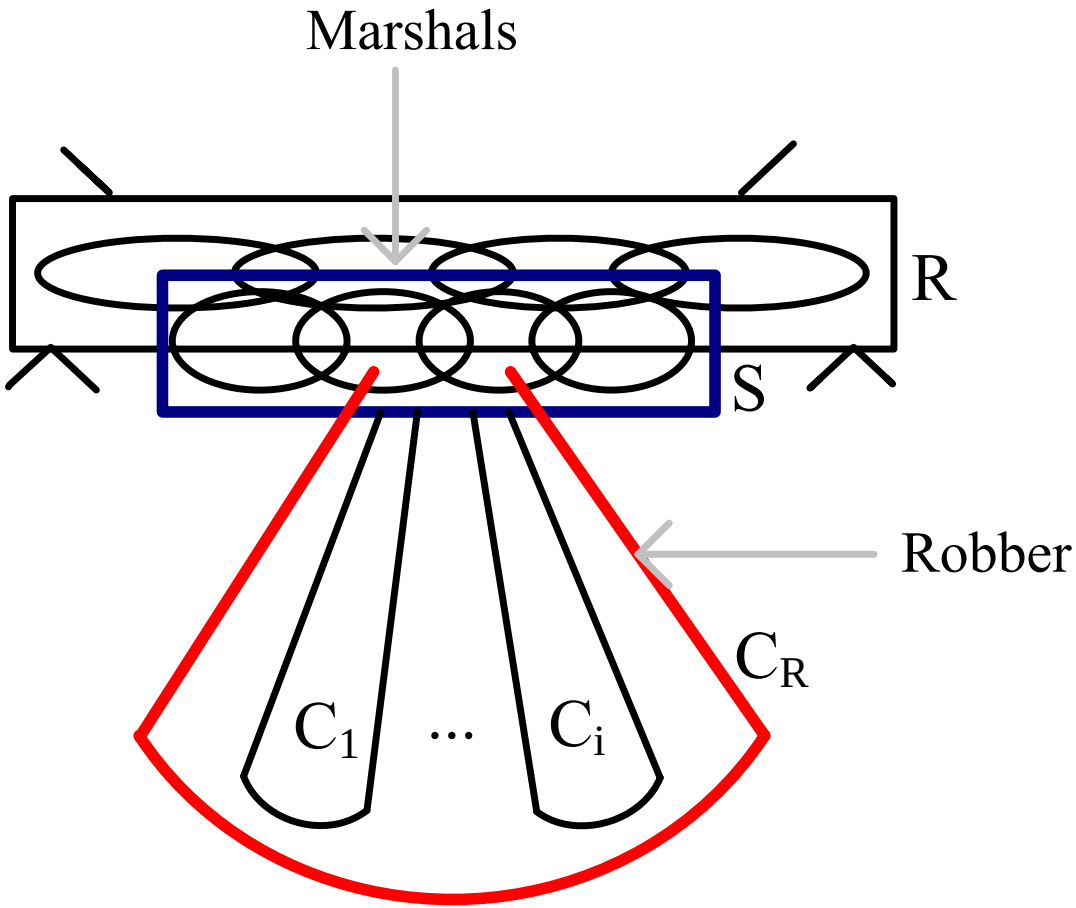


Marshals...



Polynomial algorithm: Alternating LOGSPACE

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



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

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

} LOGSPACE CHECKABLE

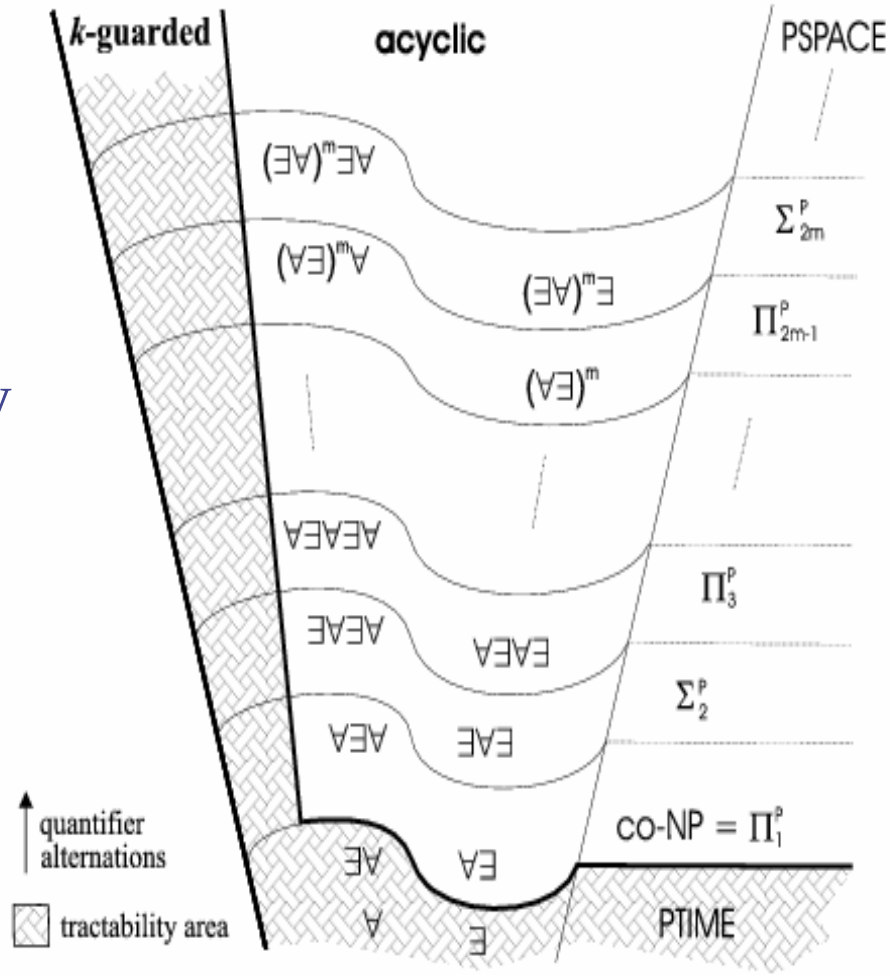
Quantified CSPs

Bad News:

- Even tree-structured QCSPs with prefix ∀∃ are intractable.
- For fixed domains, the tractability of bounded-treewidth QCSPs is optimal: even QCPS with acyclic hypergraphs and bounded treewidth incidence graphs are intractable

Good News:

- *k*-guarded QCSPs are tractable, without any restriction on domains or quantified alternations.



For furthe resultsr → [Hubie Chen]

Nasa problem

Part of relations for the Nasa problem

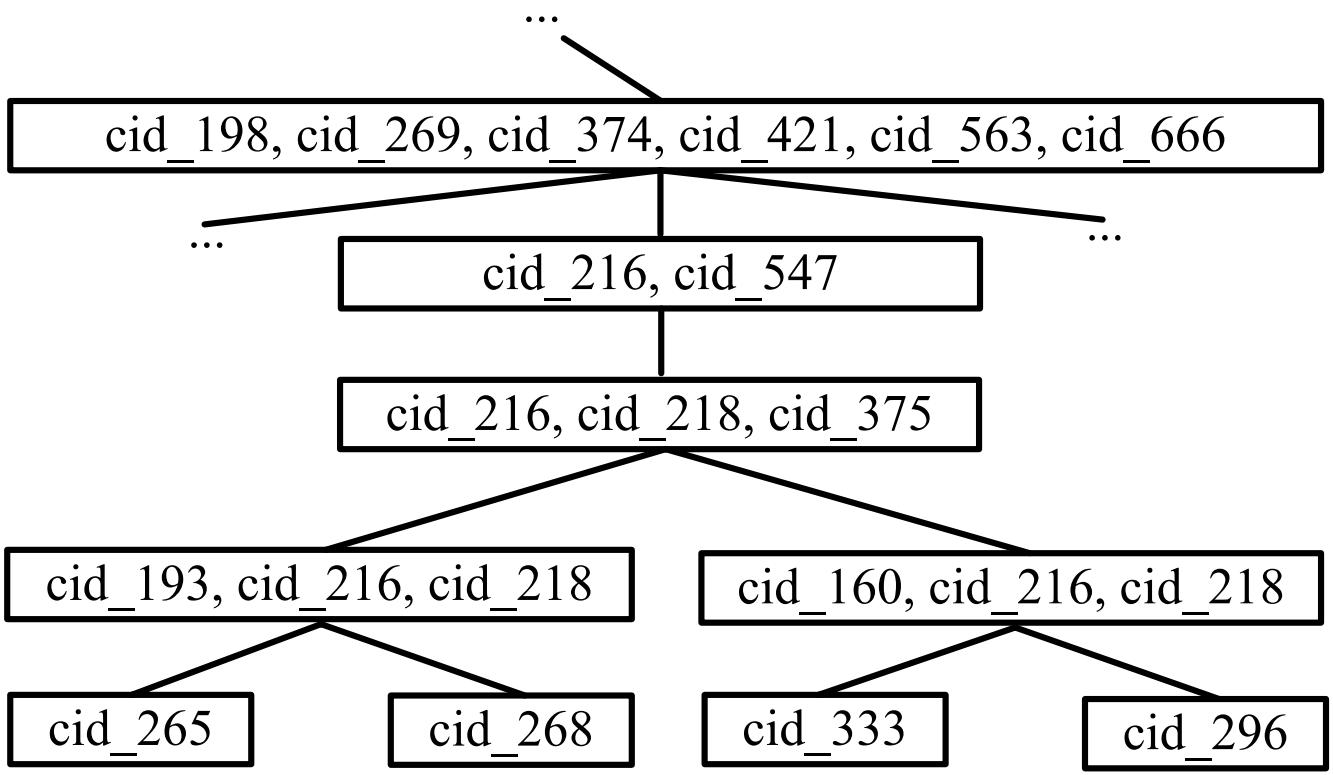
...

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/