Conjunctive Queries

Complexity & Decomposition Techniques

G. Gottlob
Technical University of Vienna, Austria

This talk reports about joint work with
I. Adler, M. Grohe, N. Leone and F. Scarcello

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

Three Problems:

CSP: Constraint satisfaction problem

BCQ: Boolean conjunctive query evaluation

HOM: The homomorphism problem

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

But actually: CSP = BCQ = HOM

CSP

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

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

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

scope — $(X_{j1},\ldots,X_{jr})$

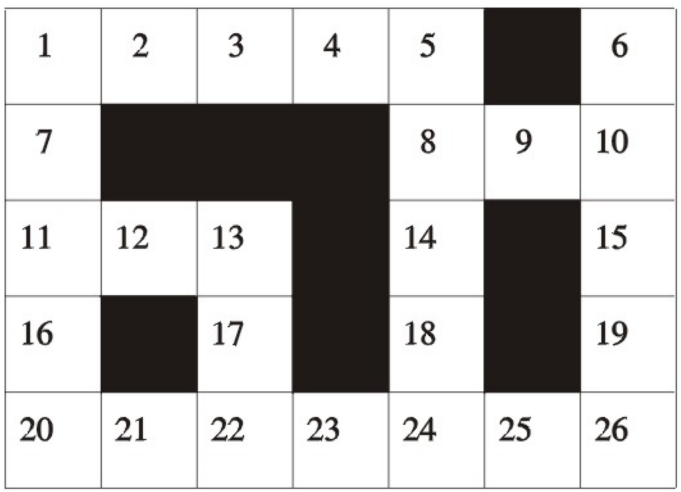
relation

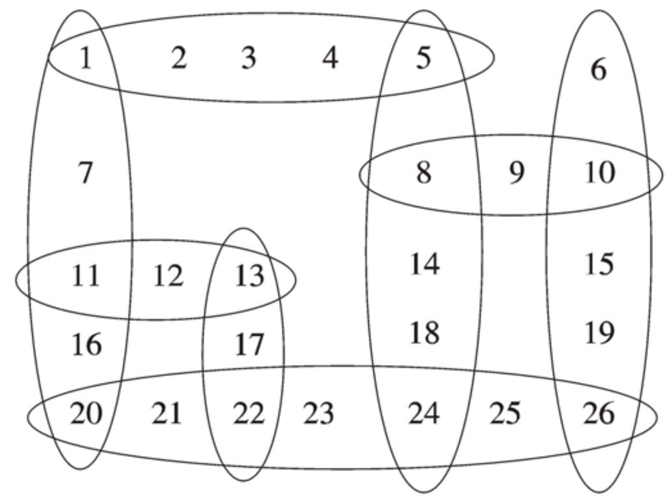
1	6	7	3
1	5	3	9
2	4	7	6
3	5	4	7

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

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

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:

Enrolled

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

Teaches

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

Parent

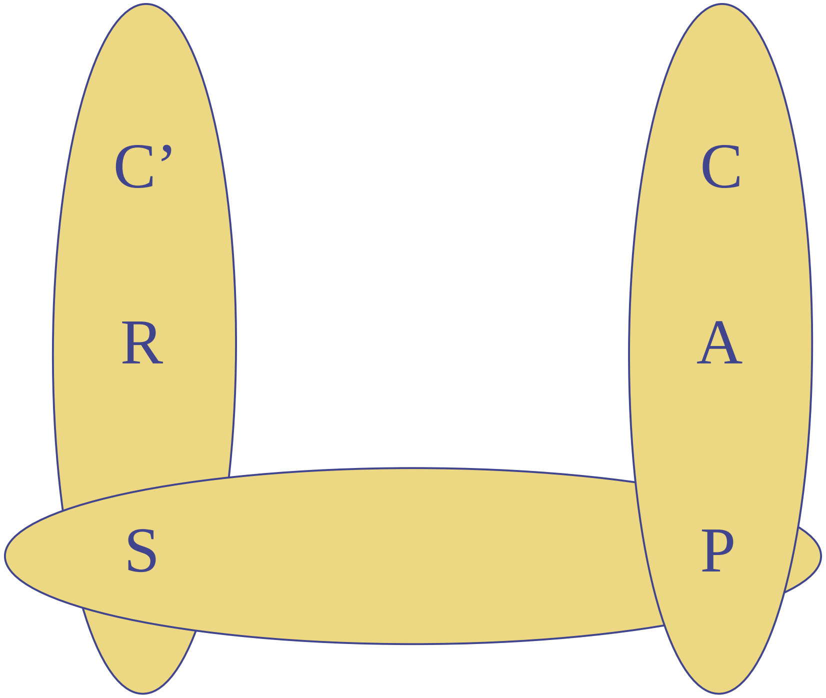
McLane	Lisa
Kolaitis	Robert
Rahm	Mary
....	.....

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

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

Queries and Hypergraphs

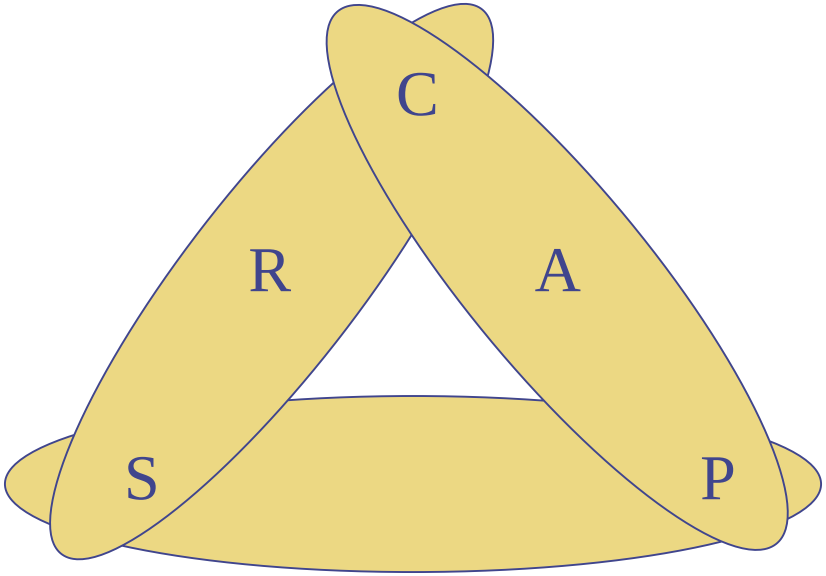
ans ← *Enrolled(S,C',R)* ∧ *Teaches(P,C,A)* ∧ *Parent(P,S)*



Queries, CSPs, and Hypergraphs

Is there a teacher whose child attends some course?

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



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*

$$h : U \longrightarrow V$$

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

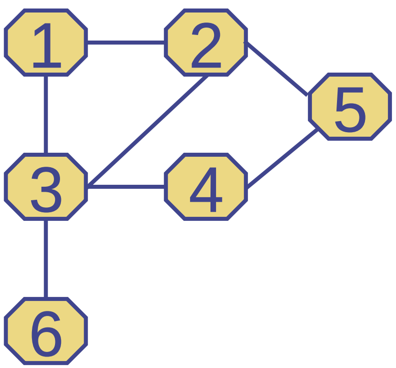
$$\mathbf{x} \in R_i \quad \Rightarrow \quad h(\mathbf{x}) \in S_i$$

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

[redacted]

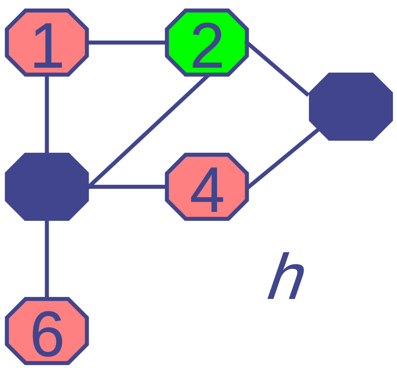
Graph 3-colourable iff HOM(*A*,*B*) yes-instance.

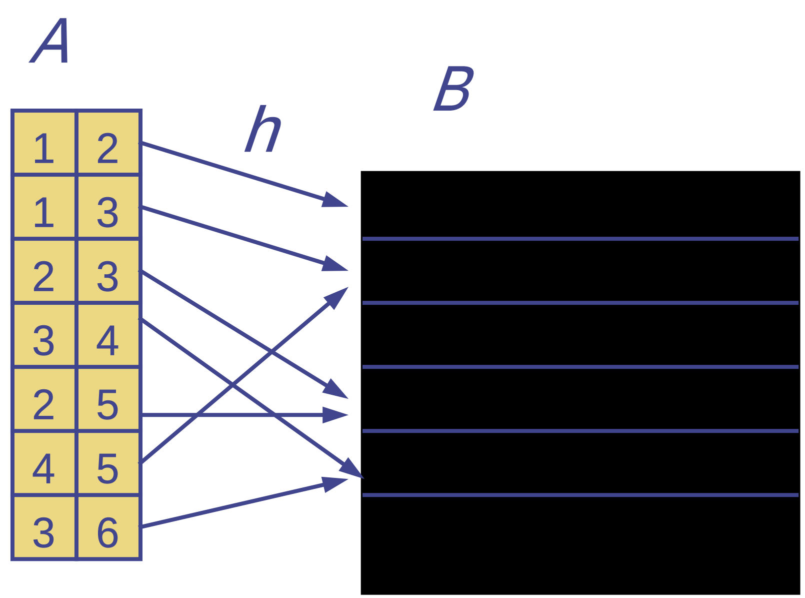
HOM is NP-complete

(well-known, independently proved in various contexts)

Membership: Obvious, guess *h.*

Hardness: Transformation from 3COL.





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

CSP= BCQ = HOM

Complexity of CSPs

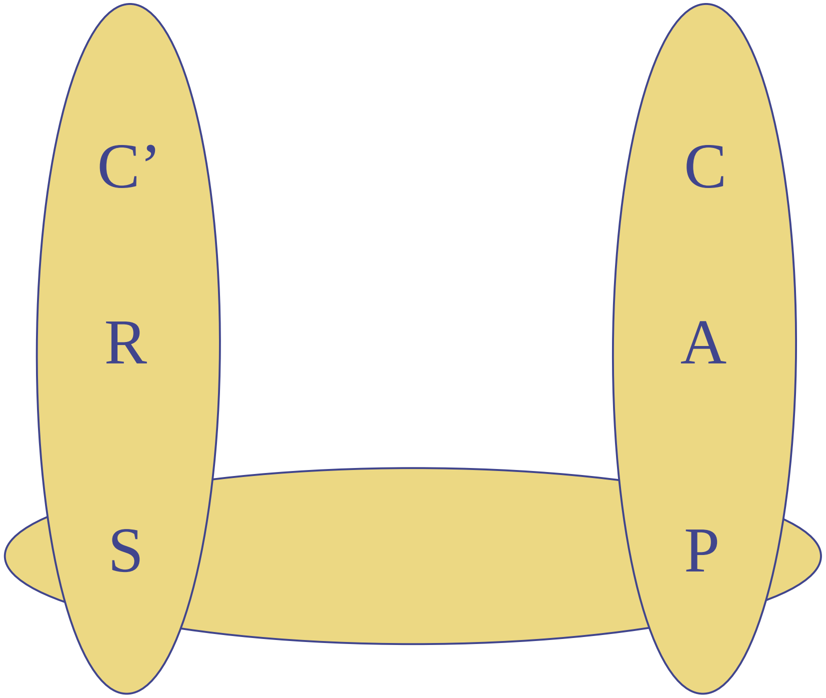
- NP-complete in the general case (Bibel, Chandra and Merlin '77, etc.)
  NP-hard even for fixed constraint relations

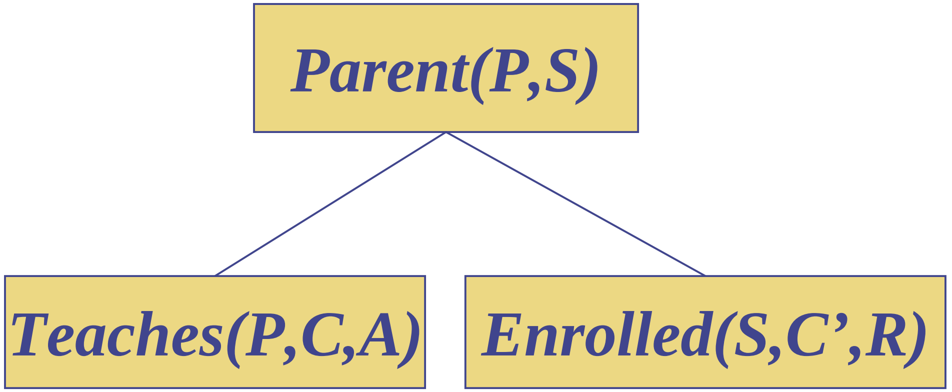
- Polynomial in case of acyclic hypergraphs (Yannakakis '81)
  LOGCFL-complete (in NC_2) (G.L.S. '98)

⇒ Interest in larger tractable classes of CQ/CSP

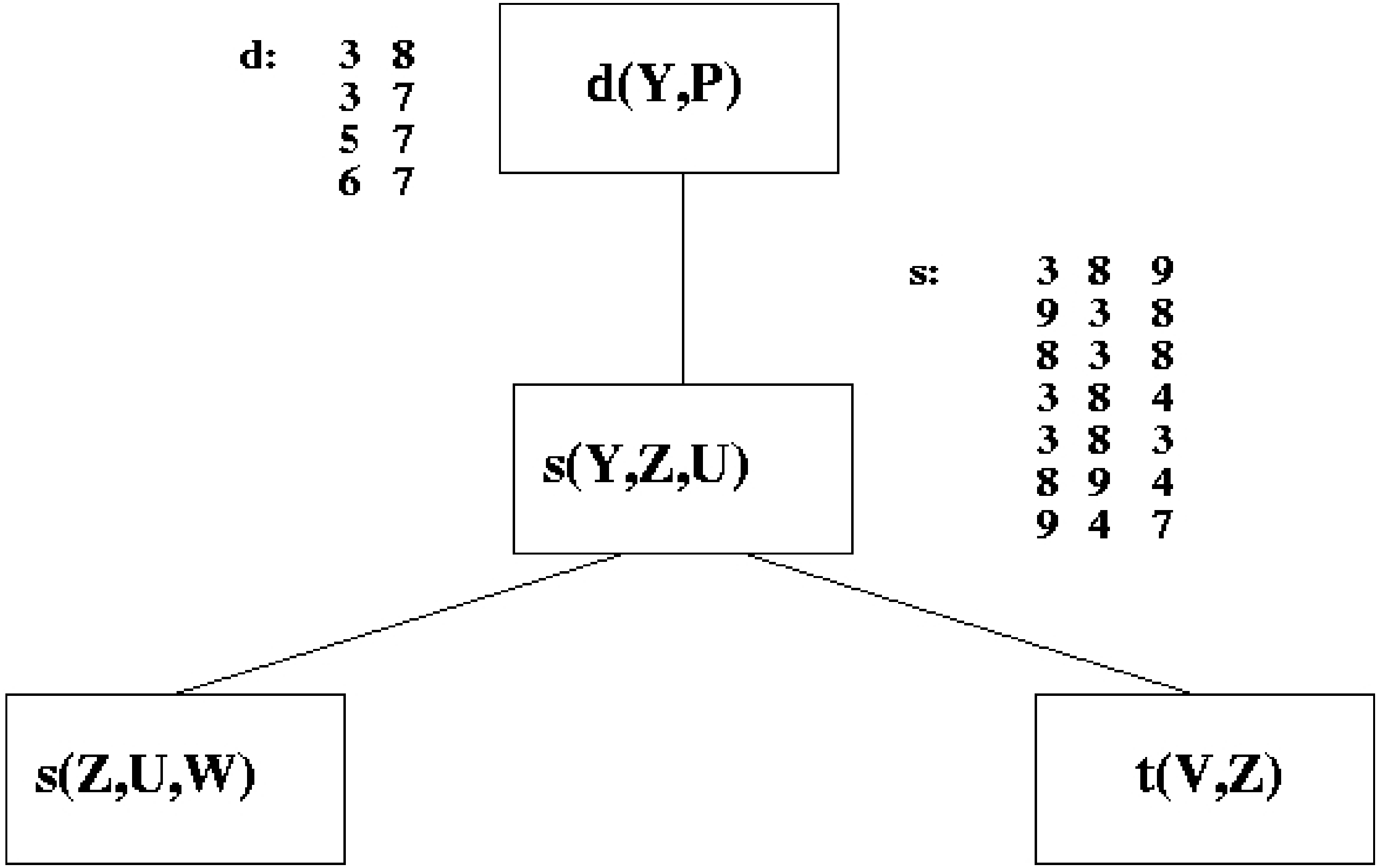
Acyclic Hypergraphs

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





Join Tree



d:

3	8
3	7
5	7
6	7

s:

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

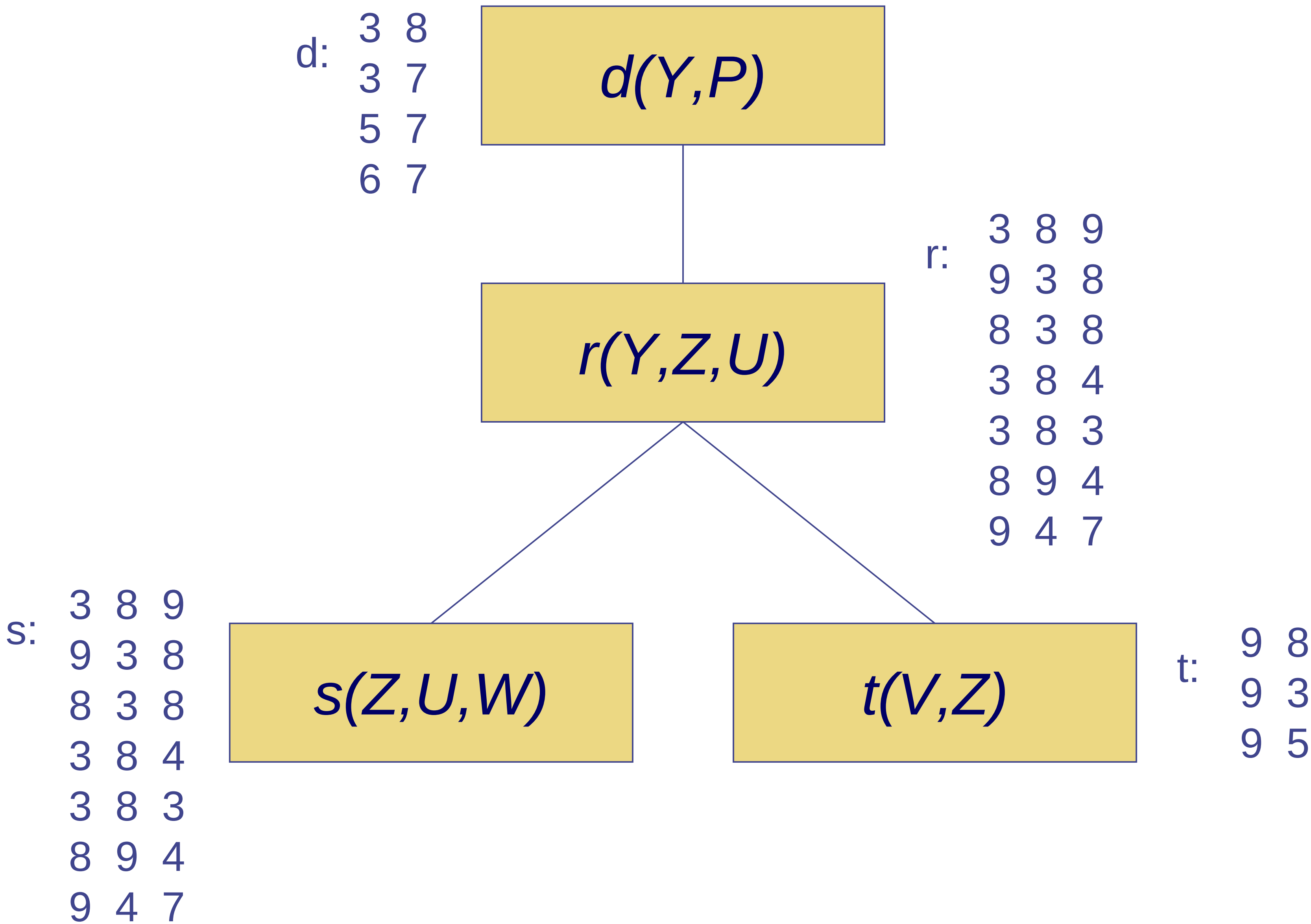
s:

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

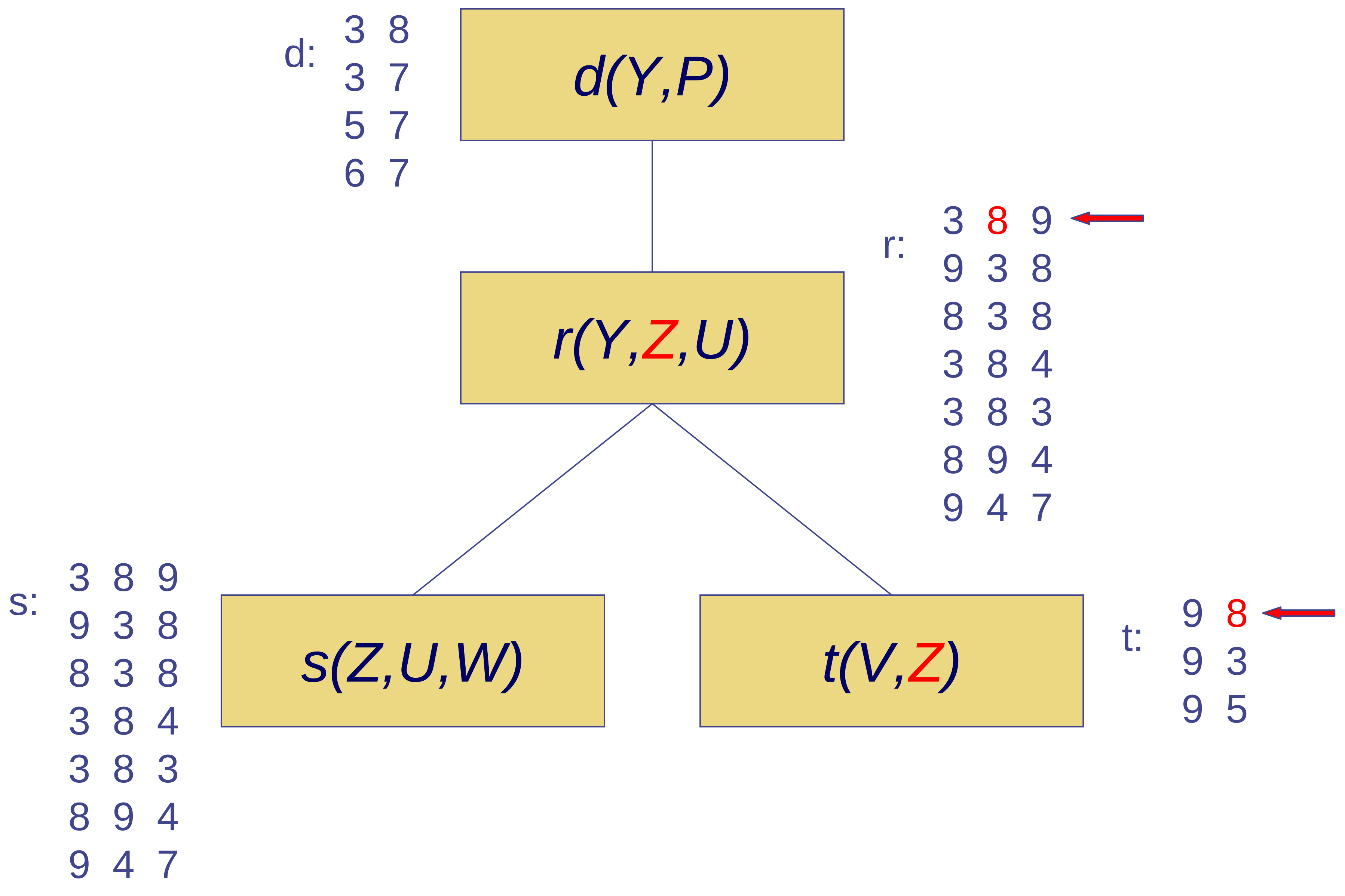
r:

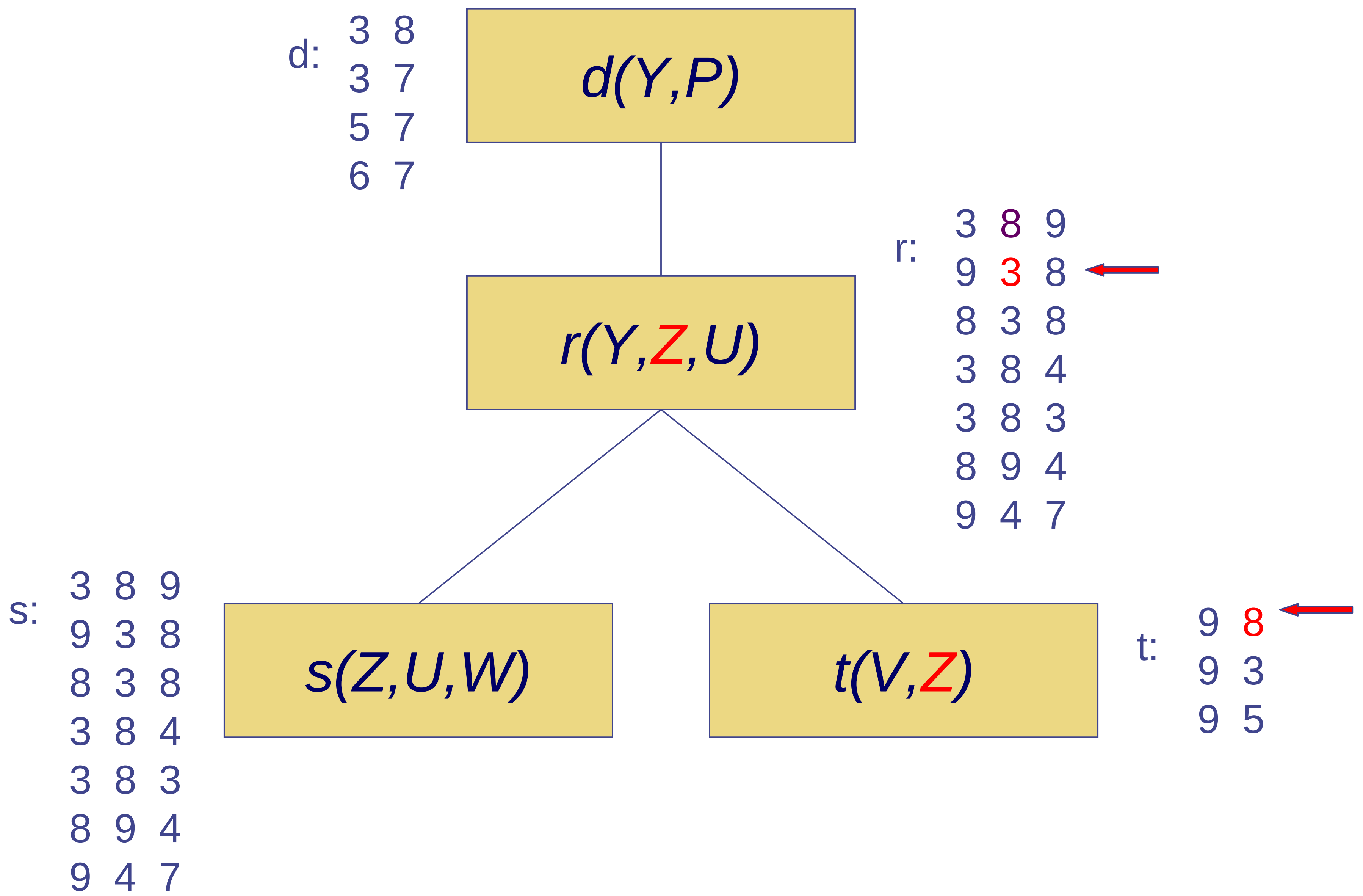
9	8
9	3
9	5

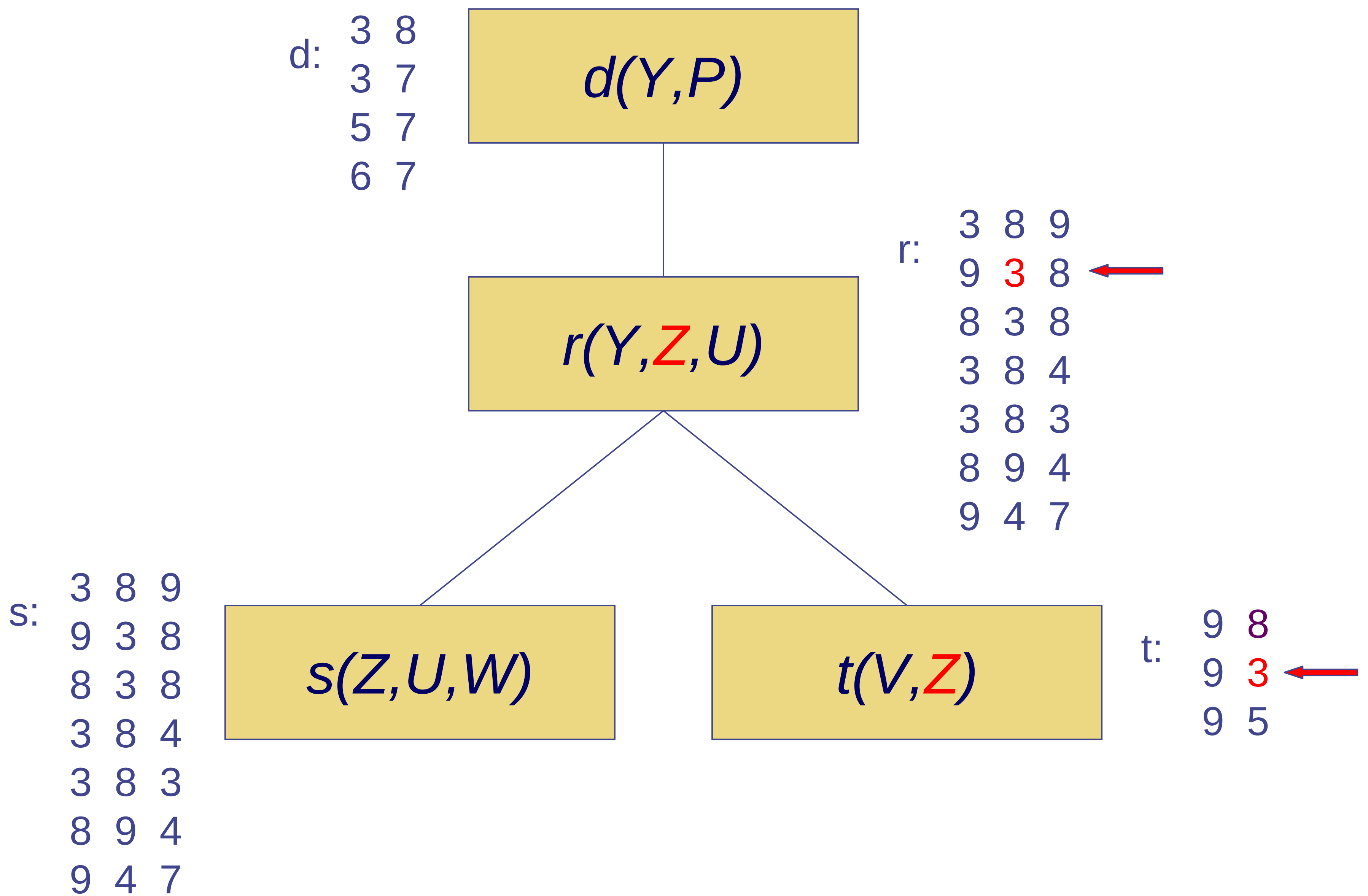
$n^2 \log n$

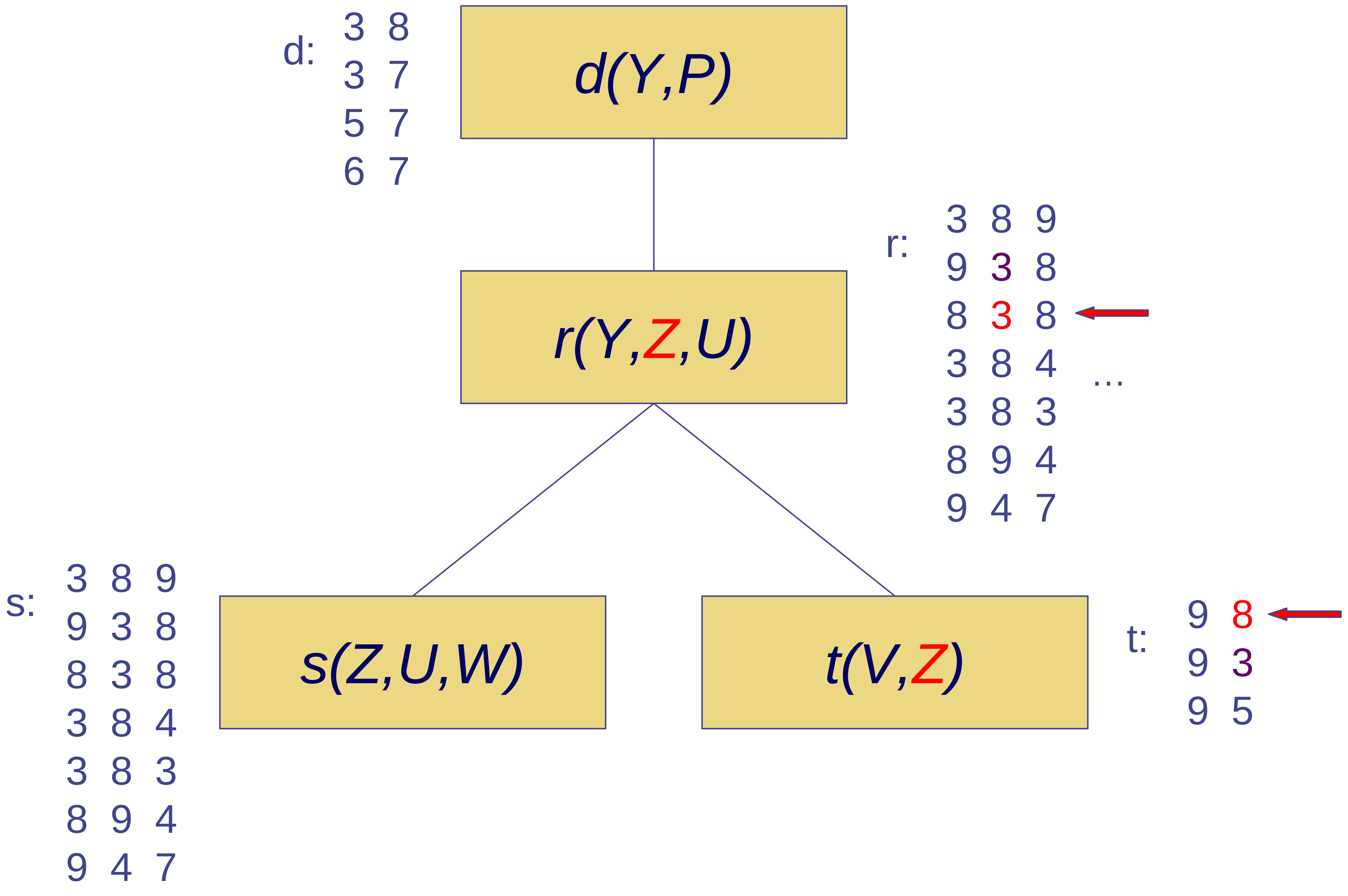


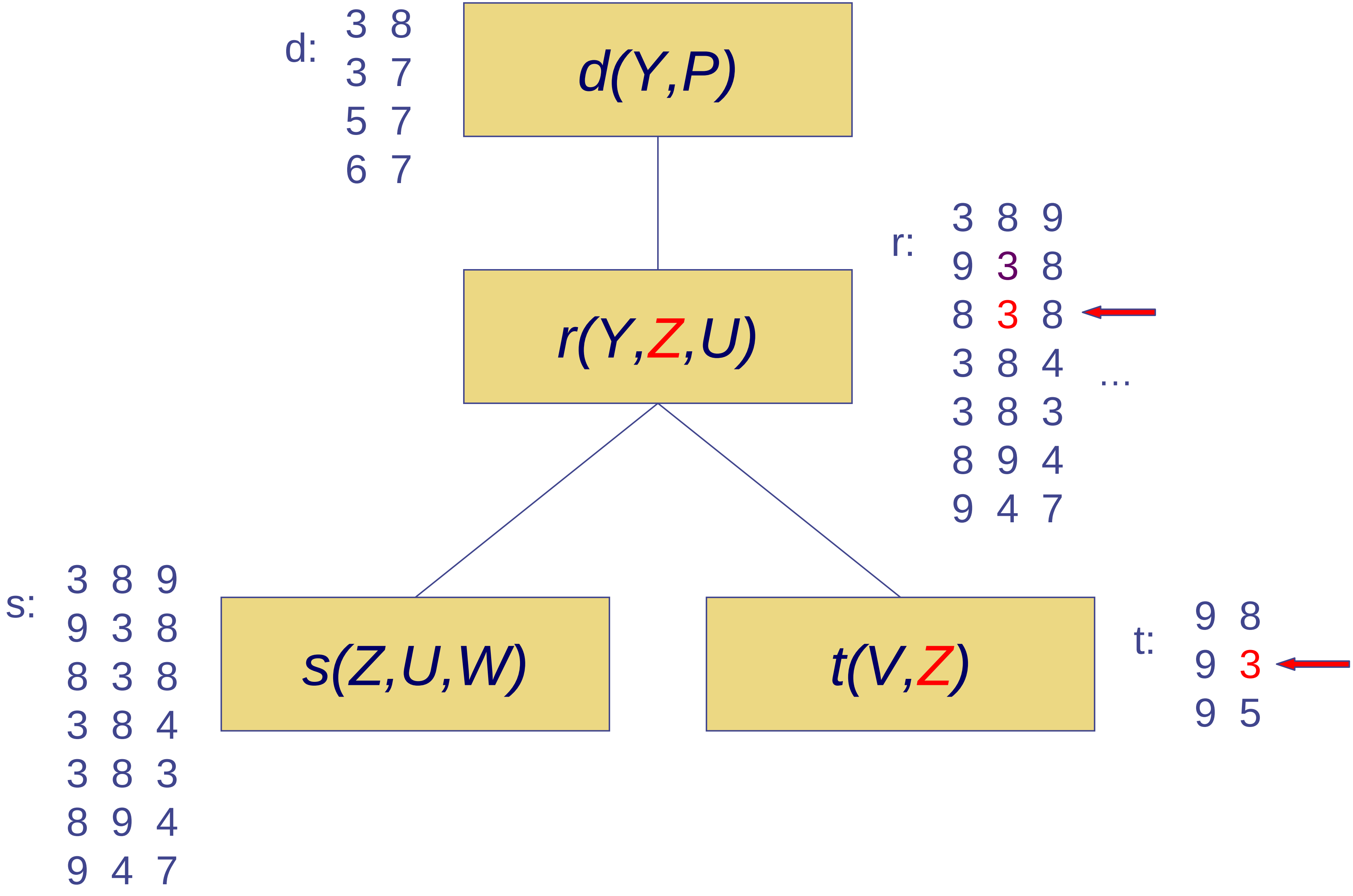
d(Y,P)
d:
3 8
3 7
5 7
6 7
r(Y,Z,U)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
t(V,Z)
t:
9 8
9 3
9 5

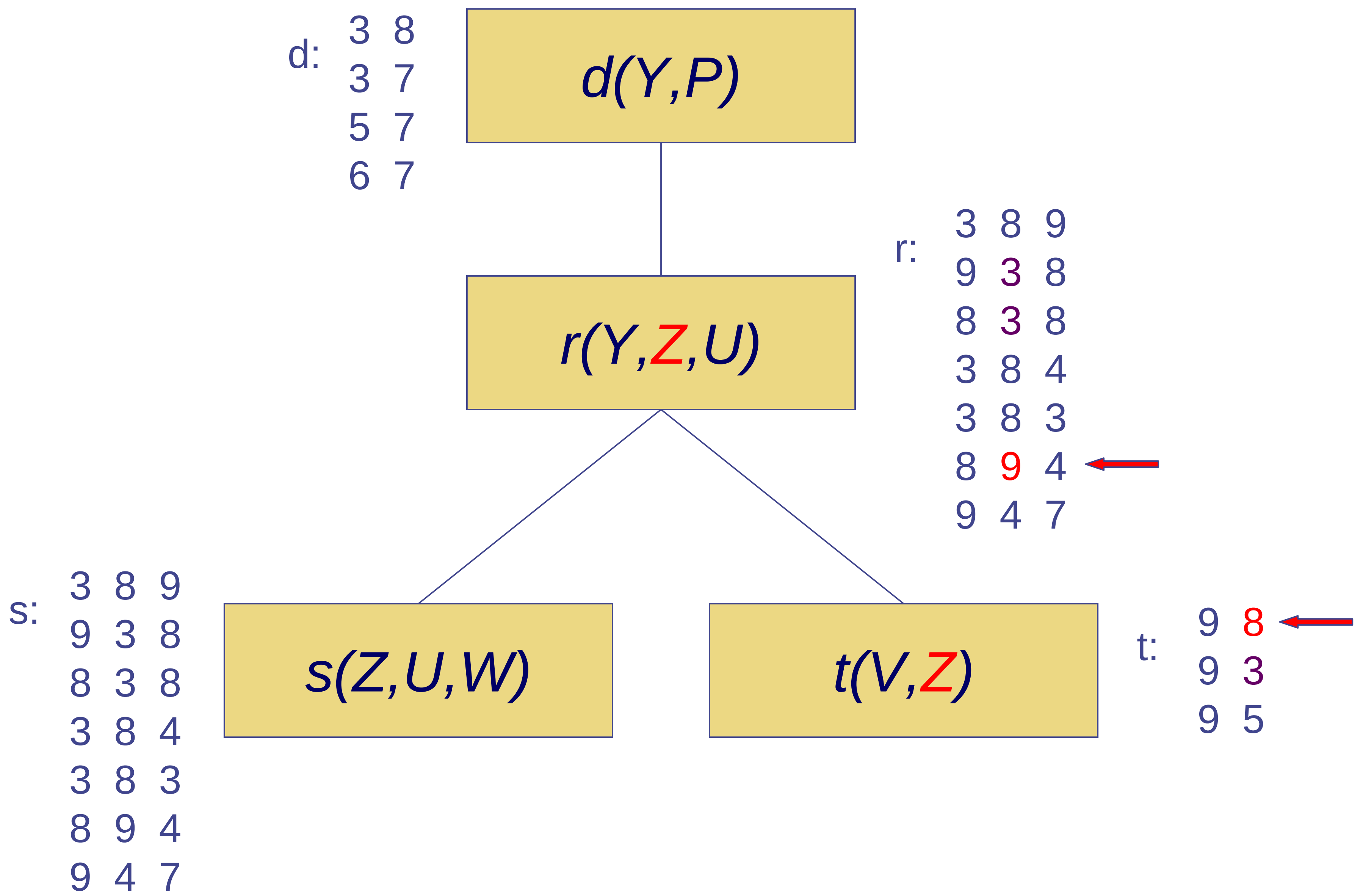

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

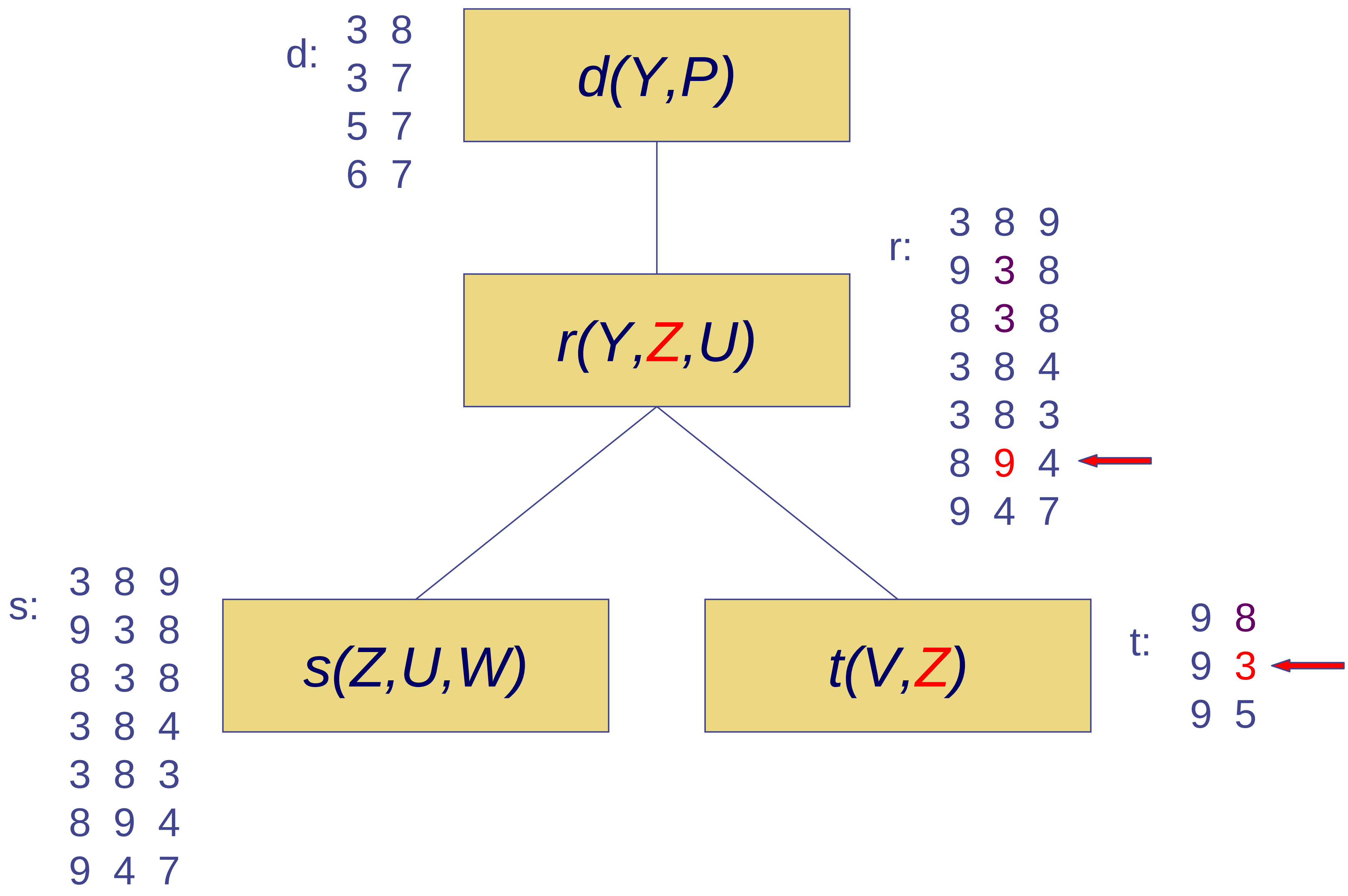

d(Y,P)
r(Y,Z,U)
s(Z,U,W)
t(V,Z)
d:
3 8
3 7
5 7
6 7
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
t:
9 8
9 3
9 5


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

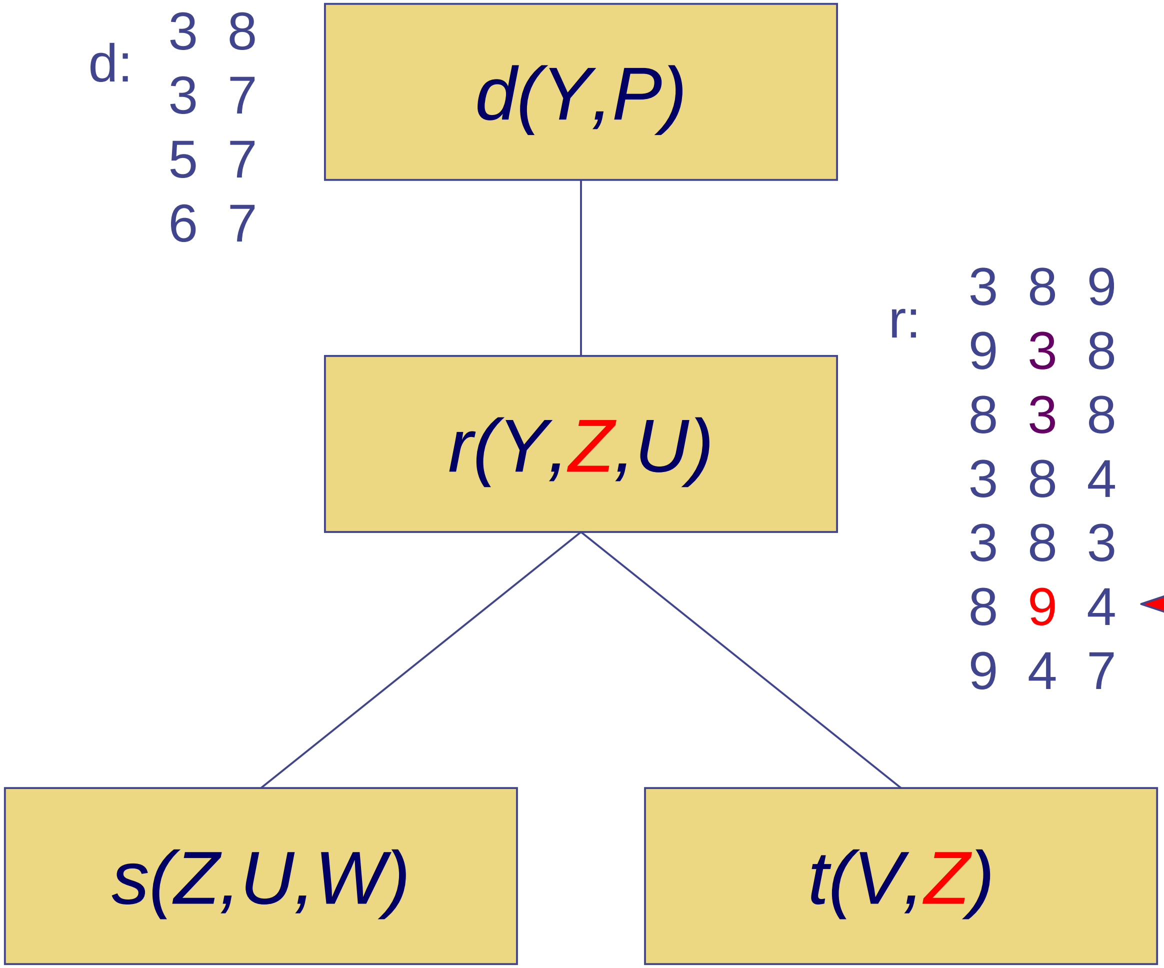

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
...
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t:
9 8
9 3
9 5
t(V,Z)


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:
9 8
9 3
9 5
t(V,Z)


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



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



d:

3	8
3	7
5	7
6	7

r:

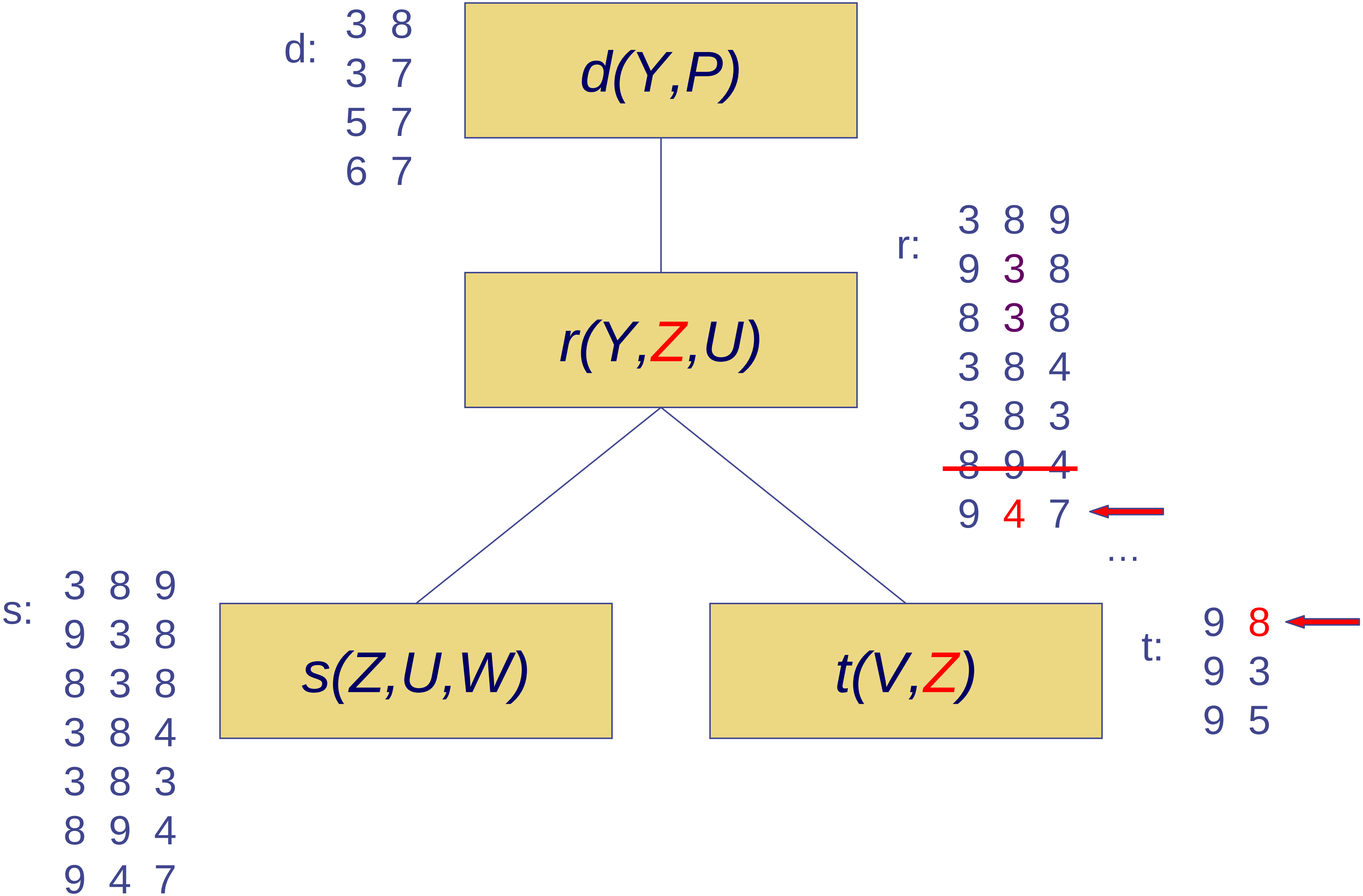
3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

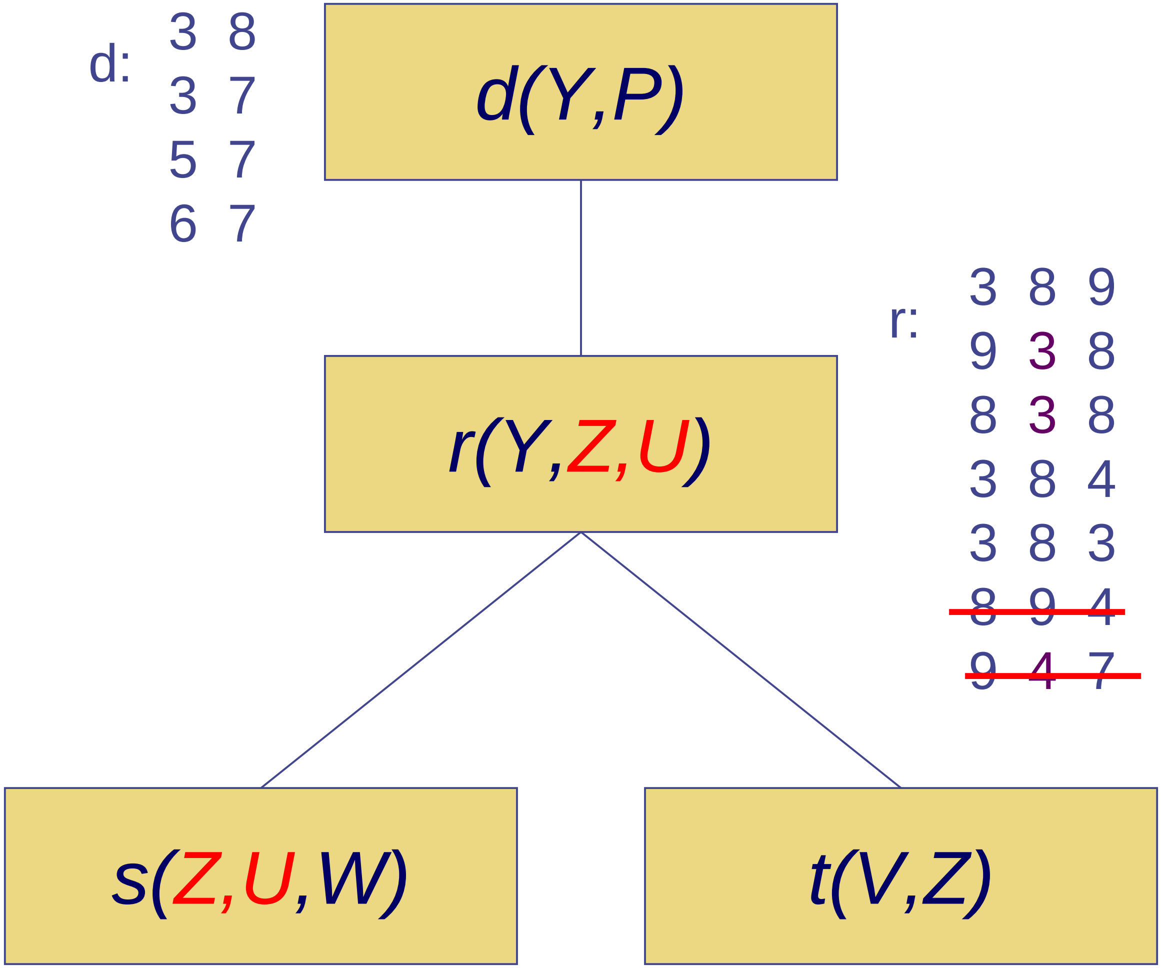
s:

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

t:

9	8
9	3
9	5


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



d:

3	8
3	7
5	7
6	7

r:

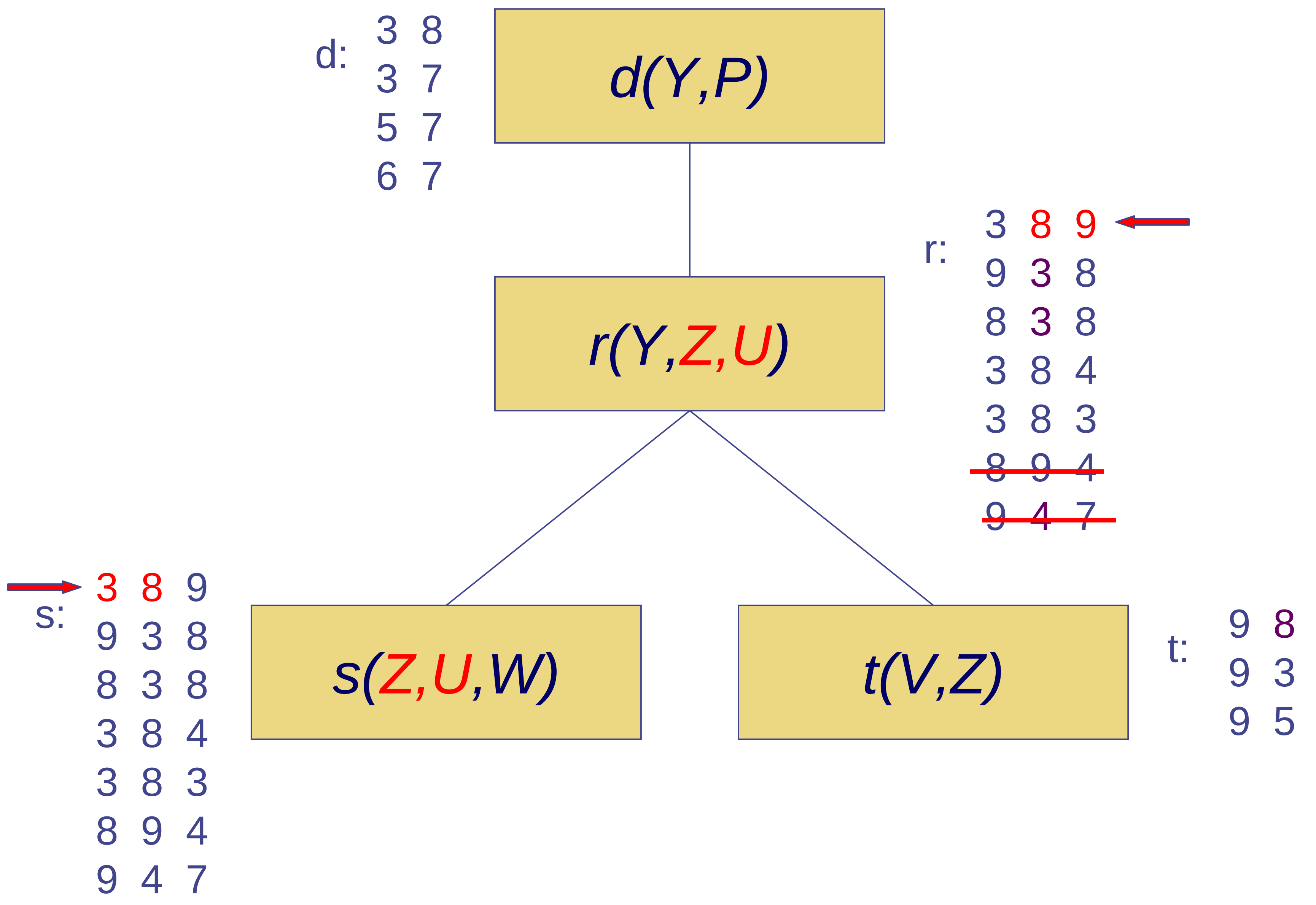
3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
~~8~~	~~9~~	~~4~~
~~9~~	~~4~~	~~7~~

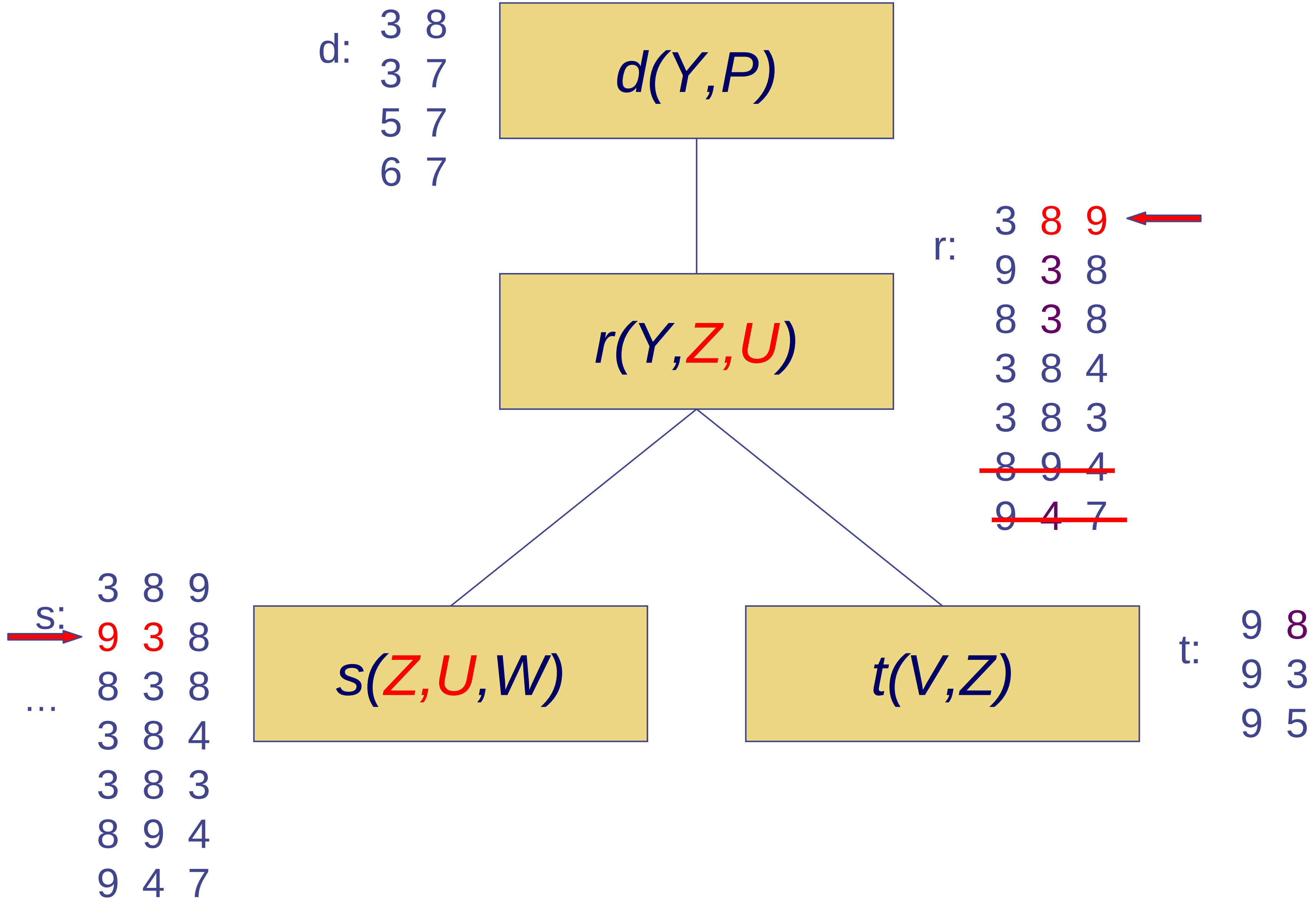
s:

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

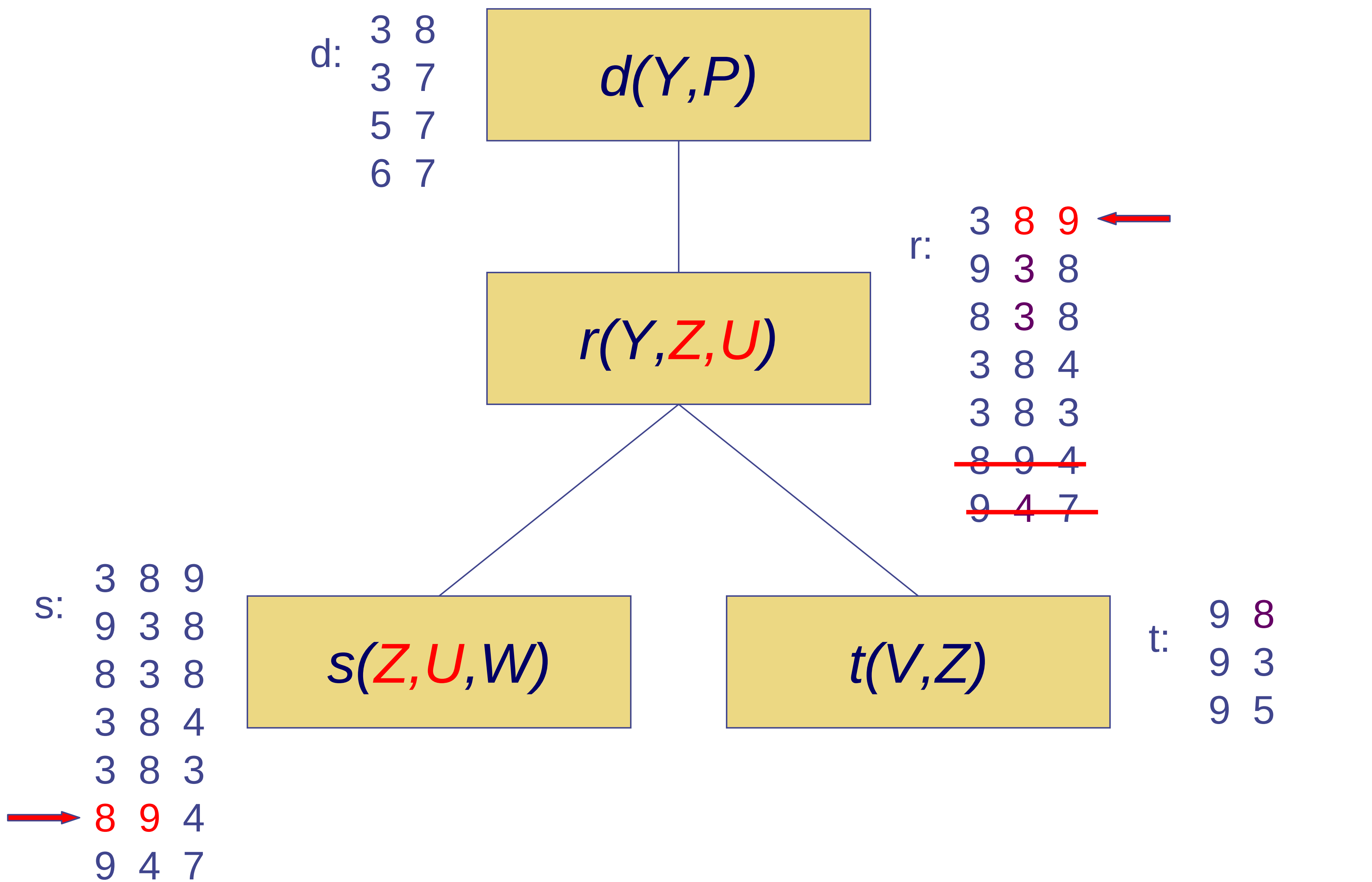
t:

9	8
9	3
9	5

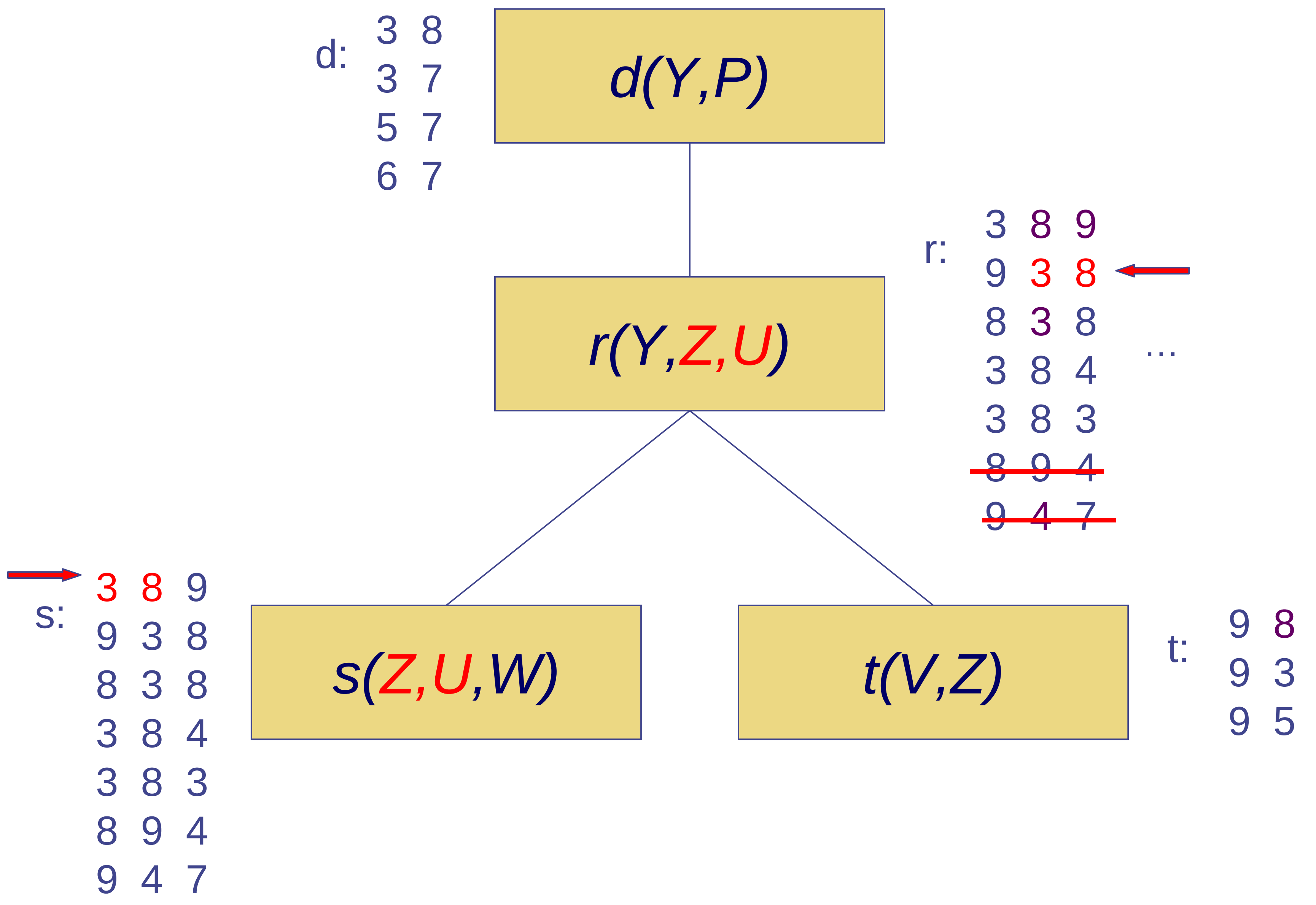

d(Y,P)
d:
3 8
3 7
5 7
6 7
r(Y,Z,U)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
t(V,Z)
t:
9 8
9 3
9 5

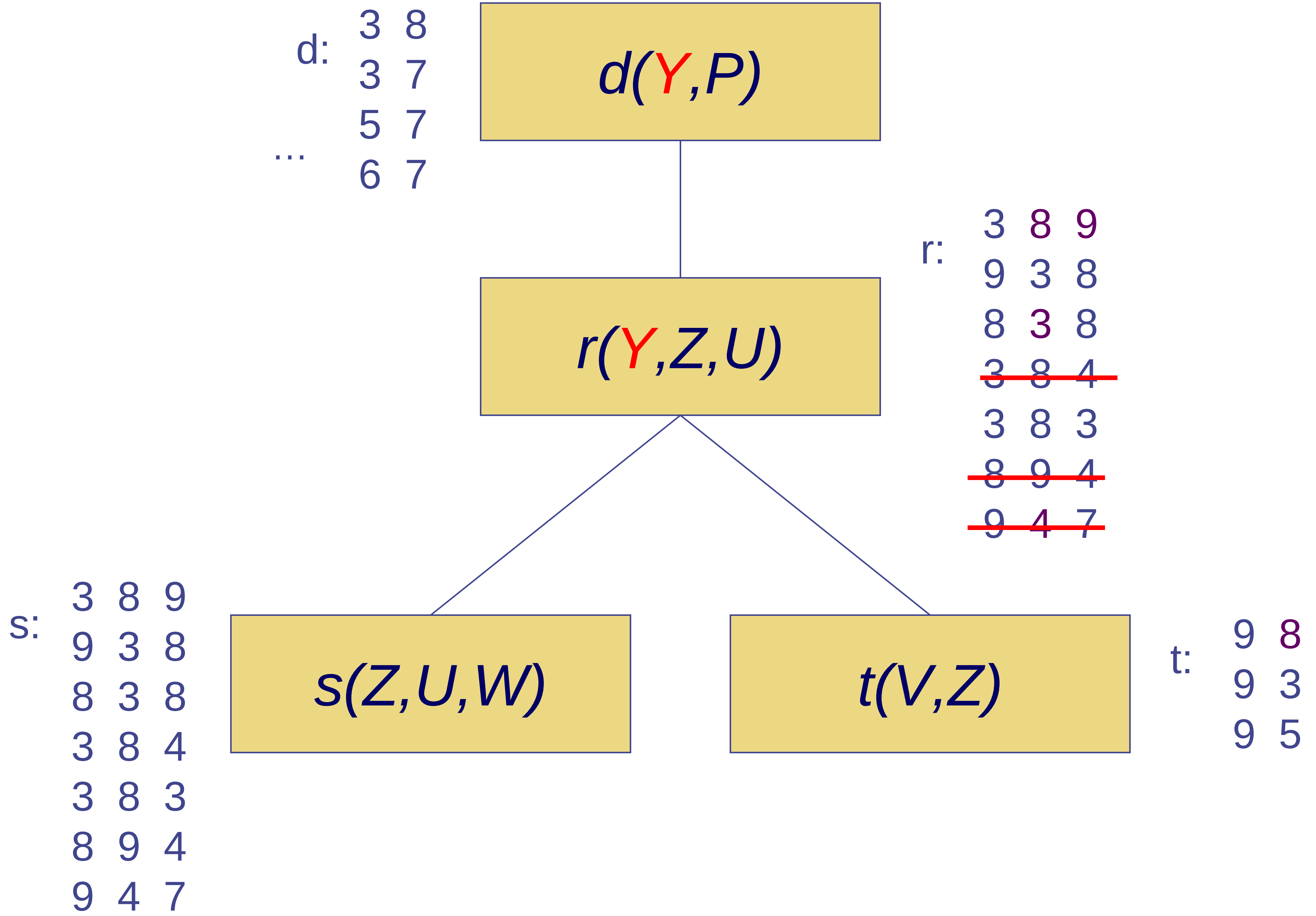


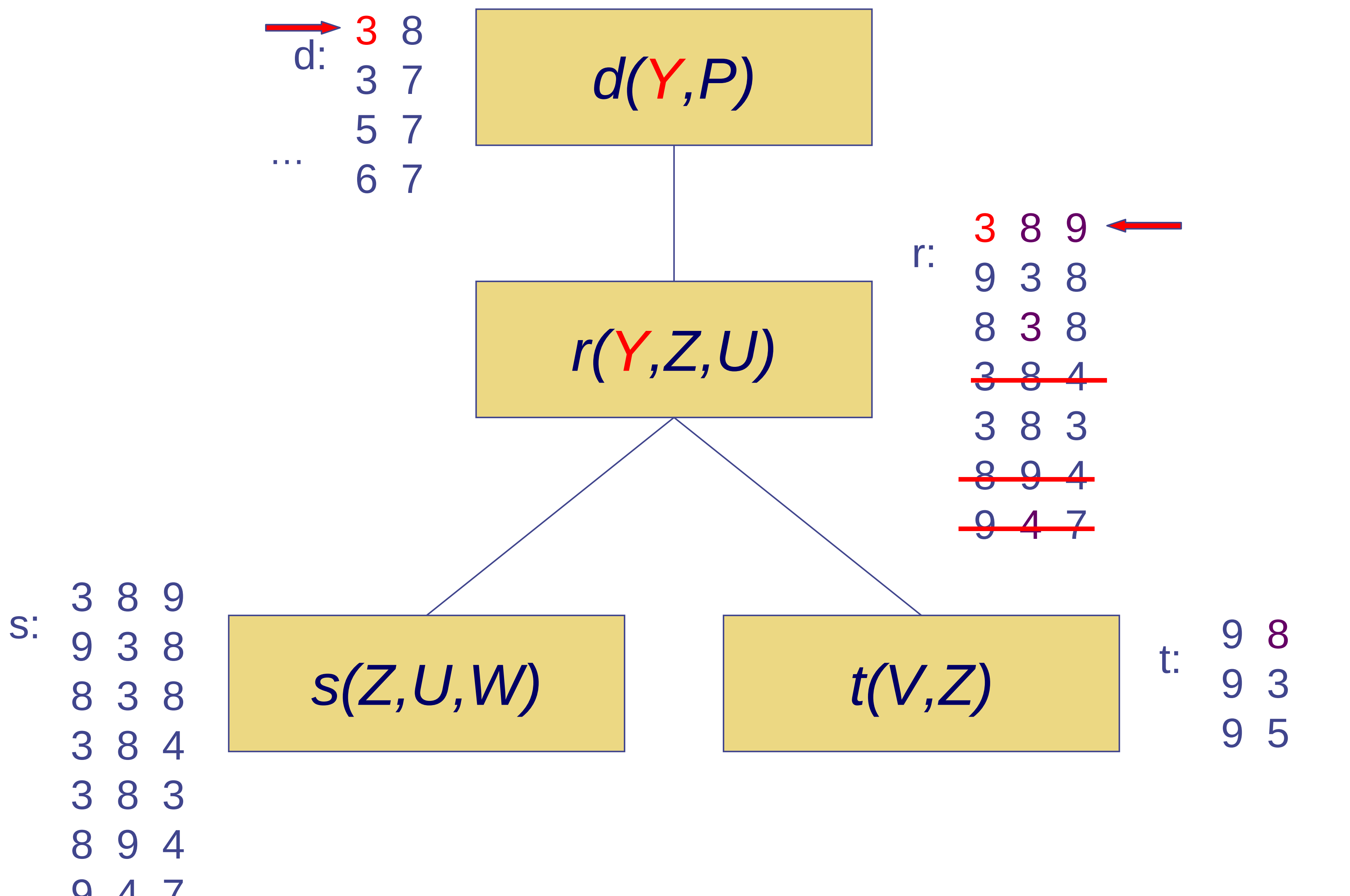
d(Y,P)
r(Y,Z,U)
s(Z,U,W)
t(V,Z)
d:
3 8
3 7
5 7
6 7
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
~~8 9 4~~
~~9 4 7~~
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
…
t:
9 8
9 3
9 5

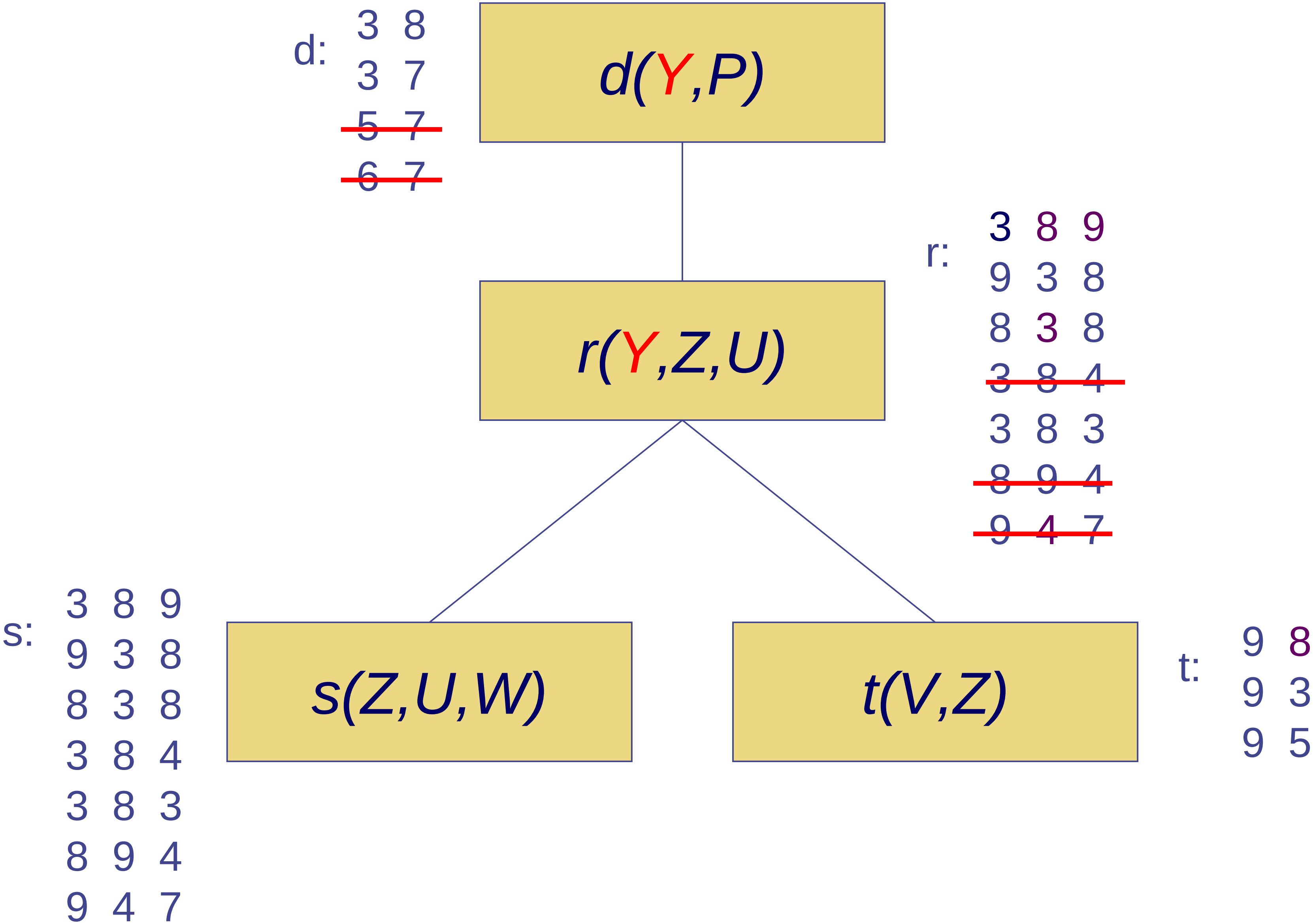


d:
3 8
3 7
5 7
6 7
d(Y,P)
r(Y,Z,U)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
~~8 9 4~~
~~9 4 7~~
s(Z,U,W)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
t(V,Z)
t:
9 8
9 3
9 5

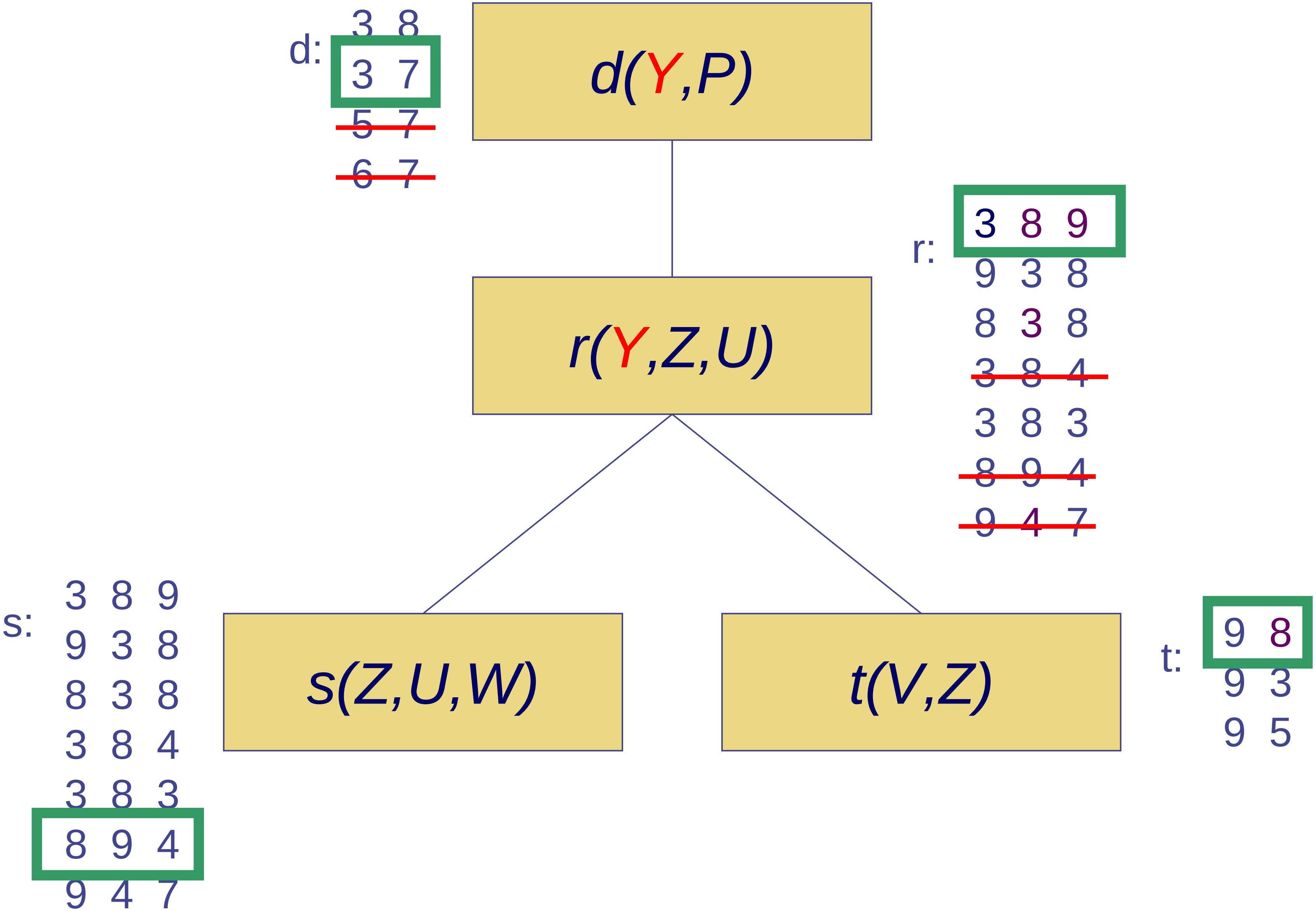

d(Y,P)
r(Y,Z,U)
s(Z,U,W)
t(V,Z)
d:
3 8
3 7
5 7
6 7
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
...
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
t:
9 8
9 3
9 5


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


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



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



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 joinoing the partial results.

Theorem [GLS99]: Answering acyclic BCQs is LOGCFL-complete

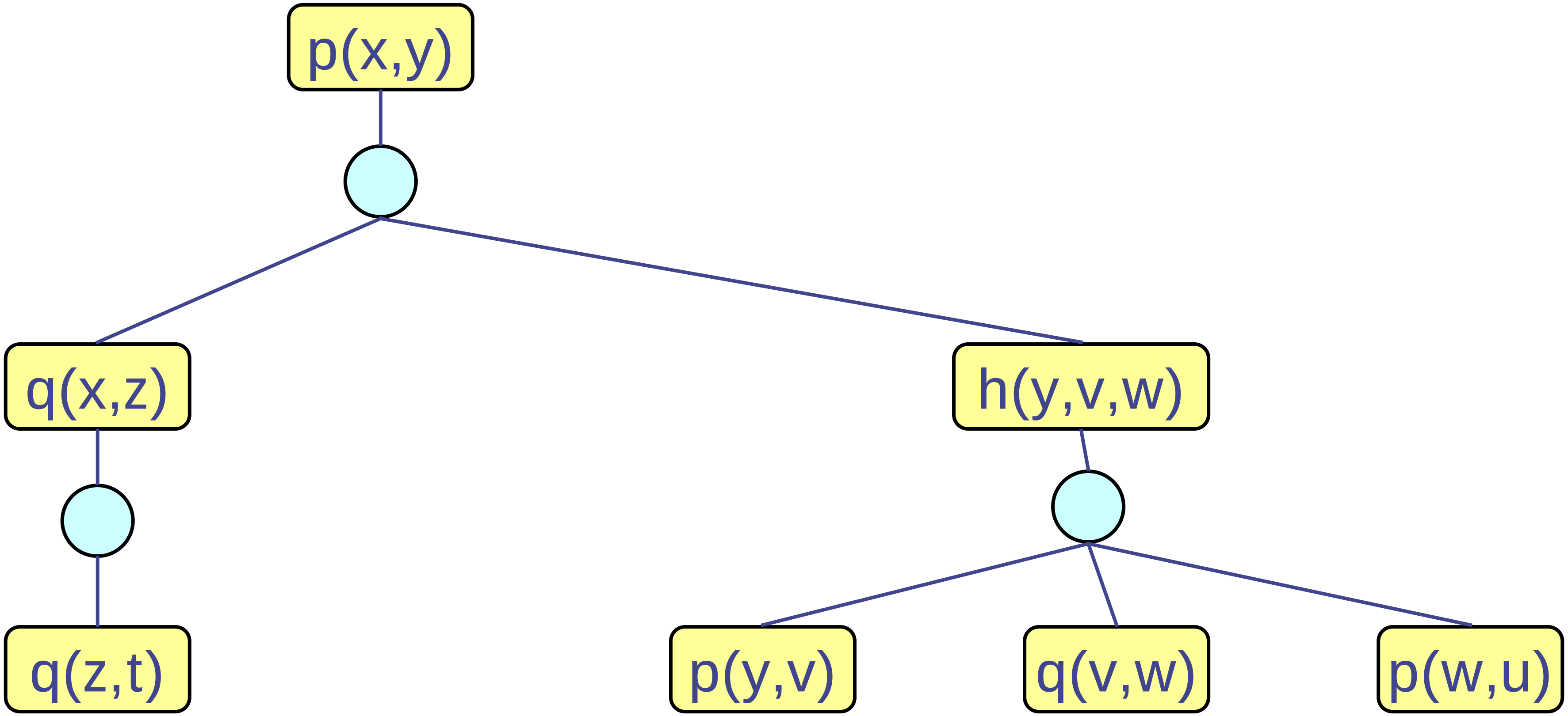
LOGCFL: class of problems/languages that are logspace-reducible to a CFL

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

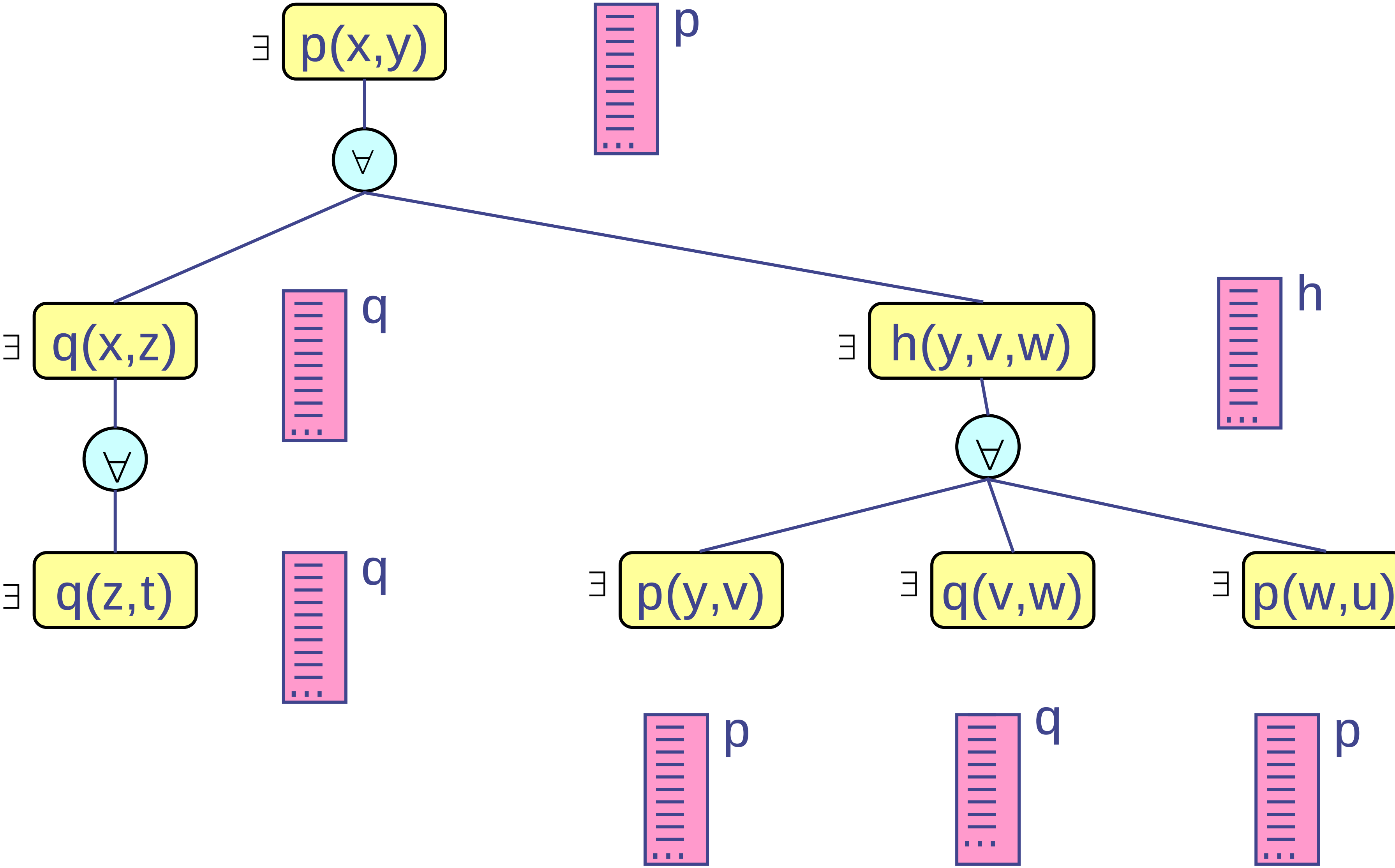
Characterization of LOGCFL [Ruzzo80]:

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

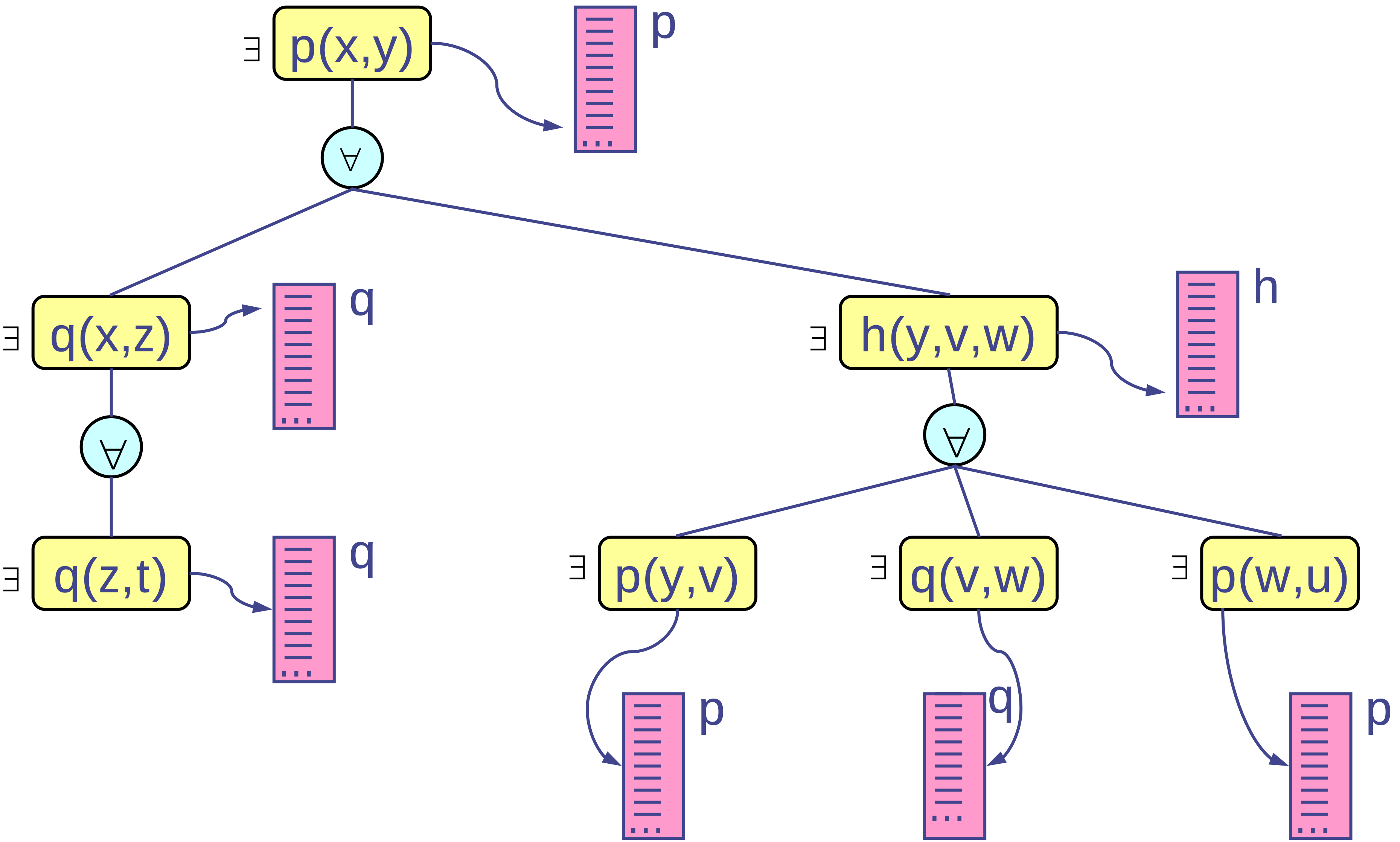
ABCQ is in LOGCFL



ABCQ is in LOGCFL



ABCQ is in LOGCFL



Is this query hard?

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

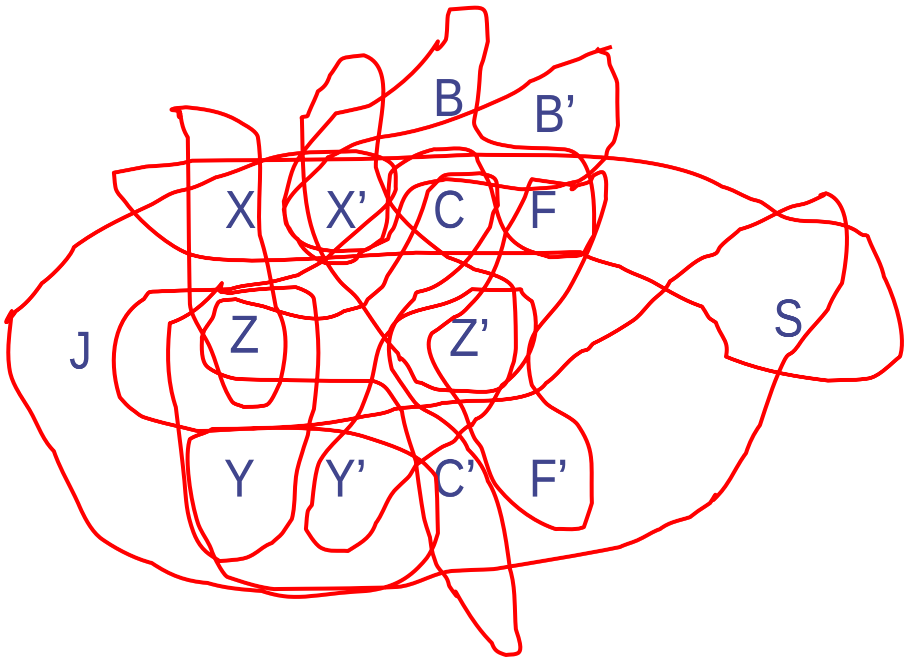
$m = 11$!

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

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

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

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



It can be evaluated in *O(m·n*^{2}*·* 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

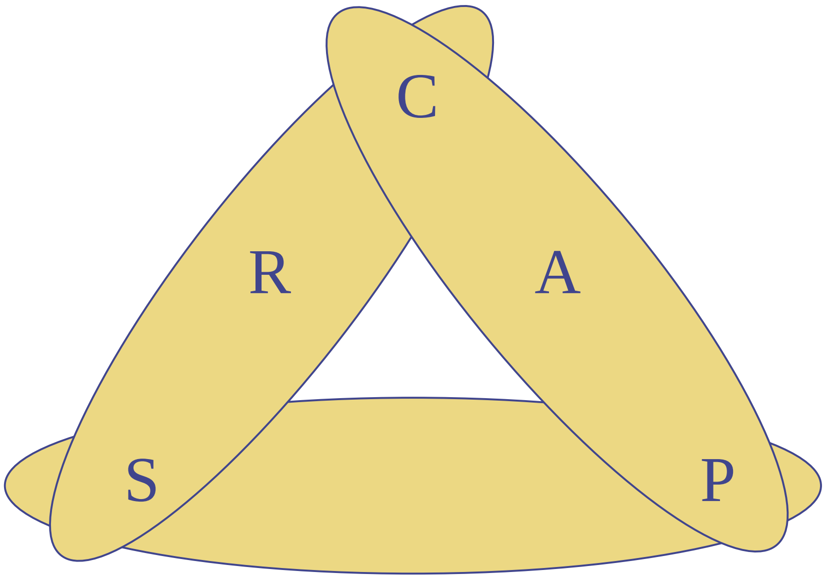
  } linear time (Bodlaender'96)
- Deciding CSP of treewidth *k*:

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

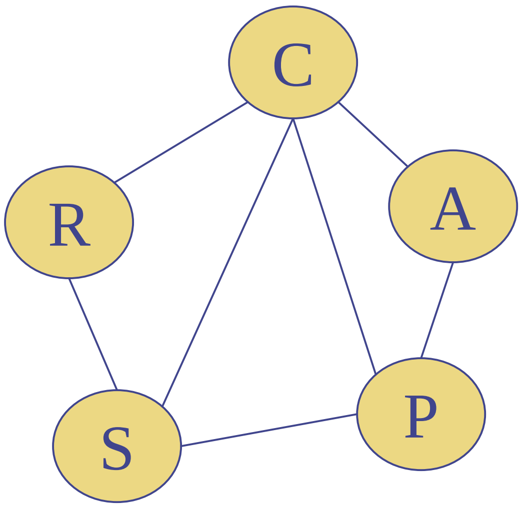
  LOGCFL-complete (G.L.S.'98)

Primal graphs of CSPs/Queries

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

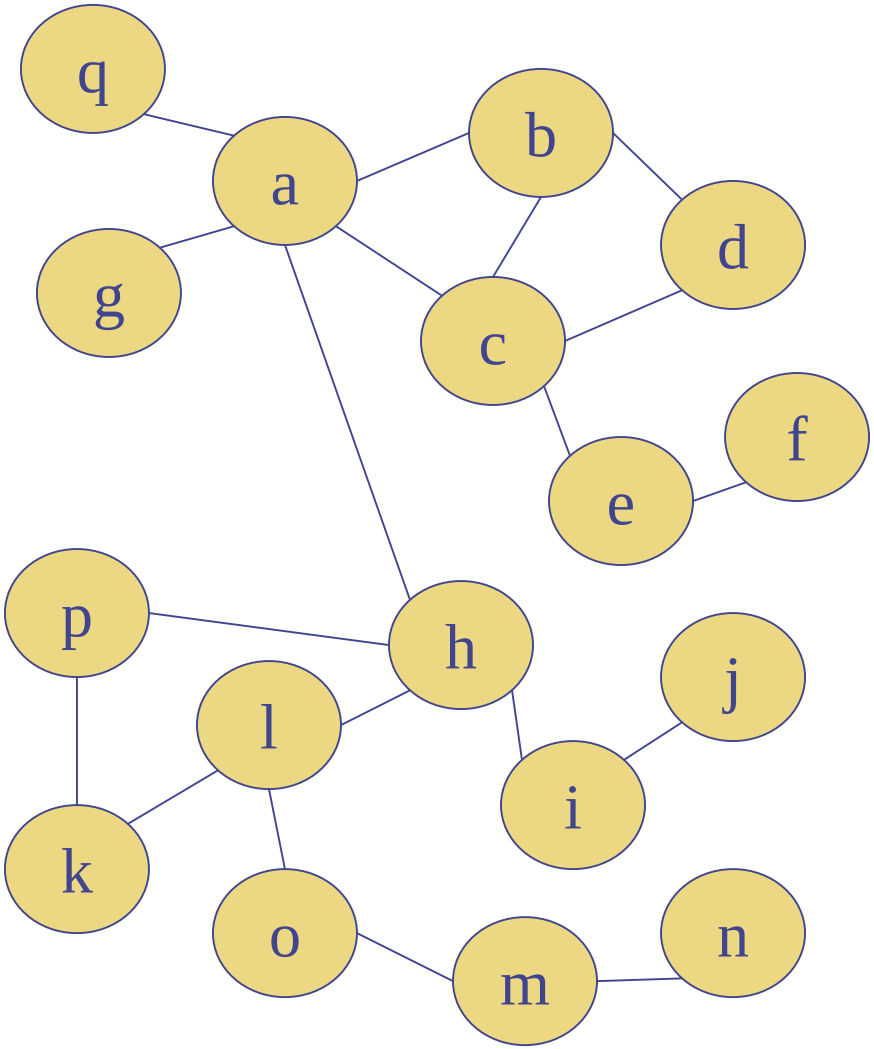


Hypergraph *H(Q)*

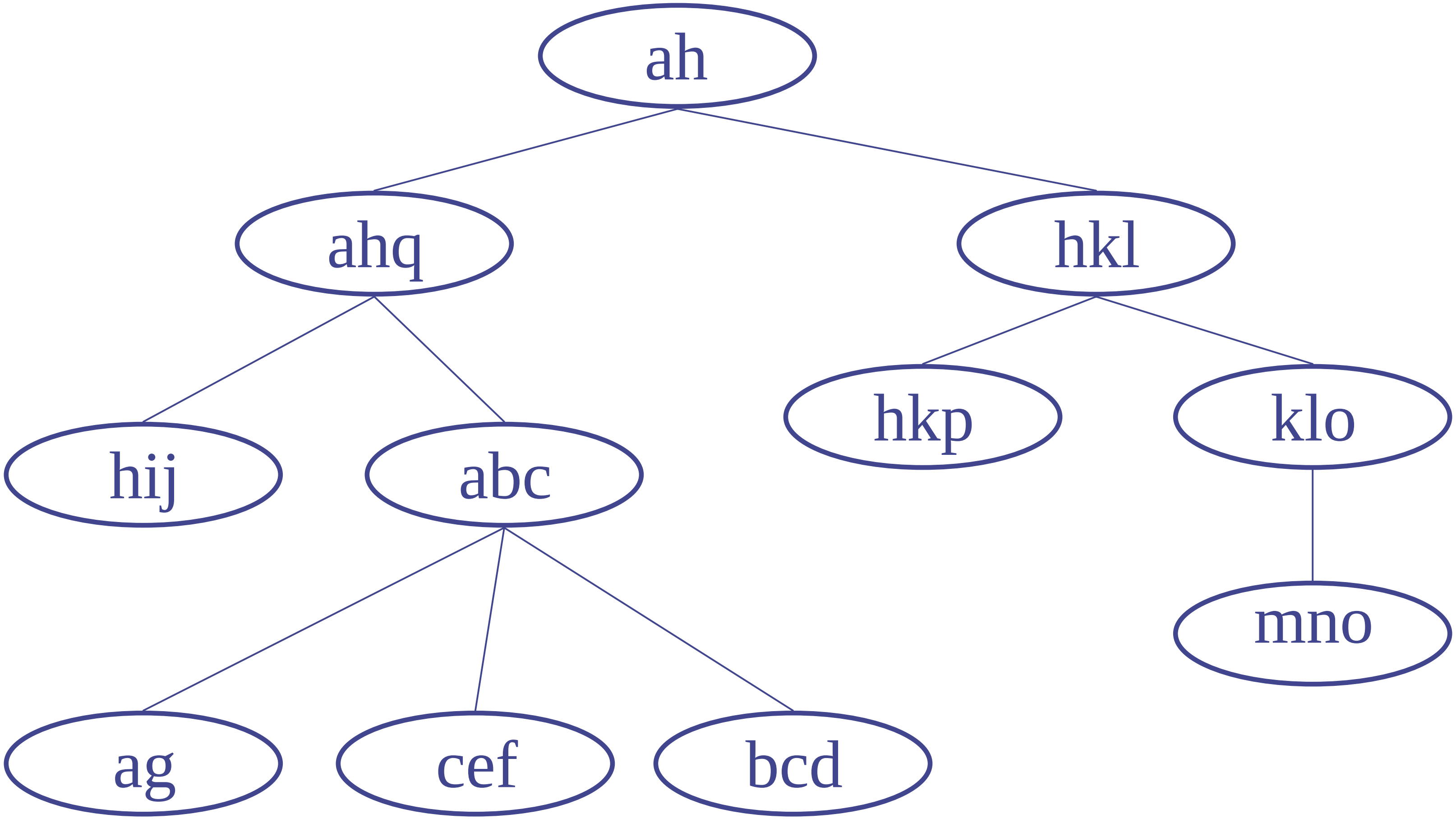


Primal graph *G(Q)*

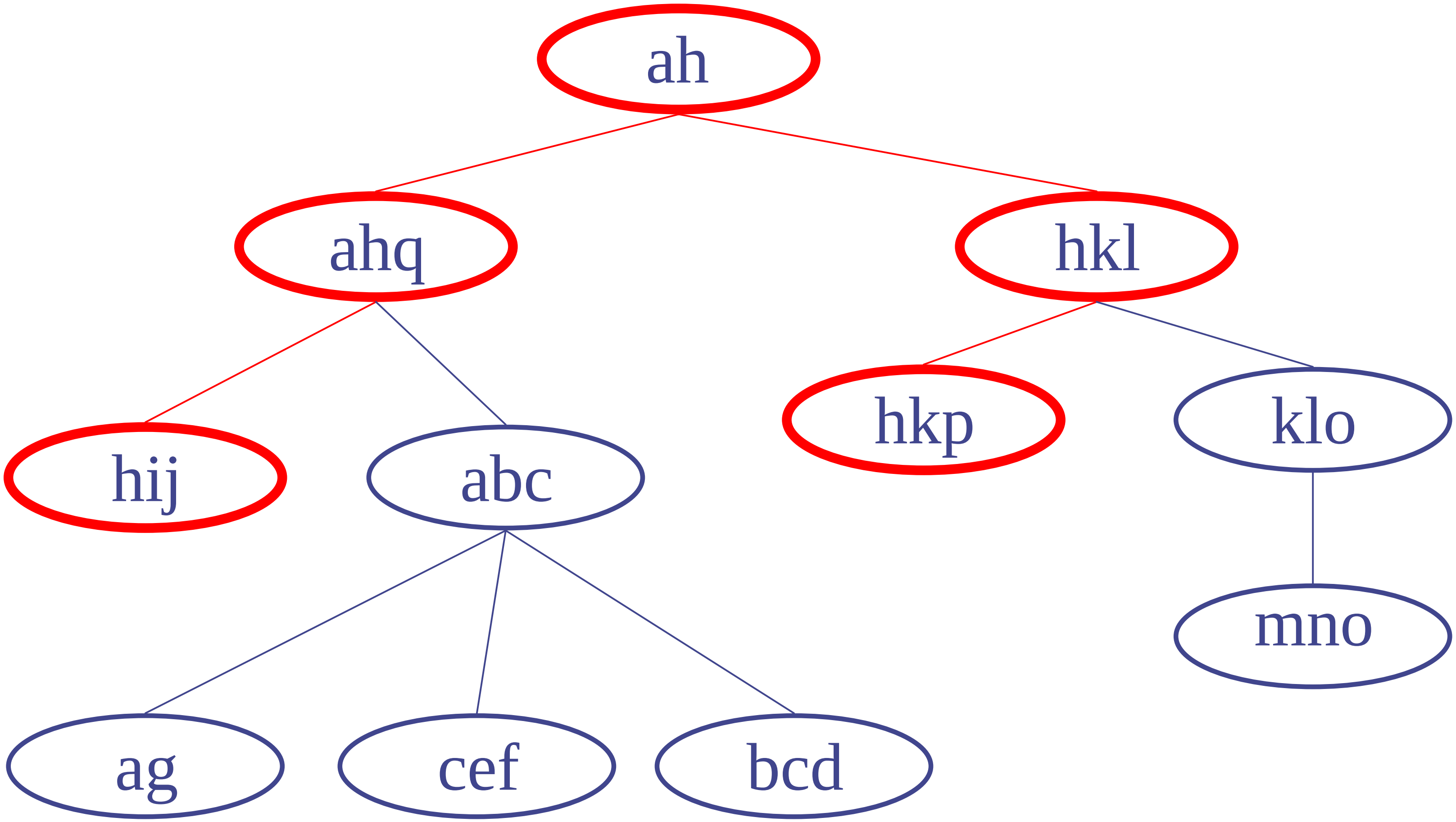
Example: a cyclic graph



A tree decomposition of width 2



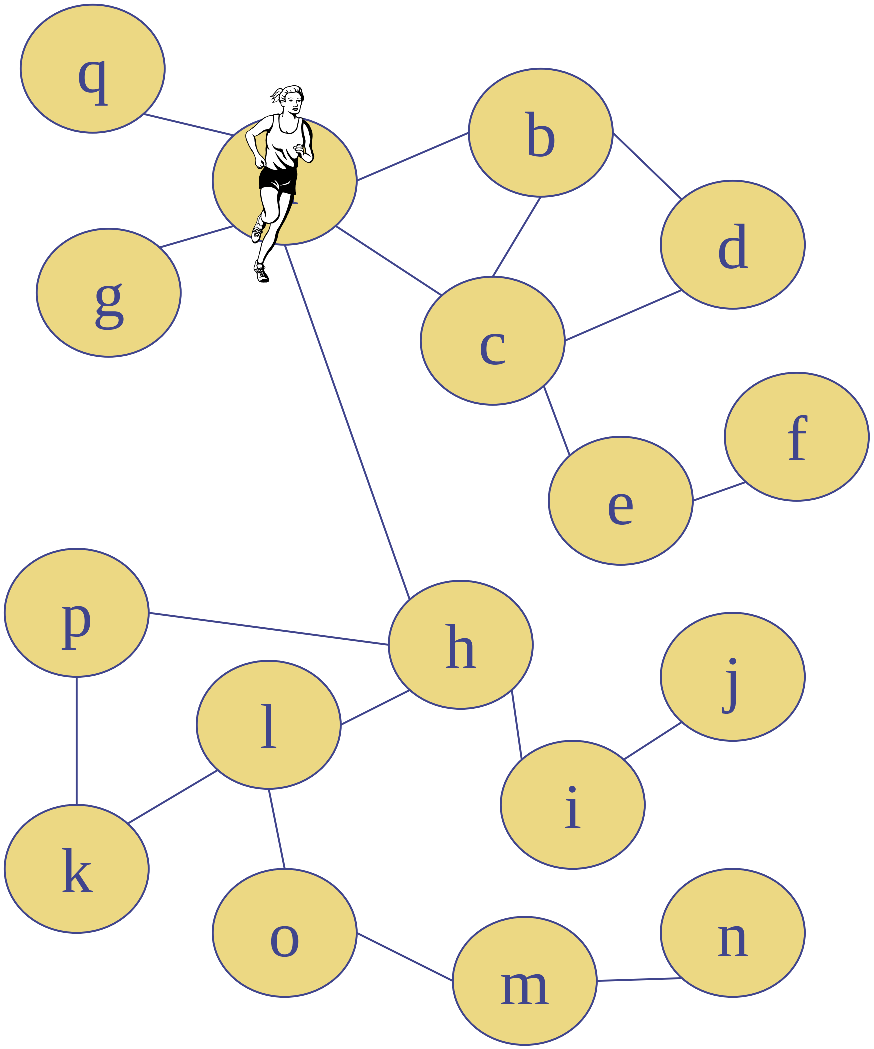
Connectedness condition for *h*



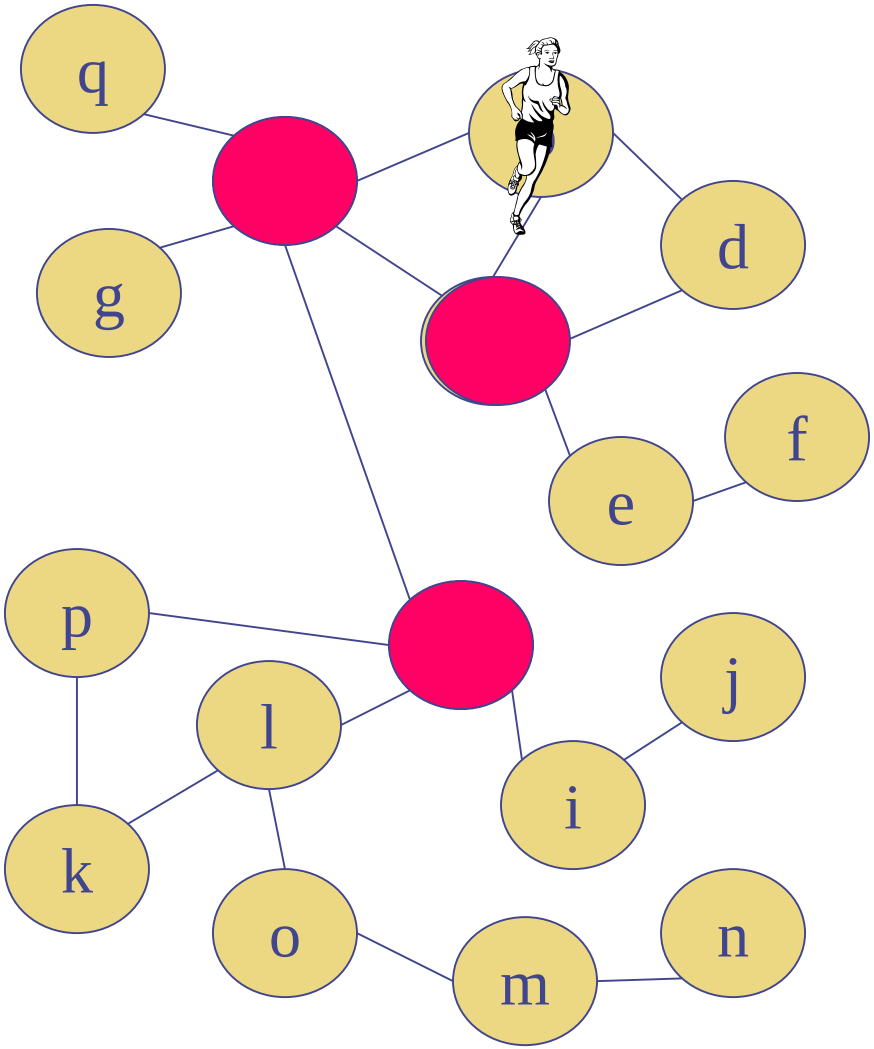
Game characterization of Treewidth

- 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

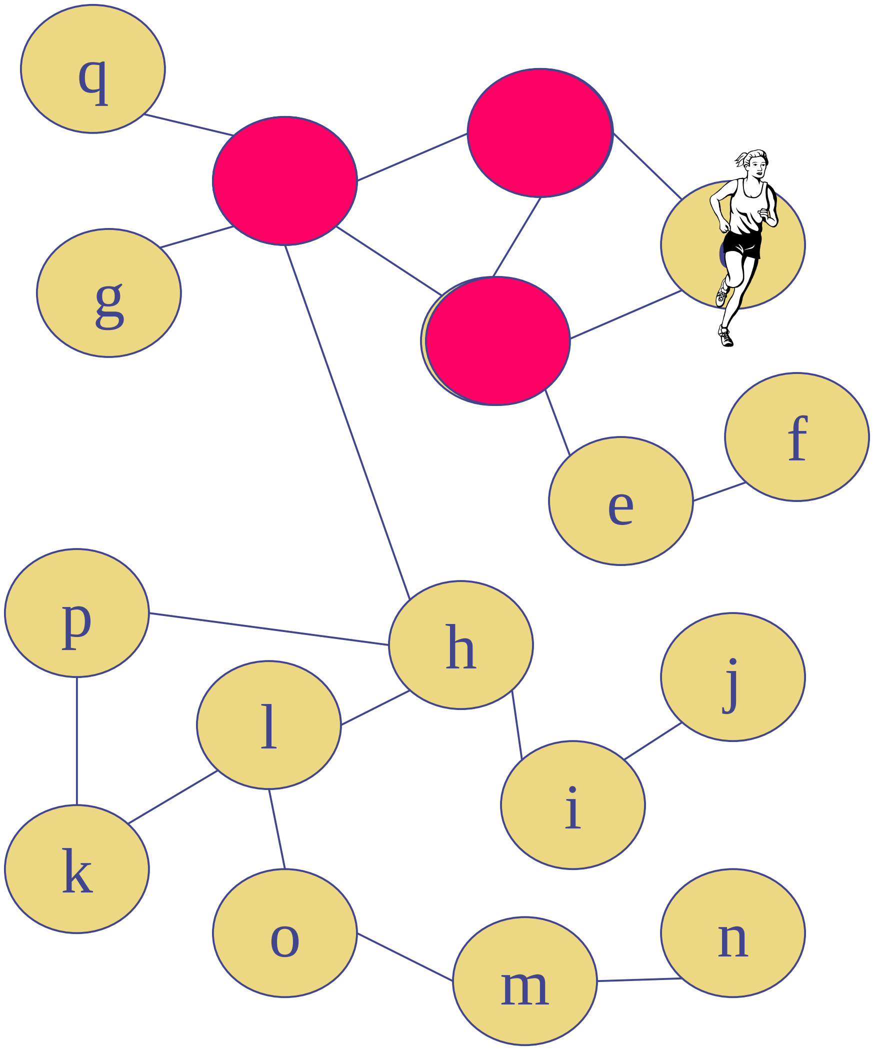
Playing the game



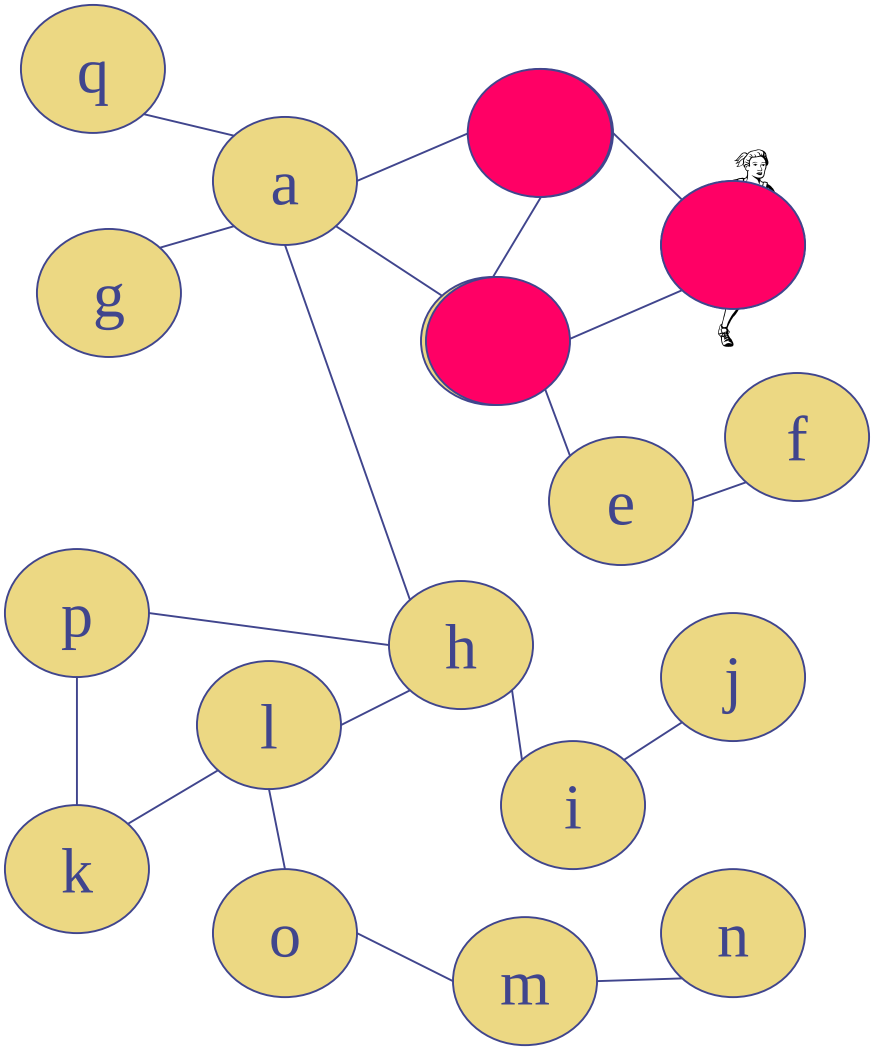
Playing the game



Playing the game



Playing the game



Logical characterization of Treewidth

Logic L : FO based on $\exists, \wedge$ ($\neg, \vee, \forall$ disallowed)

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

$$\boxed{\mathrm{TW}[k] = \mathsf{L}^{k+1}}$$ (Kolaitis & Vardi '98)

L^{k+1} : L with at most $k+1$ vars.

Logical characterization of Treewidth

A generalization:

$$\text{NRS-DATALOG-TW}[k] = \text{FO}^{k+1}$$

(Flum, Frick, and Grohe ‘01)

When is the evaluation of conjunctive queries tractable?

In case they are characterized by graphs e.g., through primal or Gaifman graphs:

$\mathcal{G}$: class of graphs

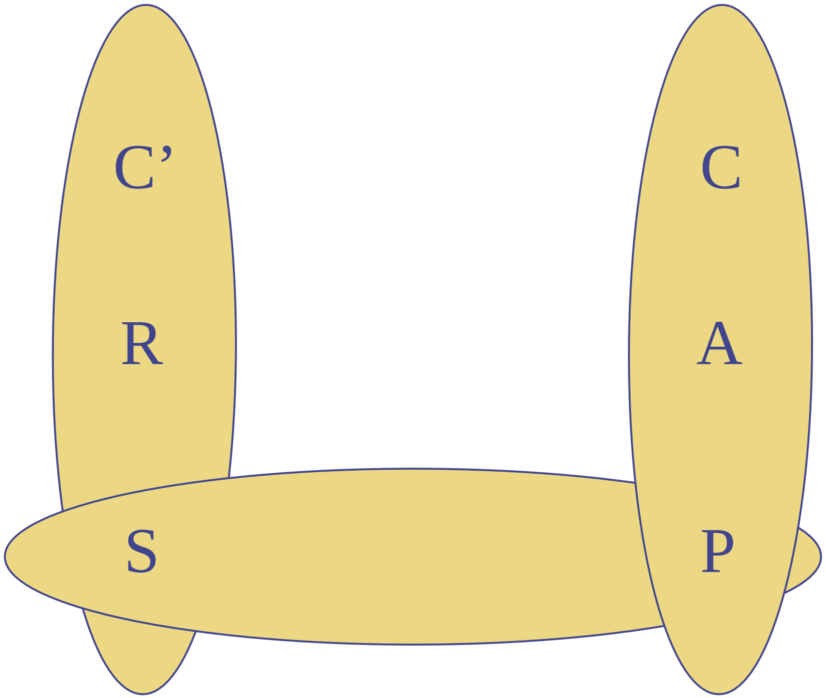
$Q(\mathcal{G})$: all queries characterized by graphs in $\mathcal{G}$

$Q(\mathcal{G})$ tractable iff $\mathcal{G}$ has bounded treewidth

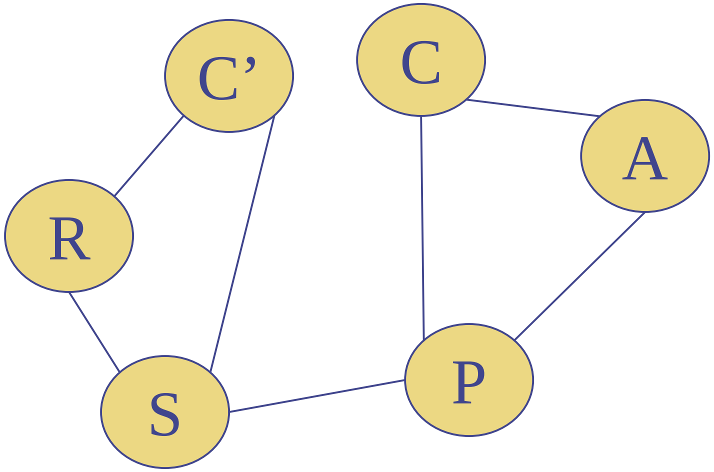
unless P = W[1] or other collapses occur

(Grohe, Schwentick, and Segoufin, '01)

Hypergraphs vs Graphs (1)

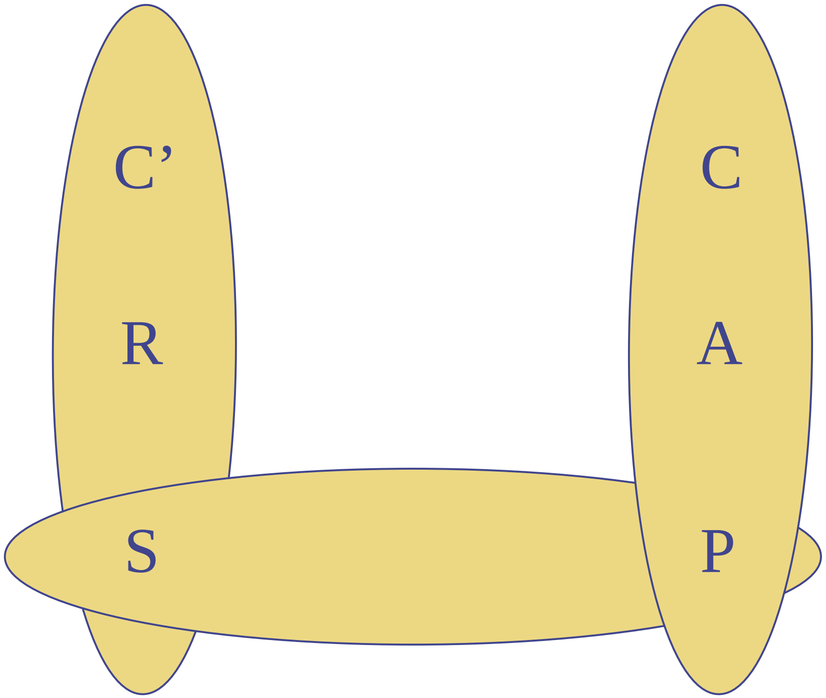


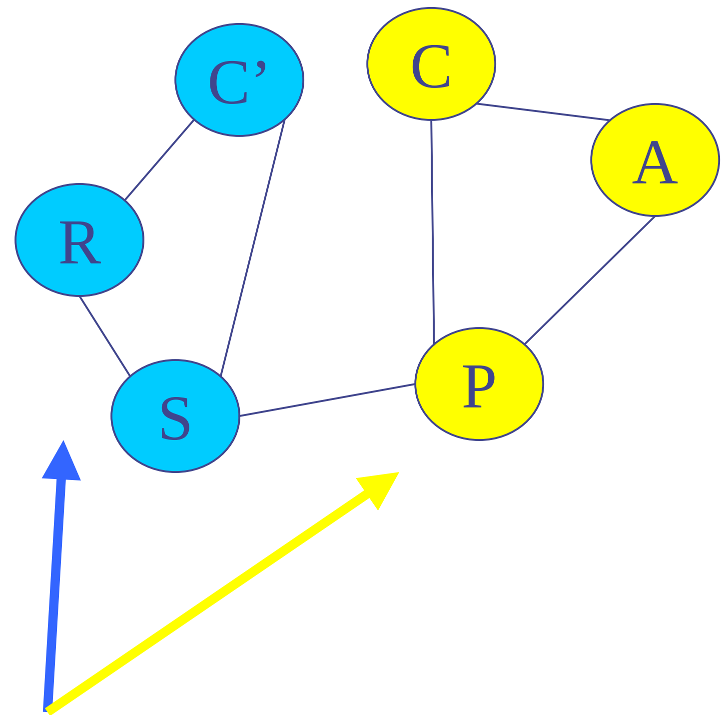
An acyclic hypergraph



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:

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

is acyclic, obviously polynomial,

but has treewidth *n-1*

Beyond treewidth

- **Bounded Degree of Cyclicity**

  (Gyssens & Paredaens ’84)

- **Bounded Query width**

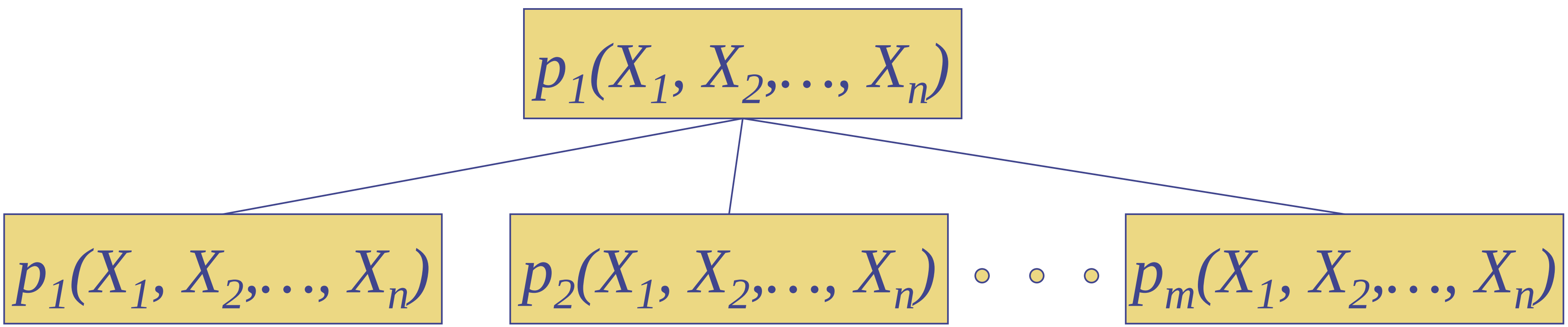
  (Chekuri & Rajaraman ’97)

Group together query atoms (hyperedges) instead of variables

Query Decomposition

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

Query width = 1

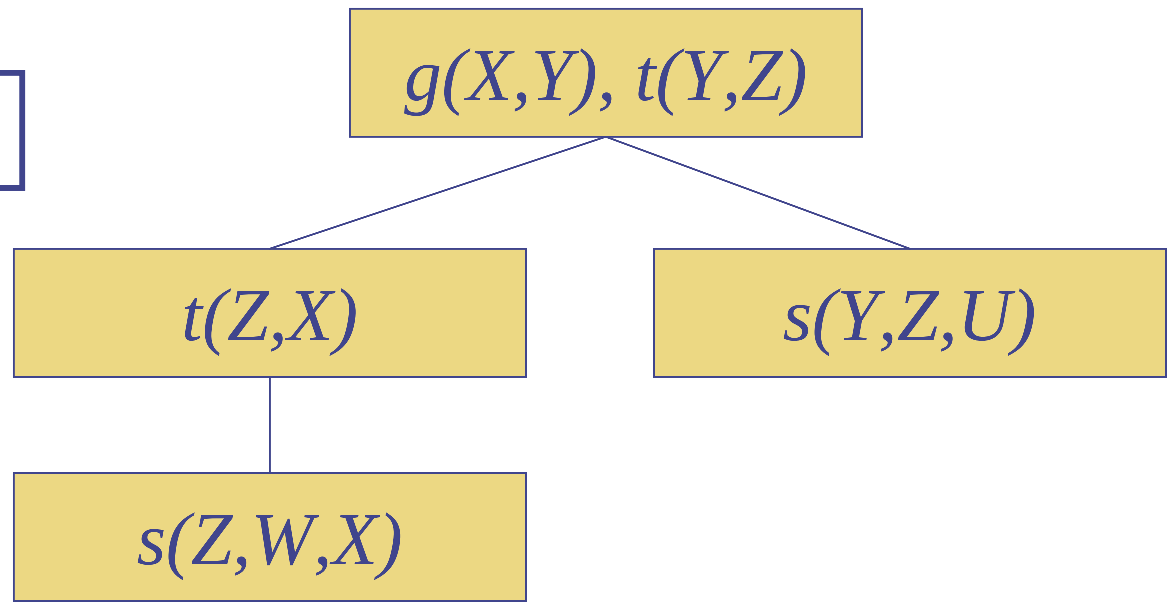


- Every atom appears in some node
- Connectedness conditions for variables and atoms

Decomposition of cyclic queries

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

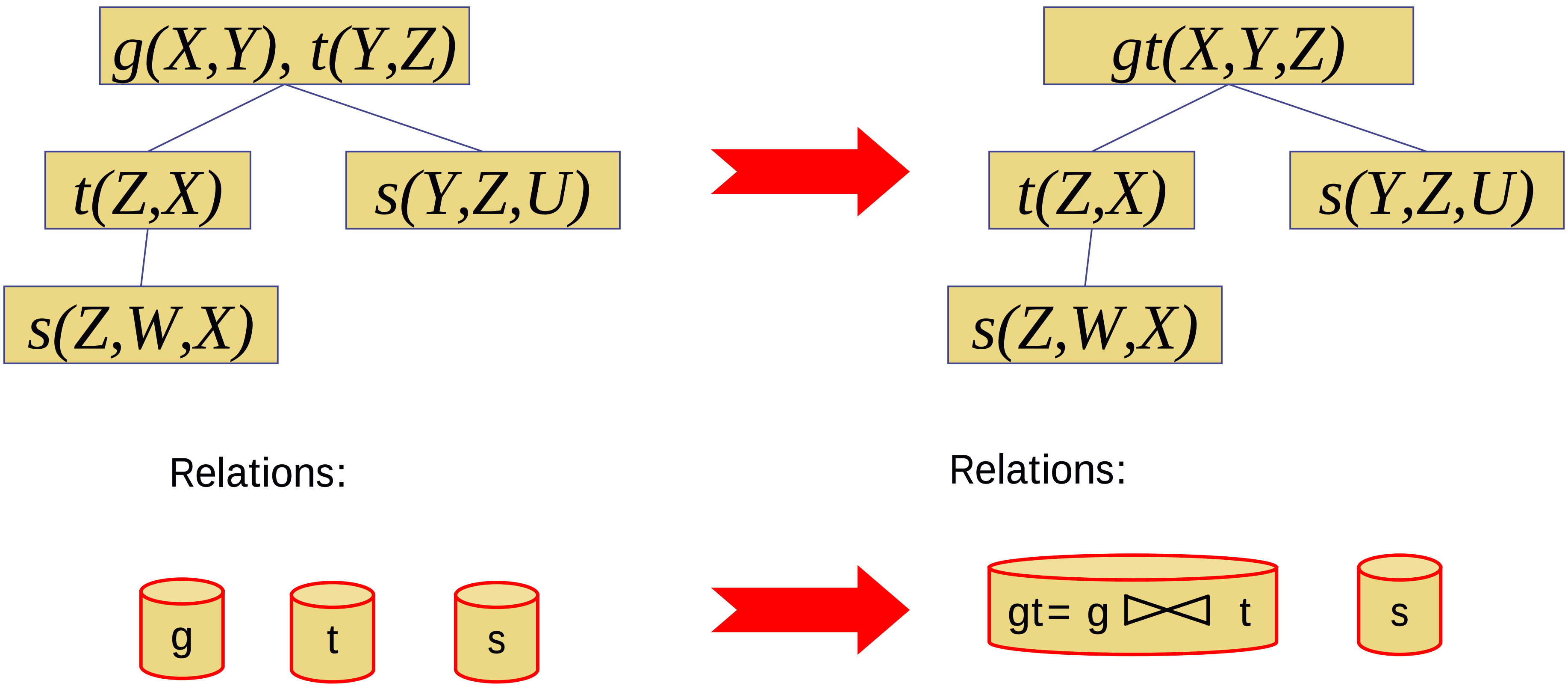
Query width = 2



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

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



Open 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

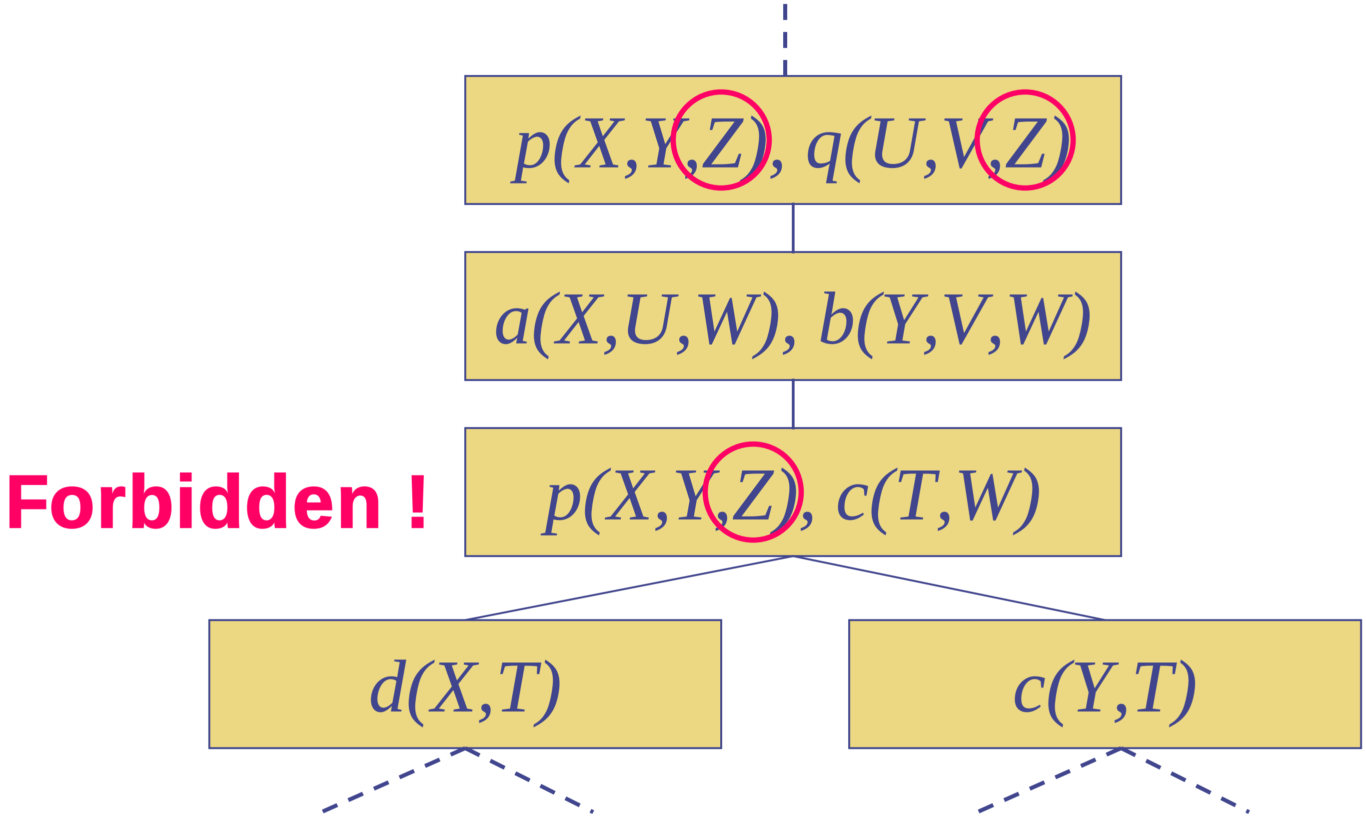
A negative answer (G.L.S. '99)

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

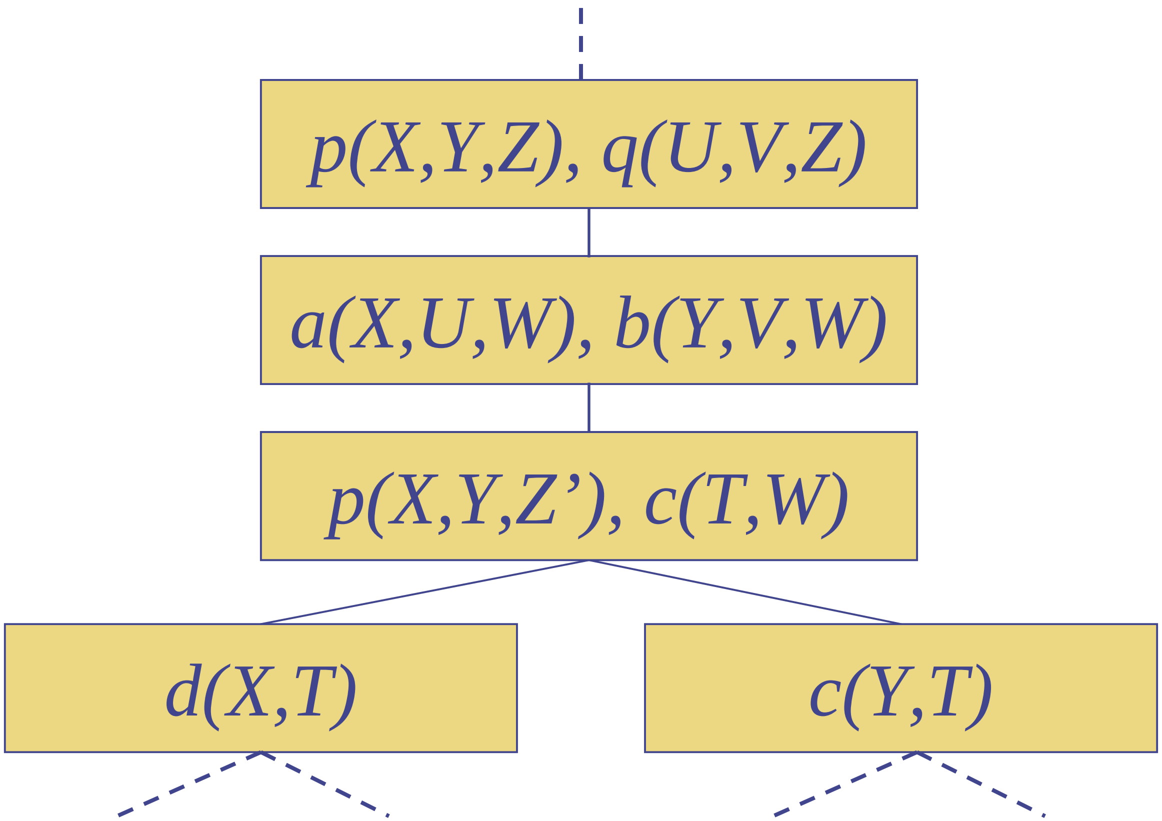
NP-hardness id due to an overly strong condition in the definition of query decomposition



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

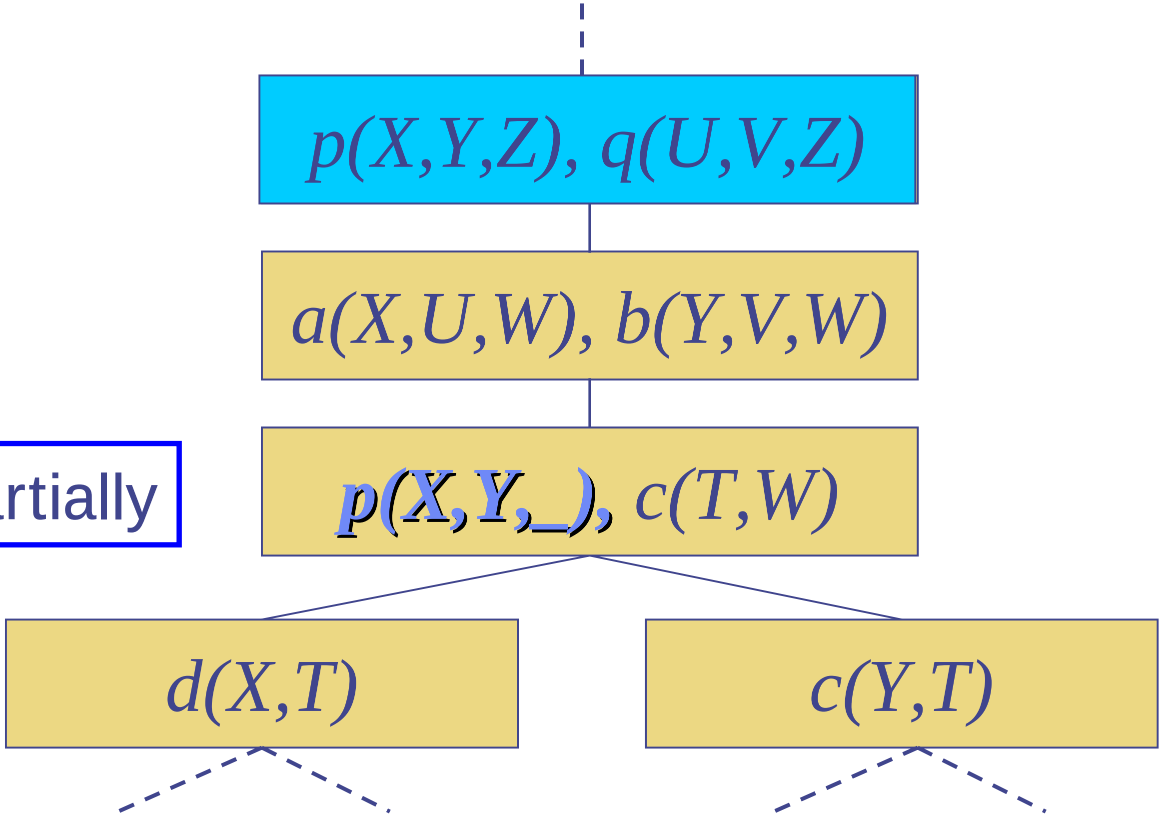
Query atoms can be used “partially” as long as the full atom appears somewhere else

More liberal than query decomposition

Grouping and Reusing Atoms

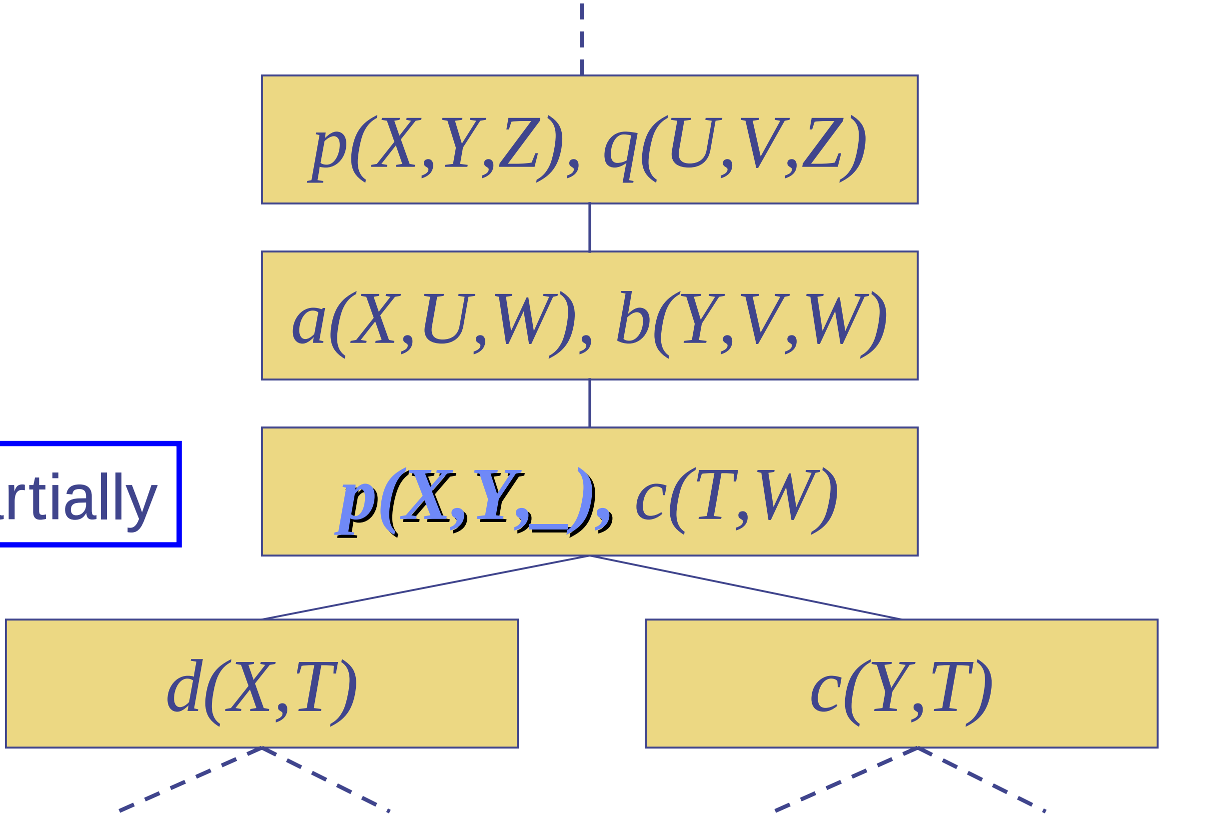
We group atoms

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

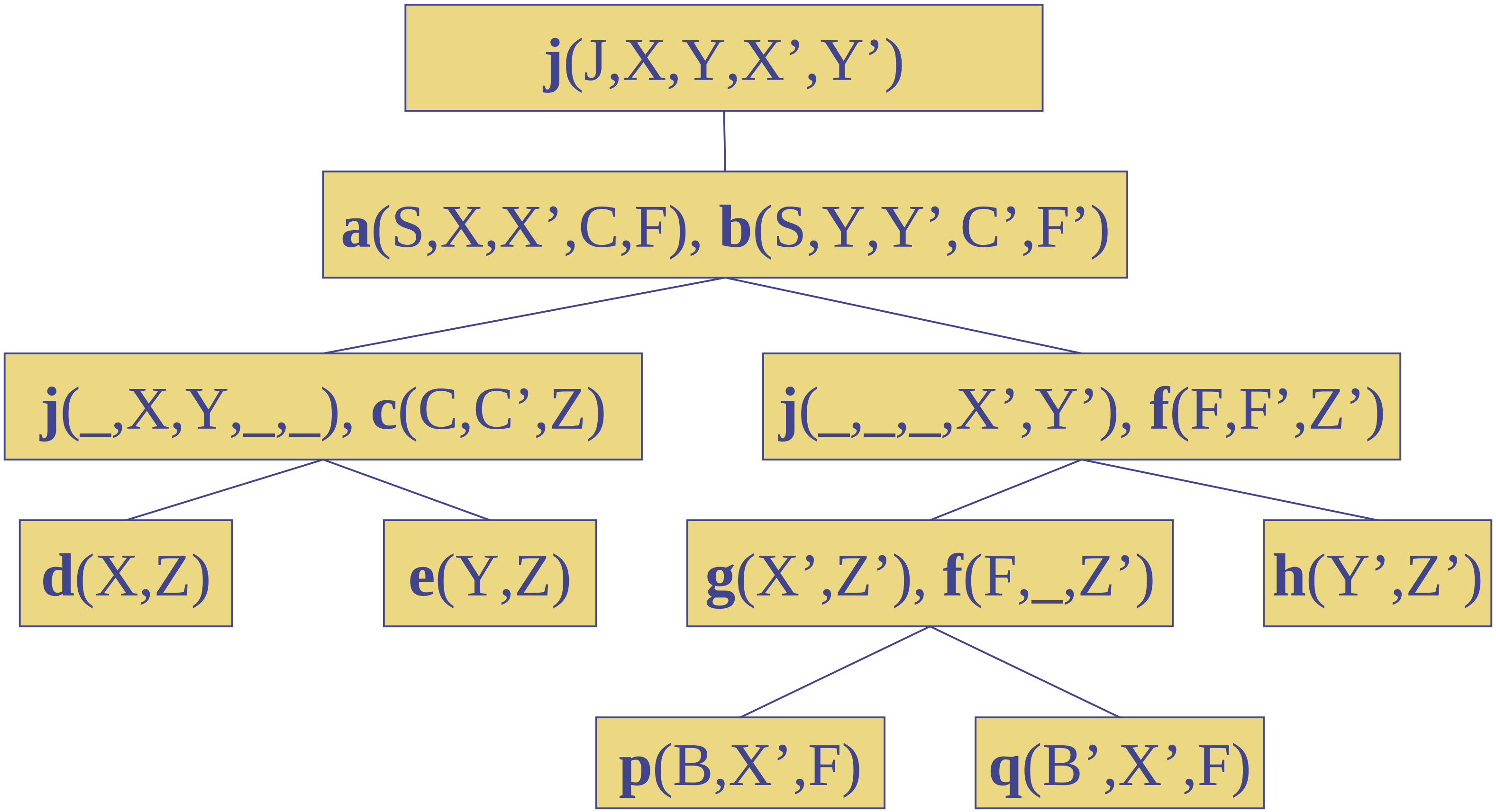


Reusing atoms

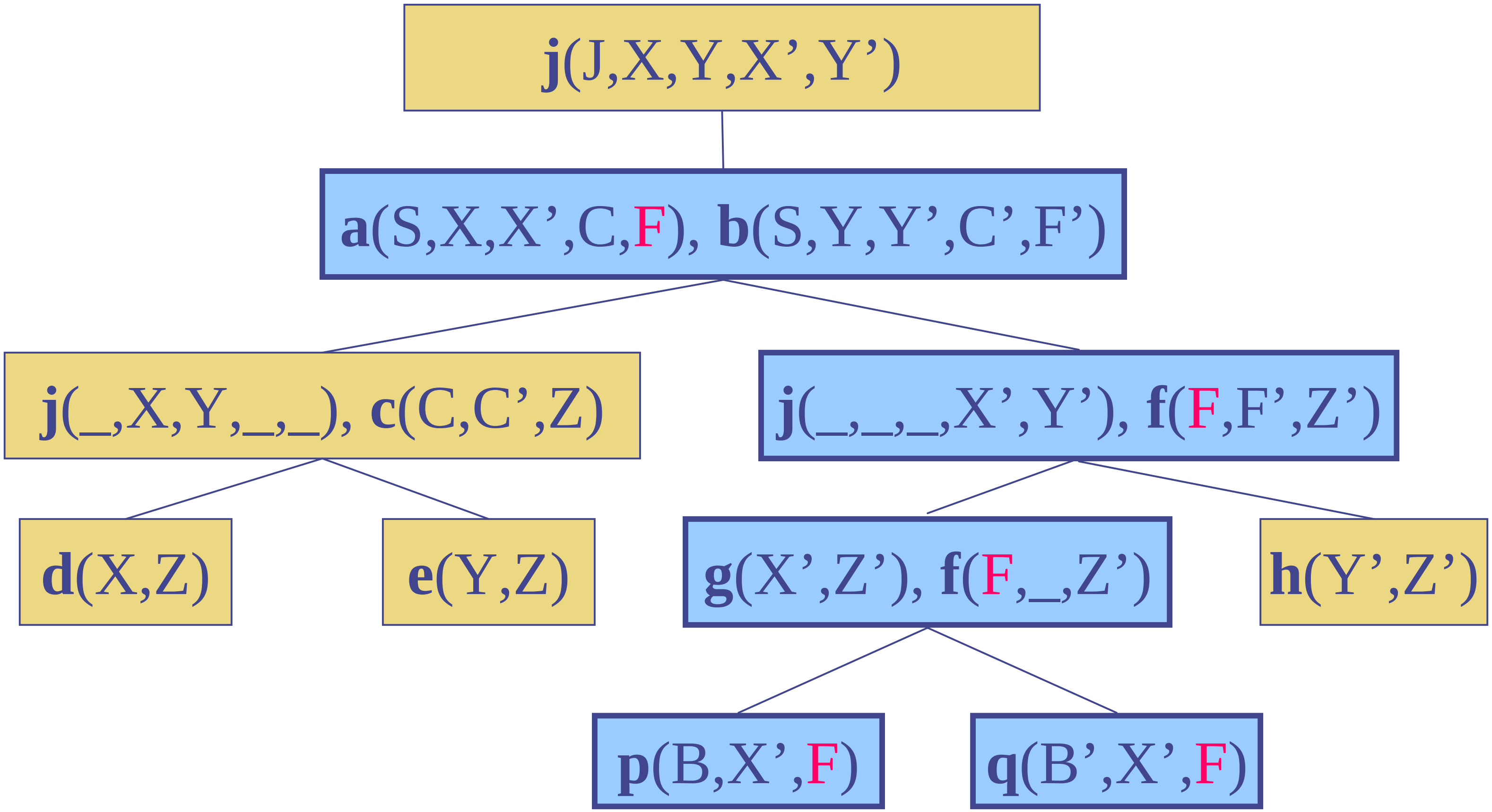
We use *p(X,Y,Z)* partially



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

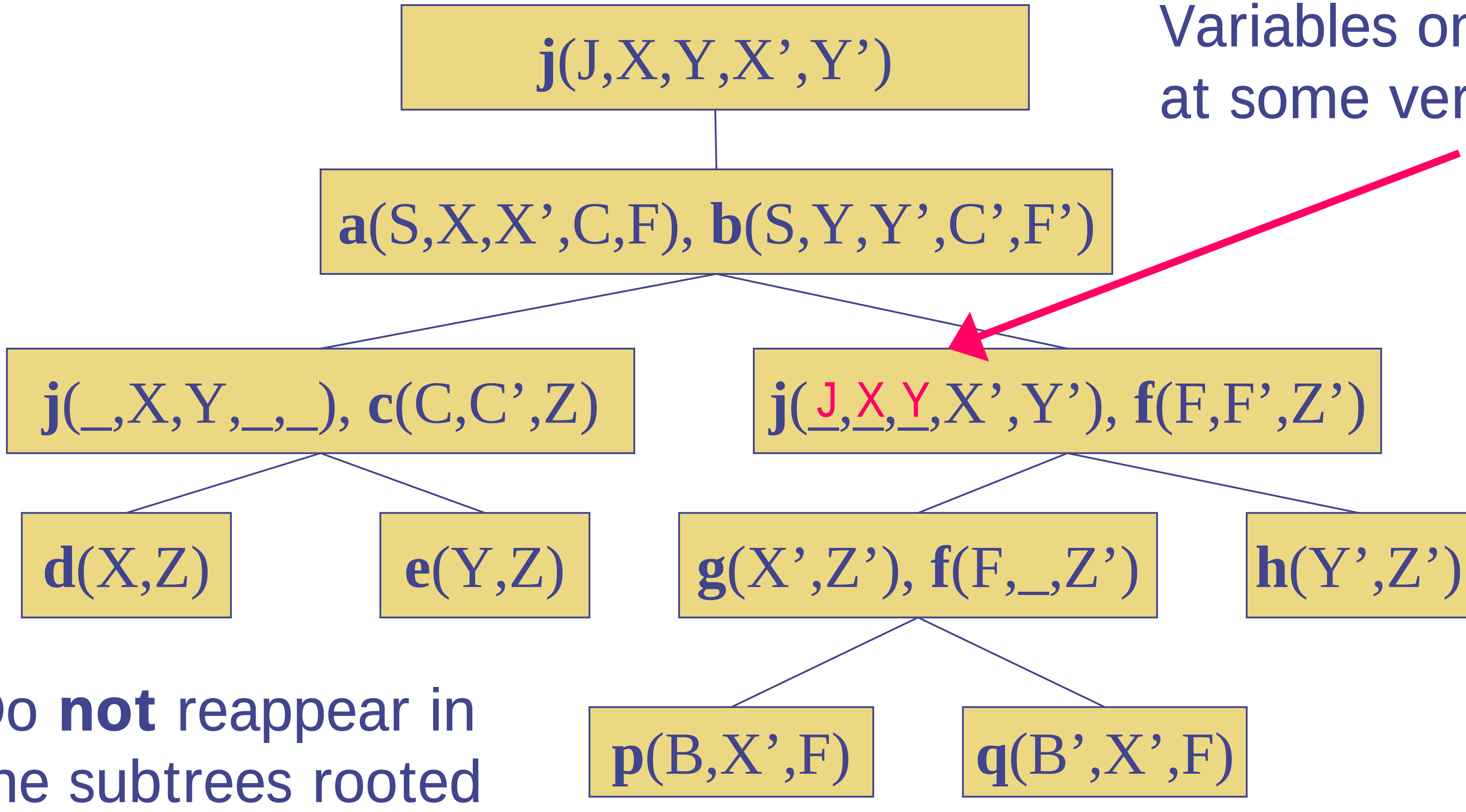


Connectedness Condition



Special Condition

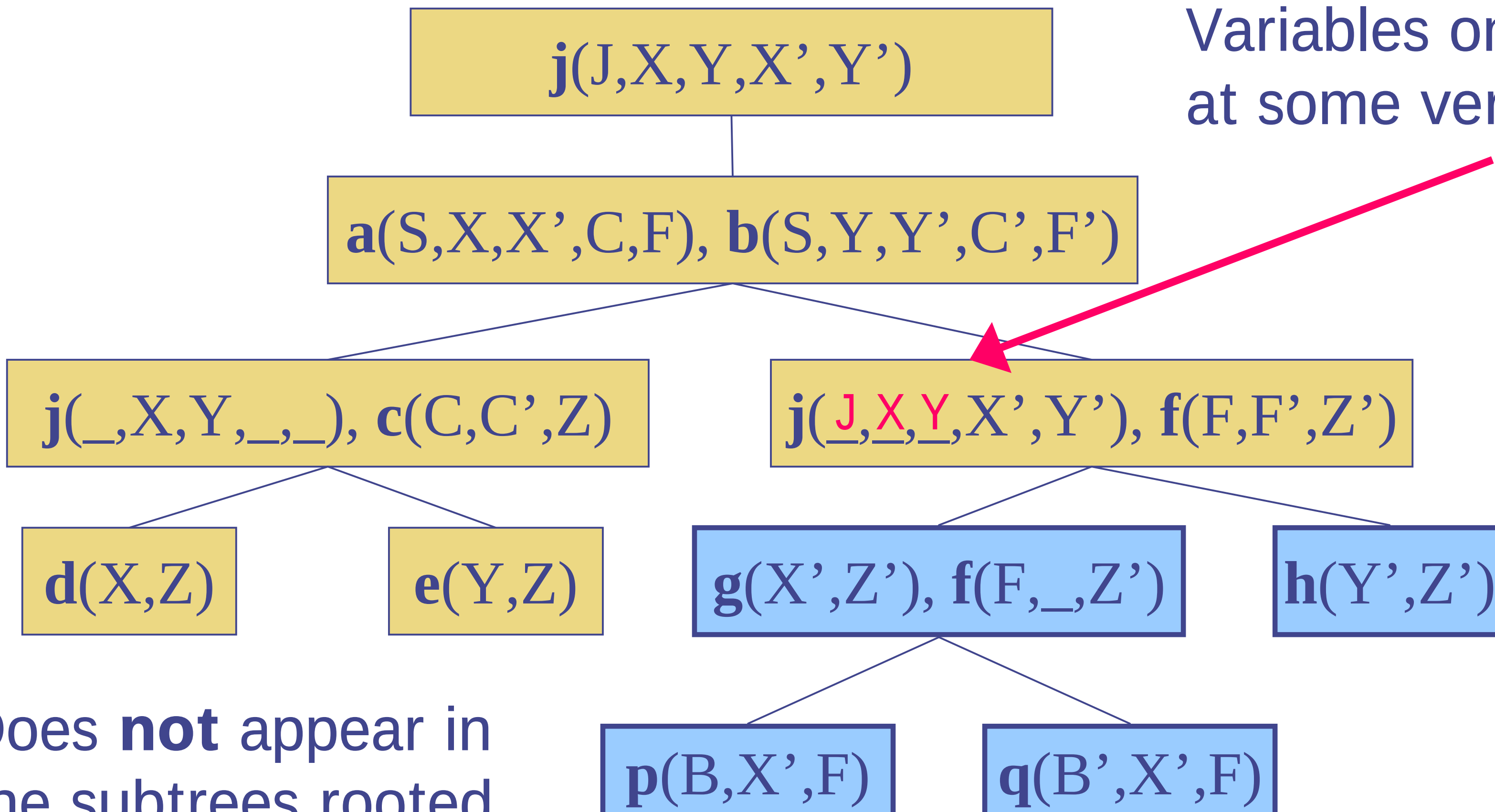
Variables omitted at some vertex *v*



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

Special Condition

Variables omitted at some vertex *v*



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

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)

Evaluating queries having bounded hypertree width

k fixed

Given:

a database db

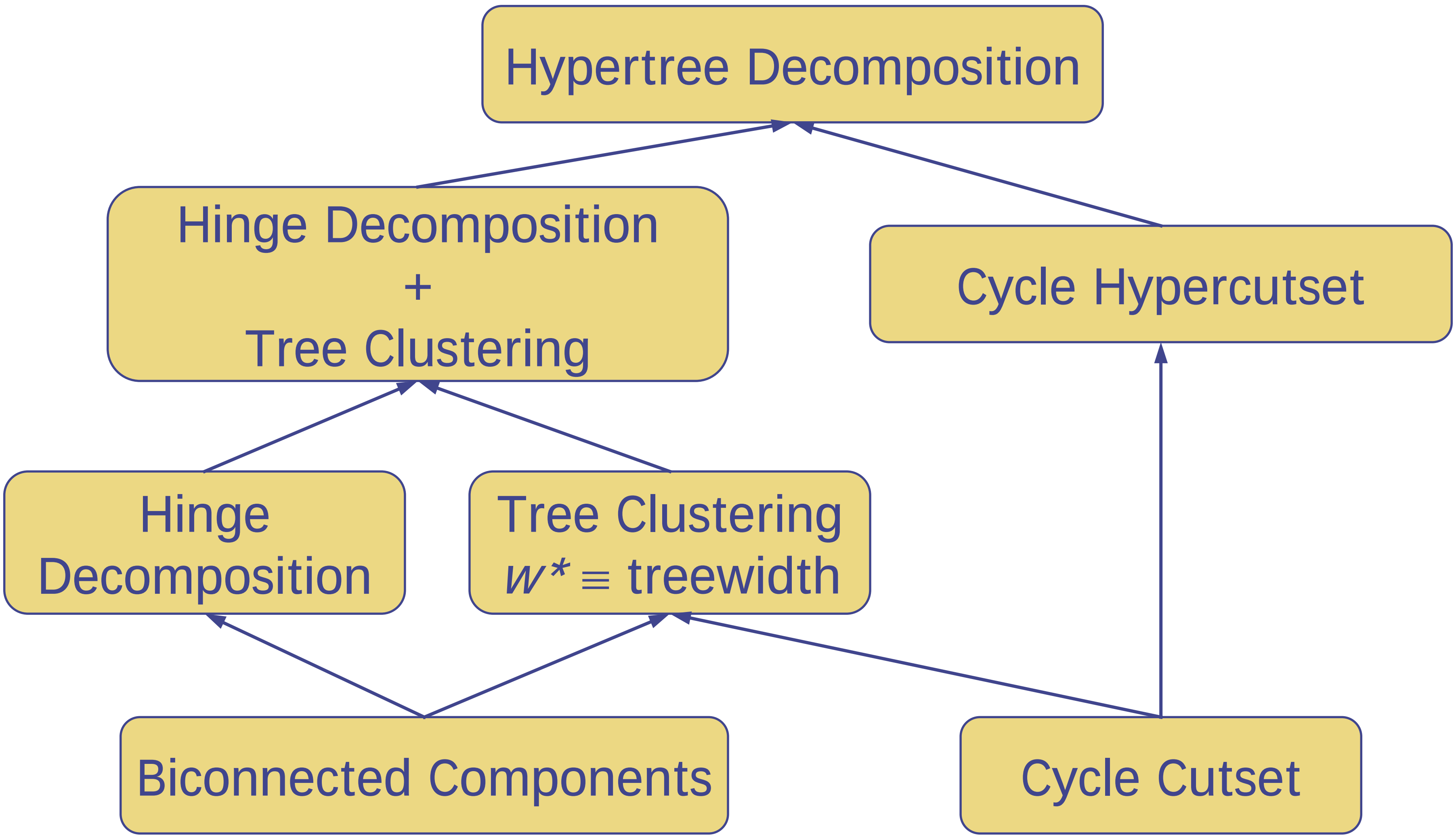
a query Q over db such that $hw(Q) \leq k$

a width k hypertree decomposition of Q

- Deciding whether $Q(db)$ is not empty is in $O(n^{k+1} \log n)$ and complete for LOGCFL

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

Comparison results



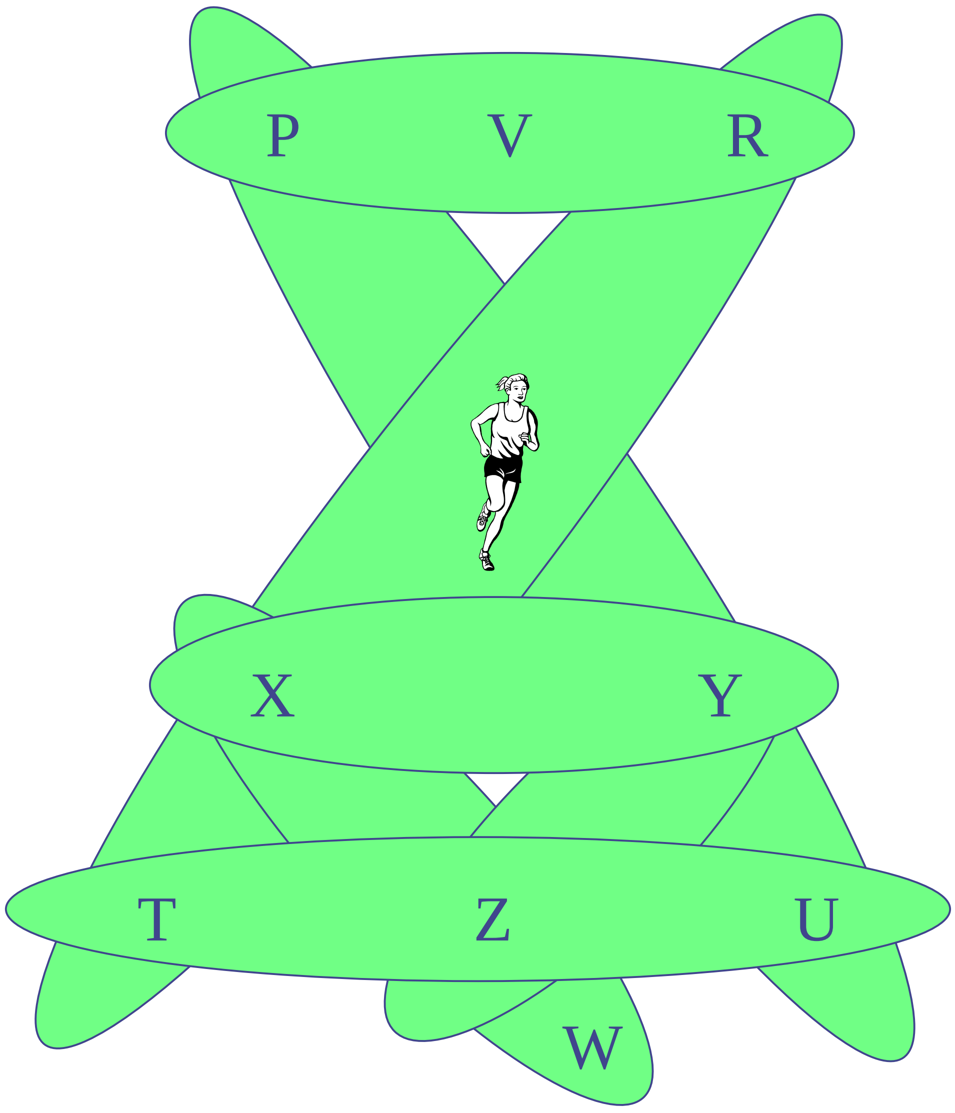
Game characterization: Robber and Marshals

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

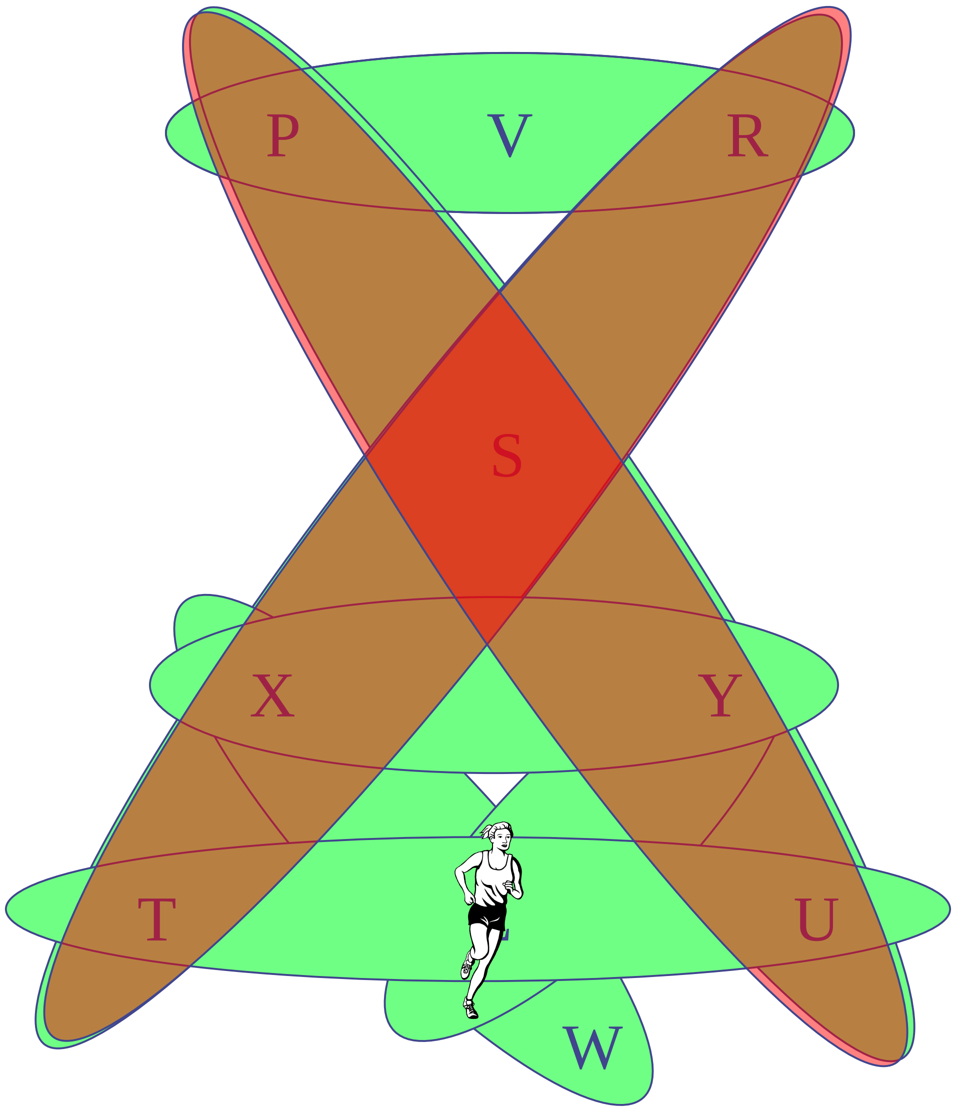
Robbers and Marshals: the rules

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

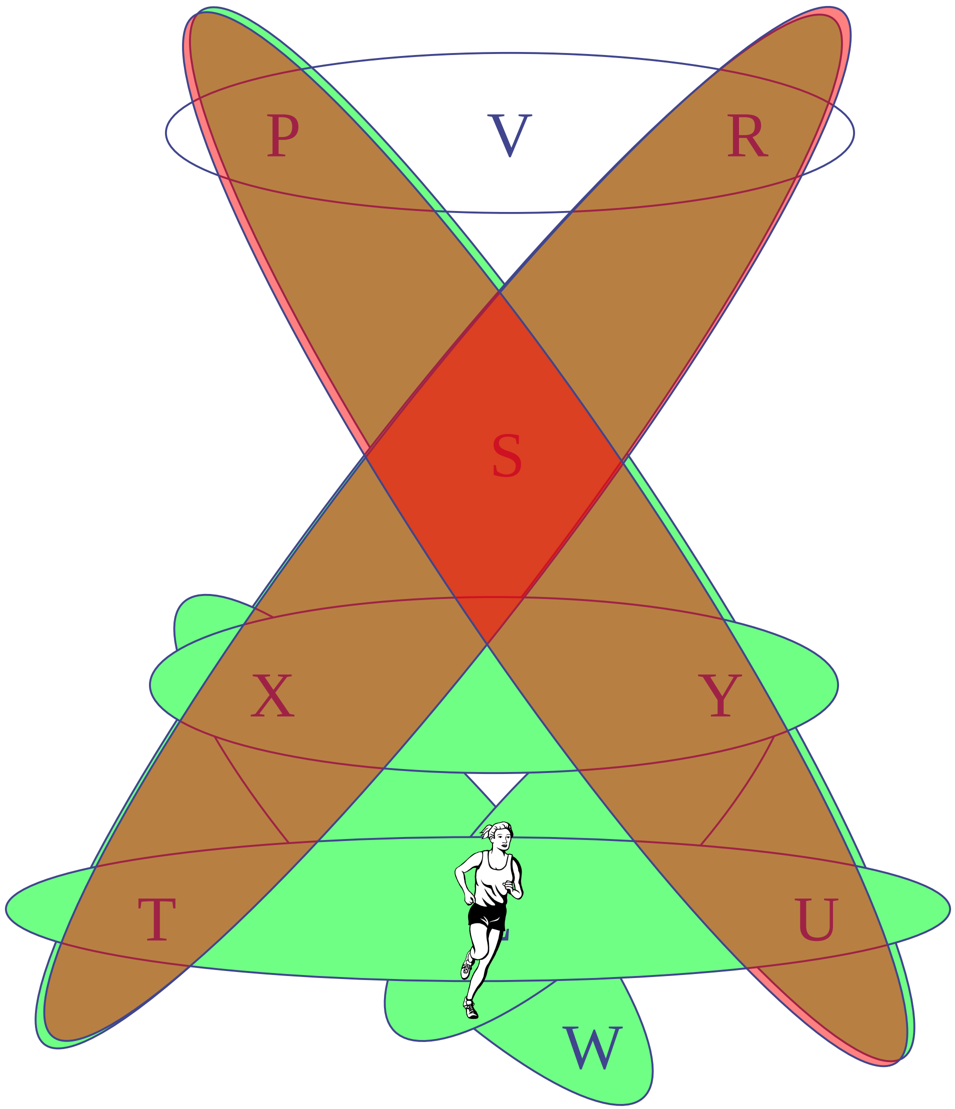
Step 0: the empty hypergraph



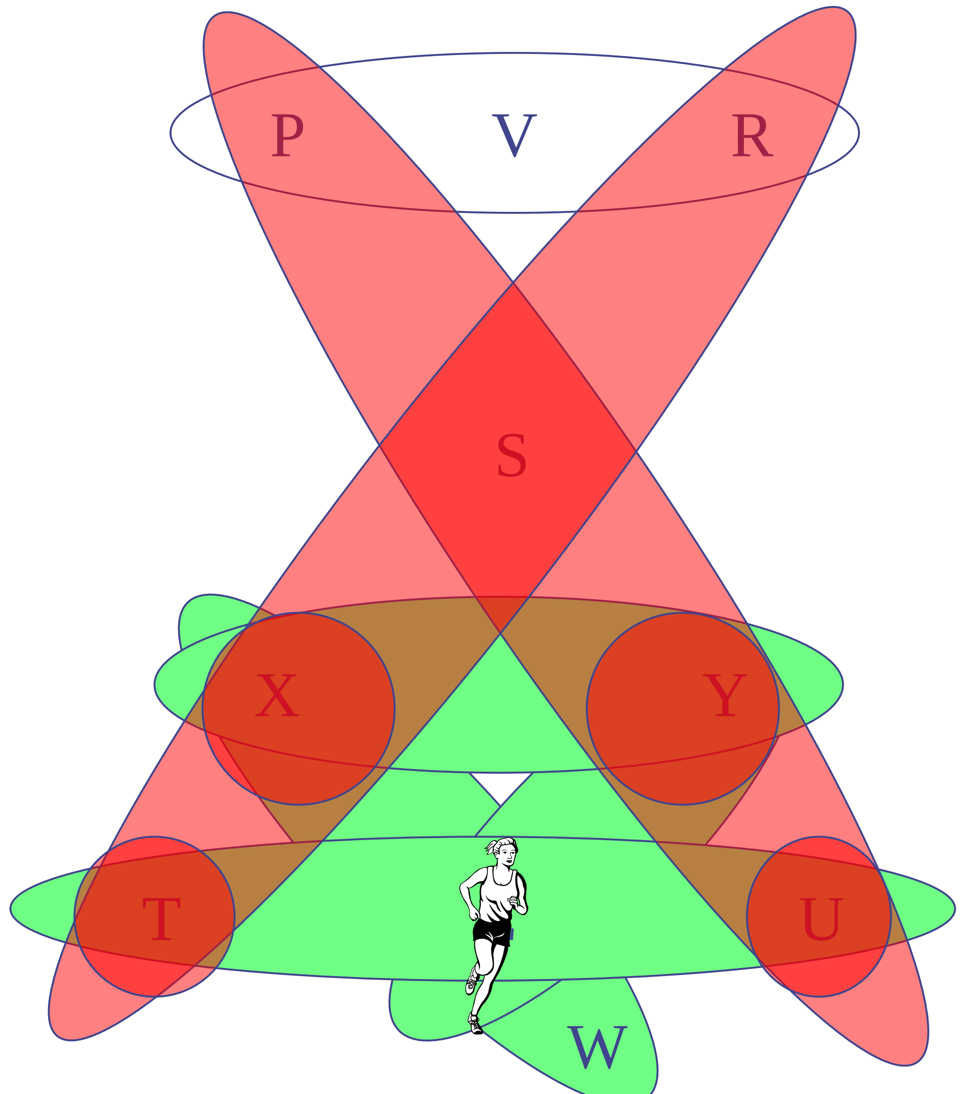
Step 1: first move of the marshals



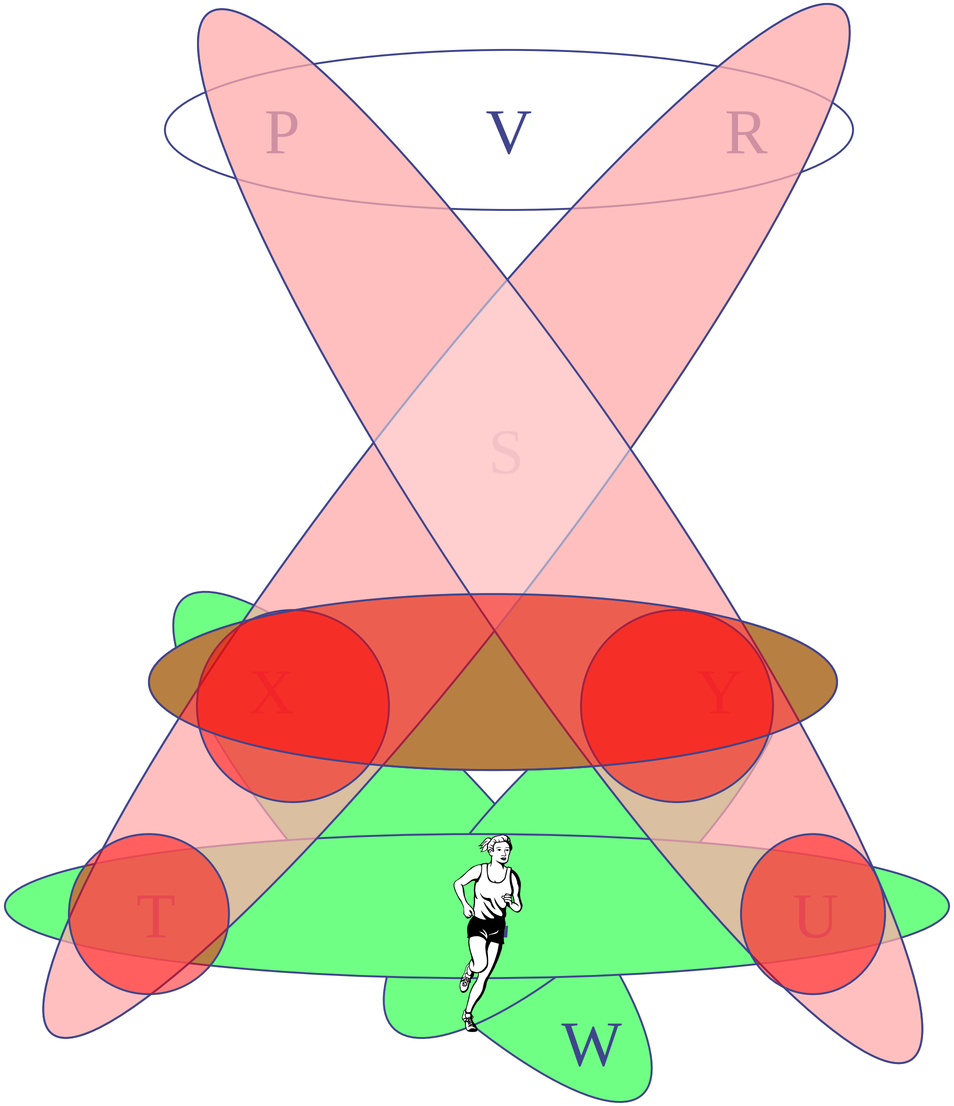
Step 1: first move of the marshals



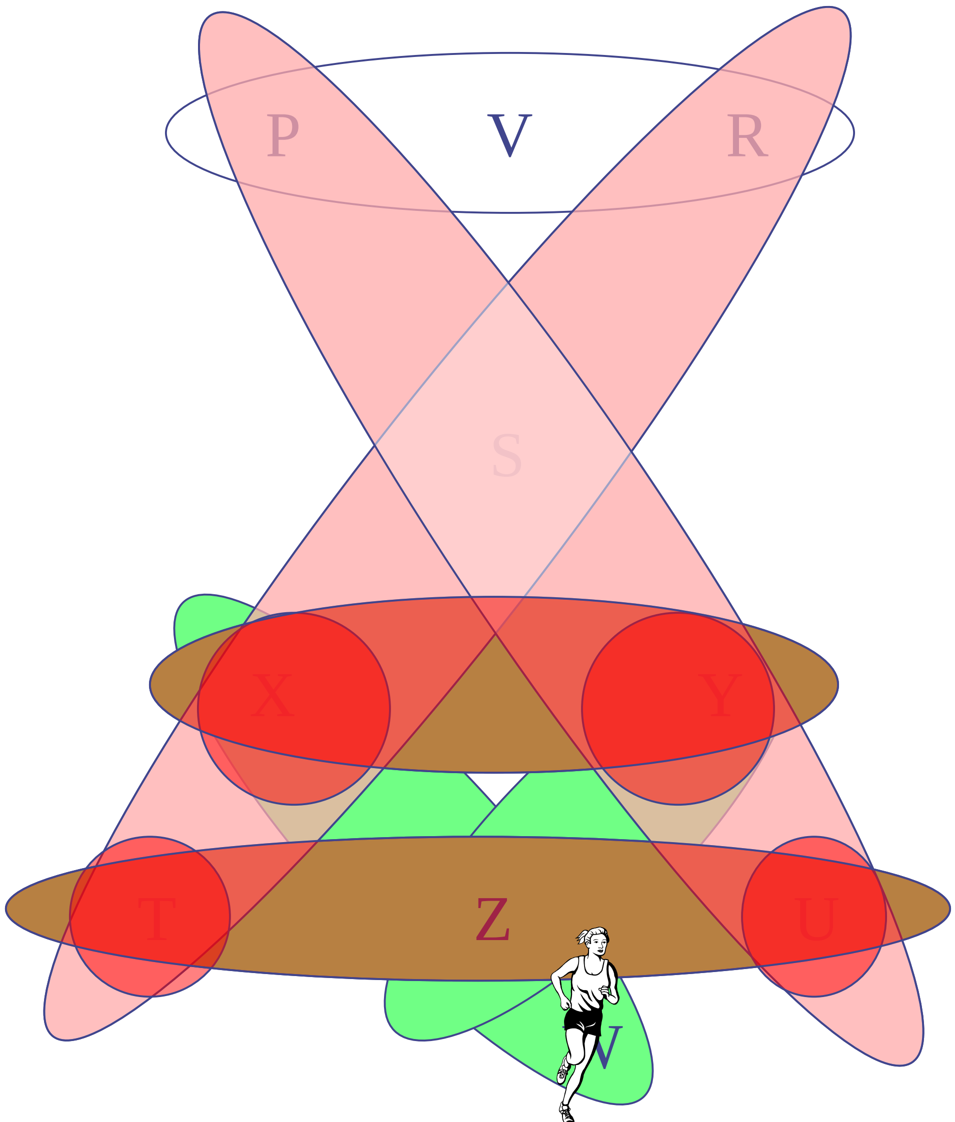
Step 2a: shrinking the space



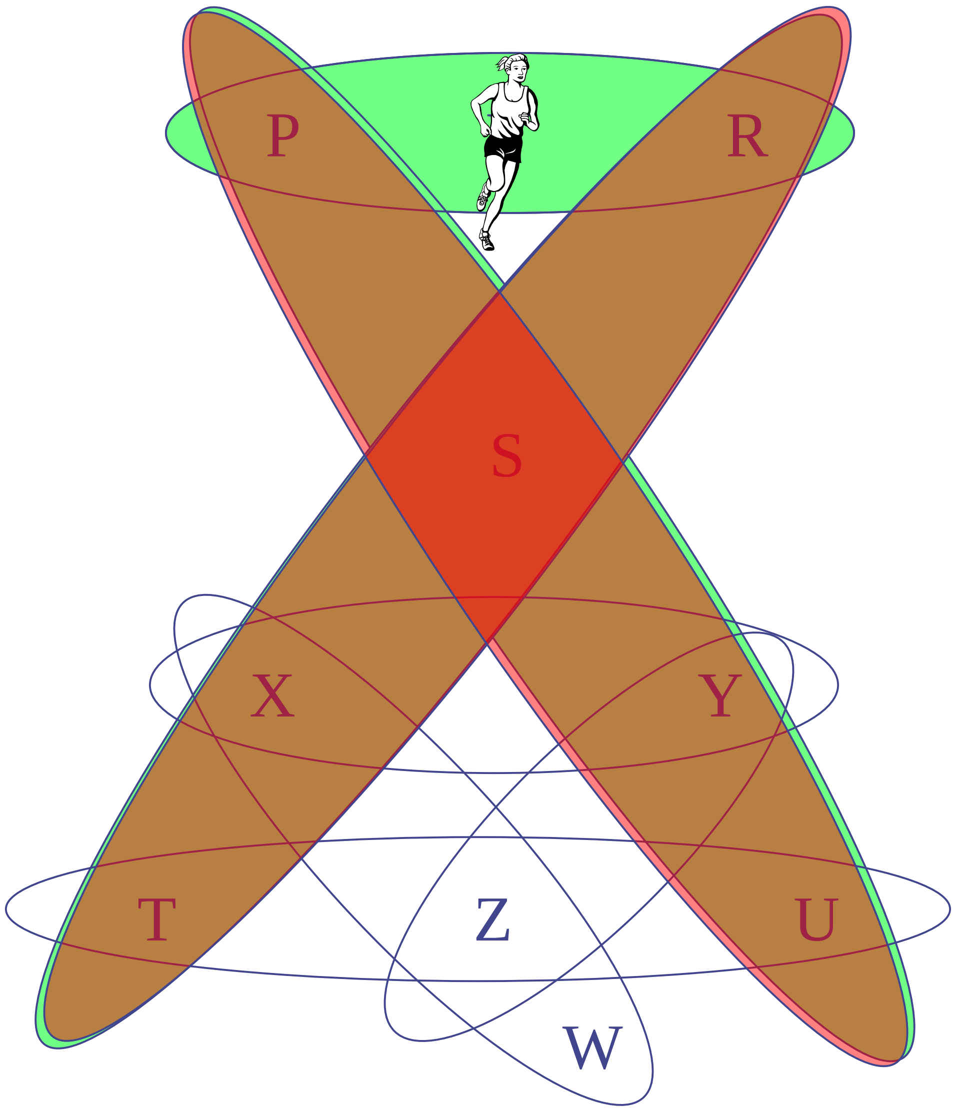
Step 2a: shrinking the space



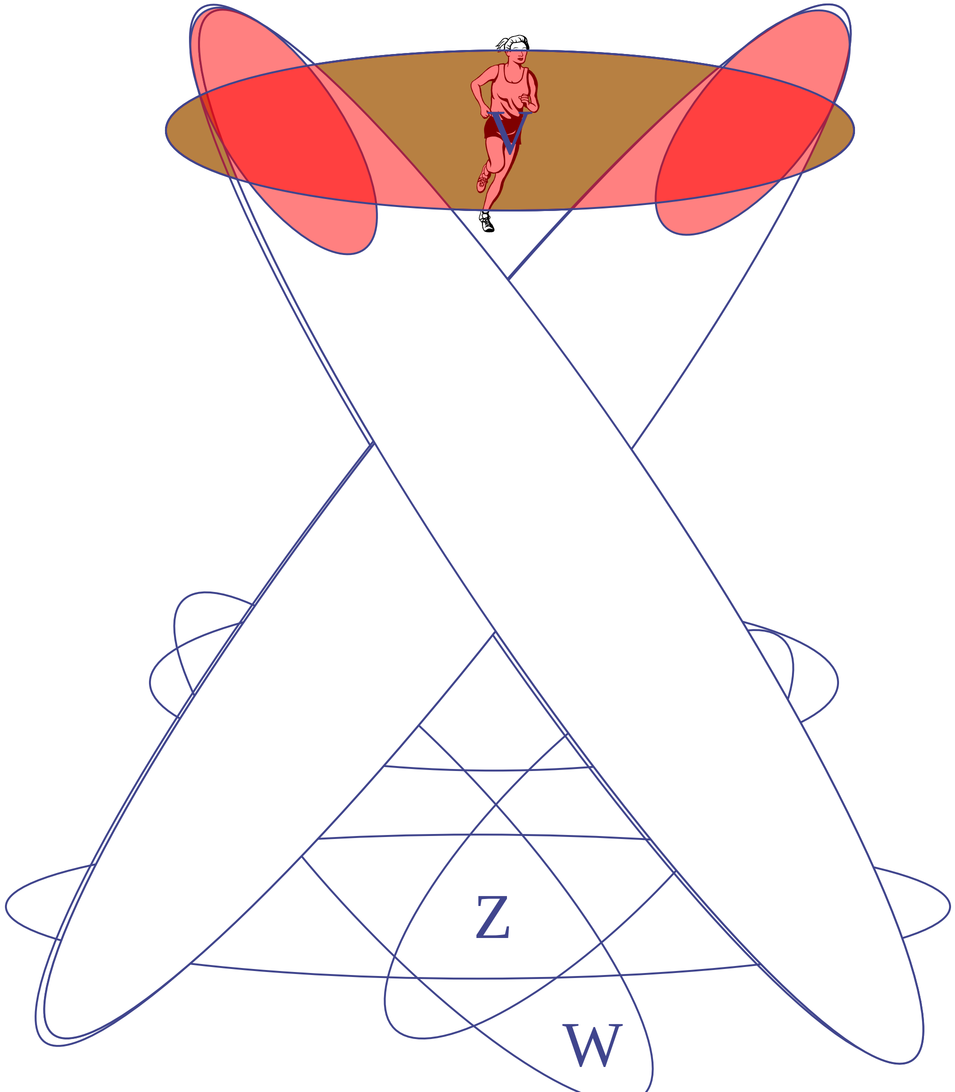
Step 2a: shrinking the space



A different robber’s choice



Step 2b: the capture



R&M Game and Hypertree Width

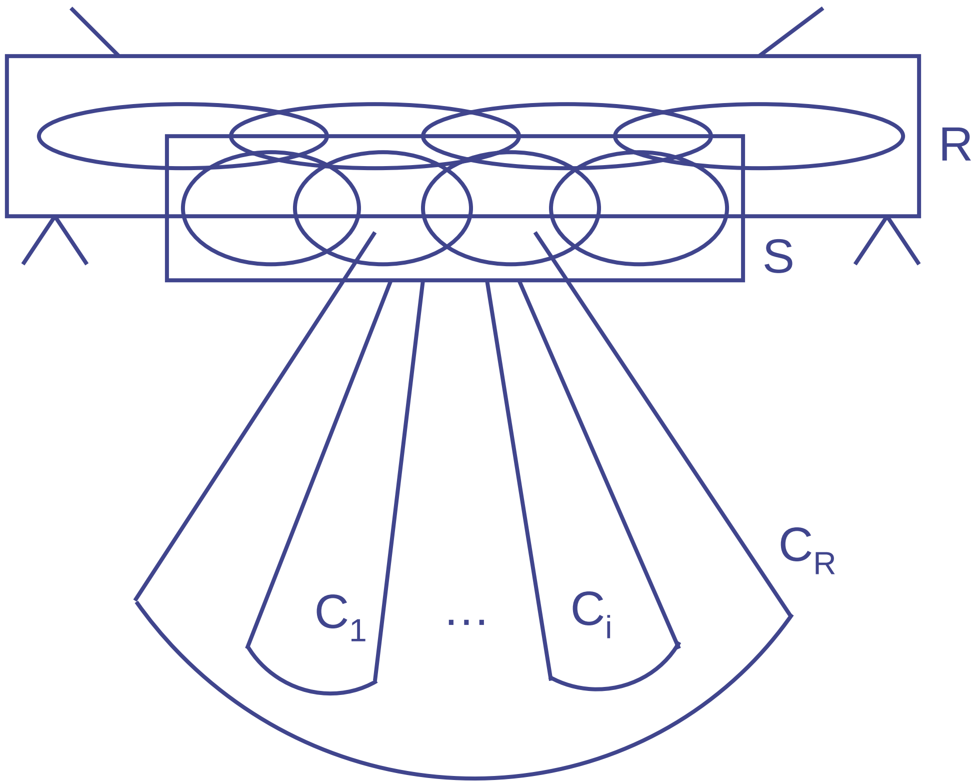
Let H be a hypergraph.

- **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.

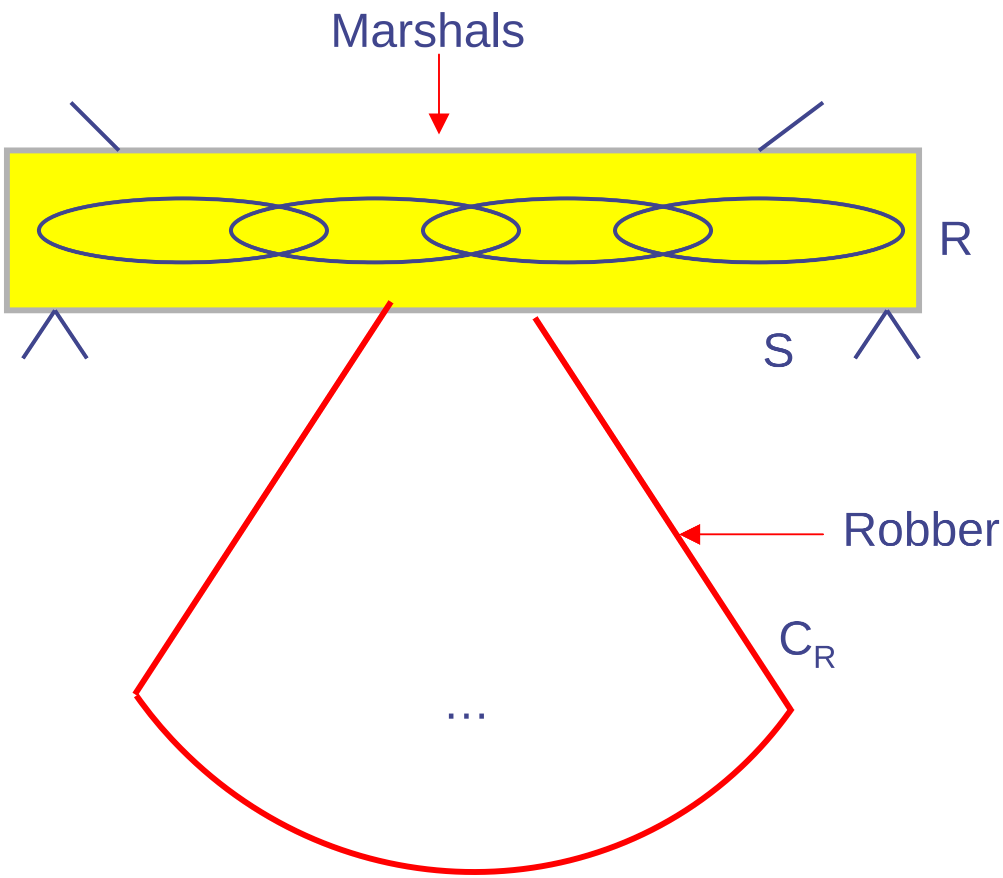
Polynomial algorithm: Alternating LOGSPACE

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



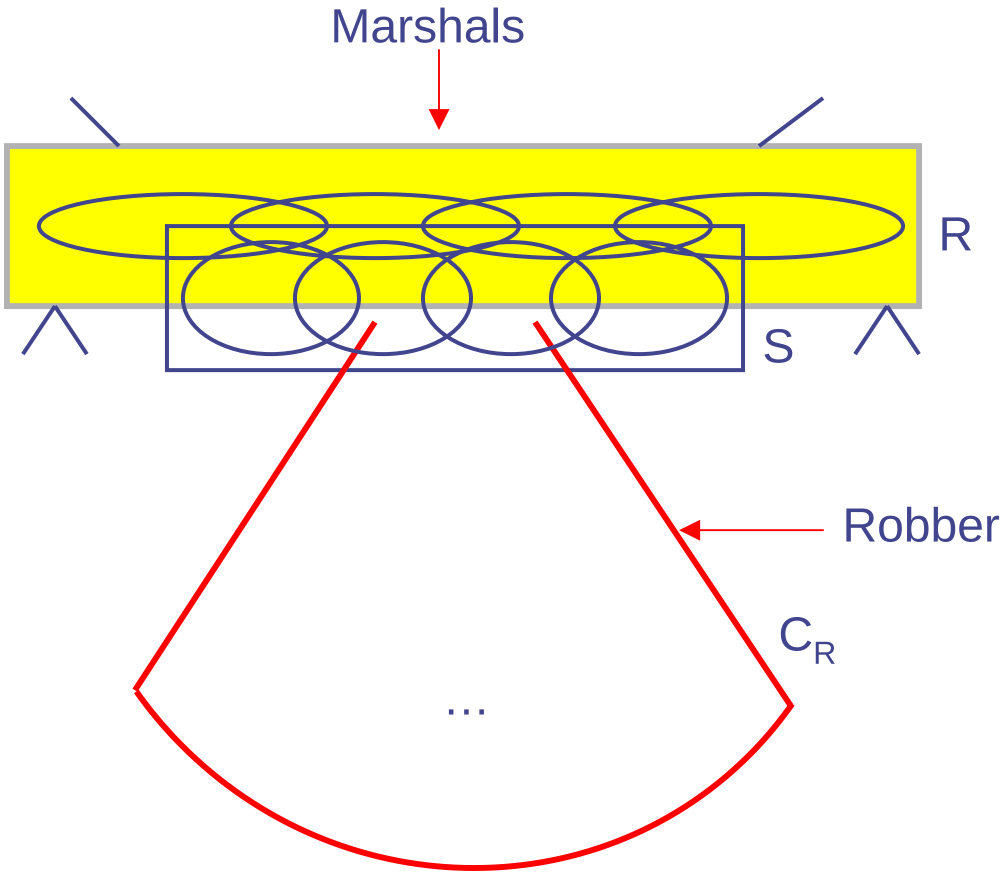
Polynomial algorithm: Alternating LOGSPACE

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



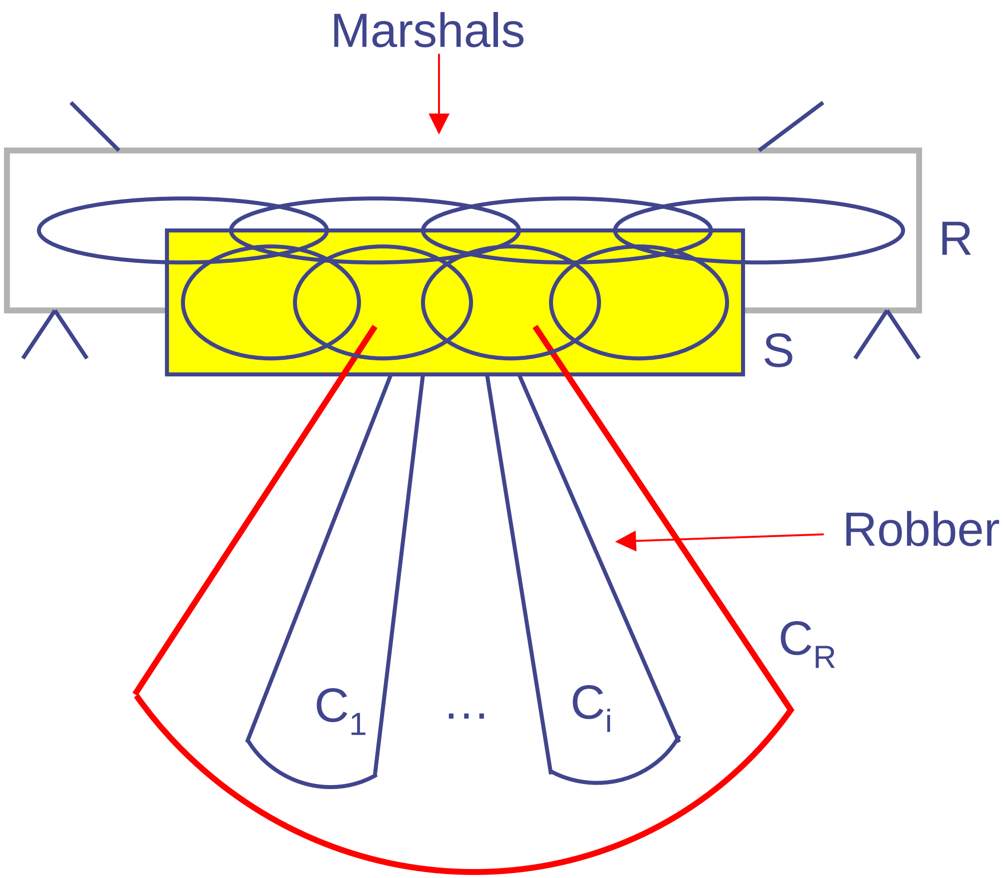
Polynomial algorithm: Alternating LOGSPACE

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



Polynomial algorithm: Alternating LOGSPACE

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



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

Strict shrinking: (US) $\cap$ $C_{\mathbf{R}} \neq \emptyset$

LOGSPACE CHECKABLE

Logical Characterization of Hypertree width

Loosely guarded logic

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: $HW_k = GF_k(L\ \ )$

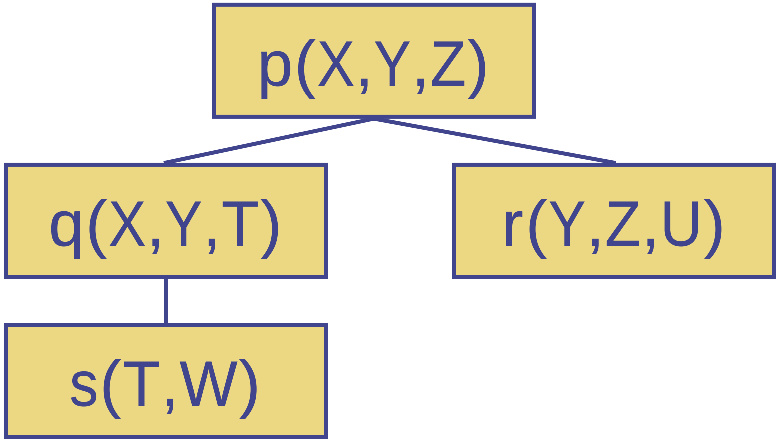
From this general result, we also get a nice logical characterization of acyclic queries:

Corollary: $HW_1 = ACYCLIC = GF(L\ \ )$

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:

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

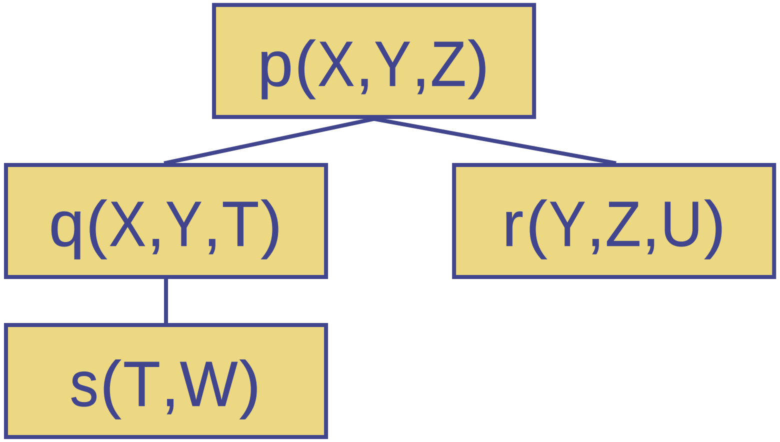
Guard

Guarded subformula

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:

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

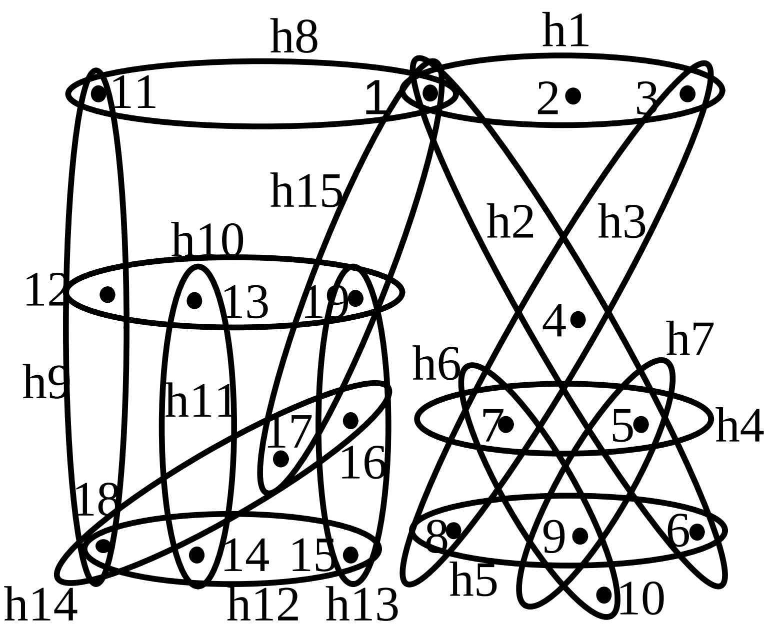
Guard

Guarded subformula

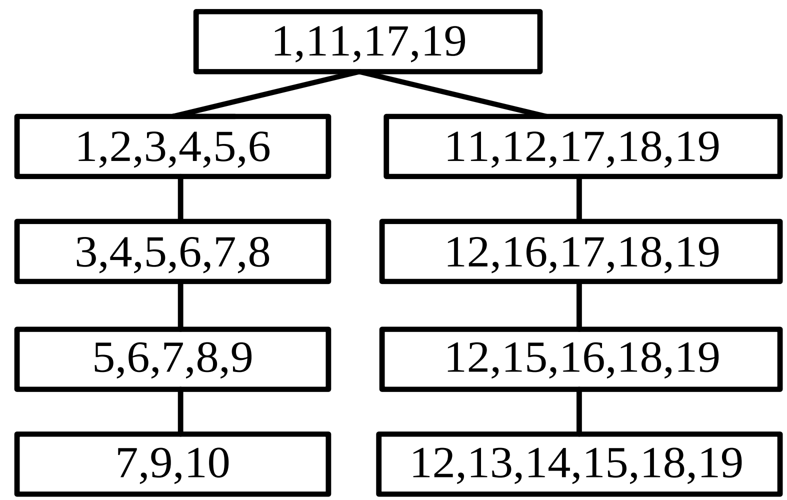
Alternative view and generalized HT-Decompositions

Tree decomposition of hypergraph

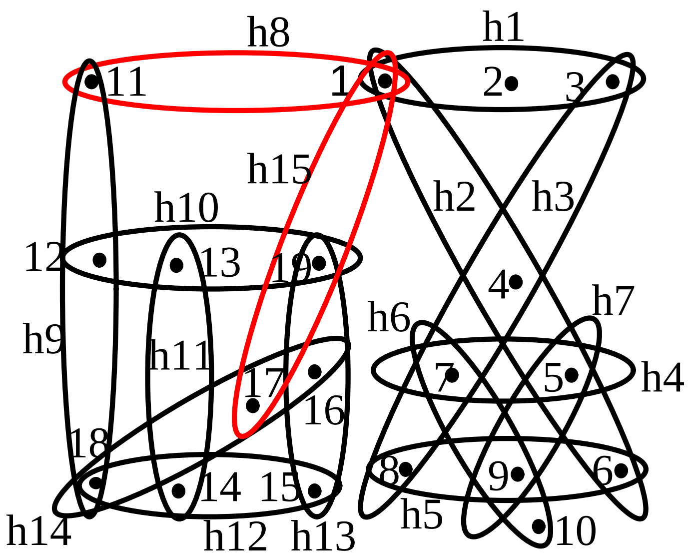
H

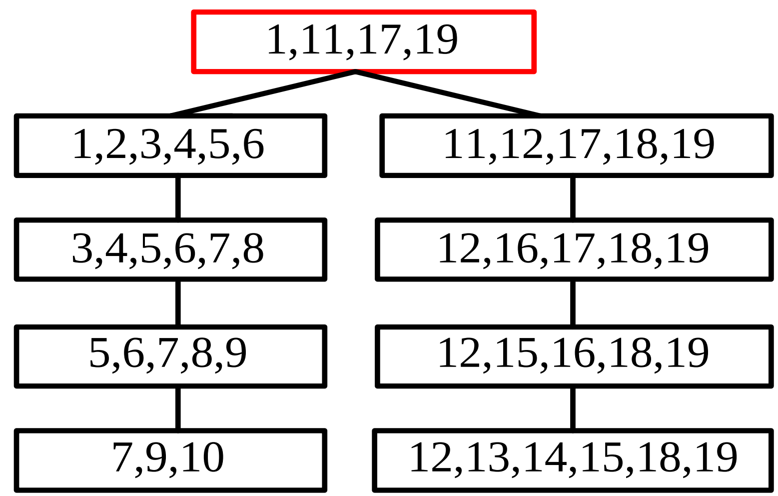


Tree decomp of G(H)

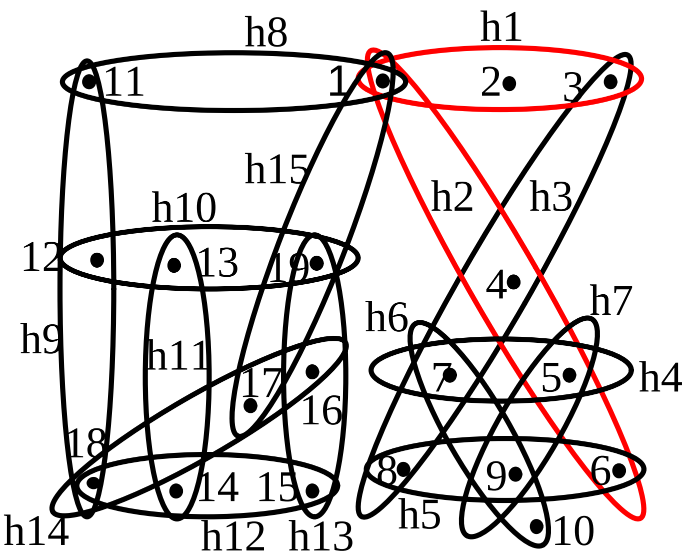


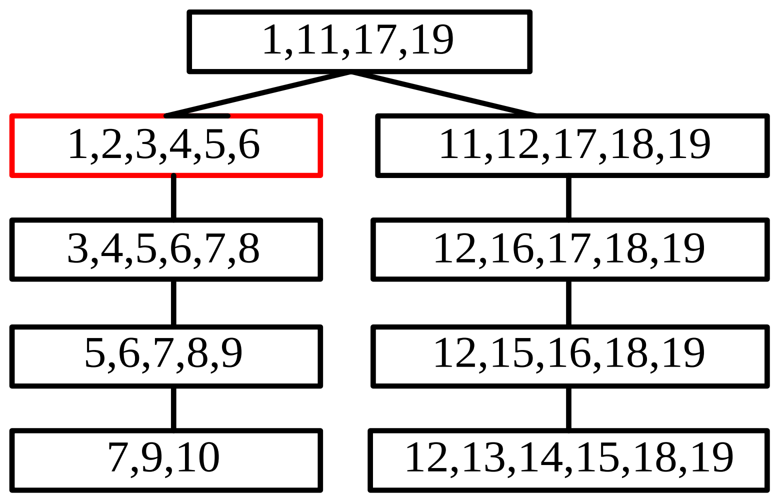
Hypergraph and the tree decomposition



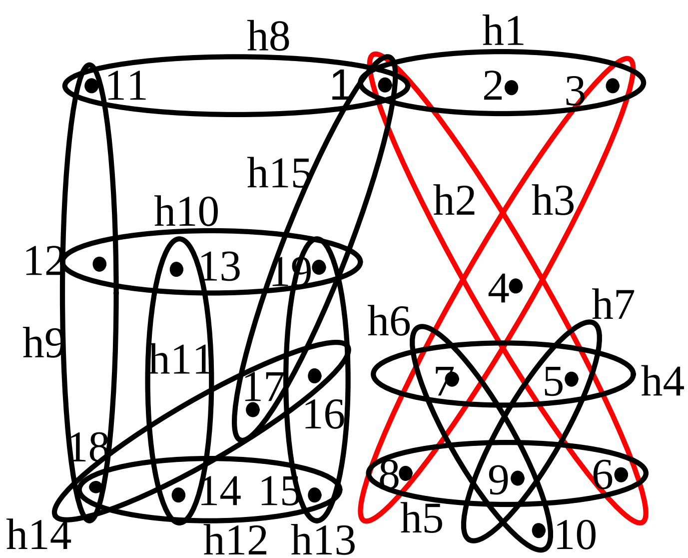


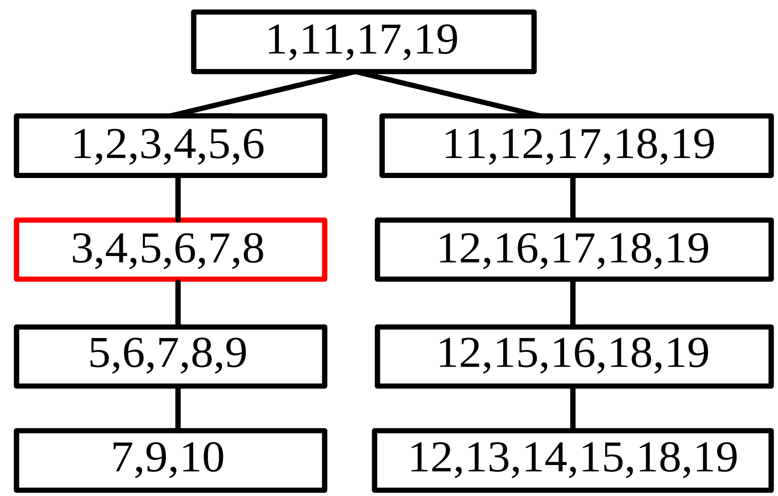
Hypergraph and the tree decomposition



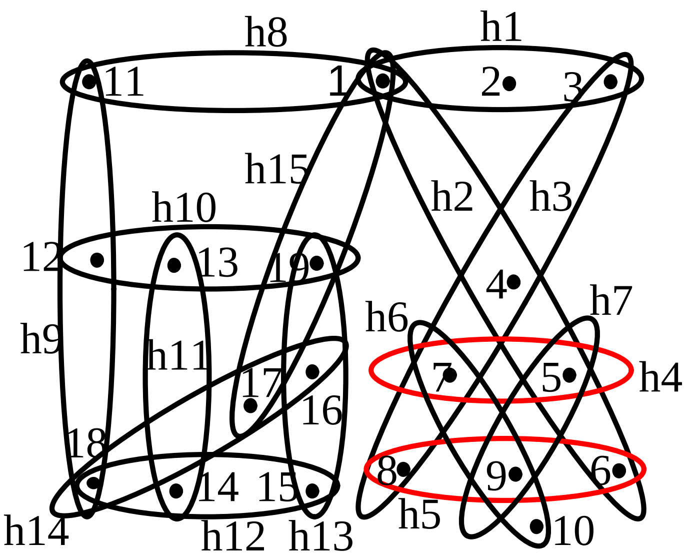


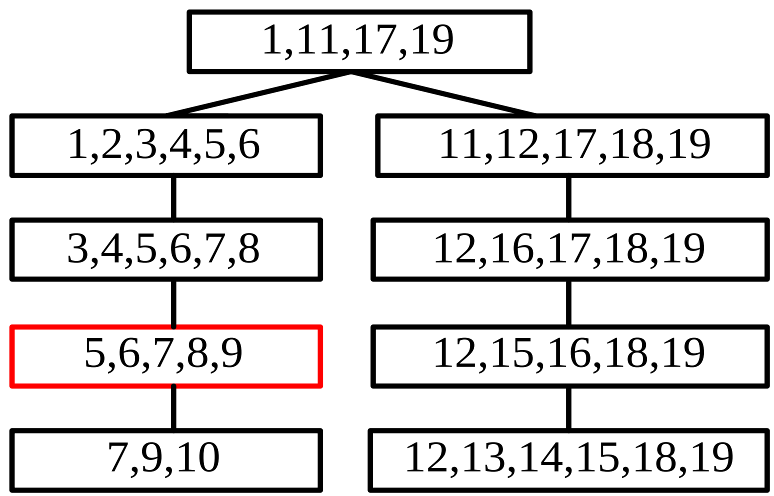
Hypergraph and the tree decomposition



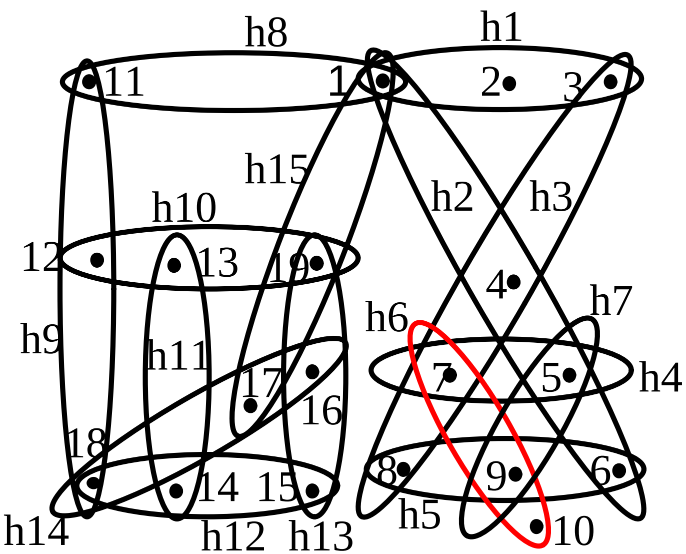


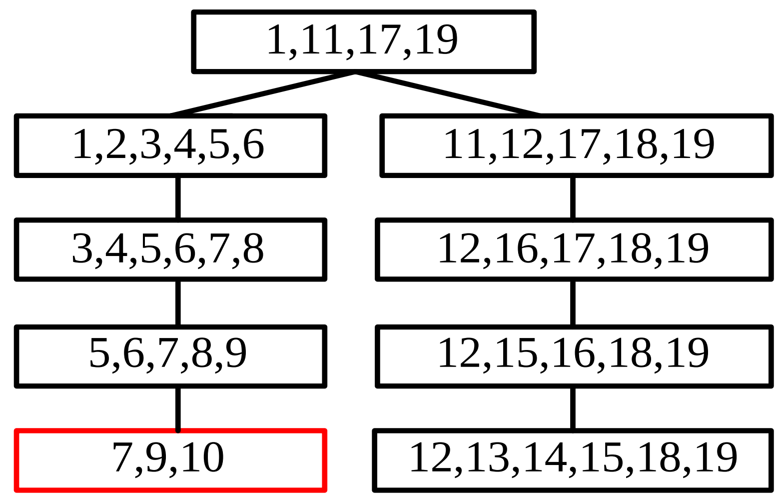
Hypergraph and the tree decomposition



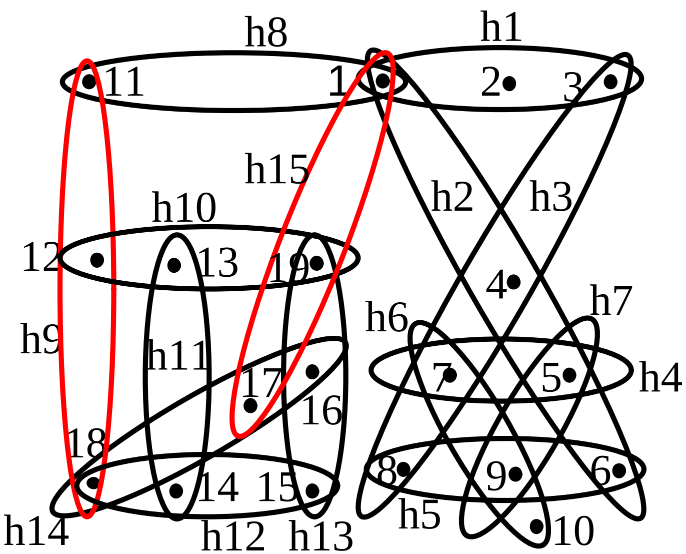


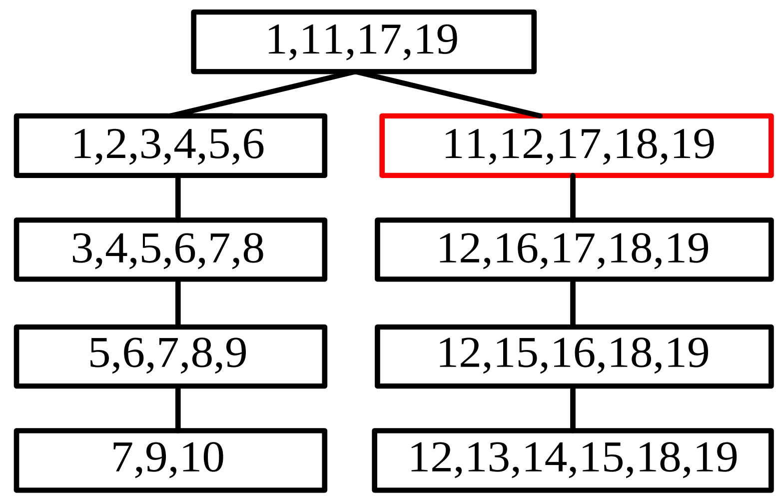
Hypergraph and the tree decomposition



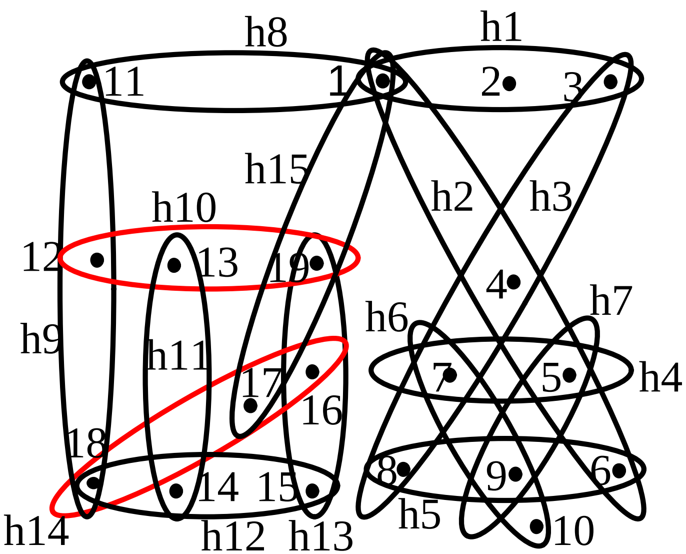


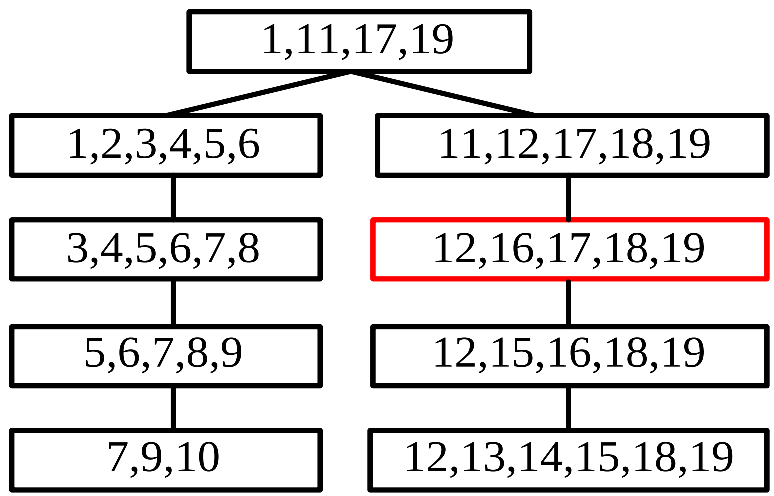
Hypergraph and the tree decomposition



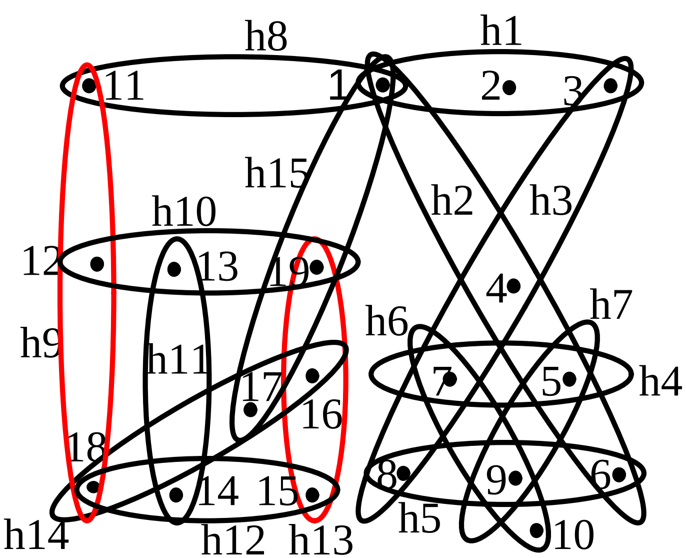


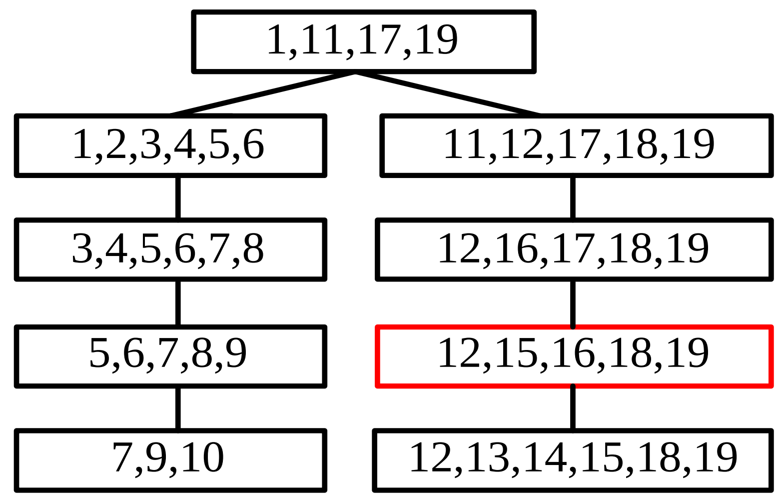
Hypergraph and the tree decomposition



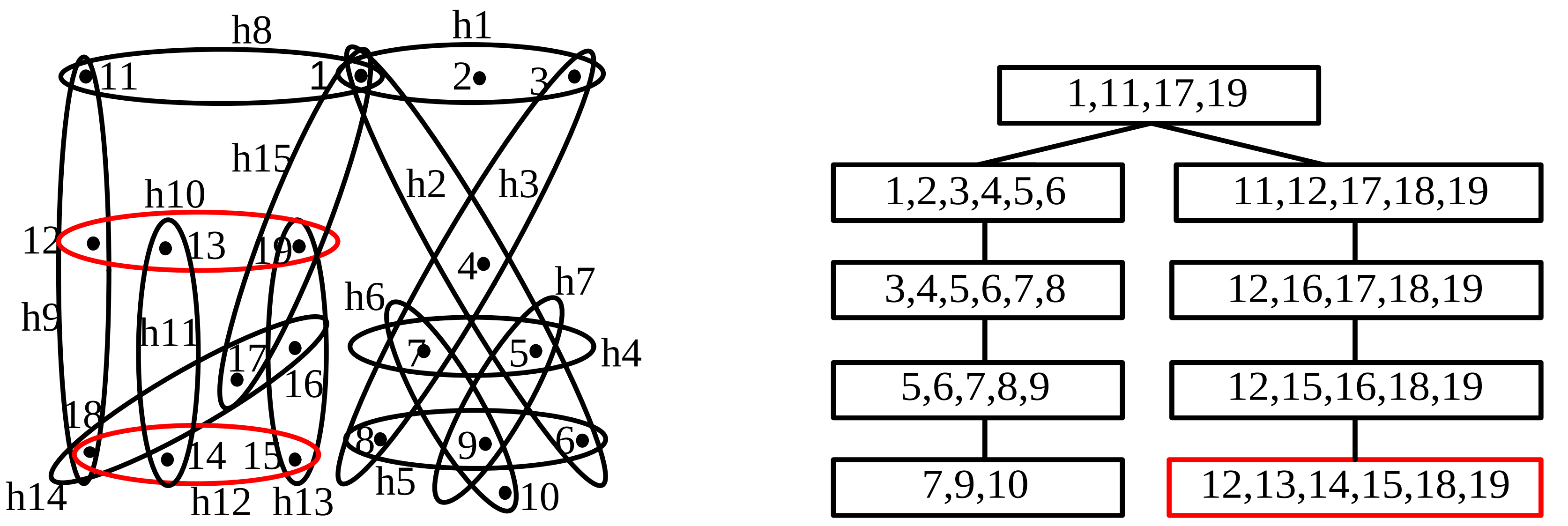


Hypergraph and the tree decomposition





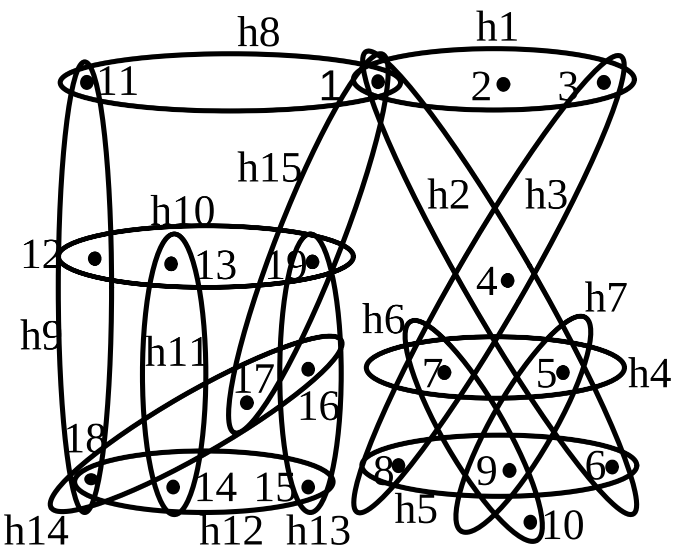
Hypergraph and the tree decomposition

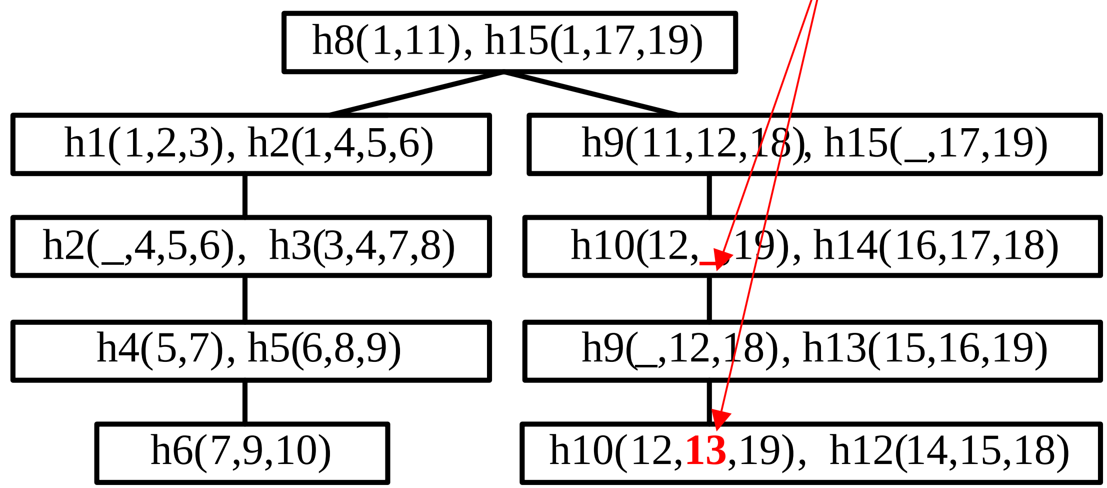


Generalized HT-Decomp of H of width 2

Generalized hypertree decomposition

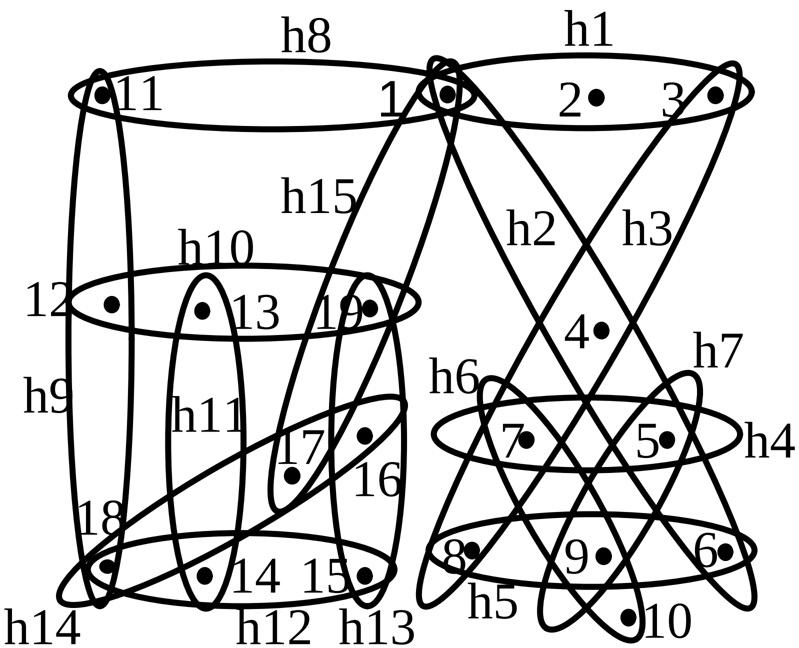
Special condition violated

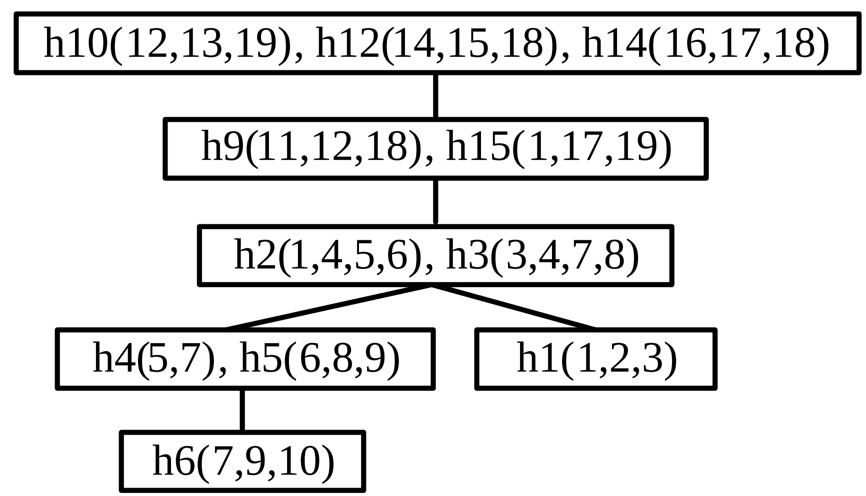




Generalized hypetree decomposition of width 2

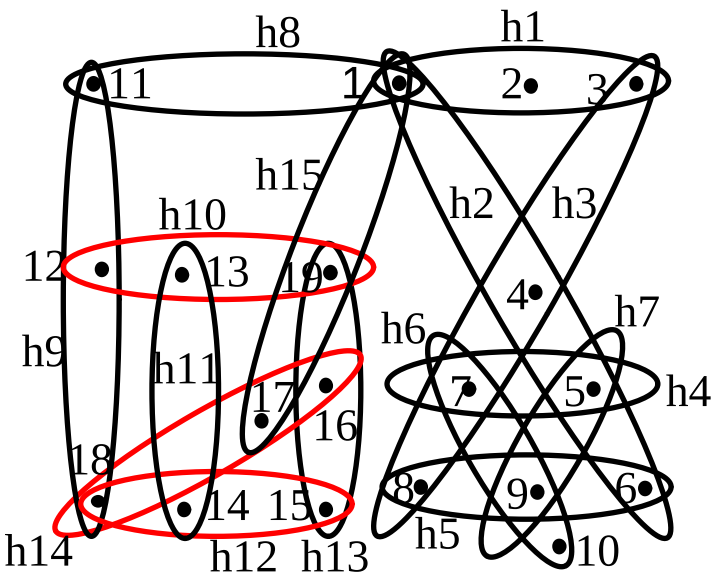
Hypertree decomposition

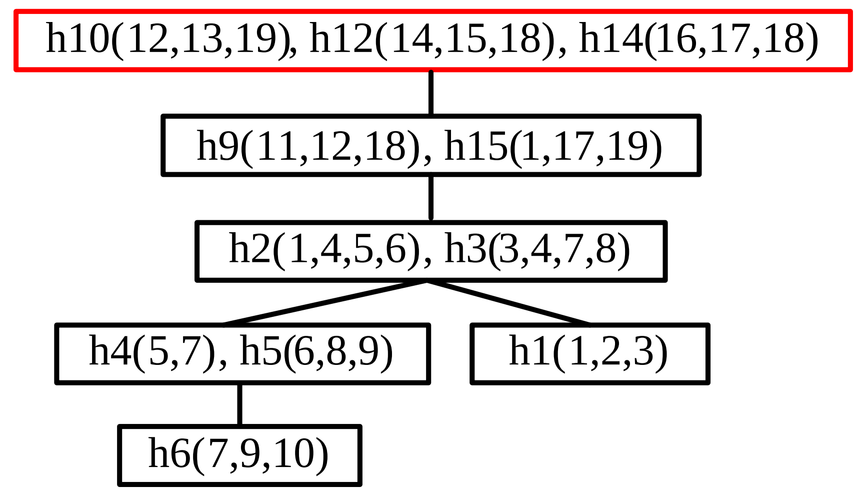




Hypertee decomposition of width 3

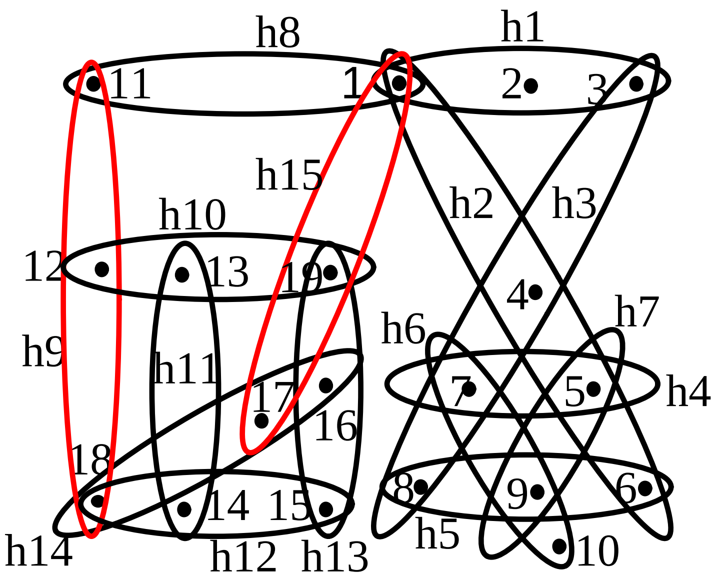
Hypertree decomposition

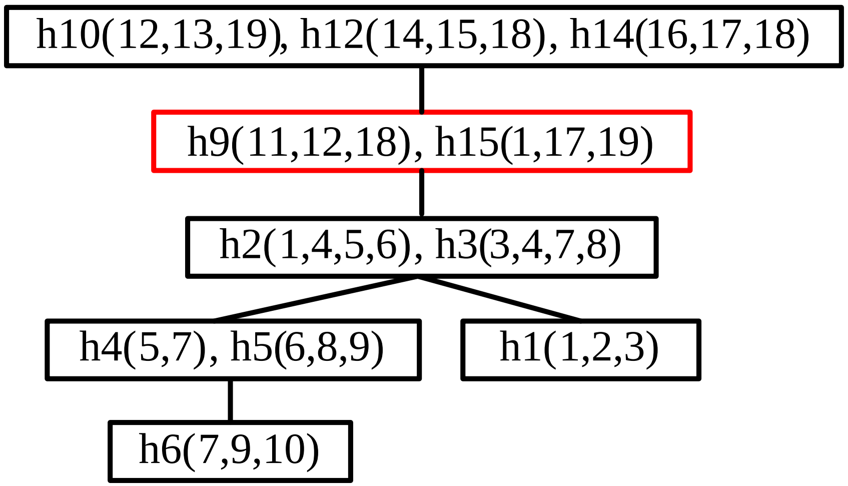




Hypertee decomposition of width 3

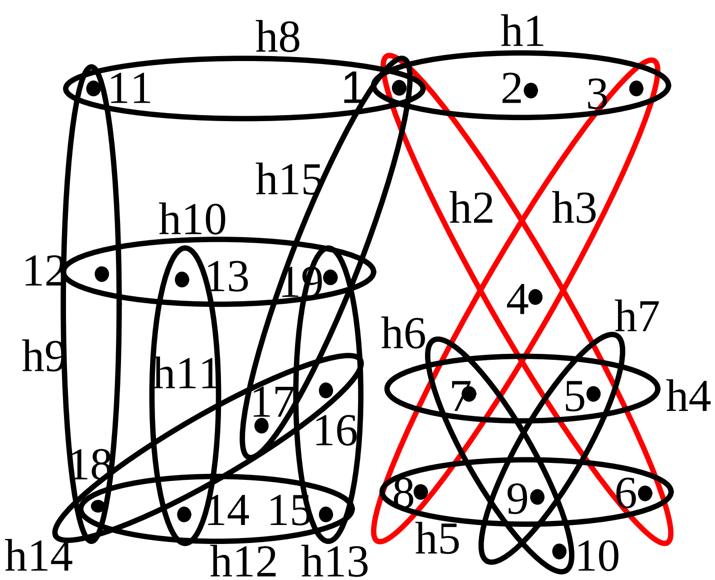
Hypertree decomposition

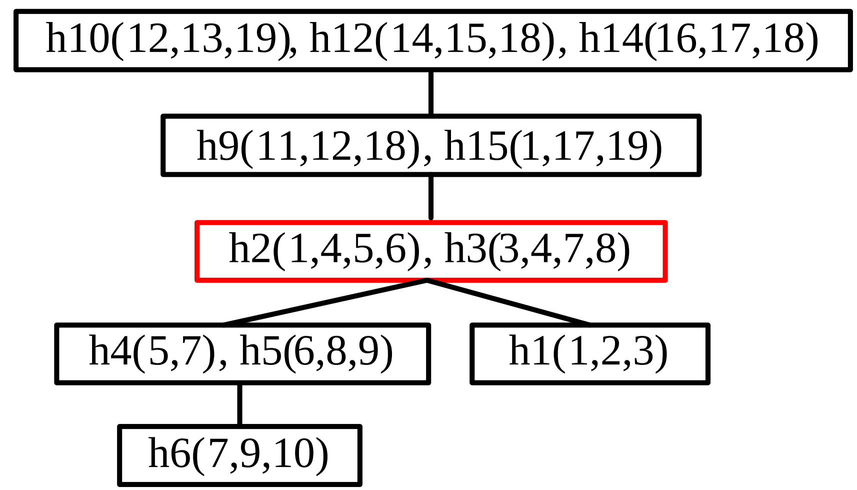




Hypertee decomposition of width 3

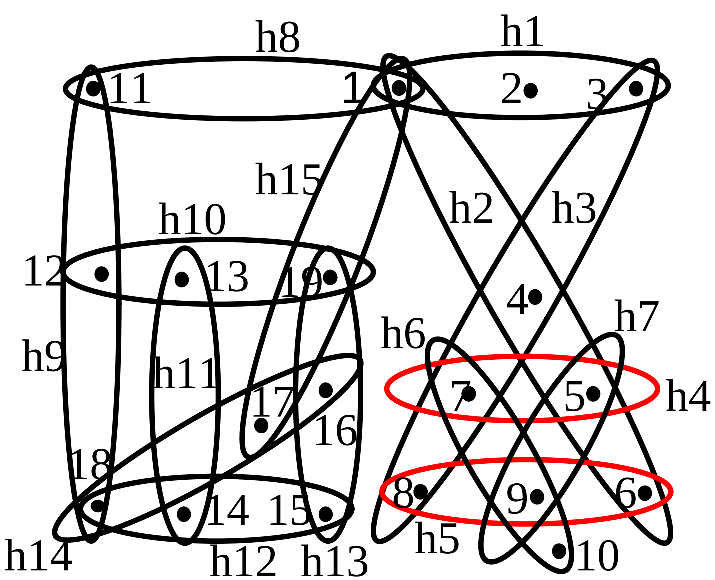
Hypertree decomposition

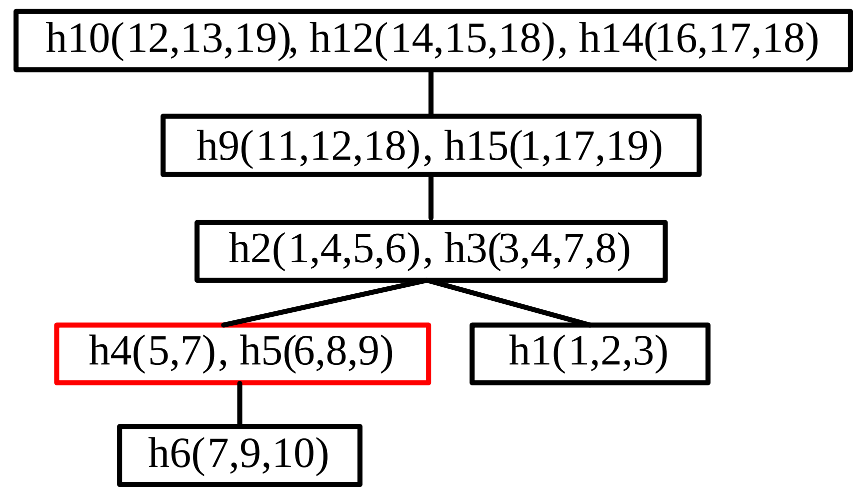




Hypertee decomposition of width 3

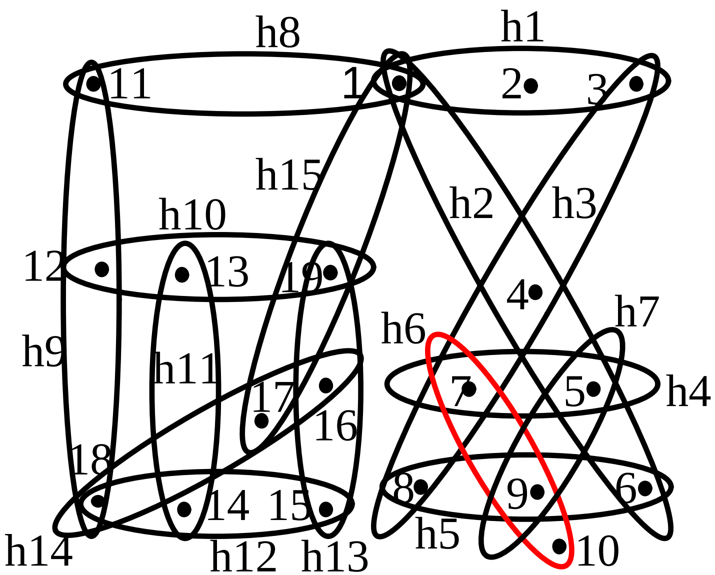
Hypertree decomposition

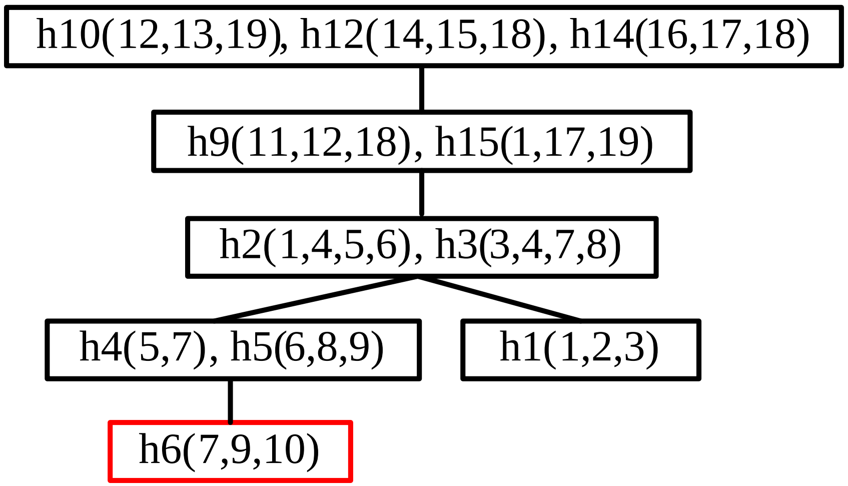




Hypertee decomposition of width 3

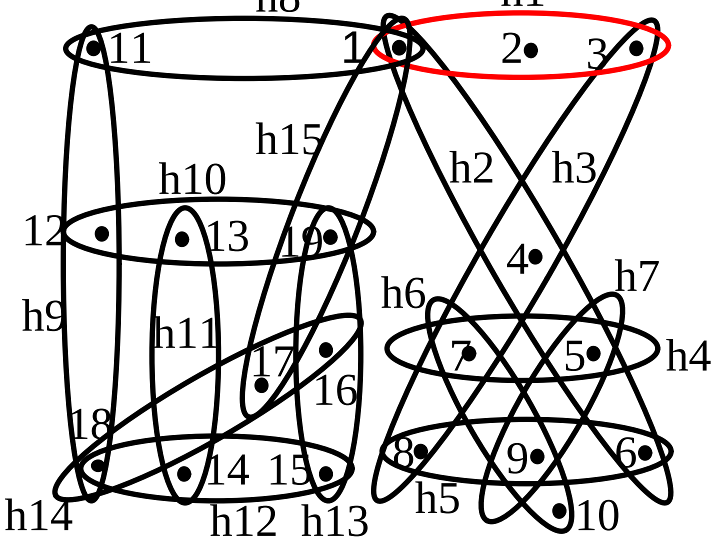
Hypertree decomposition

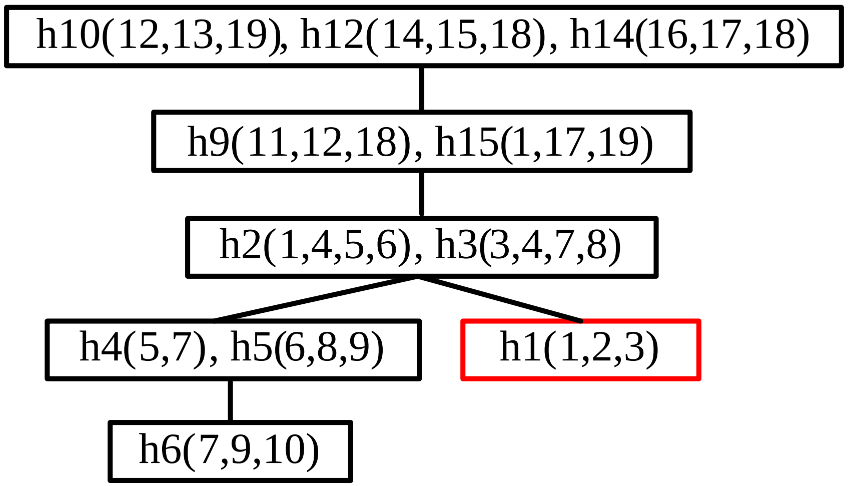




Hypertee decomposition of width 3

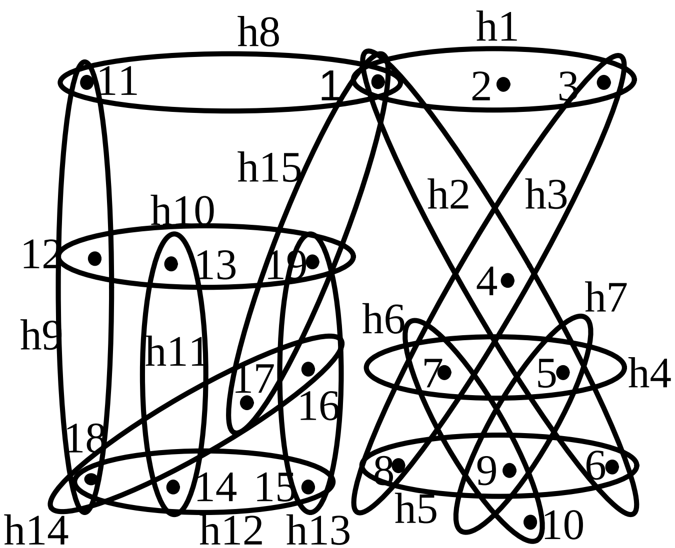
Hypertree decomposition

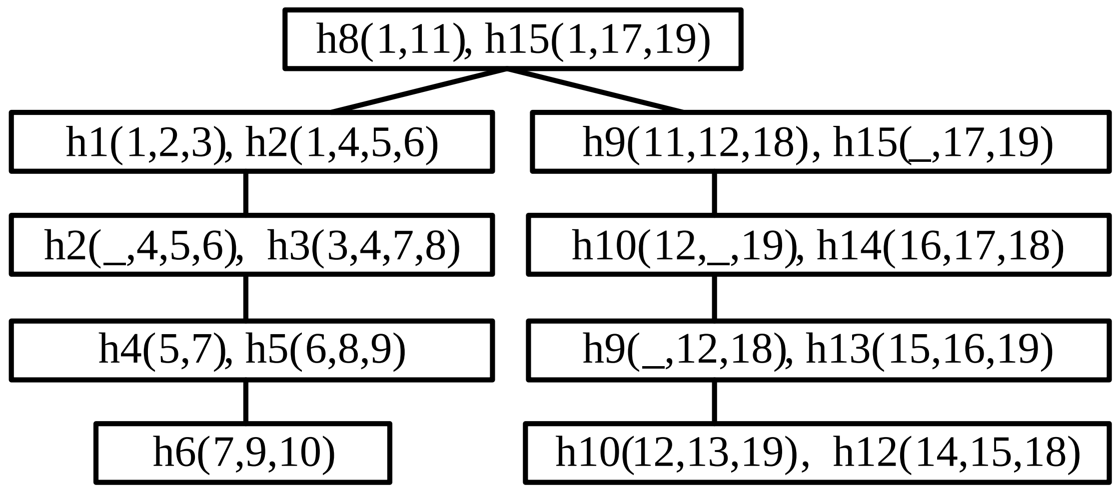




Hypertee decomposition of width 3

Generalized hypertree decomposition





Generalized hypetree decomposition of width 2

QUESTION:

Can we determine in polynomial time

Whether ghw(H) $<$ k for constant k ?

$\cup$

Observation: ghw(H) = hw(H*)

Where H* = H $\cup$ {E´ | $\exists$E in edges(H): E´ $\subseteq$ E}

Recent result [AGG] : ghw(H) <= 3hw(H)+1

Connection to the Hypergraph Sandwich problem!

Nasa problem

Part of relations for the Nasa problem

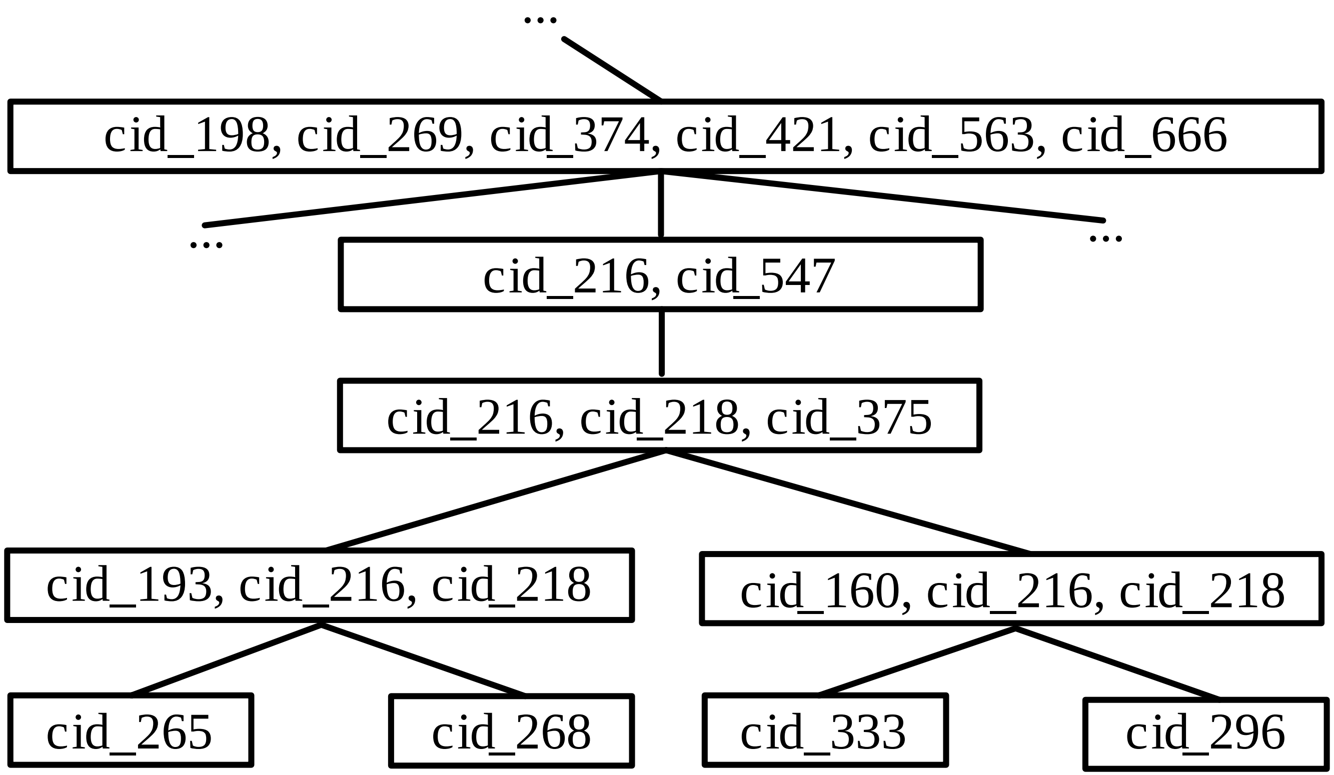
…

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

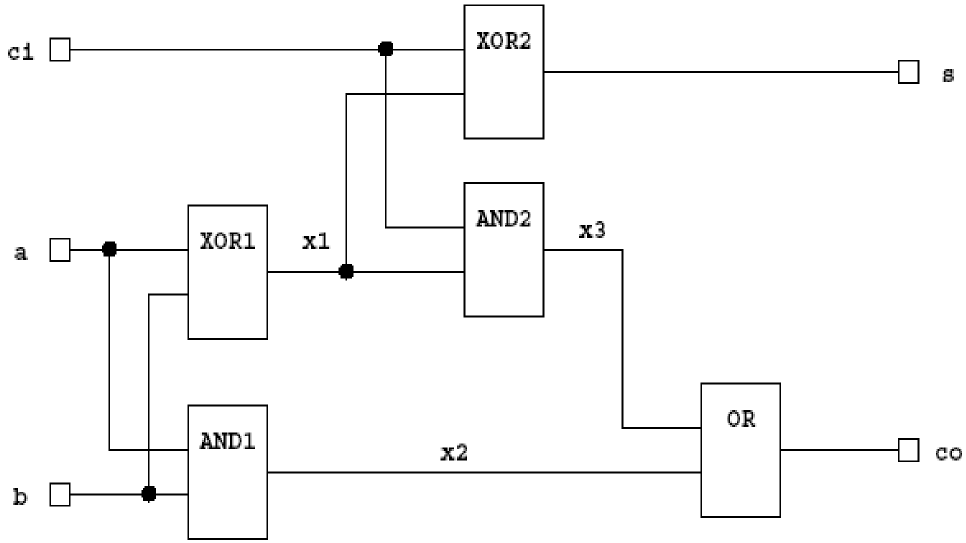
Nasa problem: hypertree



Part of hypertree for the Nasa problem

Best known hypertree-width for the Nasa problem is 22

Electronic Circuits: Adder



Adder circuit examples consist of a certain number of adder cells connected in a line

Adder_1:
6 relations and 8 variables

Adder_2:
11 relations and 15 variables
...

The basic cell of a adder circuit

The legal values for inputs and outputs in adder should be found

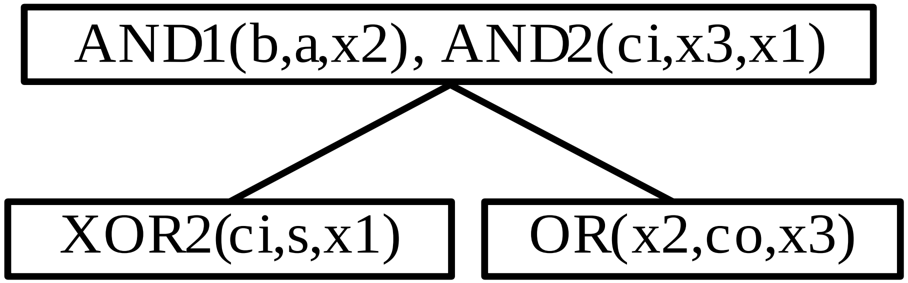
Adder_99:
496 relations and 694 variables
...

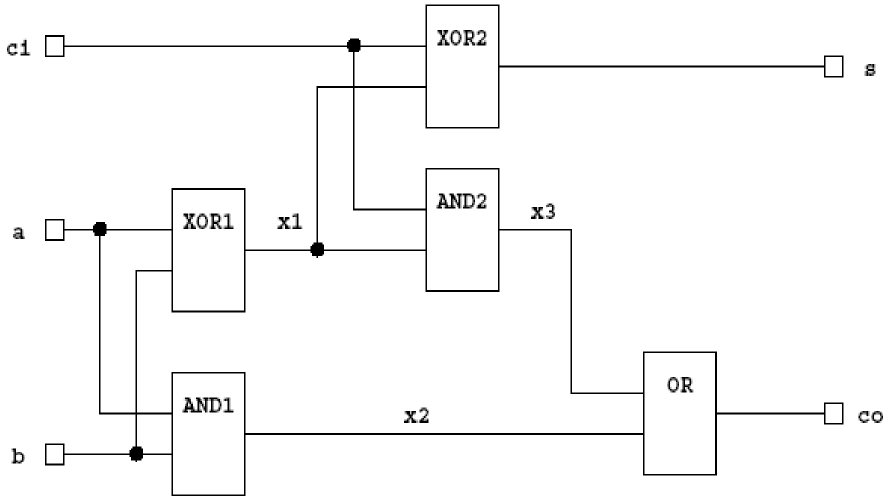
Hypertree width for all examples is 2

Hypertree for the adder circuit with one cell

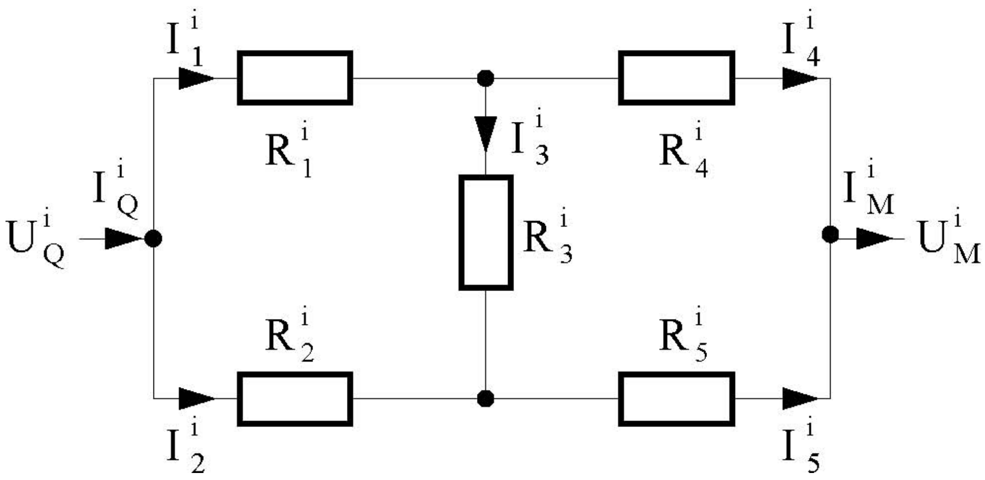
Constraints

init(ci)
AND1(b, a, x2)
XOR1(b, a, x1)
AND2(ci, x3, x1)
OR(x2, co, x3)
XOR2(ci, s, x1)





Electronic Circuits: Bridges



The basic cell of a bridge circuit

The legal values for variables (tensions and currents) in circuit should be found

Bridge circuit examples consist of a certain number of bridge cells connected in a line

Bridge_1:
11 relations and 11 variables

Bridge_2:
20 relations, 20 variables

...

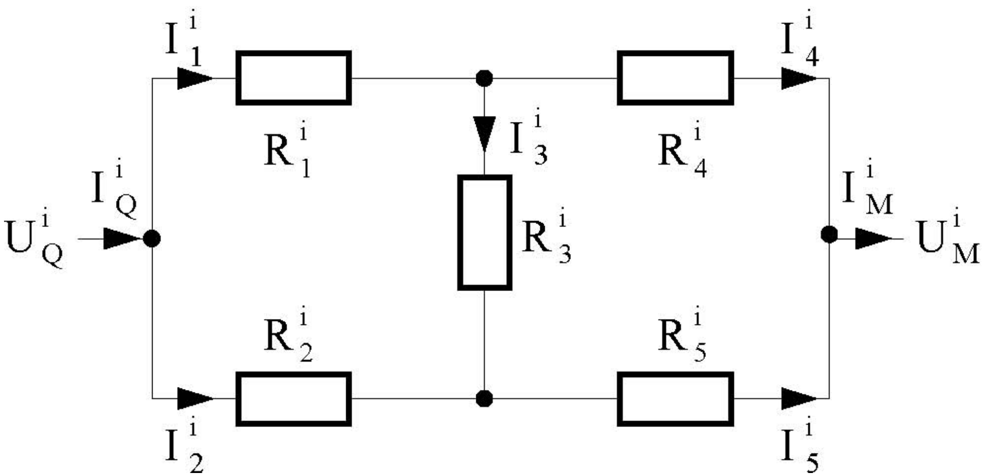
Bridge_99:
893 relations and 893 variables

...

Hypertree width for all examples is 2

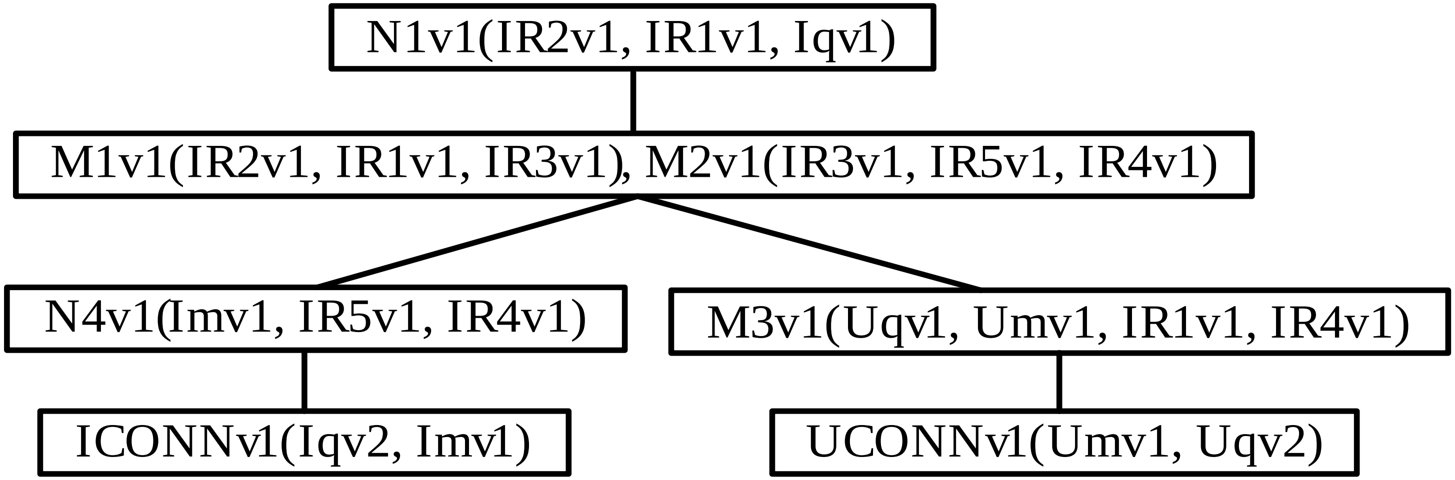
The bridge circuit with one cell

Constraints are obtained by Kirchhoff Laws, Ohm's Law and others



Init (Uqv1)
M1v1 (IR2v1, IR1v1, IR3v1)
M2v1 (IR3v1, IR5v1, IR4v1)
M3v1 (Uqv1, Umv1, IR1v1, IR4v1)
N1v1 (IR2v1, IR1v1, Iqv1)
N2v1 (IR4v1, IR1v1, IR3v1)
N3v1 (IR2v1, IR5v1, IR3v1)
N4v1 (Imv1, IR5v1, IR4v1)
UCONNv1 (Umv1, Uqv2)
ICONNv1 (Iqv2, Imv1)
Term (Uqv2)

Hypertree for the bridge circuit with one cell



The structure of many problems can be described by graphs or hypergraphs.

Many NP-hard problems become tractable for instances whose associated graphs or hypergraphs have bounded treewidth.

An important class of problems is tractable even in case of large treewidth. This class is hypergraph-based.

We described such problems (CQ,CSP,HOM) and developed an appropriate notion of width:

HYPERTREE WIDTH

Conclusions and open questions

- There are some interesting open questions:
  - Special condition
- Hypertree decomposition is a very natural notion
  - Issues of fixed-parameter tractability
  - Nonmonotonic capturing vs monotonic capturing

For papers and further material see:

http://ulisse.deis.unical.it/~frank/Hypertrees/