Hypertree Decompositions

G. Gottlob
Technical University of Vienna, Austria
N. Leone and **F. Scarcello**
University of Calabria, Italy

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

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*

$$h : U \longrightarrow V$$

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

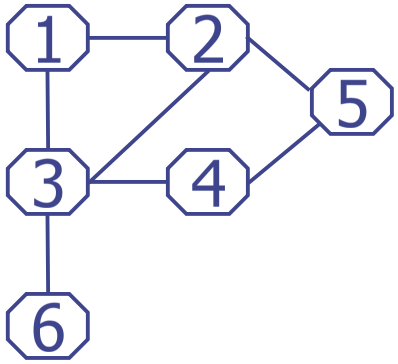
$$\mathbf{x} \in R_i \quad \Rightarrow \quad h(\mathbf{x}) \in S_i$$

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

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

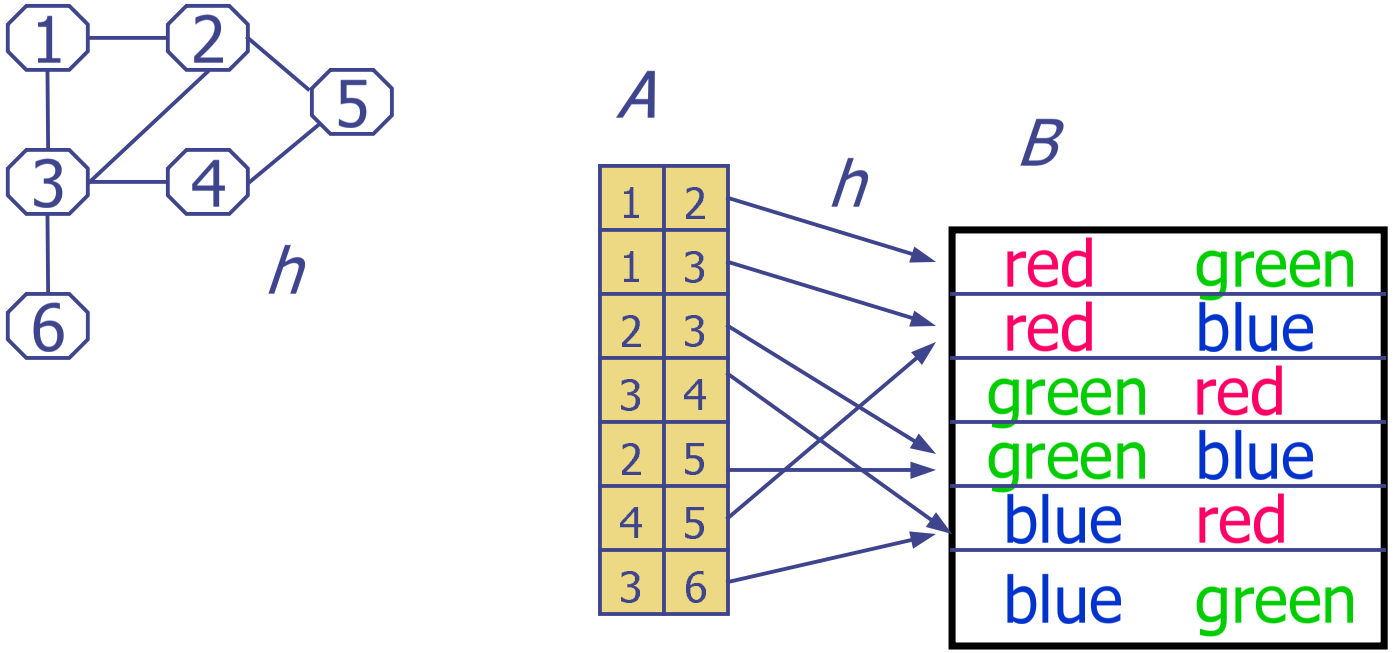
Graph 3-colourable iff HOM(*A*,*B*) yes-instance.

HOM is NP-complete

(well-known, independently proved in various contexts)

Membership: Obvious, guess *h*.

Hardness: Transformation from 3COL.



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

Conjunctive Queries, CSPs

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

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

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

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?

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

BCQ = HOM

View query Q (=scopes) 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.

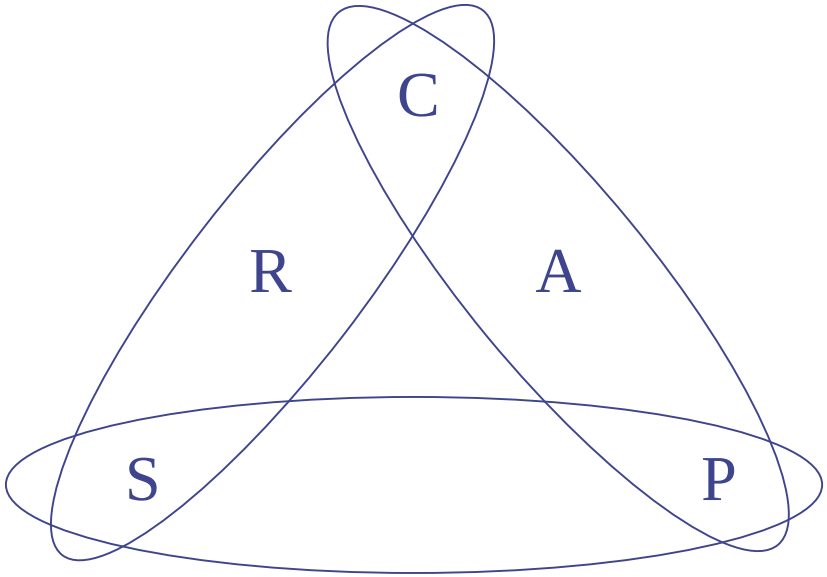
The Boolean conjunctive query (CSP) is equivalen to the HOM instance HOM(Q,D).

Vive-versa, every HOM instance can be reformulated as a Boolean Conjunctive Query (CSP).

This talk will mainly concentrate on BCQ

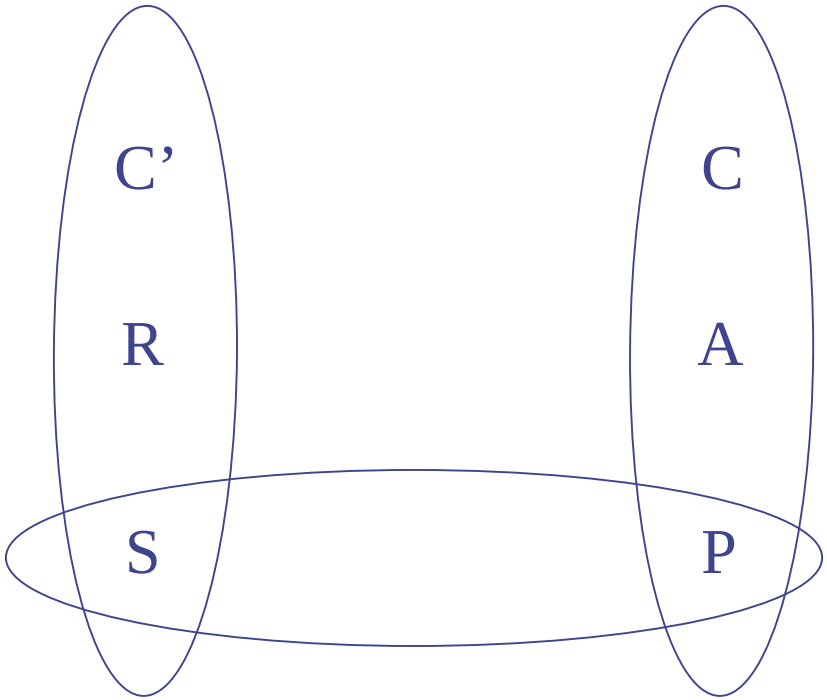
Queries, CSPs, and Hypergraphs

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



Queries, CSPs, and Hypergraphs

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

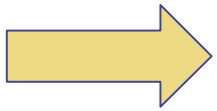


Boolean Conjunctive Queries

The problem BCQ (= constraint satisiability)

Instance: < **DB, Q**> (= <Relations, Scope>)

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



Combined Complexity

(Vardi ’82)

Problems Equivalent to BCQ

- Conjunctive Query Containment

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

- Query of Tuple Problem

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

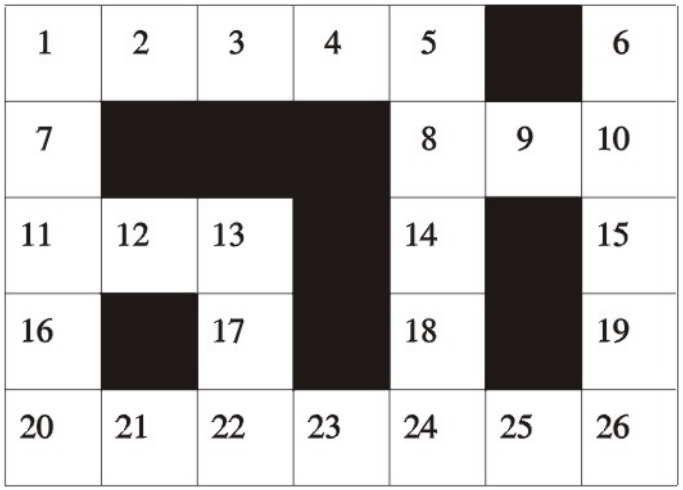
- Constraint Satisfaction in AI

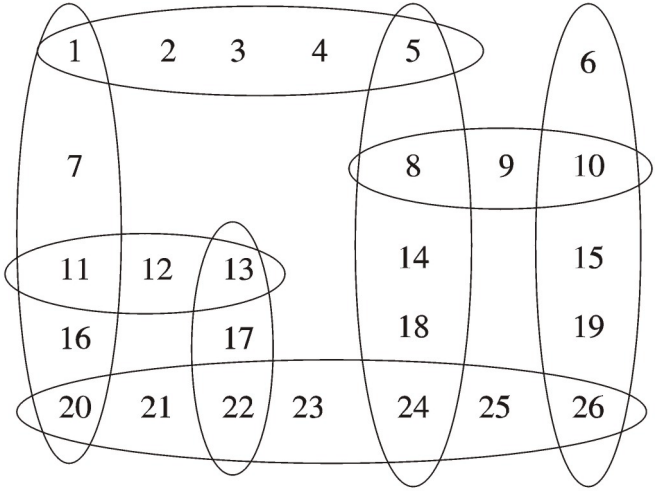
$$\text{CSP} \;\; \approx \;\; \text{BCQ}$$

- Clause Subsumption in Theorem Proving

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

Example of CSP: Crossword Puzzle

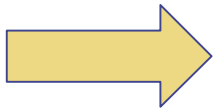




Complexity of BCQ

- NP-complete in the general case
  (Chandra and Merlin '77)
  NP-hard even for fixed database

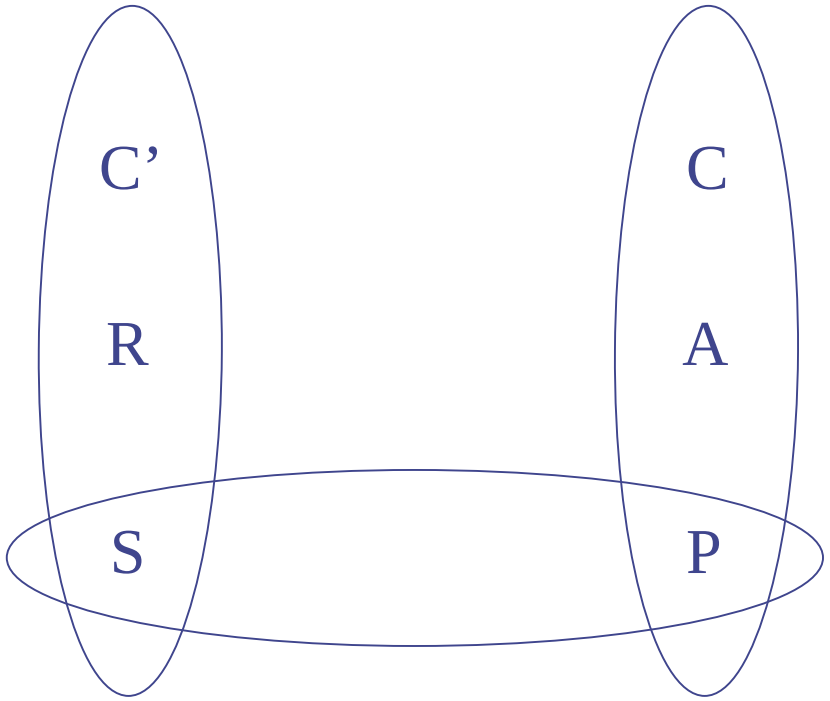
- Polynomial if Q has an acyclic hypergraph
  (Yannakakis '81)
  LOGCFL-complete (in NC_2)
  (G.L.S. '98)

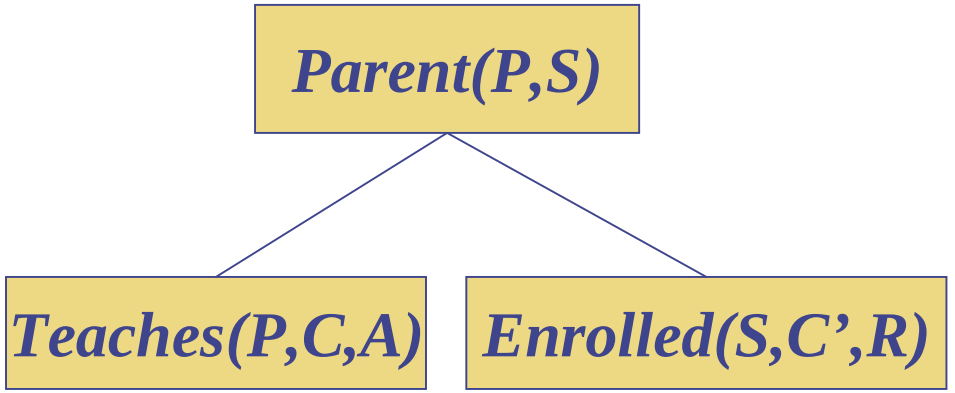


Interest in larger tractable classes of CQS

Acyclic queries or CSPs

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





Join Tree

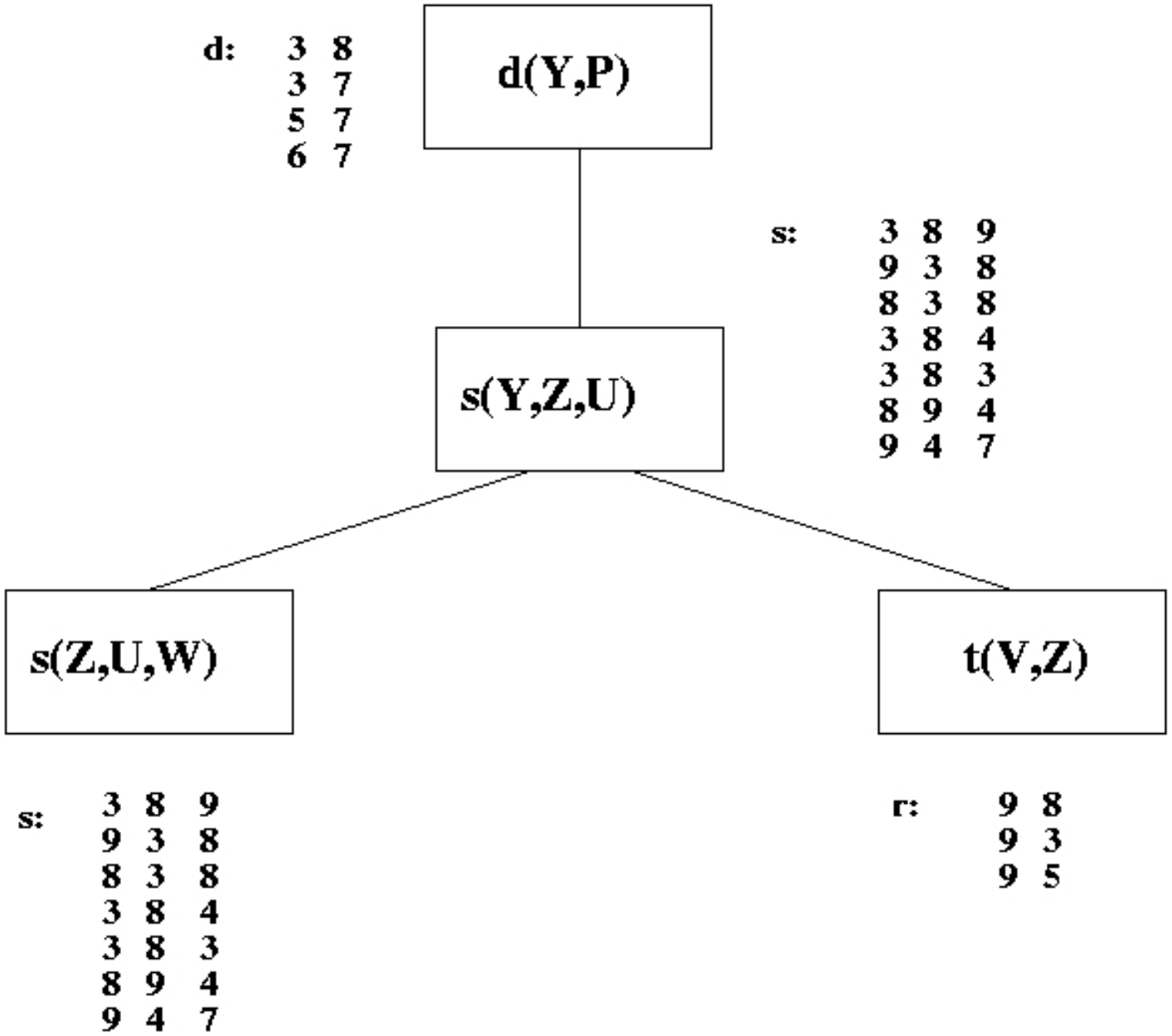
Theorem [GLS99]: Answering acyclic BCQs is LOGCFL-complete

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


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

Is this query hard?

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

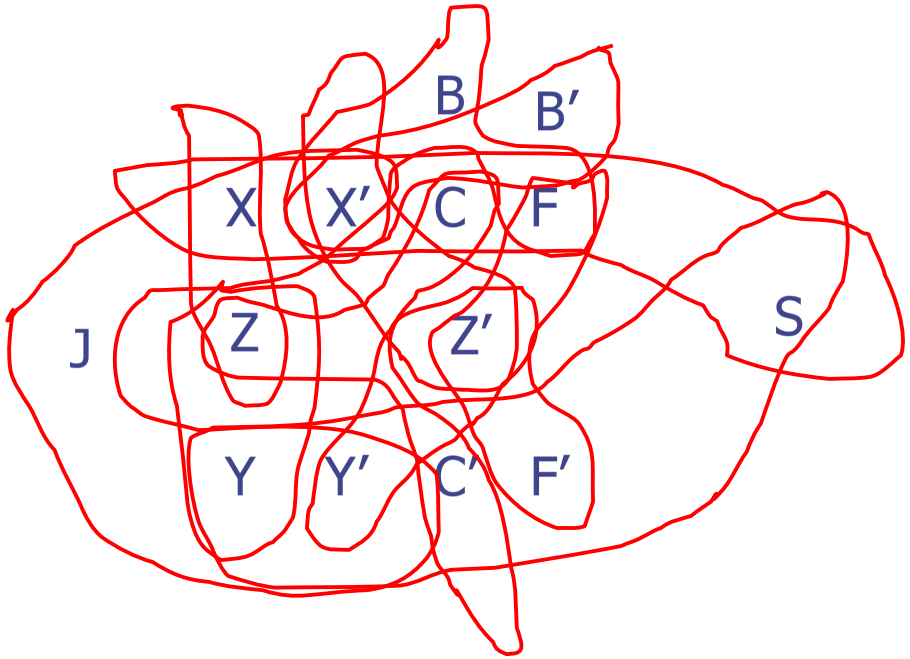
n size of the database
m number of atoms in the query

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 $tw(Q) \leq k$
  - Computing a tree decomposition

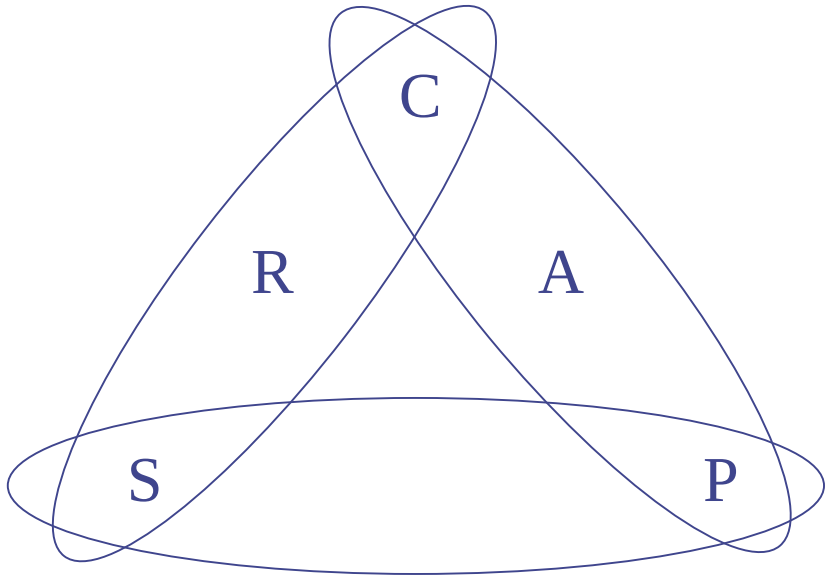
  } linear time (Bodlaender'96)
- Answering BCQ of treewidth k:

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

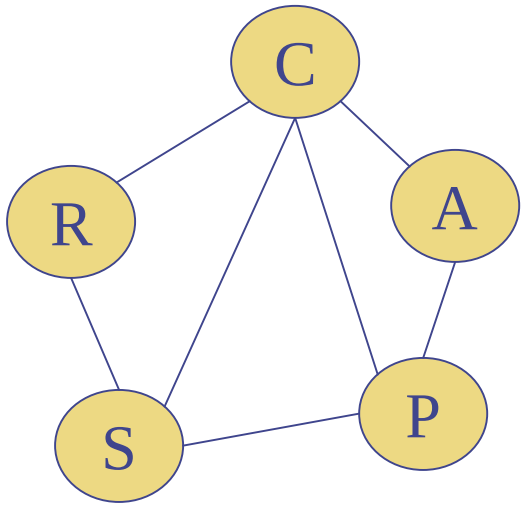
  LOGCFL-complete (G.L.S.'98)

Primal graphs of Queries

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

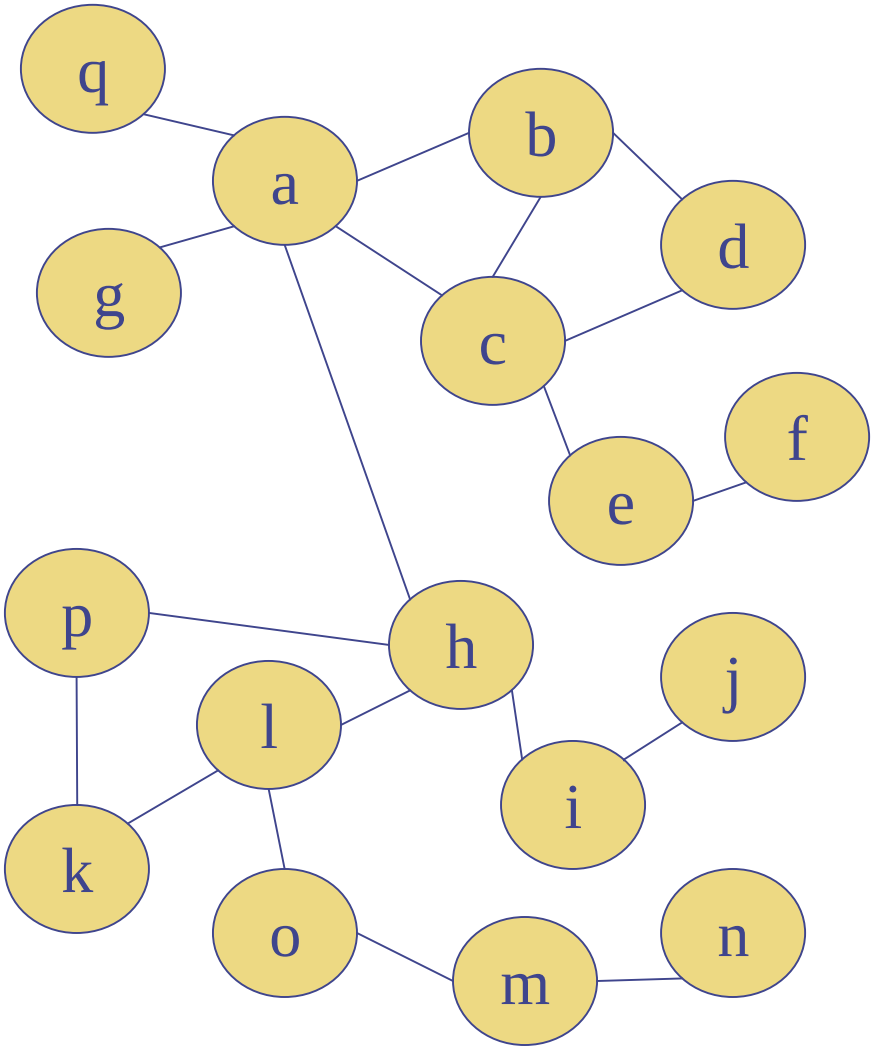


Hypergraph *H(Q)*

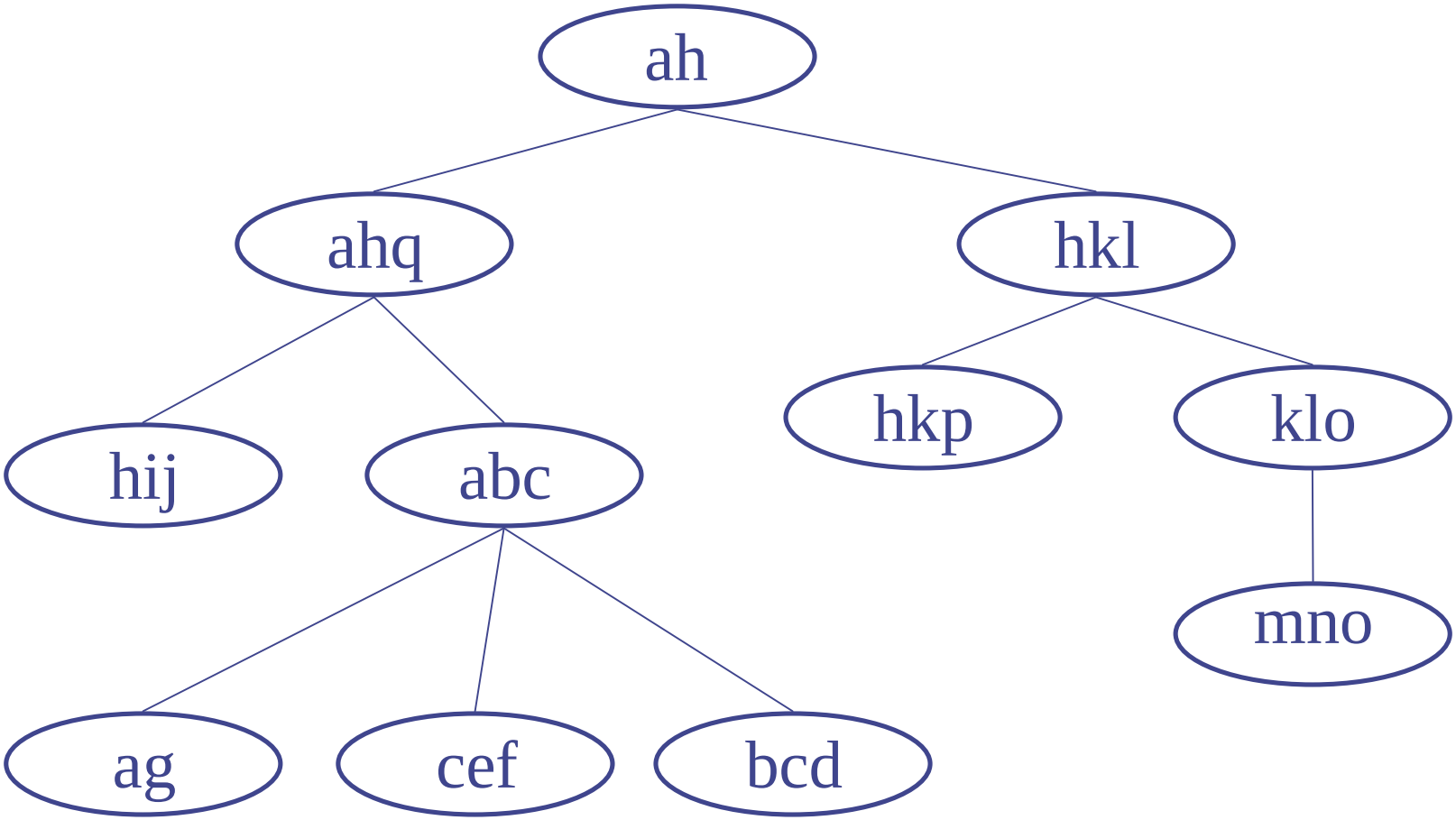


Primal graph *G(Q)*

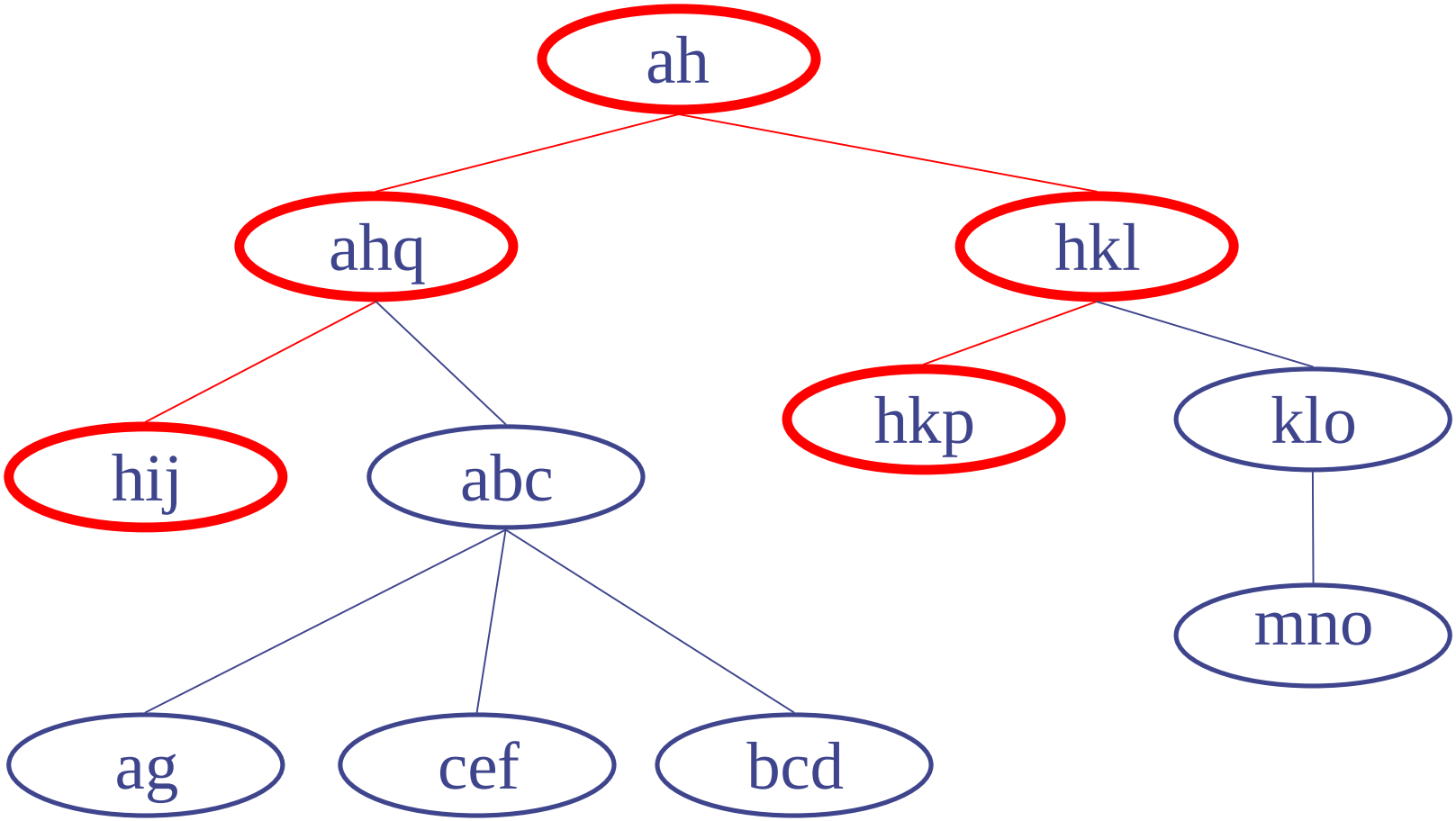
Example: a cyclic graph



A tree decomposition of width 2



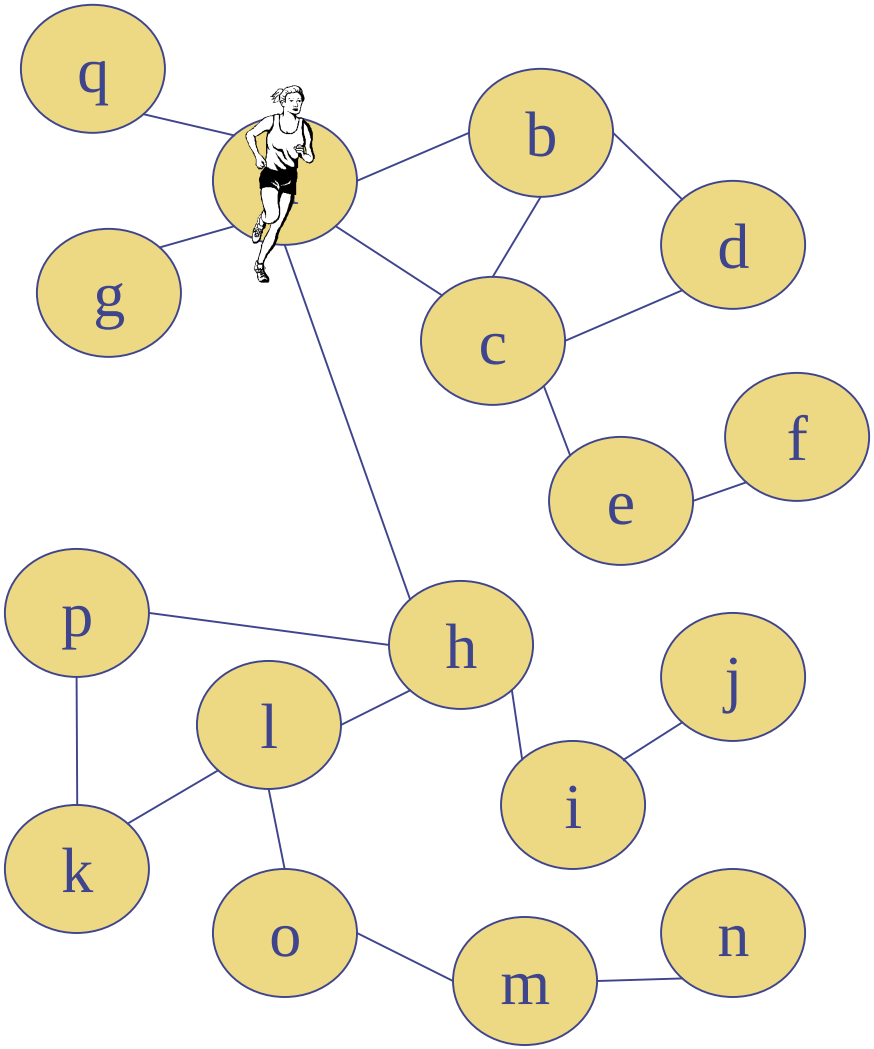
Connectedness condition for *h*



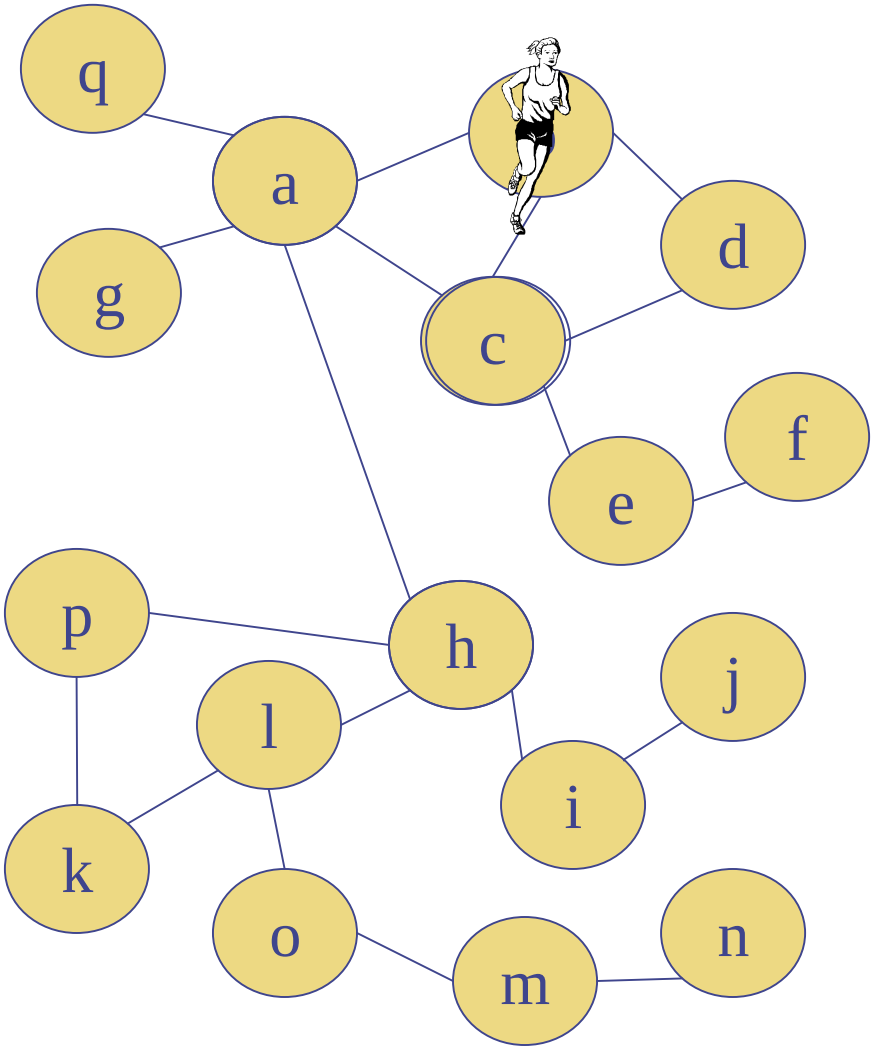
Game characterization of Treewidth

- 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

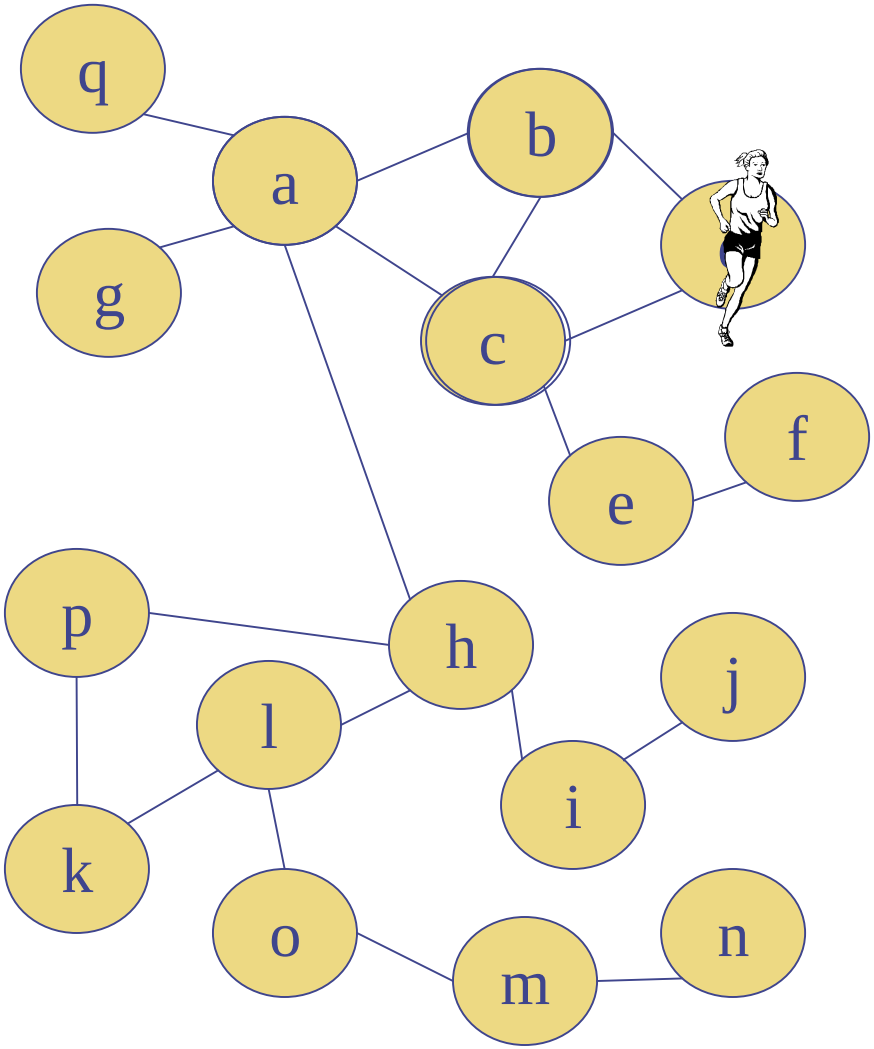
Playing the game



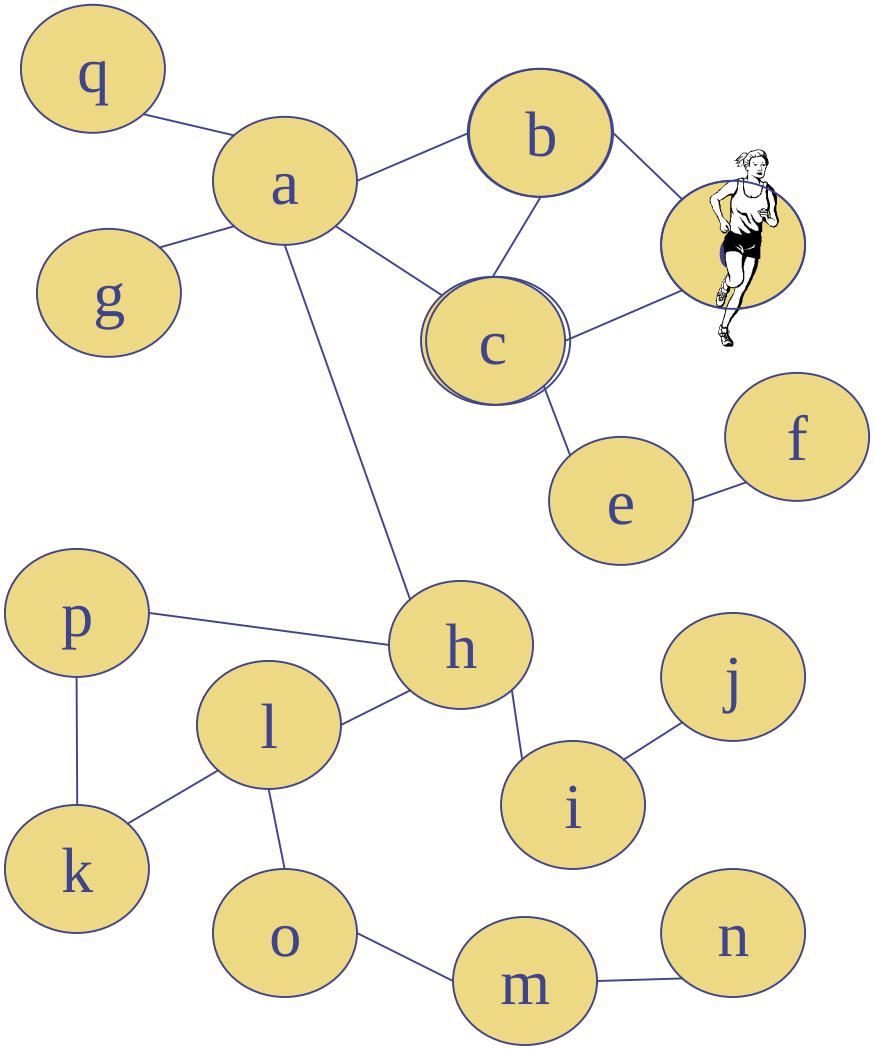
Playing the game



Playing the game



Playing the game



Logical characterization of Treewidth

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

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

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

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

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:

$\mathscr{G}$: class of graphs

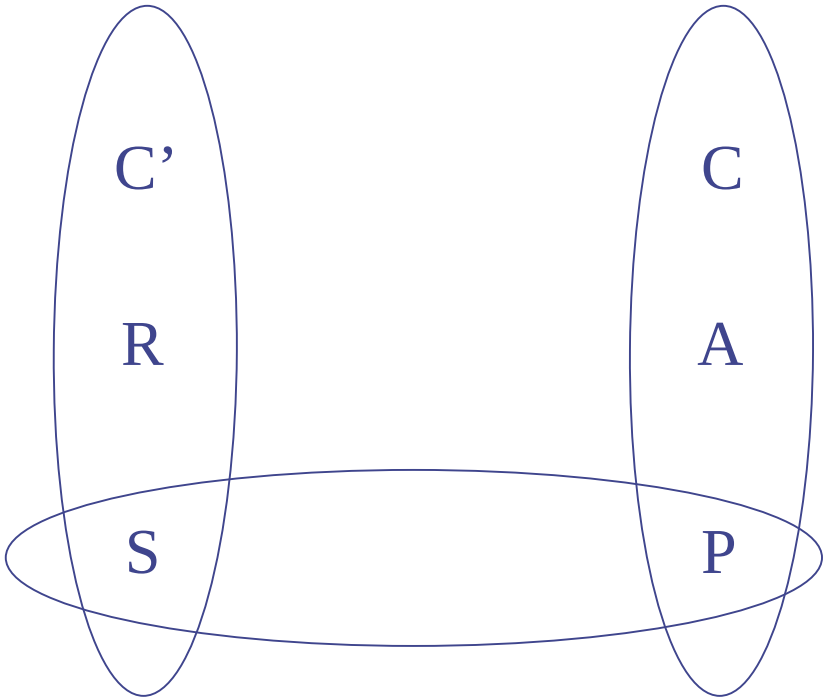
Q($\mathscr{G}$) : all queries characterized by graphs in $\mathscr{G}$

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

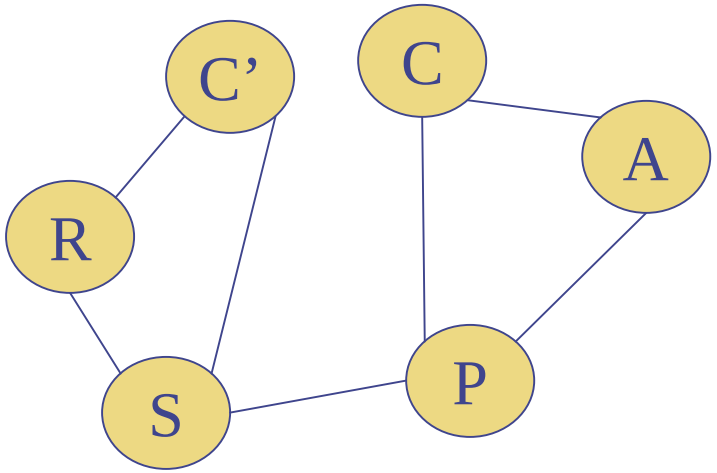
unless P = W[1] or other collapses occur

(Grohe, Schwentick, and Segoufin, '01)

Hypergraphs vs Graphs (1)

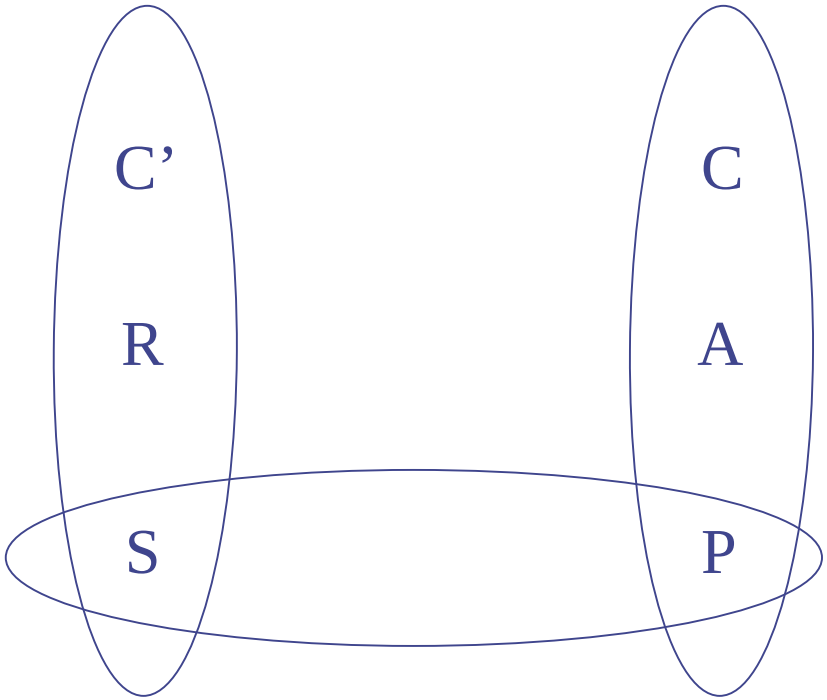


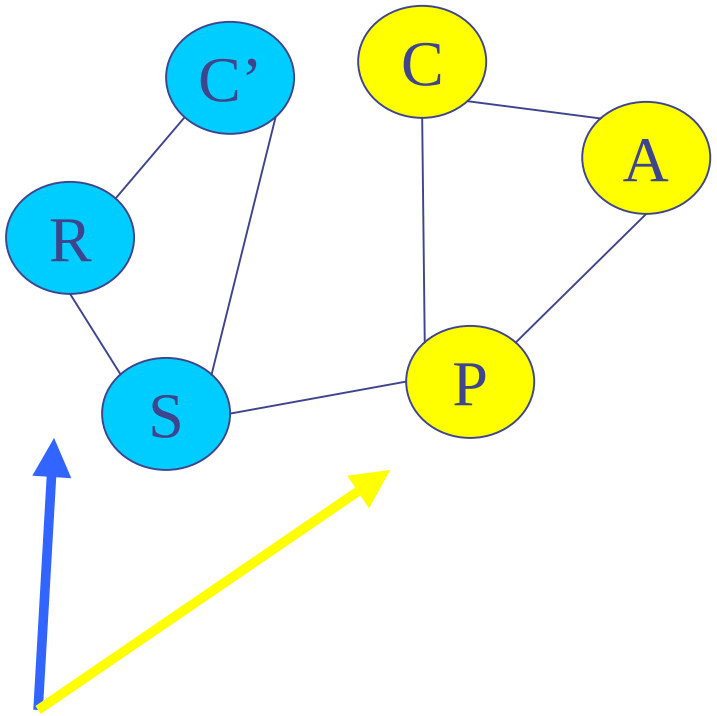
An acyclic hypergraph



Its cyclic primal graph

Hypergraphs vs Graphs (1)





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

Drawbacks of treewidth

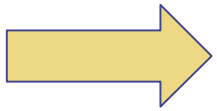
Acyclic queries may have unbounded TW!

Example:

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

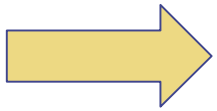
is acyclic, obviously polynomial,
but has treewidth *n-1*

Beyond treewidth



Bounded Degree of Cyclicity

(Gyssens & Paredaens ’84)



Bounded Query width

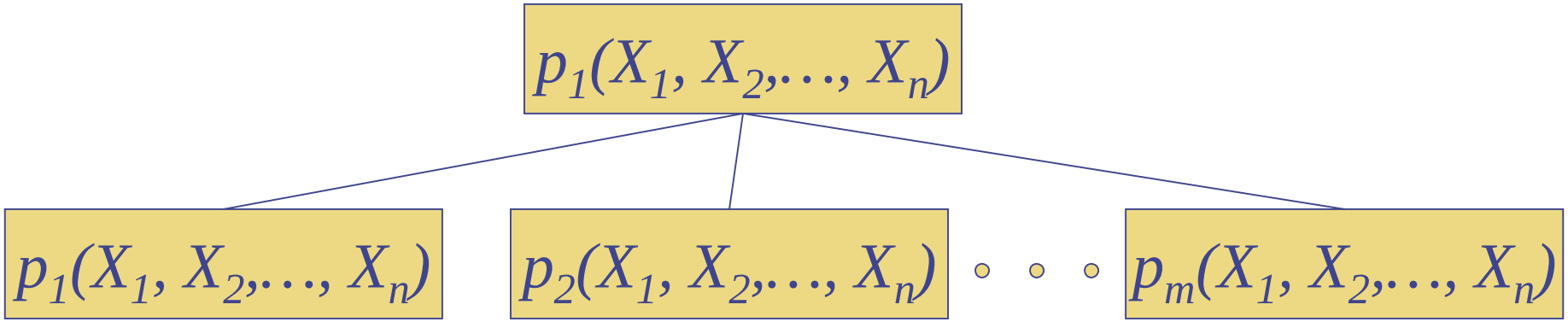
(Chekuri & Rajaraman ’97)

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

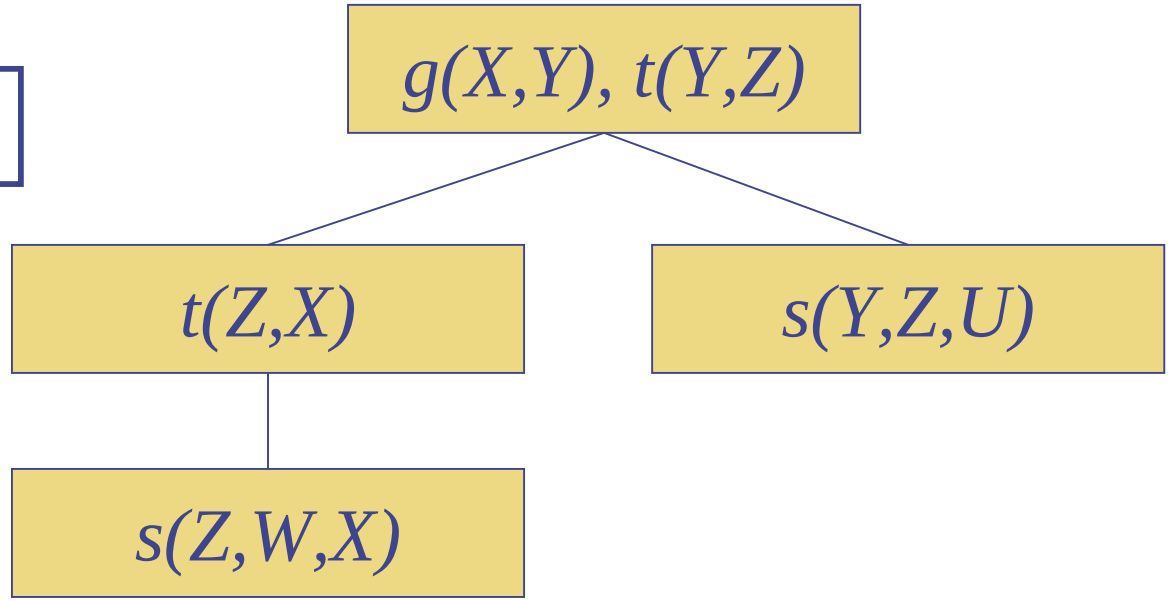


- Every atom appears in some node
- Connectedness conditions for variables and atoms

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

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

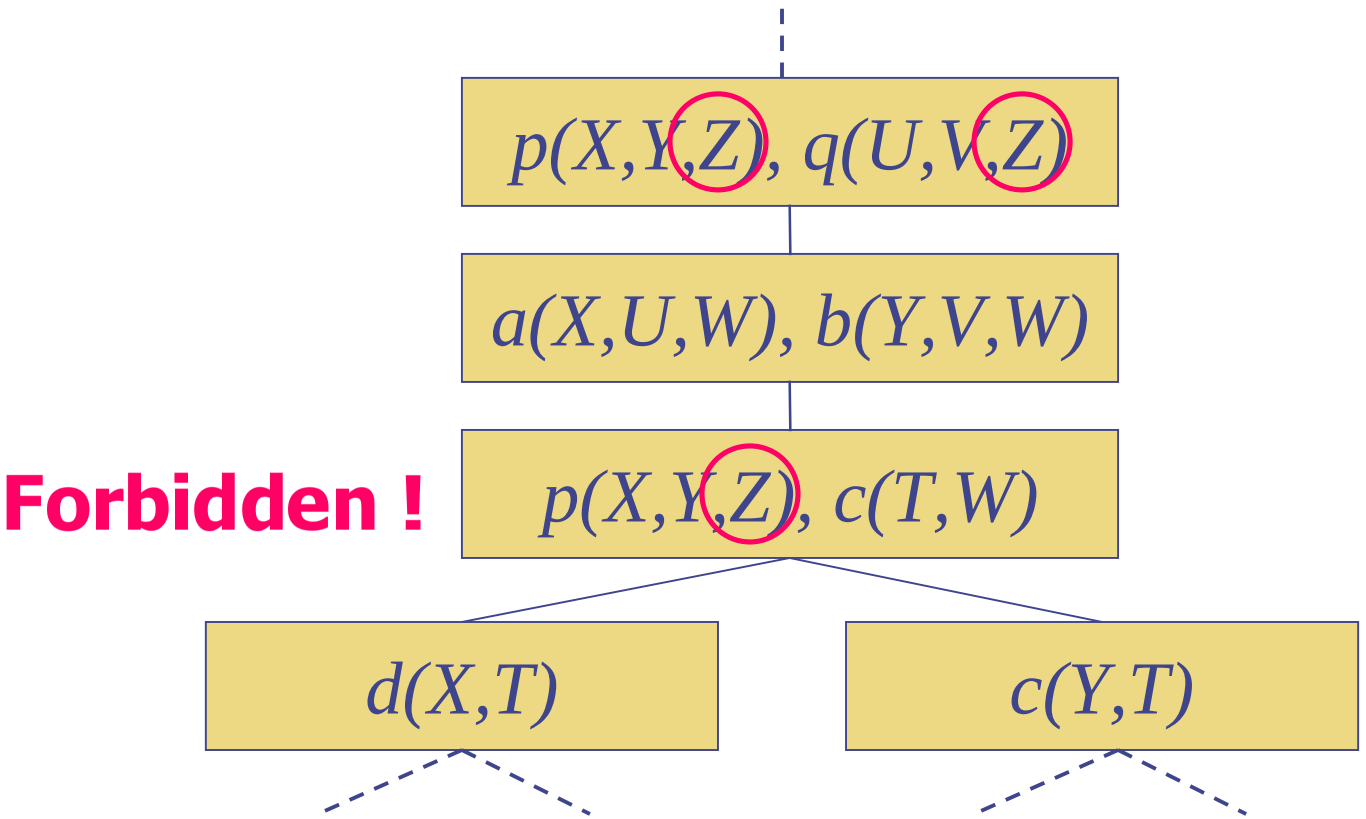
A negative answer (G.L.S. '99)

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

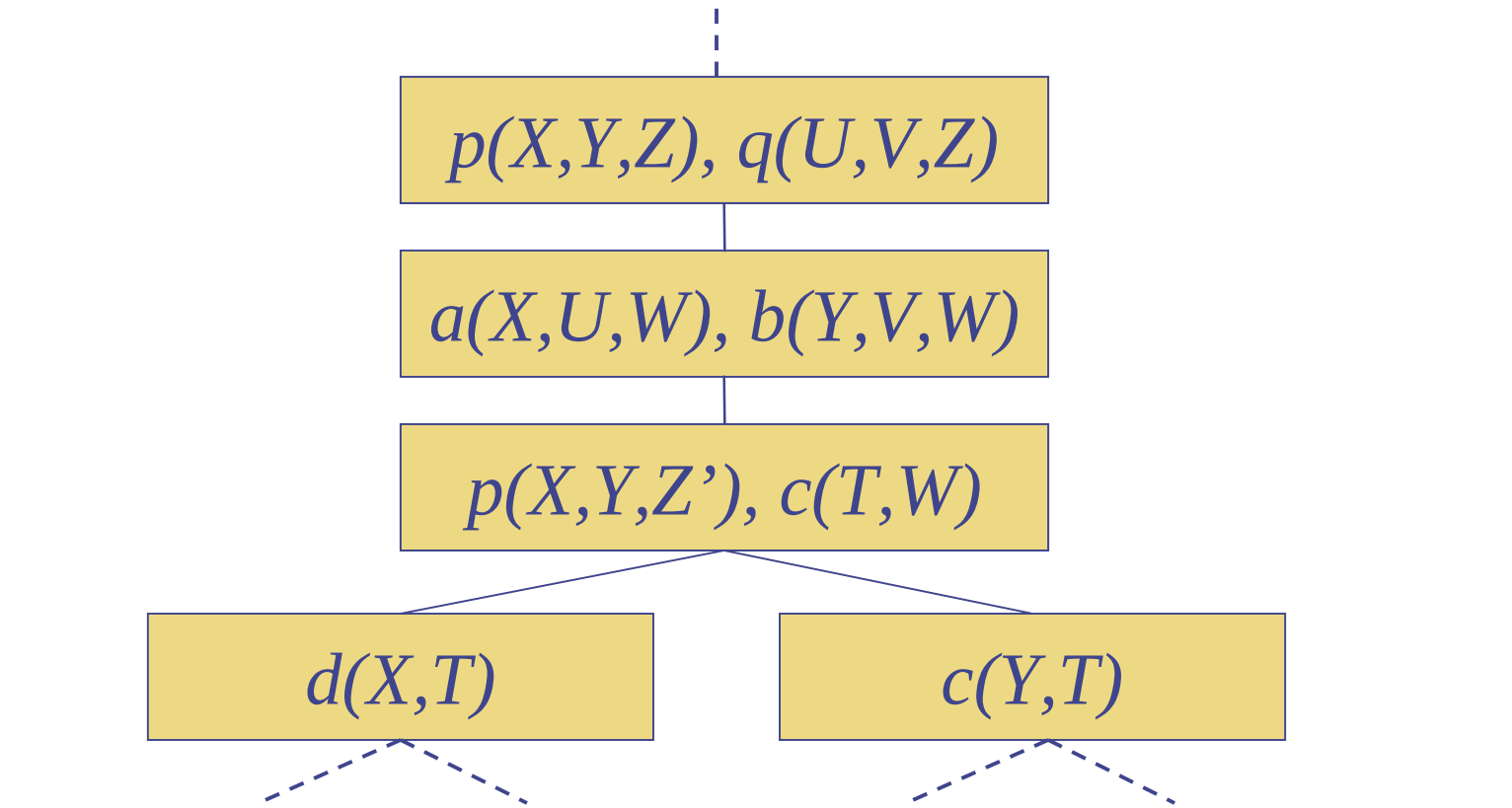
NP-hardness id due to an overly strong condition in the definition of query decomposition



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

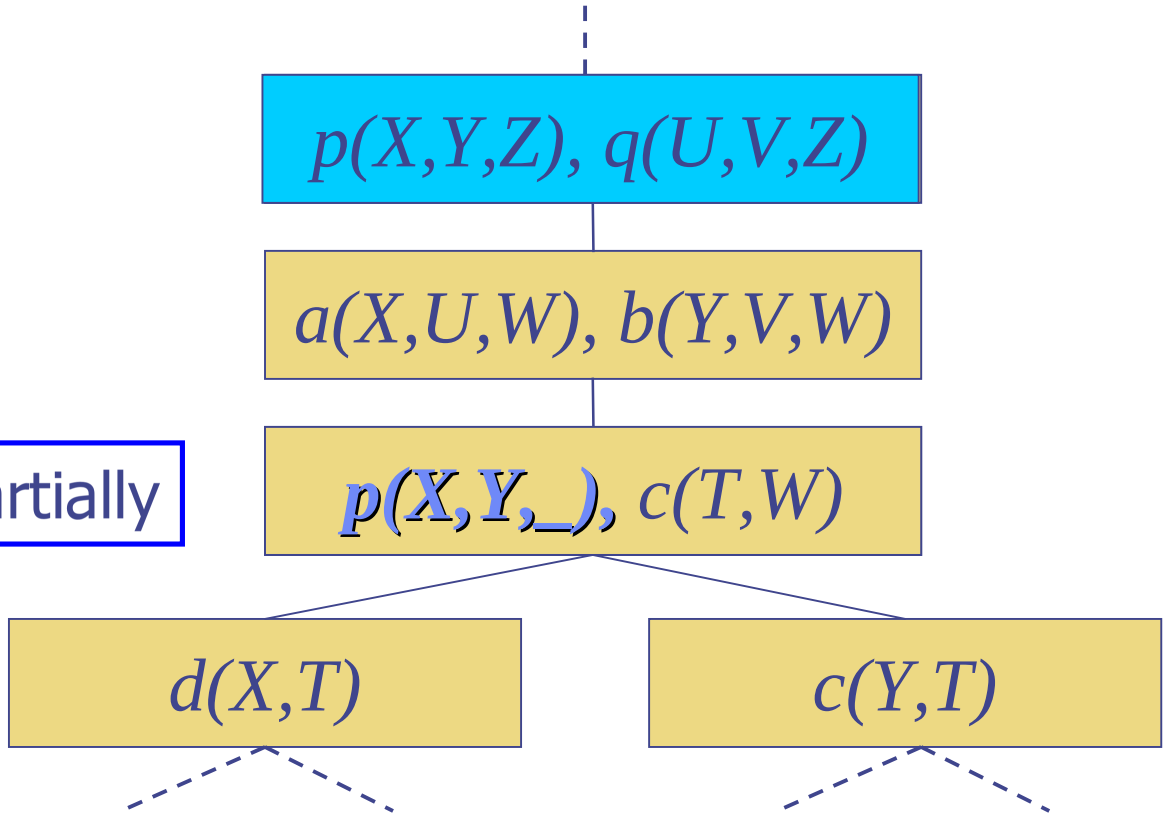
Query atoms can be used “partially” as long as the full atom appears somewhere else

More liberal than query decomposition

Grouping and Reusing Atoms

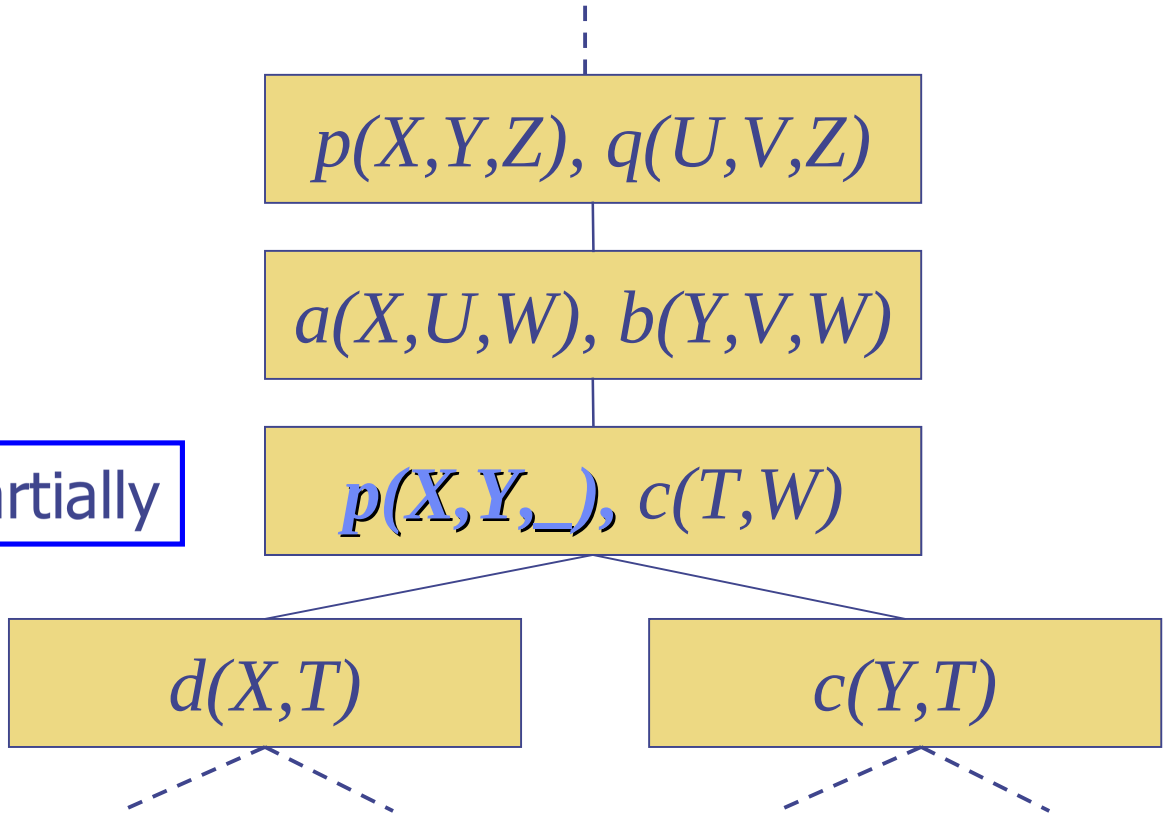
We group atoms

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

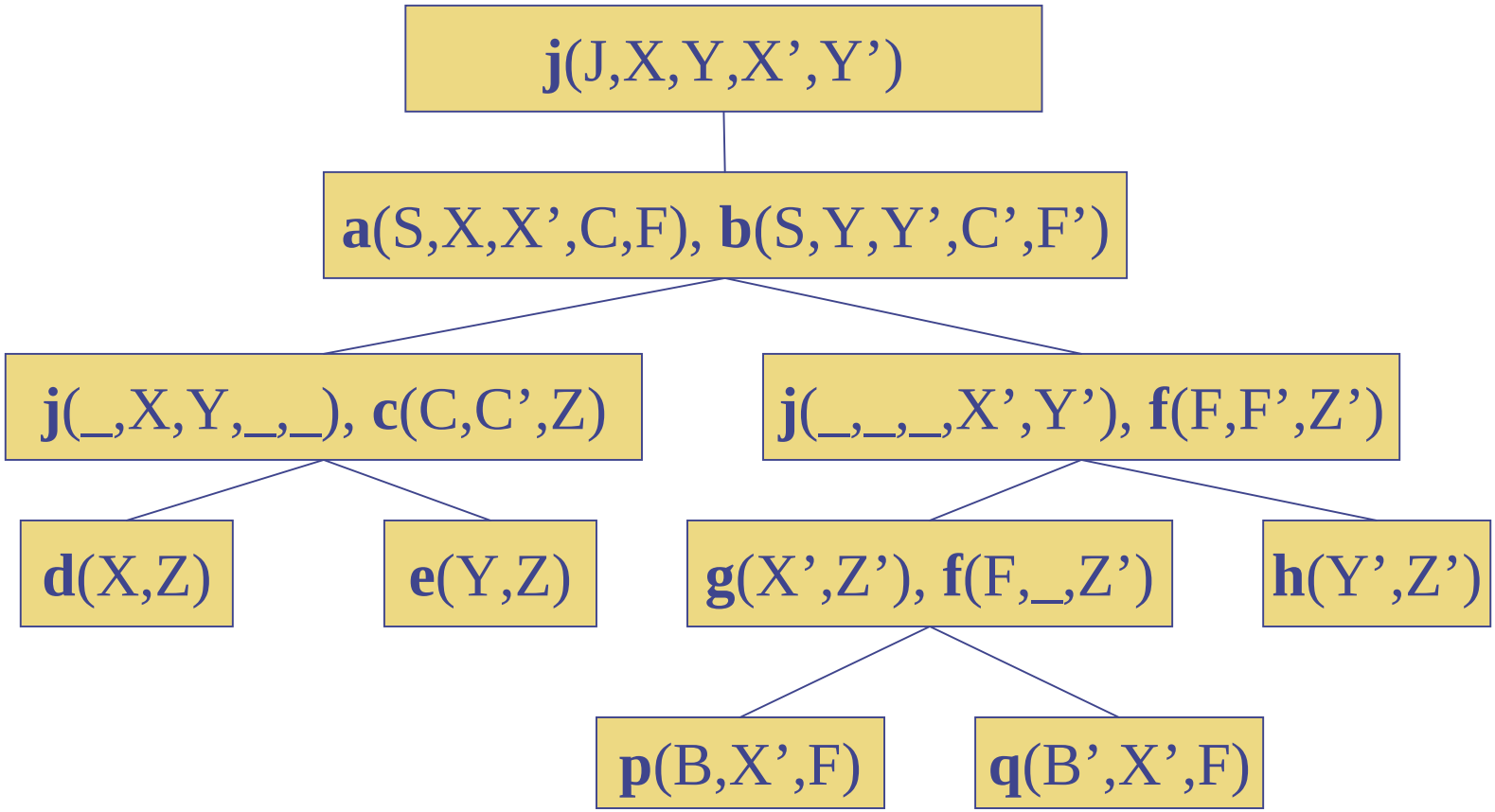


Reusing atoms

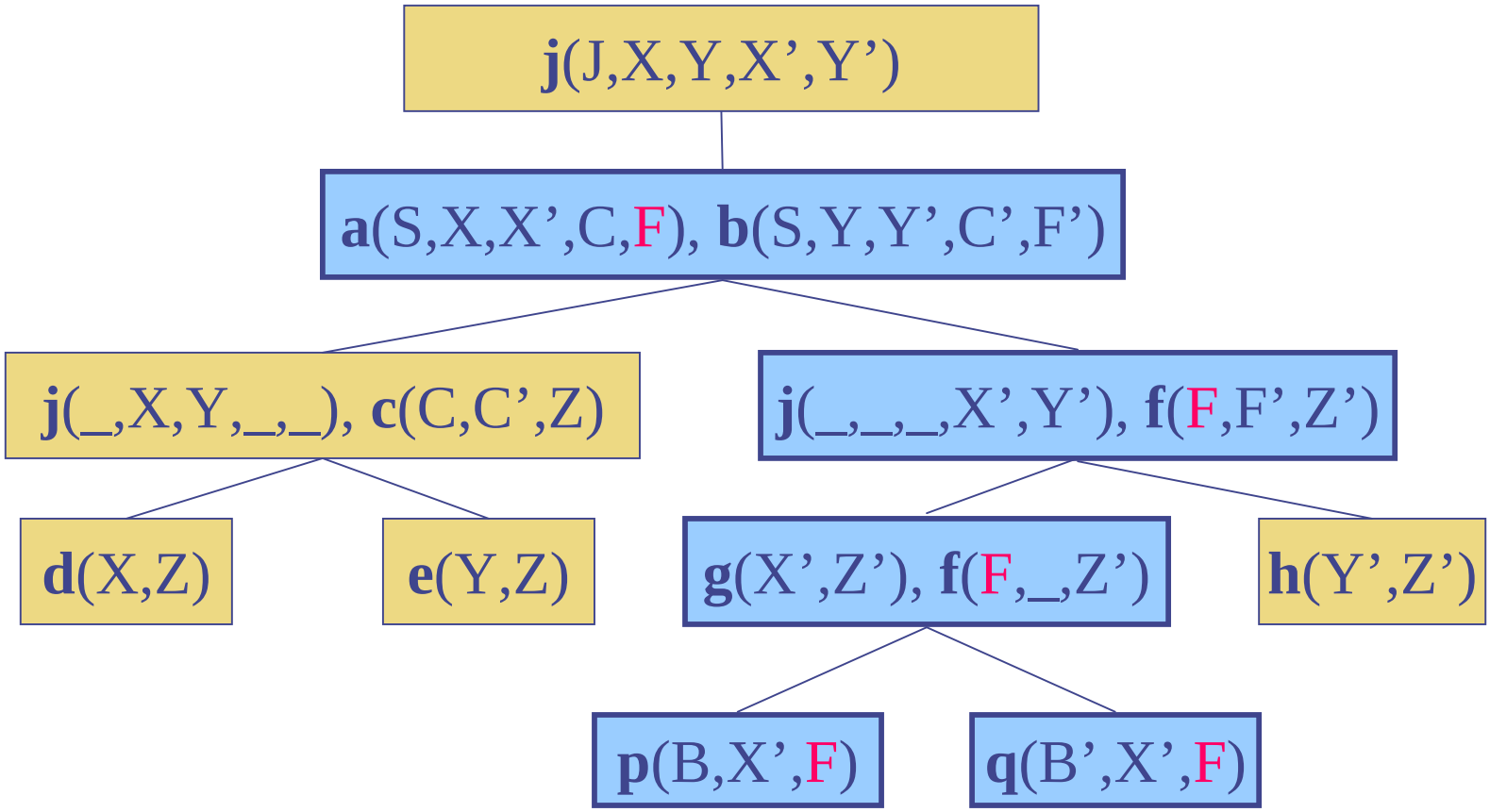
We use *p(X,Y,Z)* partially



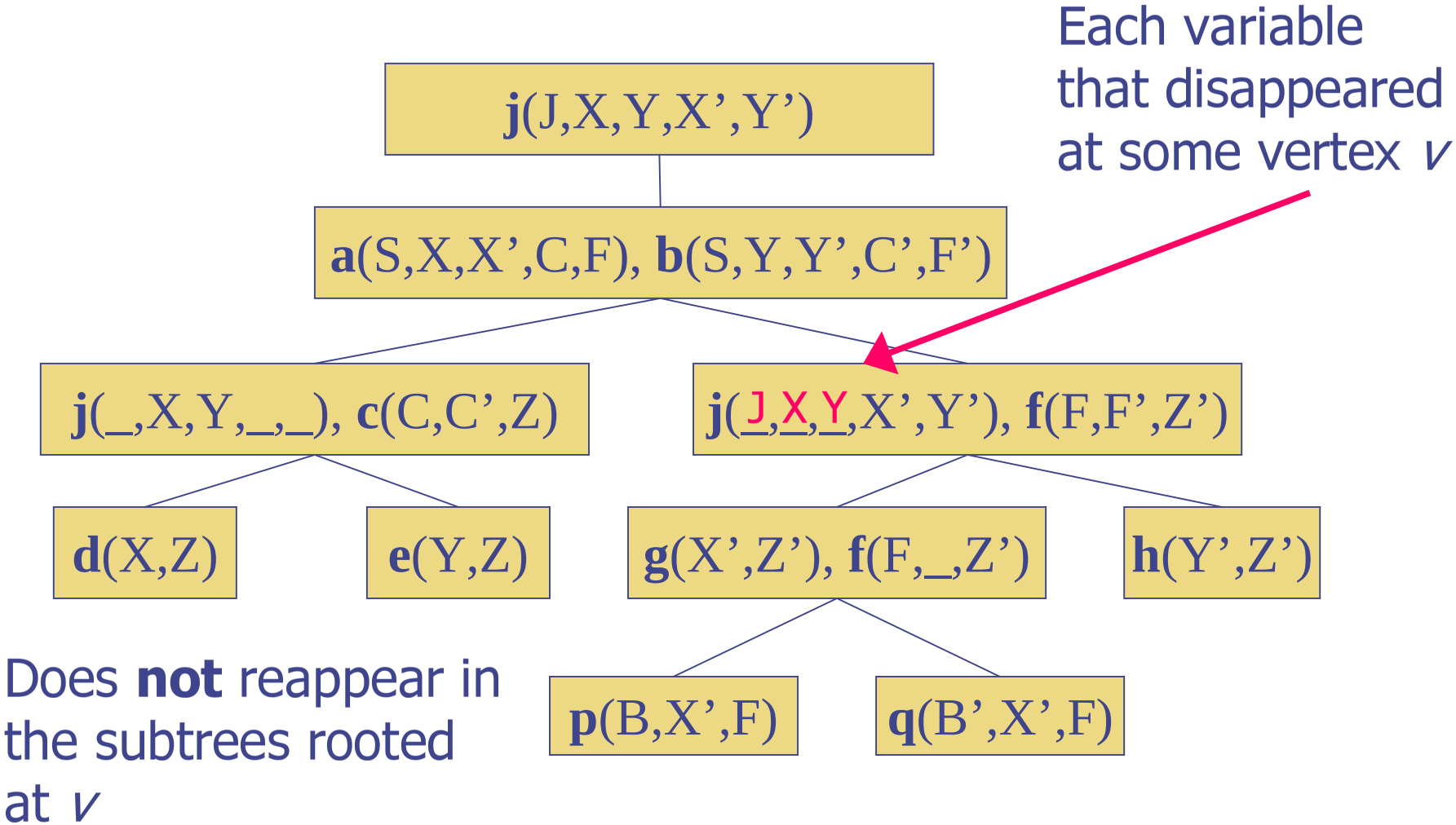
$$ans \leftarrow a(S,X,X',C,F) \wedge b(S,Y,Y',C',F') \wedge c(C,C',Z) \wedge d(X,Z) \wedge$$
$$e(Y,Z) \wedge f(F,F',Z') \wedge g(X',Z') \wedge h(Y',Z') \wedge$$
$$j(J,X,Y,X',Y') \wedge p(B,X',F) \wedge q(B',X',F)$$



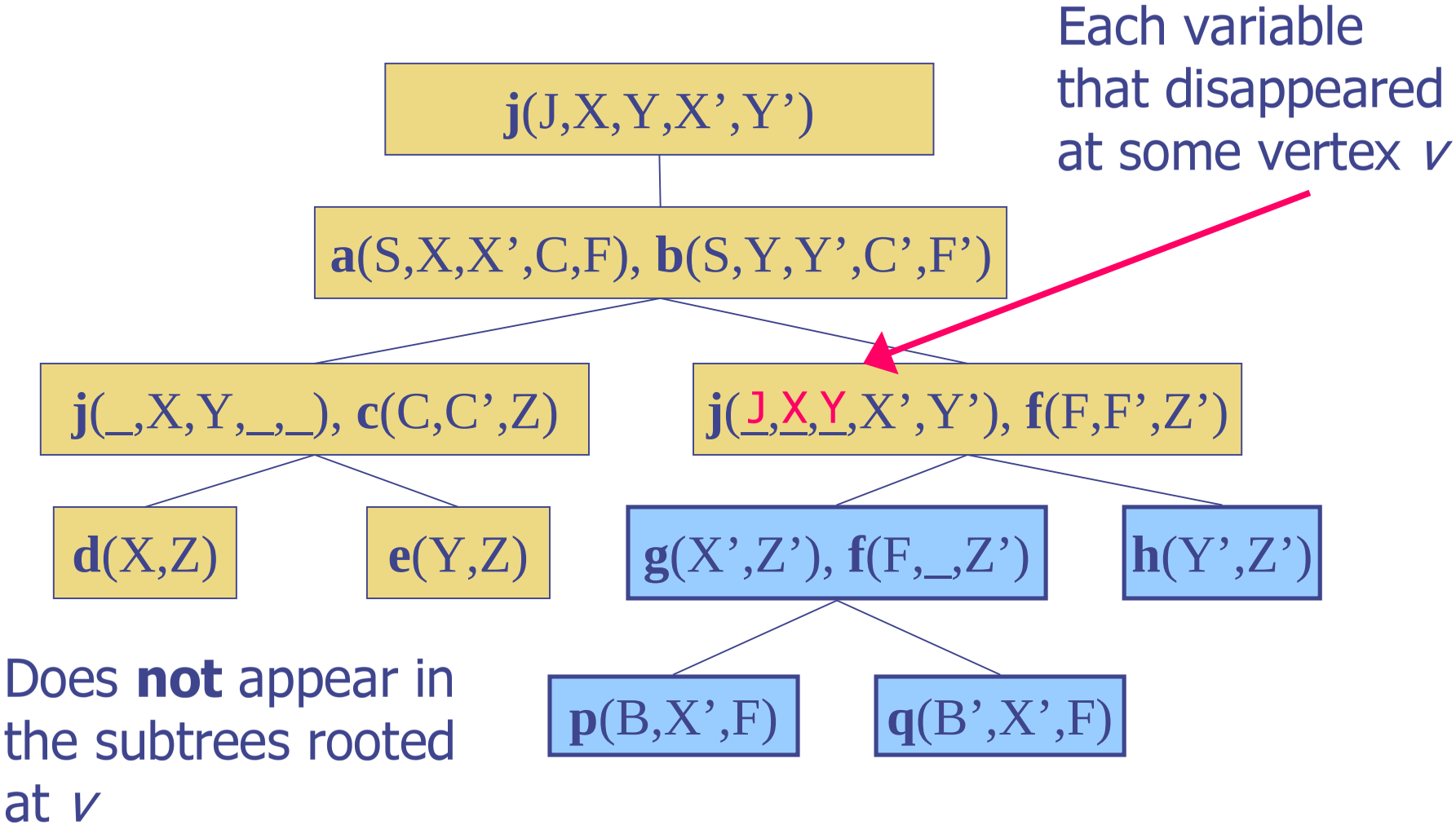
Connectedness Condition



Special Condition



Special Condition



Formal Definition of Hypertree Decomposition

Labeled tree T

Each vertex p of T has labels:

-
-

such that

-
-
-
-

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

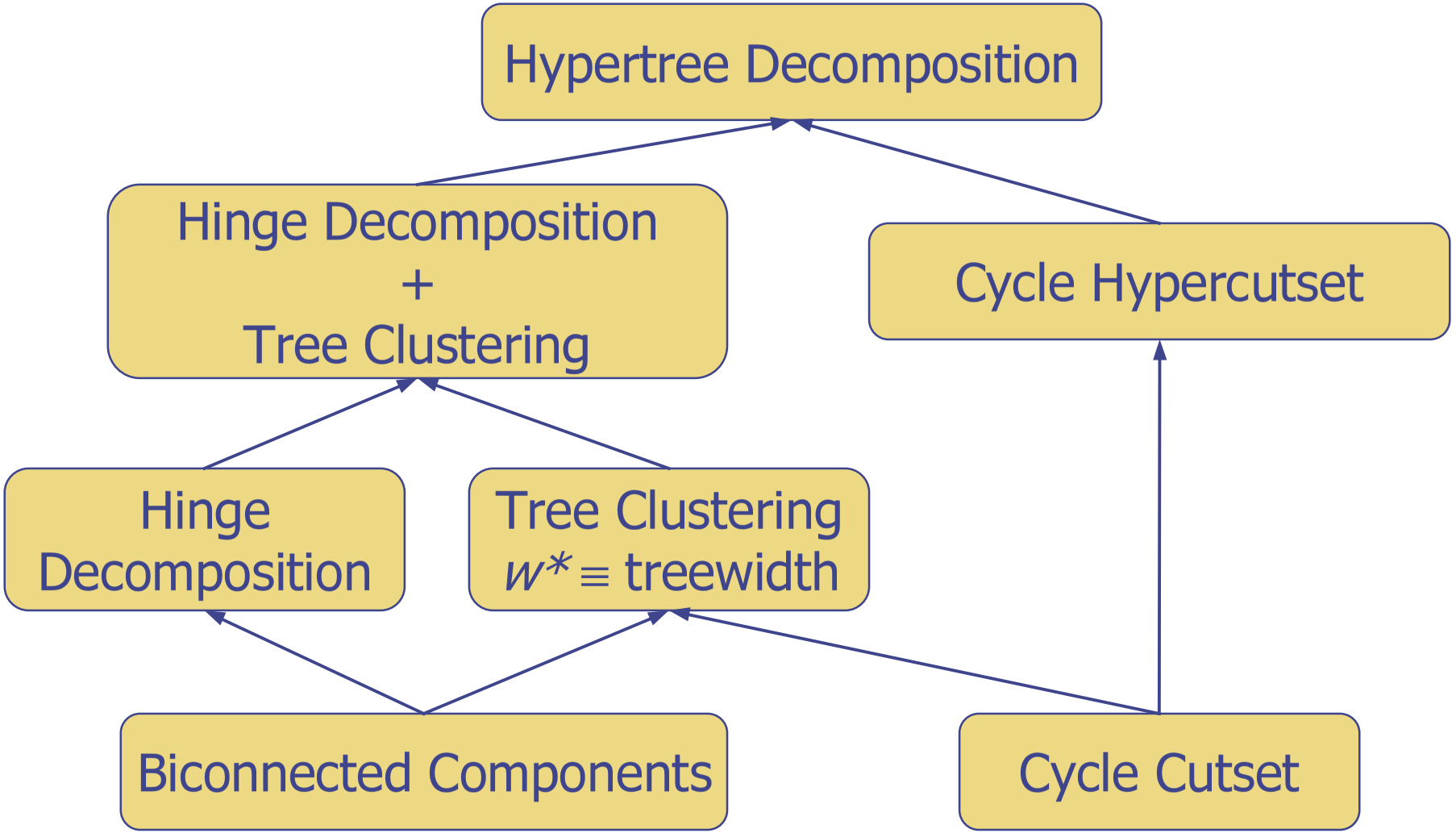
a query *Q* over db such that $hw(Q) \leq k$

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



Characterizations of Hypertree width

- Is hypertree width
  - A natural concept?
  - A natural generalization of hypergraph acyclicity?
- Are there nice characterizations
  - In terms of logic?
  - In terms of games?

Yes !

Characterizations of Hypertree width

- Logical characterization:
  Loosely guarded logic

- Game characterization:
  The robber and marshals game

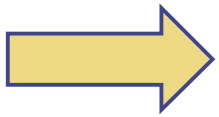
Guarded Formulas

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

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

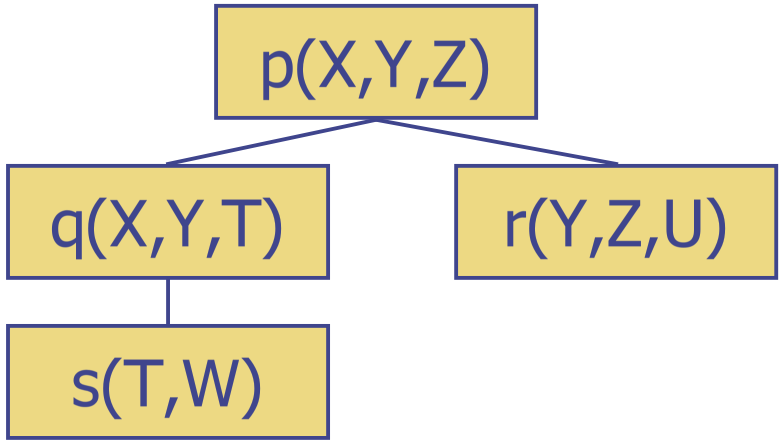
From this general result, we also get a nice logical characterization of acyclic queries:

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

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:

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

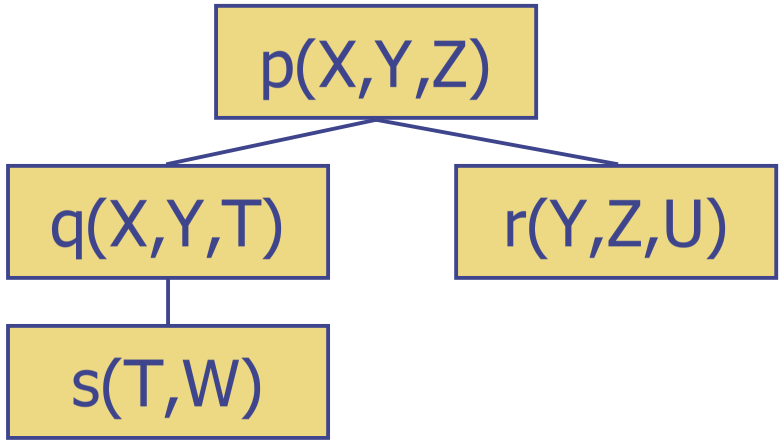
Guard

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:

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

Guard

Guarded subformula

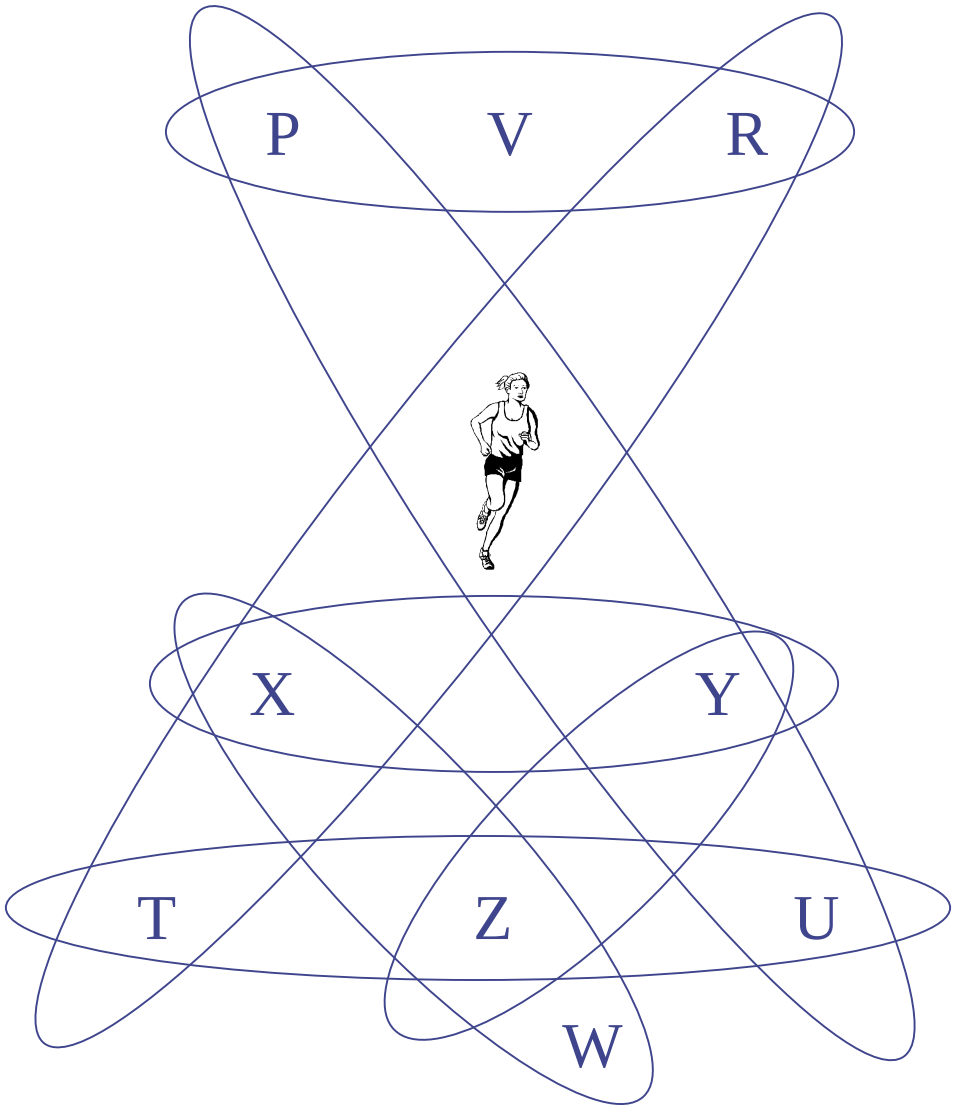
Game characterization: Robber and Marshals

- 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

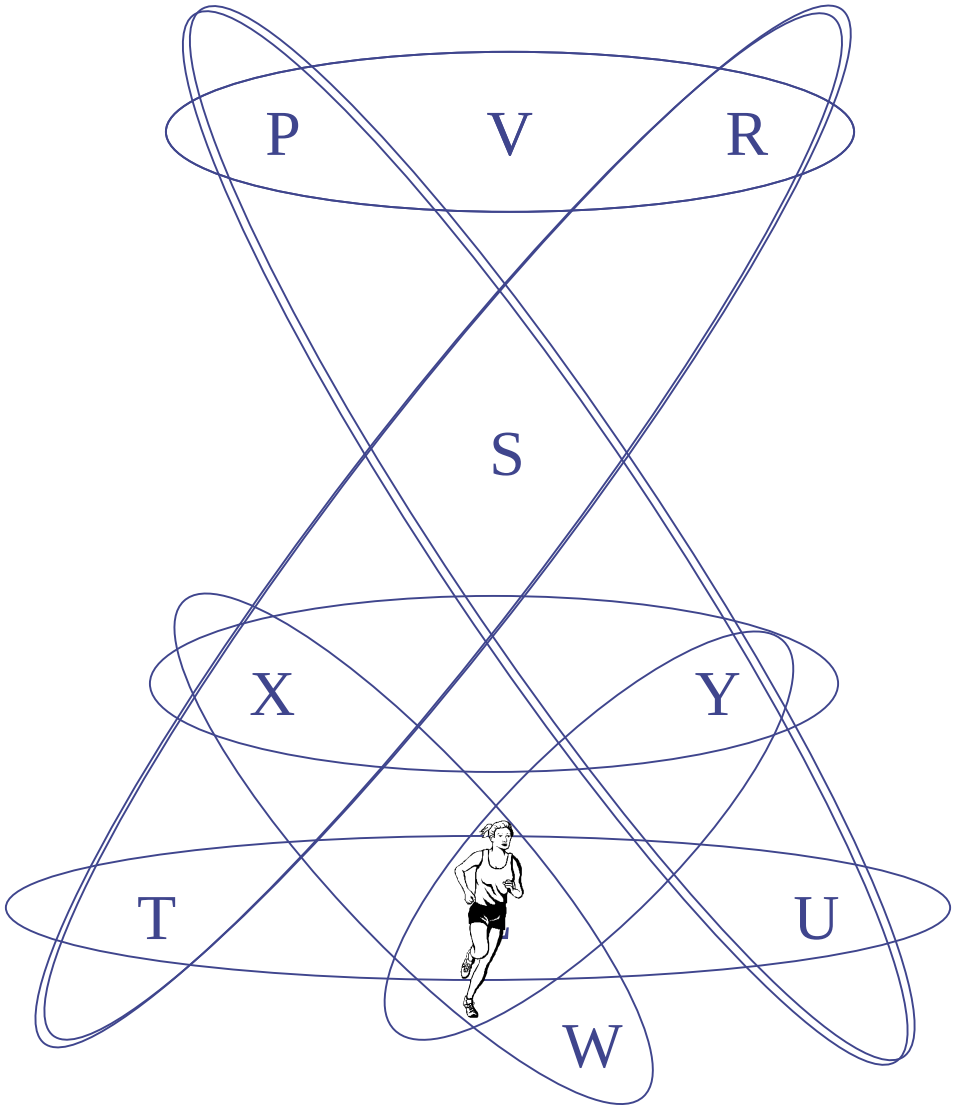
Robbers and Marshals: the rules

- 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

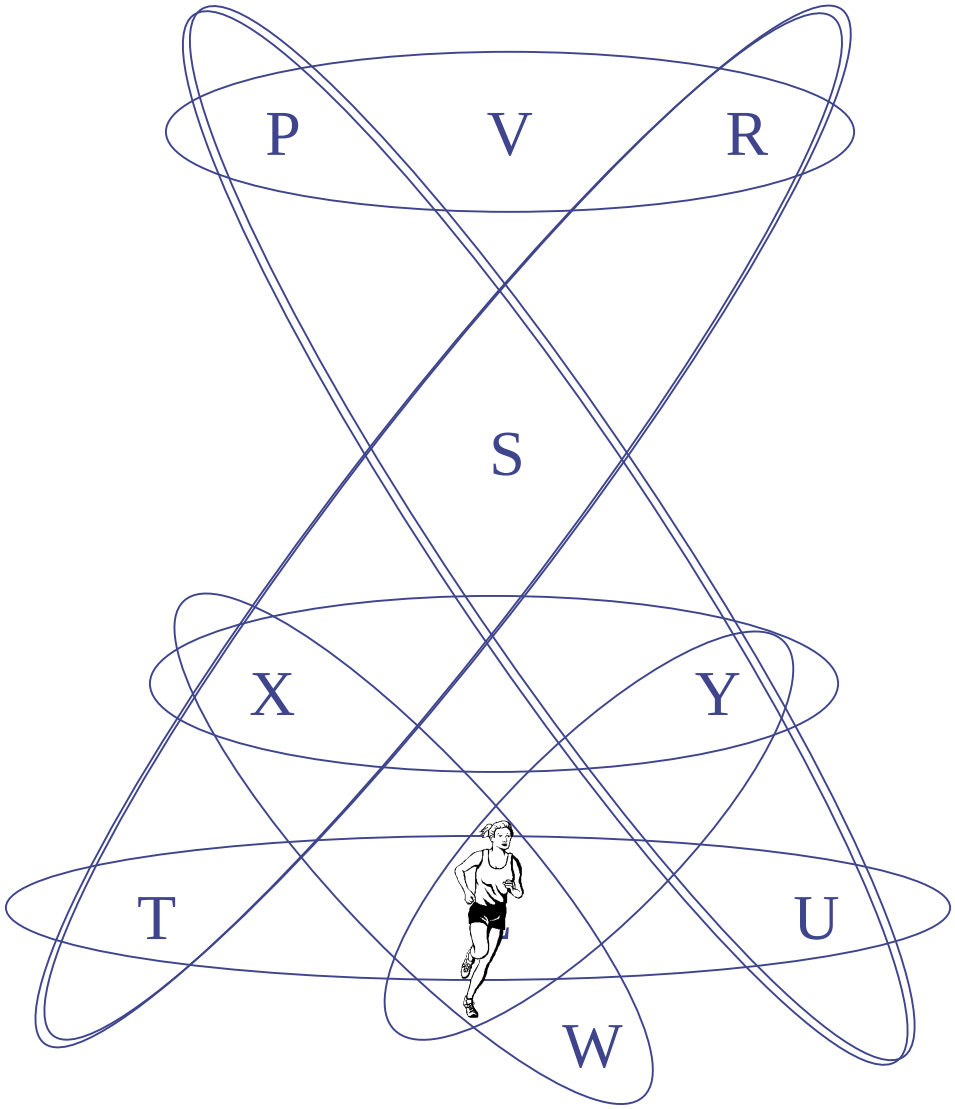
Step 0: the empty hypergraph



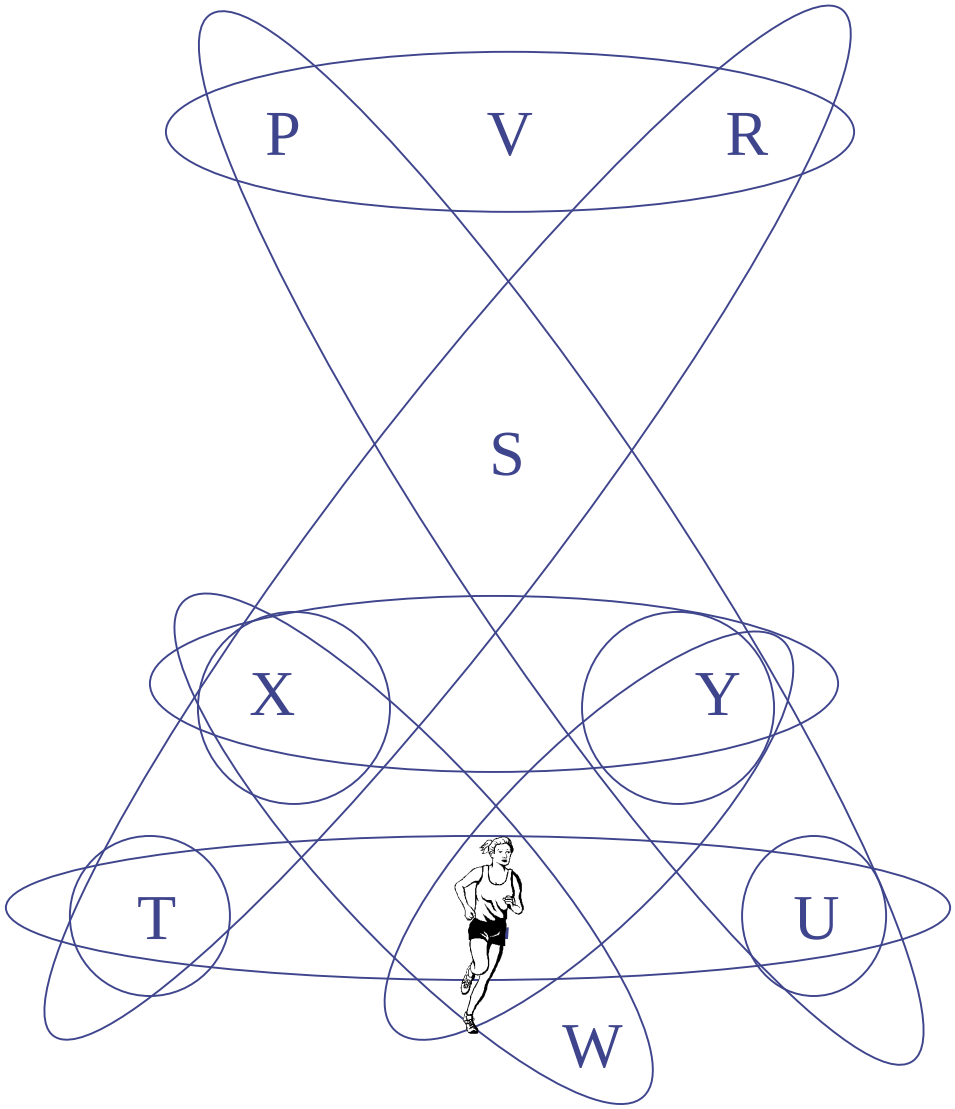
Step 1: first move of the marshals



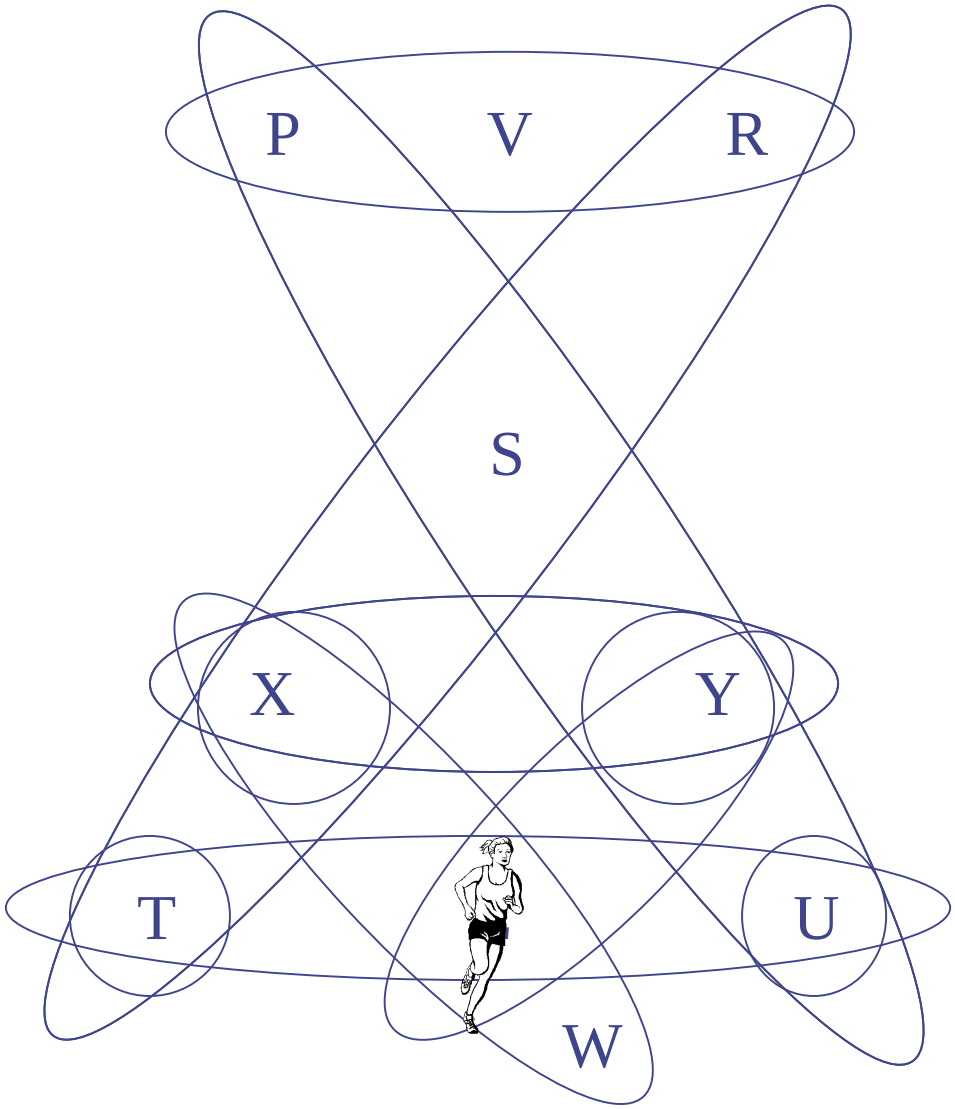
Step 1: first move of the marshals



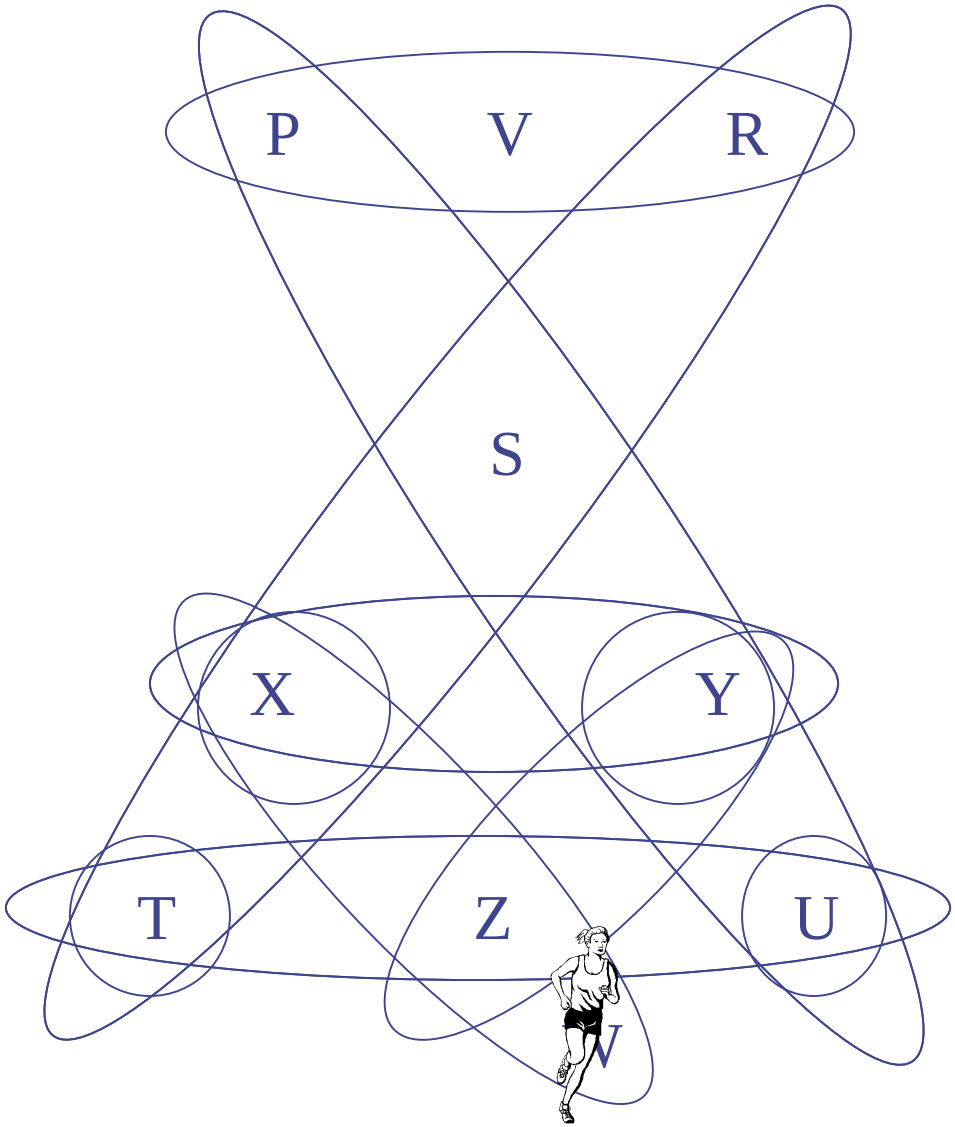
Step 2a: shrinking the space



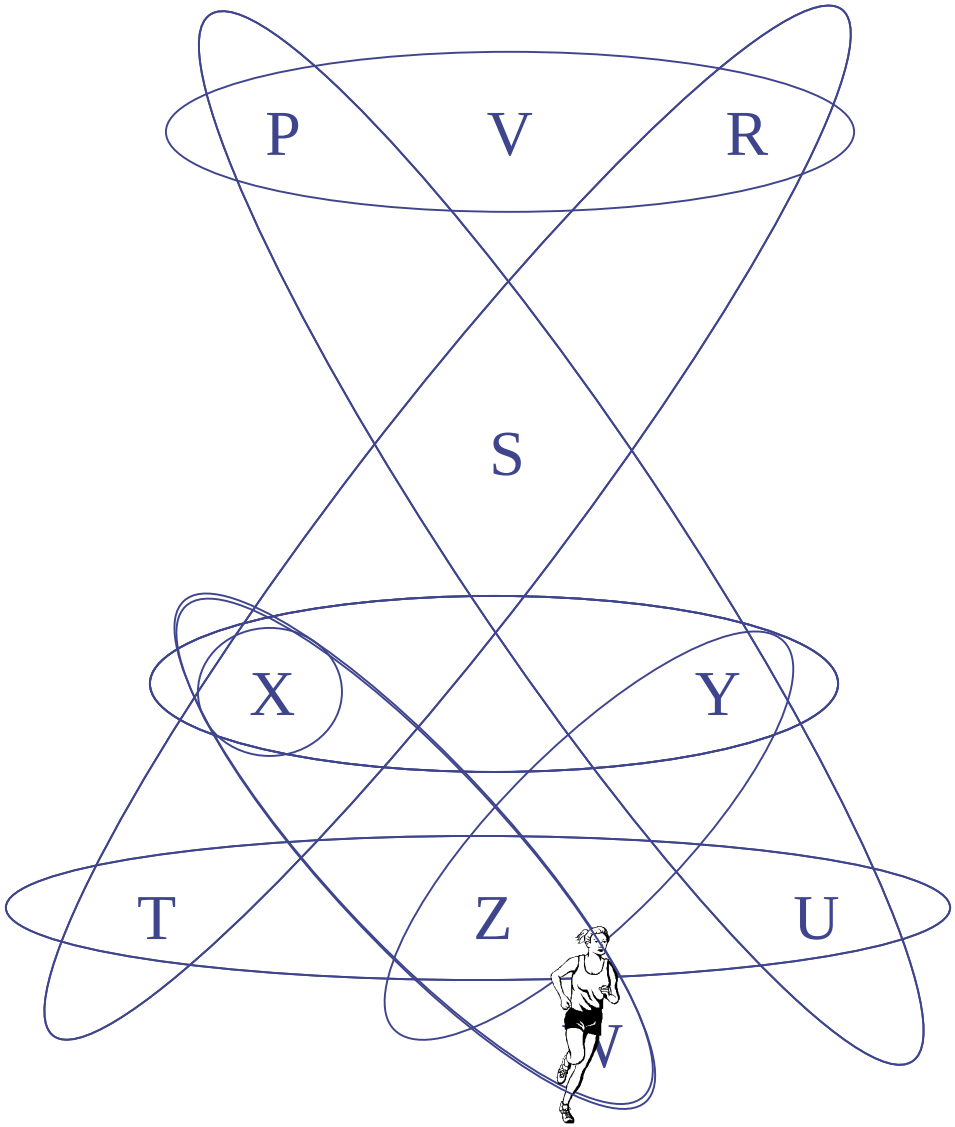
Step 2a: shrinking the space



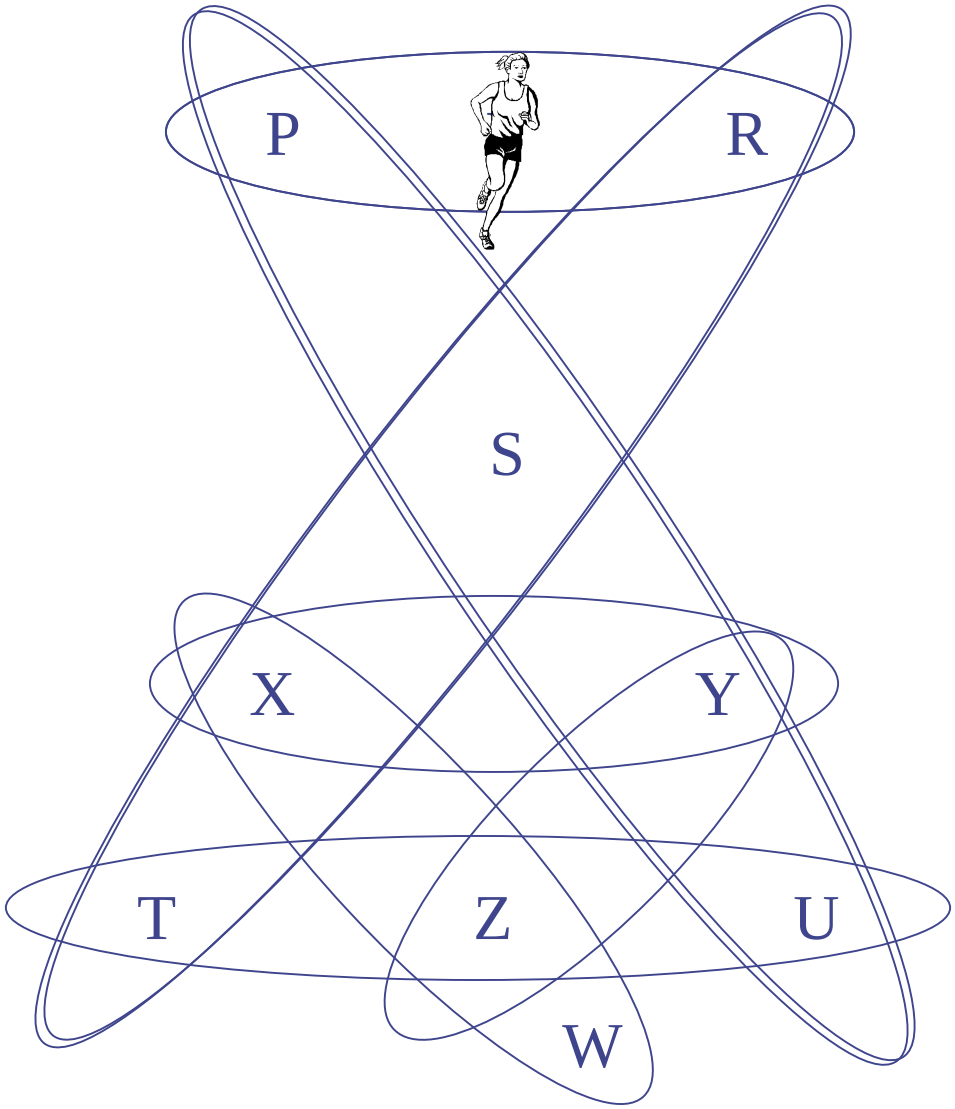
Step 2a: shrinking the space



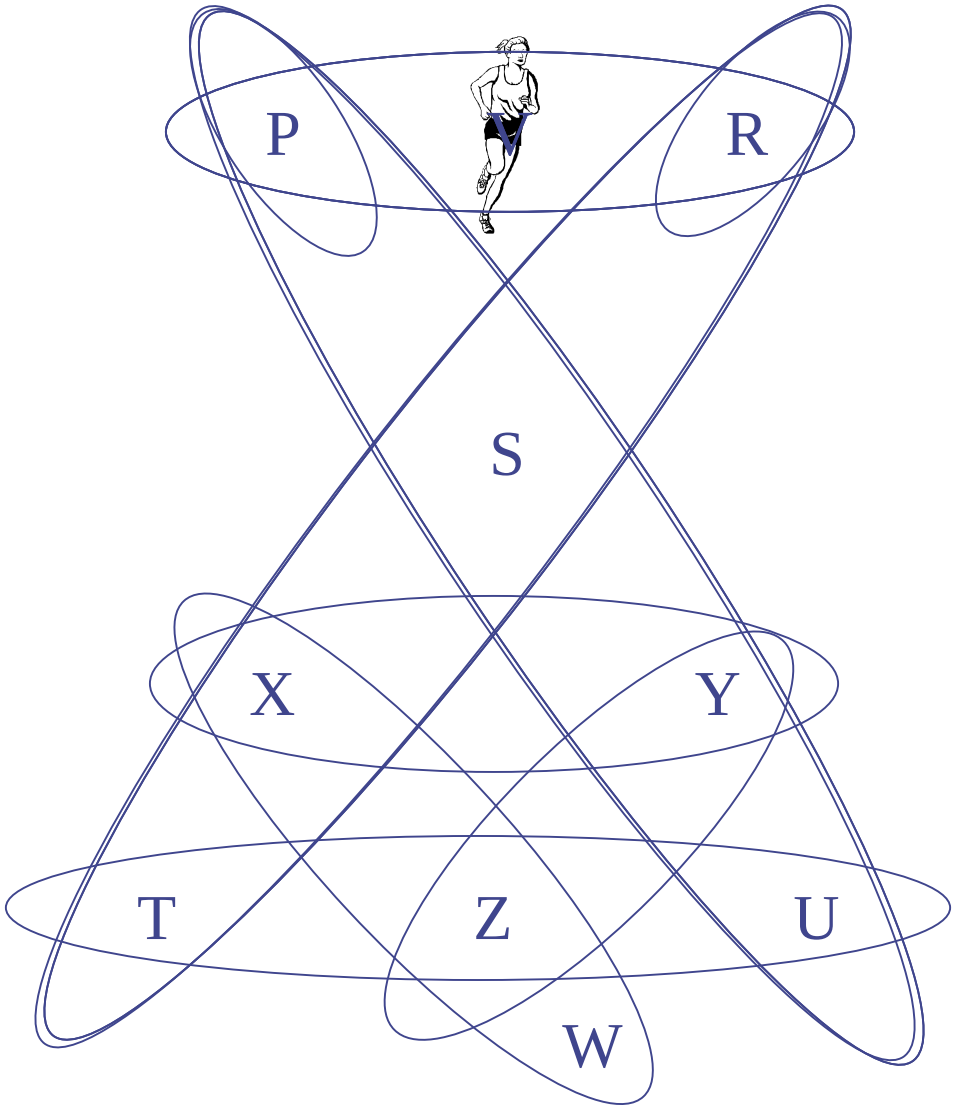
The capture



A different robber’s choice



Step 2b: the capture



R&M Game and Hypertree Width

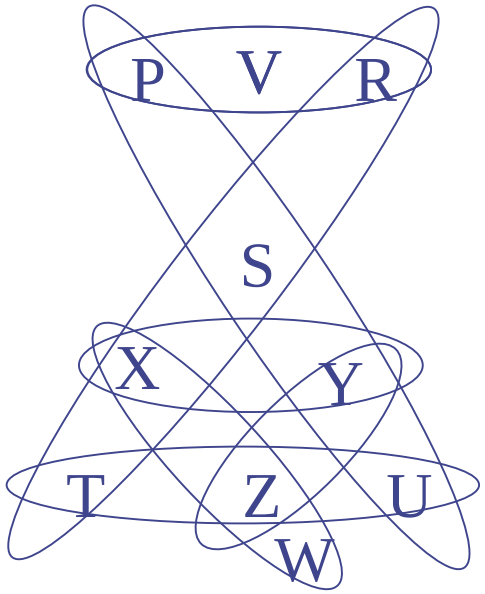
Let H be a hypergraph.

- **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.

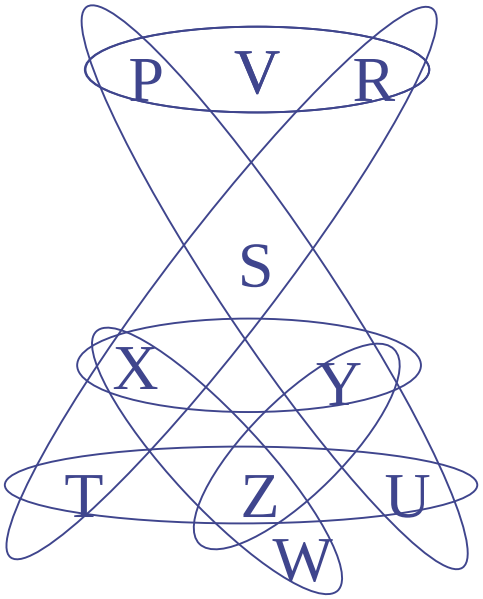
Strategies and Decompositions

$$ans \leftarrow a(S,X,T,R) \wedge b(S,Y,U,P) \wedge c(T,U,Z) \wedge e(Y,Z) \wedge$$
$$g(X,Y) \wedge f(R,P,V) \wedge \wedge d(W,X,Z)$$



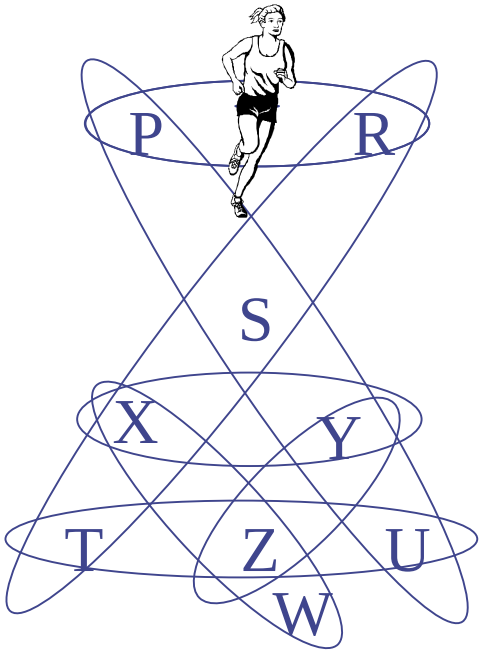
First choice of the two marshals

a(S,X,T,R), **b**(S,Y,U,P)



A possible choice for the robber

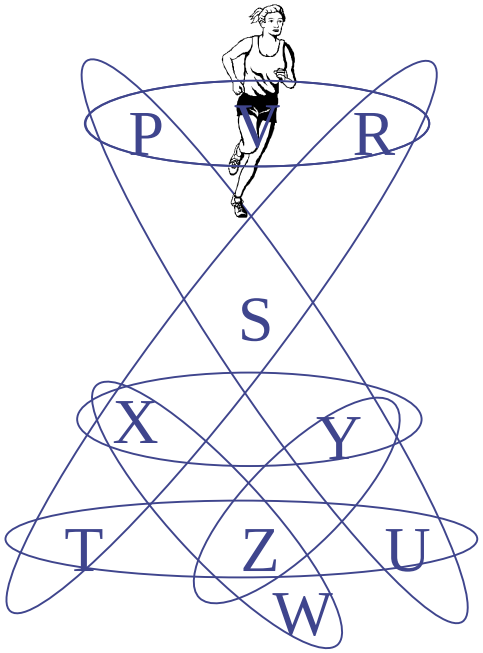
a(S,X,T,R), **b**(S,Y,U,P)



The capture

a(S,X,T,R), **b**(S,Y,U,P)

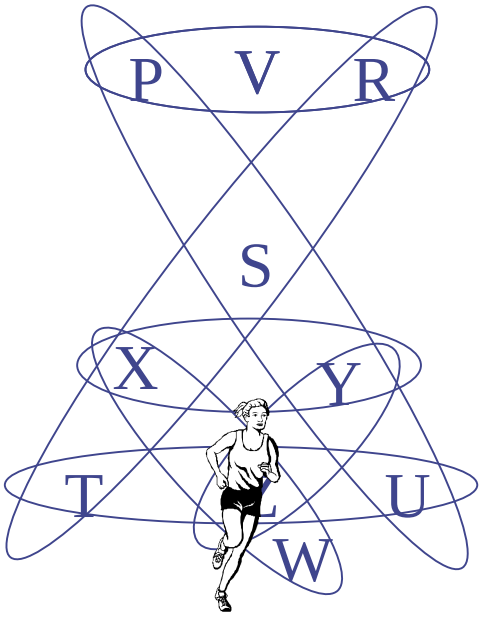
f(R,P,V)



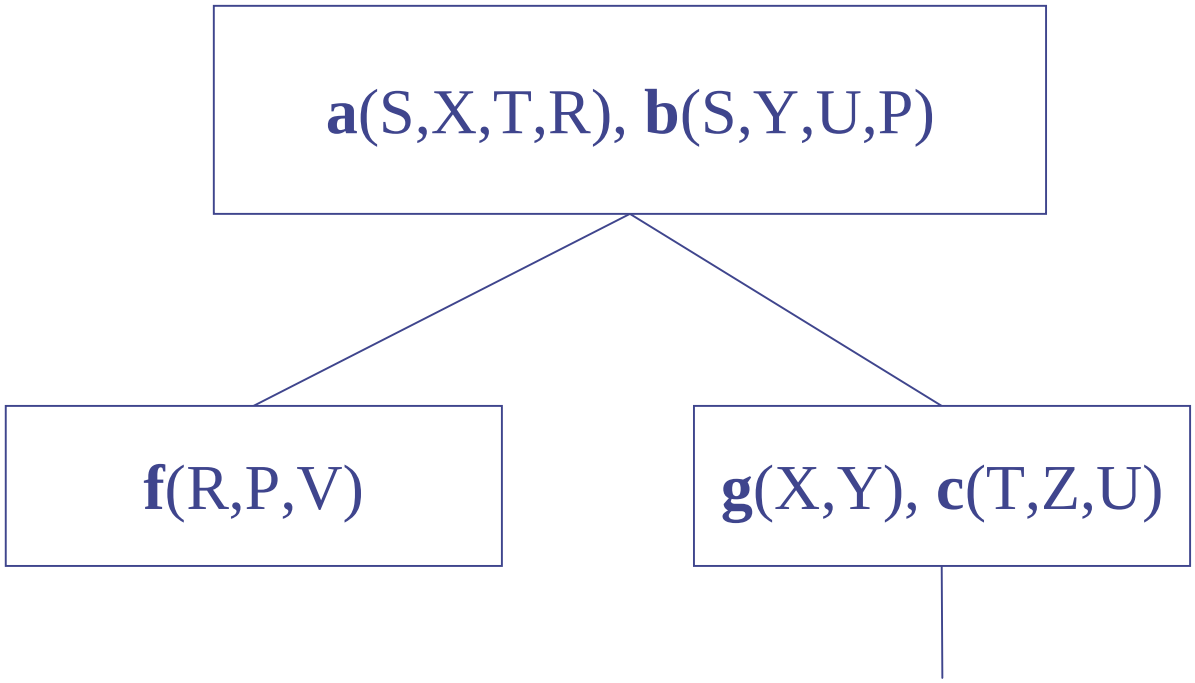
The second choice for the robber

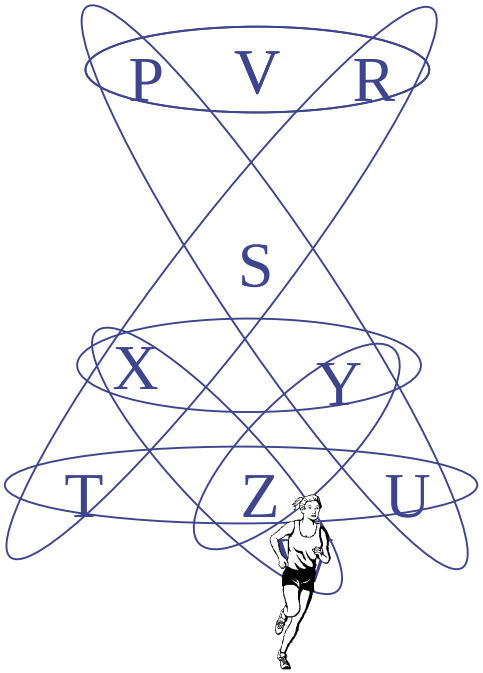
a(S,X,T,R), **b**(S,Y,U,P)

f(R,P,V)

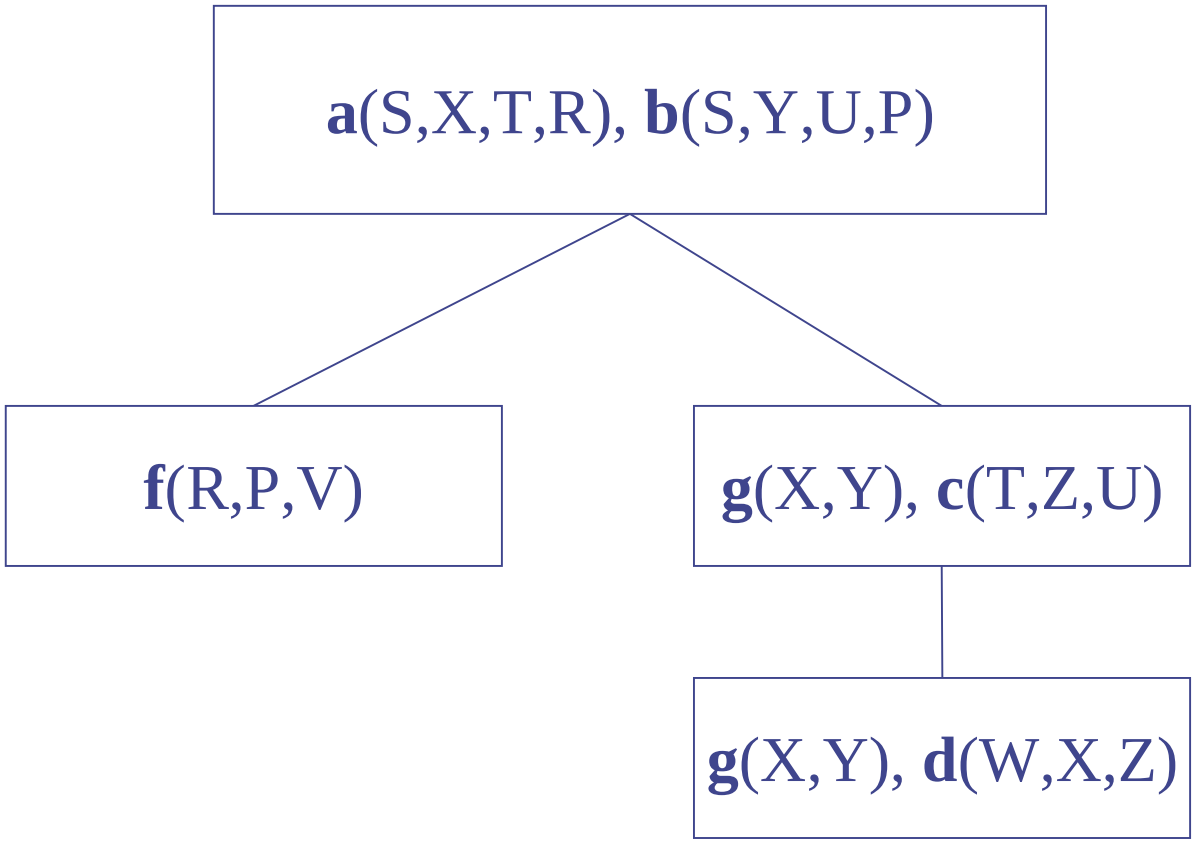


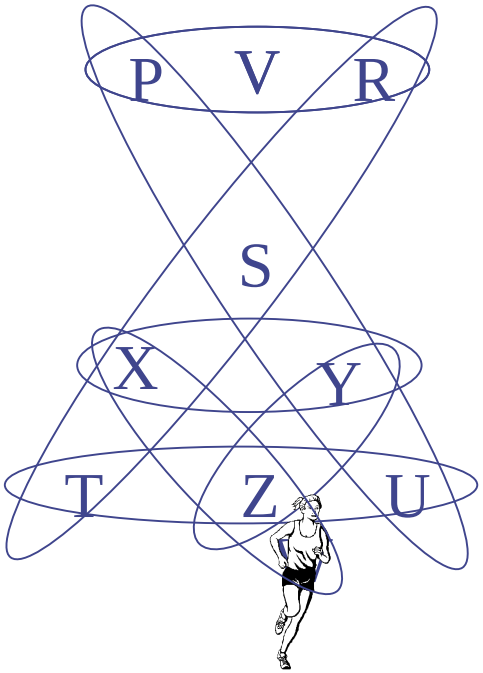
The marshals corner the robber





The capture





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/