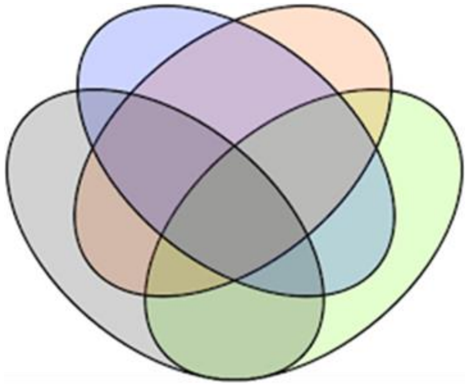
FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

dbai

Acyclity and Notions of "Width" of Hypergraphs



By Rupert Millard after Venn

Reinhard Pichler
TU Wien

Joint work with several co-authors
(Fischl/Gottlob/Lanzinger/Longo/Okulmus from TU Wien + Razgon from Birkbeck U.)

Simons Institute, 18 September, 2023

Roadmap

- 3 Problems: HOM, CSP, BCQ
- Hypergraphs and acyclicity
- Well-known width notions: tw, hw, ghw, fhw
- hw vs. ghw
- Tractable cases of ghw (and fhw) computation
- NP-hardness of the Check-problem for ghw and fhw
- A glimpse beyond fhw

Roadmap

- 3 Problems: HOM, CSP, BCQ
- Hypergraphs and acyclicity
- Well-known width notions: tw, hw, ghw, fhw
- hw vs. ghw
- Tractable cases of ghw (and fhw) computation
- NP-hardness of the Check-problem for ghw and fhw
- A glimpse beyond fhw

Three Problems

- BCQ: Boolean Conjunctive Query Evaluation
- CSP: Constraint Satisfaction Problem
- HOM: Homomorphism Problem

All these problems are essentially the same.

All these problems are based on hypergraphs.

HOM: Homomorphism Problem

Given two relational structures

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

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

Decide whether there exists a homomorphism *h* from *A* to *B*

$$h: U \longrightarrow V$$

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

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

[Feder and Vardi 1993] Relationship to CSP, restrictions on B
[Kolaitis and Vardi 1998] Relationship to Query Containment, restrictions on A, B

CSP: Constraint Satisfaction Problem

Set of variables $V=\{X_1,\ldots,X_n\}$, domain D, and 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 the CSP: A substitution $h: V \rightarrow D$ such that $\forall i: h(S_i) \in R_i$

BCQ: Boolean Conjunctive Query Evaluation

DATABASE:

Enrolled

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

Teaches

McLane	Algebra	March
Verdi	Logic	May
Lausen	DB	June
Rahm	DB	May
………	…..	…….

Parent

McLane	Lisa
Verdi	Robert
Rahm	Mary
………	…..

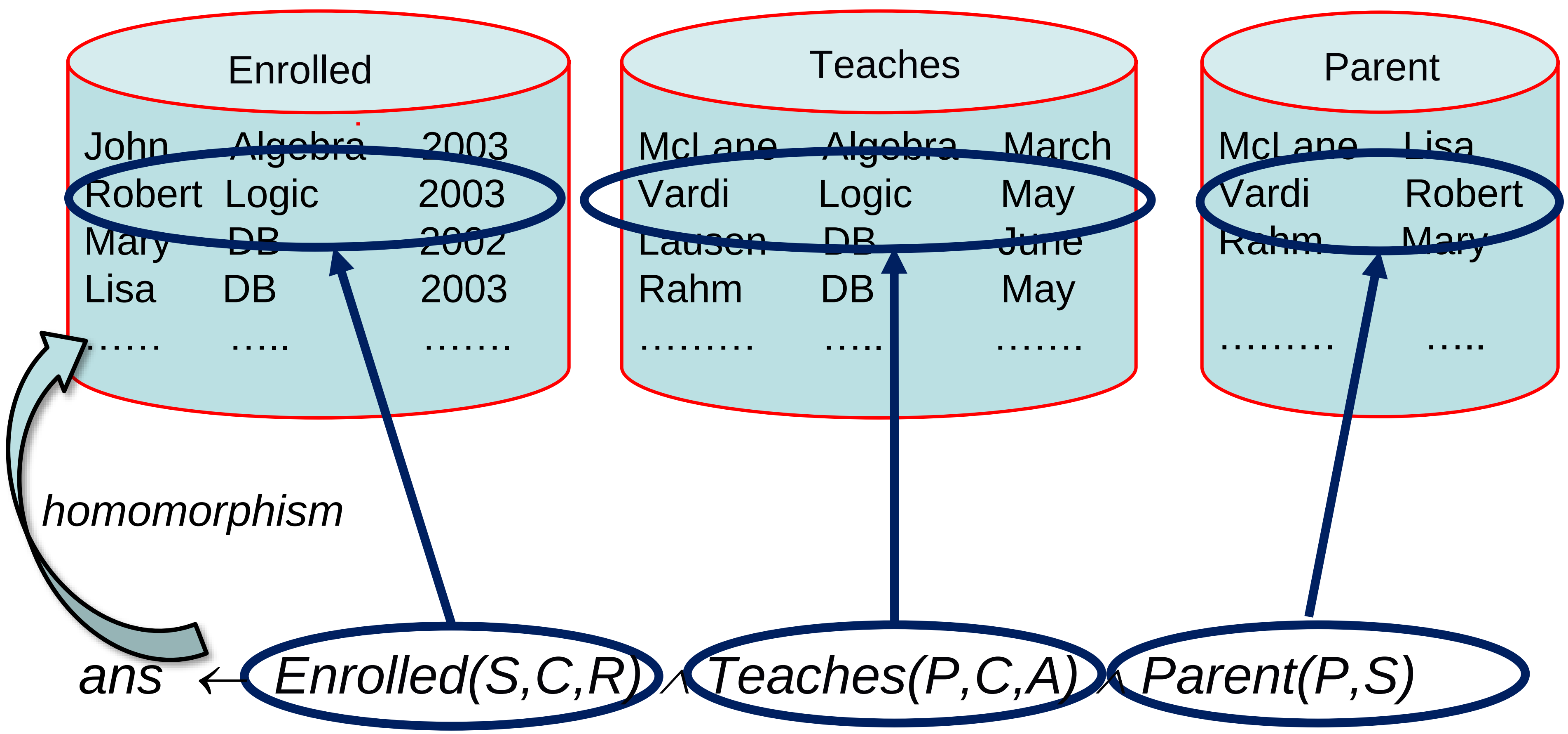
QUERY:

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

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

BCQ: Boolean Conjunctive Query Evaluation

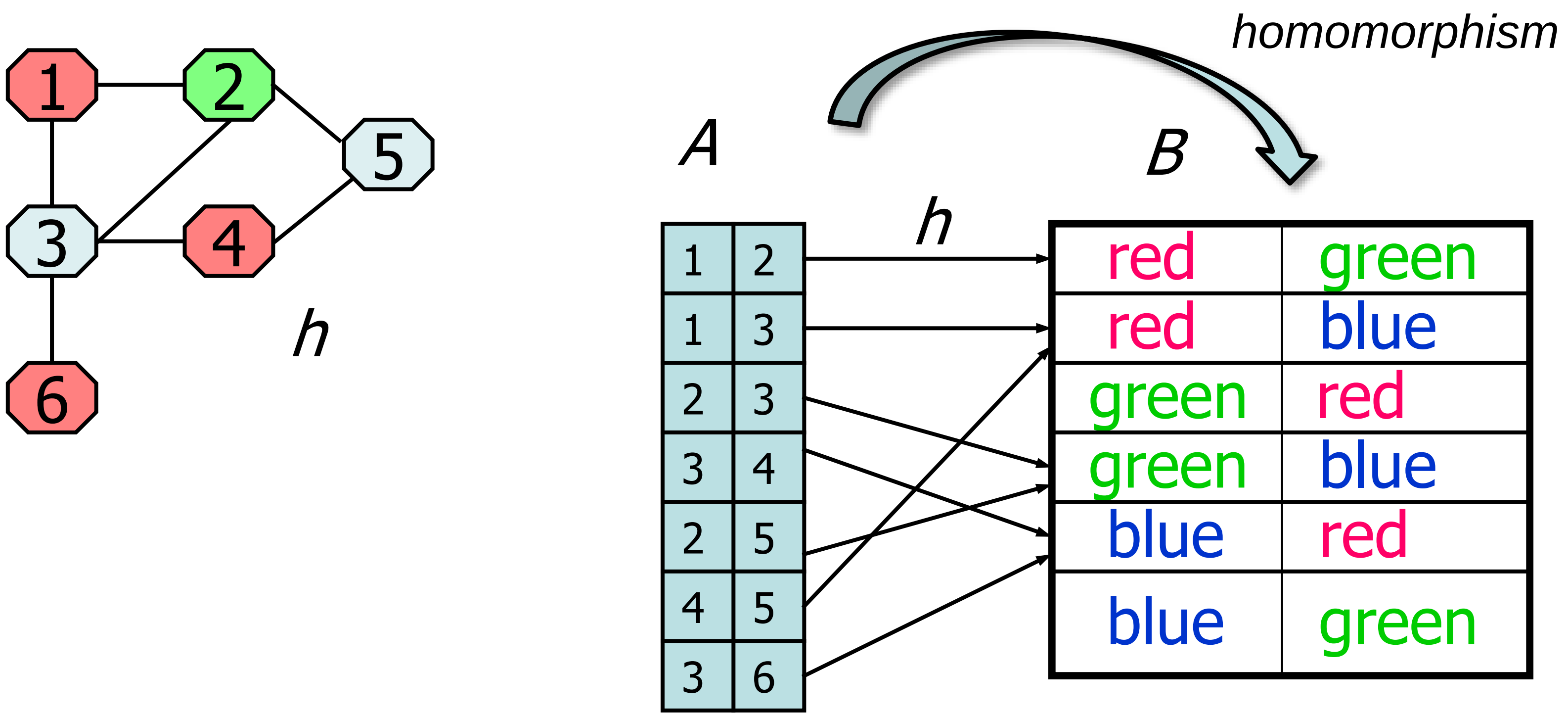
DATABASE:



NP-Completeness of HOM

Membership: Obvious, guess *h.*

Hardness: Reduction from 3COL.



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

Roadmap

- 3 Problems: HOM, CSP, BCQ
- Hypergraphs and acylicity
- Well-known width notions: tw, hw, ghw, fhw
- hw vs. ghw
- Tractable cases of ghw (and fhw) computation
- NP-hardness of the Check-problem for ghw and fhw
- A glimpse beyond fhw

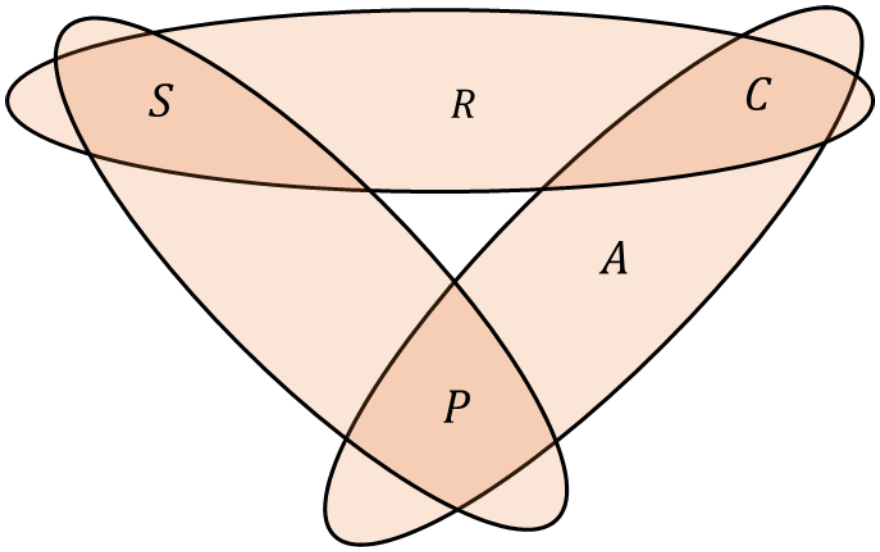
Hypergraph of a CQ

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

Hypergraph H = (V,E):

vertices V: variables of the CQ

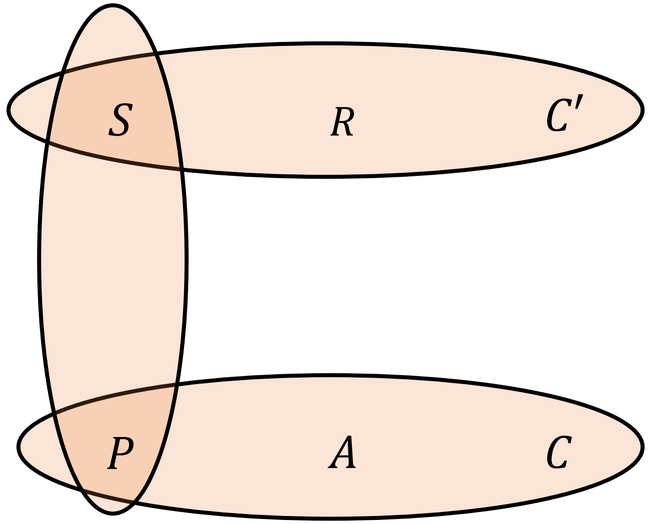
edges E: atoms of the CQ



Acyclic CQs (ACQs)

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

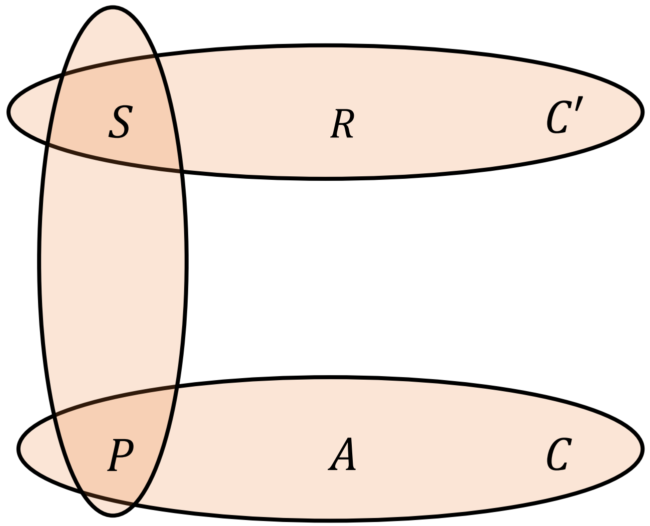
$$ans \leftarrow Enrolled(S, C', R) \wedge Teaches(P, C, A) \wedge Parent(P, S)$$

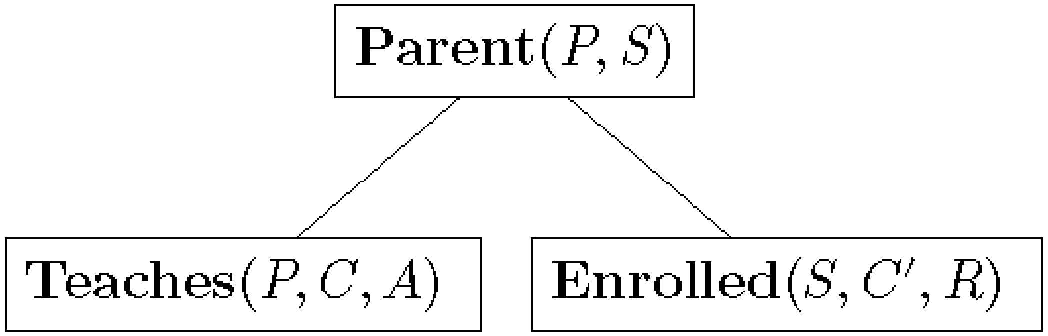


Acyclic CQs (ACQs)

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

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





Join Tree

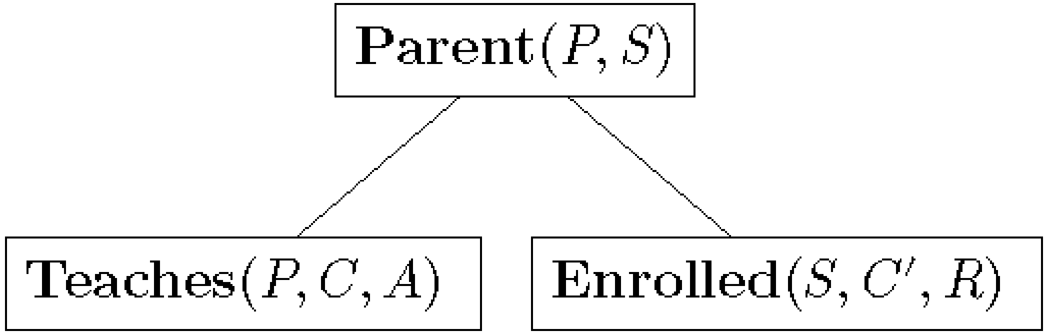
Acyclic CQs (ACQs)

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

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

Properties of a Join Tree:

- Nodes correspond to atoms
- For each query variable V, the tree-nodes containing V span a connected subtree (*connectednes condition*)



Join Tree

Complexity of CQ Answering

- NP-complete in the general case [Chandra and Merlin 1977]
- Tractable in case of acyclic CQs [Yannakakis 1981]

  even LOGCFL-complete, thus parallelizable [Gottlob,Leone,Scarcello 1998]

Complexity of CQ Answering

- NP-complete in the general case [Chandra and Merlin 1977]
- Tractable in case of acyclic CQs [Yannakakis 1981]

  even LOGCFL-complete, thus parallelizable [Gottlob,Leone,Scarcello 1998]

  Key Idea:

  - semi-joins along bottom-up and top-down traversals of the join tree
  - joins along another bottom-up traversal of the join tree

Complexity of CQ Answering

- NP-complete in the general case [Chandra and Merlin 1977]
- Tractable in case of acyclic CQs [Yannakakis 1981]

  even LOGCFL-complete, thus parallelizable [Gottlob,Leone,Scarcello 1998]

  Key Idea:
  - semi-joins along bottom-up and top-down traversals of the join tree
  - joins along another bottom-up traversal of the join tree

Time Complexity of CQ evaluation: $O(|Q| \cdot N + |Output|)$

with N = max size of the relations

Recognizing ACQs and Join Tree Construction

Theorem: ACQs can be recognized and, simultaneously,
if the CQ is acyclic, a join-tree can be built in linear time.

Algorithm: "GYO-reduction" (Graham resp. Yu and Ozsoyoglu 1979):

Recognizing ACQs and Join Tree Construction

Theorem: ACQs can be recognized and, simultaneously, if the CQ is acyclic, a join-tree can be built in linear time.

Algorithm: "GYO-reduction" (Graham resp. Yu and Ozsoyoglu 1979):

- Eliminate an atom if it shares no variables with other atoms;
  -> *This atom can be used as root node of another tree.*

Recognizing ACQs and Join Tree Construction

Theorem: ACQs can be recognized and, simultaneously, if the CQ is acyclic, a join-tree can be built in linear time.

Algorithm: "GYO-reduction" (Graham resp. Yu and Ozsoyoglu 1979):

- Eliminate an atom if it shares no variables with other atoms;
  -> *This atom can be used as root node of another tree.*
- Eliminate an atom R if there exists a *witness* R' s.t. each variable in R either appears in R only, or also appears in R';
  -> *R will be appended as child of R' in the join tree.*

Recognizing ACQs and Join Tree Construction

Theorem: ACQs can be recognized and, simultaneously,
if the CQ is acyclic, a join-tree can be built in linear time.

Algorithm: "GYO-reduction" (Graham resp. Yu and Ozsoyoglu 1979):

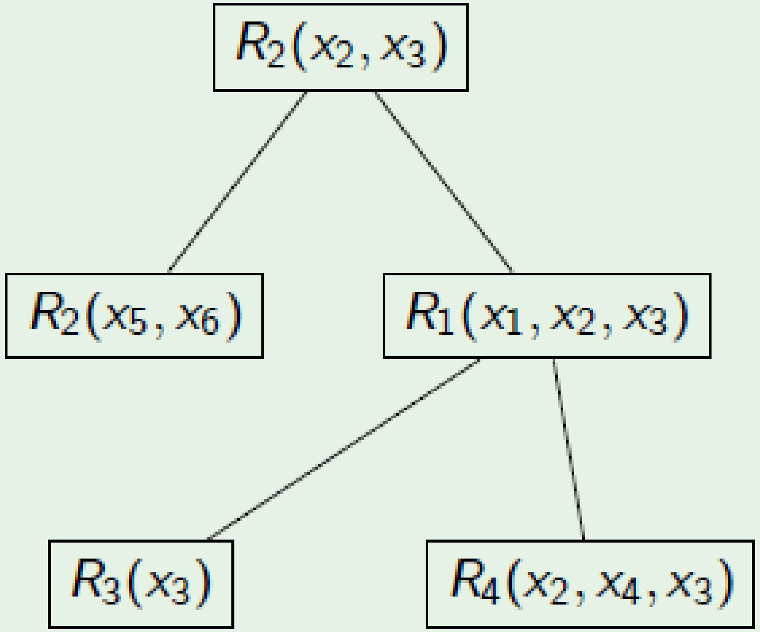
- Eliminate an atom if it shares no variables with other atoms;
  -> *This atom can be used as root node of another tree.*
- Eliminate an atom R if there exists a witness R' s.t. each variable in R either appears in R only, or also appears in R';
  -> *R will be appended as child of R' in the join tree.*

The query is acyclic iff the GYO-reduction yields the empty set of atoms.

Join Tree

Example

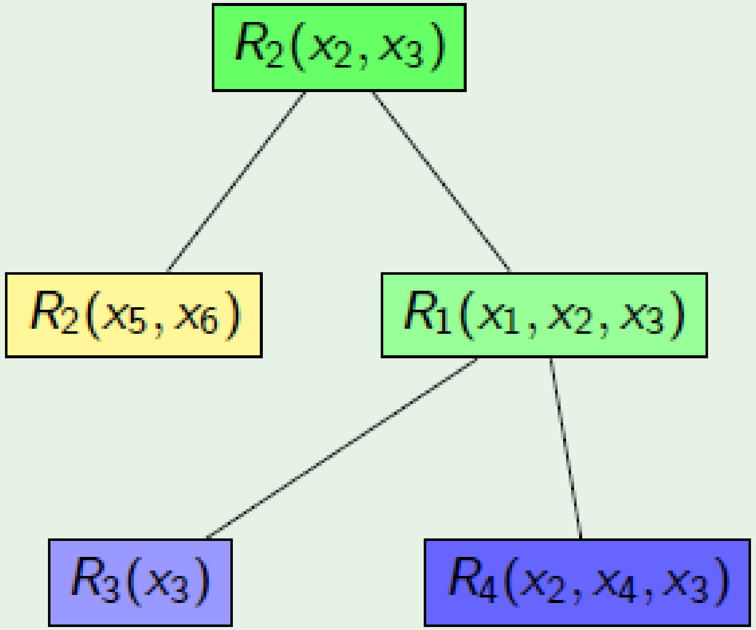
$$Q(x_1, x_2, x_3, x_4, x_5, x_6) :- R_3(x_3) \wedge R_4(x_2, x_4, x_3) \wedge R_1(x_1, x_2, x_3) \wedge R_2(x_2, x_3) \wedge R_2(x_5, x_6)$$



Join Tree

Example

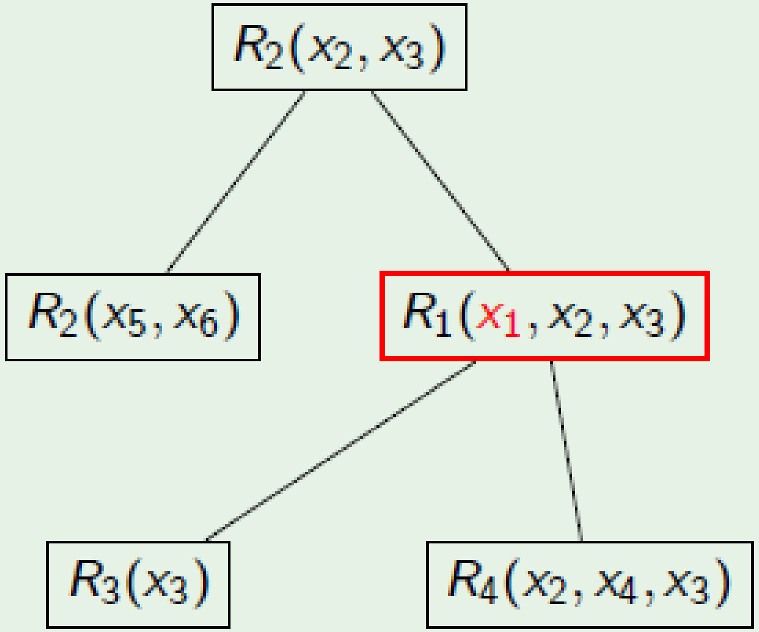
$$Q(x_1, x_2, x_3, x_4, x_5, x_6) :- R_3(x_3) \wedge R_4(x_2, x_4, x_3) \wedge R_1(x_1, x_2, x_3) \wedge R_2(x_2, x_3) \wedge R_2(x_5, x_6)$$



Join Tree

Example

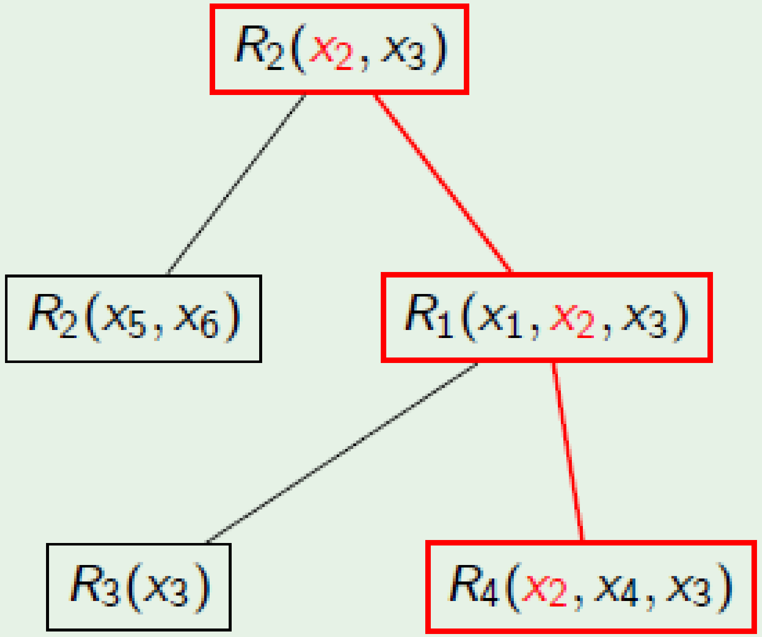
$$Q(x_1, x_2, x_3, x_4, x_5, x_6) :- R_3(x_3) \wedge R_4(x_2, x_4, x_3) \wedge R_1(x_1, x_2, x_3) \wedge R_2(x_2, x_3) \wedge R_2(x_5, x_6)$$



Join Tree

Example

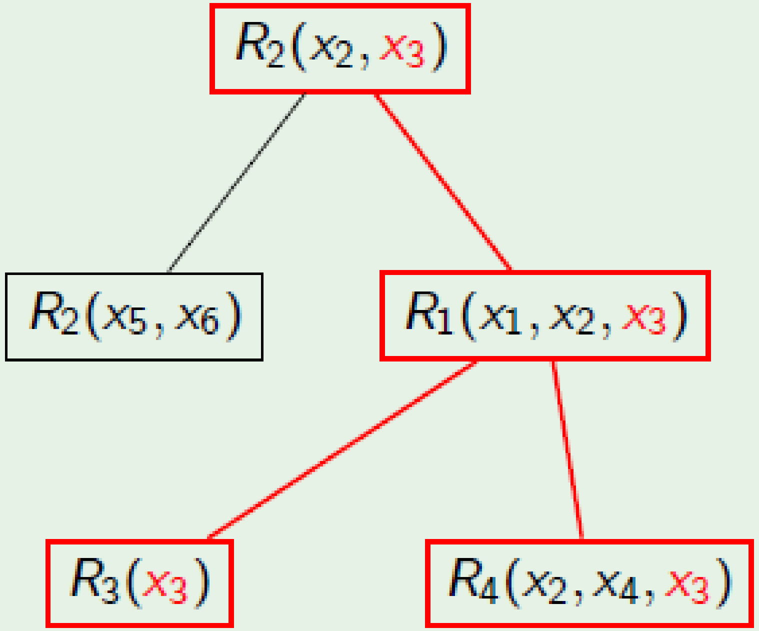
$$Q(x_1, x_2, x_3, x_4, x_5, x_6) :- R_3(x_3) \wedge R_4(x_2, x_4, x_3) \wedge R_1(x_1, x_2, x_3) \wedge R_2(x_2, x_3) \wedge R_2(x_5, x_6)$$



Join Tree

Example

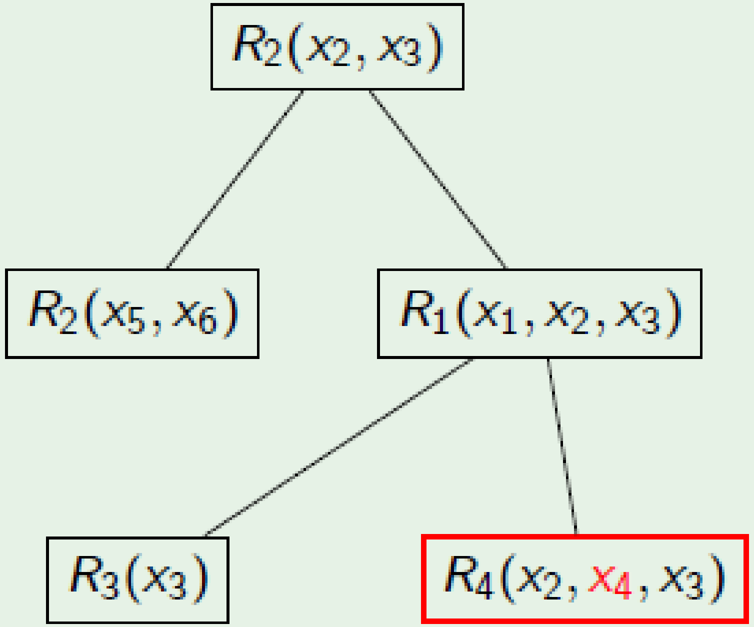
$$Q(x_1, x_2, x_3, x_4, x_5, x_6) :- R_3(x_3) \wedge R_4(x_2, x_4, x_3) \wedge R_1(x_1, x_2, x_3) \wedge R_2(x_2, x_3) \wedge R_2(x_5, x_6)$$



Join Tree

Example

$Q(x_1, x_2, x_3, x_4, x_5, x_6)$:−
$R_3(x_3) \wedge R_4(x_2, x_4, x_3) \wedge R_1(x_1, x_2, x_3) \wedge R_2(x_2, x_3) \wedge R_2(x_5, x_6)$

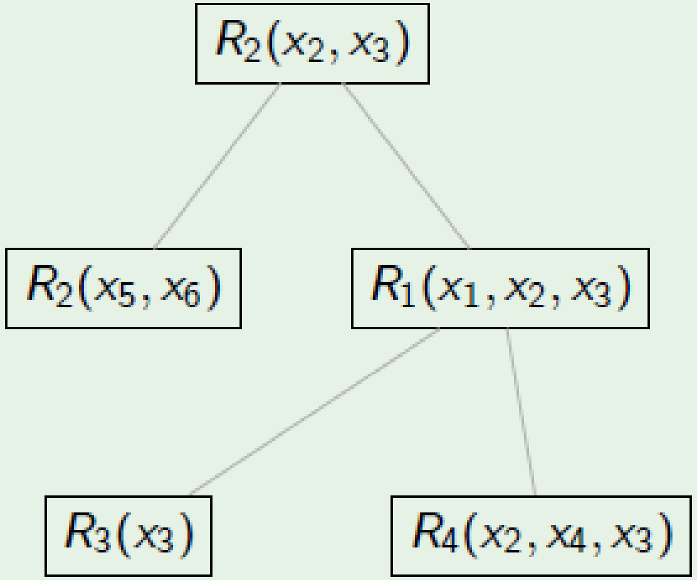


Join Tree Construction

Example

Consider again $Q(x_1, x_2, x_3, x_4, x_5, x_6)$:–

$$\underset{r_1}{R_3(x_3)} \wedge \underset{r_2}{R_4(x_2, x_4, x_3)} \wedge \underset{r_3}{R_1(x_1, x_2, x_3)} \wedge \underset{r_4}{R_2(x_2, x_3)} \wedge \underset{r_5}{R_2(x_5, x_6)}$$



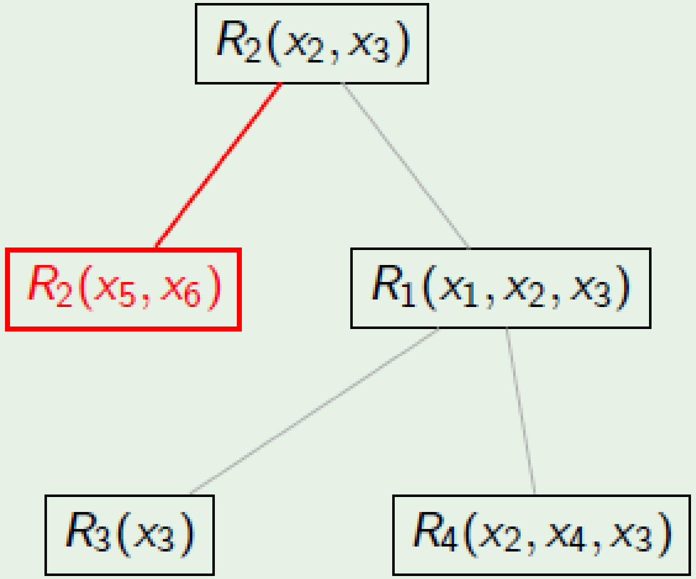
Join Tree Construction

Example

Consider again $Q(x_1, x_2, x_3, x_4, x_5, x_6)$:–

$$\underset{r_1}{R_3(x_3)} \wedge \underset{r_2}{R_4(x_2, x_4, x_3)} \wedge \underset{r_3}{R_1(x_1, x_2, x_3)} \wedge \underset{r_4}{R_2(x_2, x_3)} \wedge \underset{r_5}{R_2(x_5, x_6)}$$

$$\mathcal{A}_0 = \{r_1, r_2, r_3, r_4, r_5\}$$



Join Tree Construction

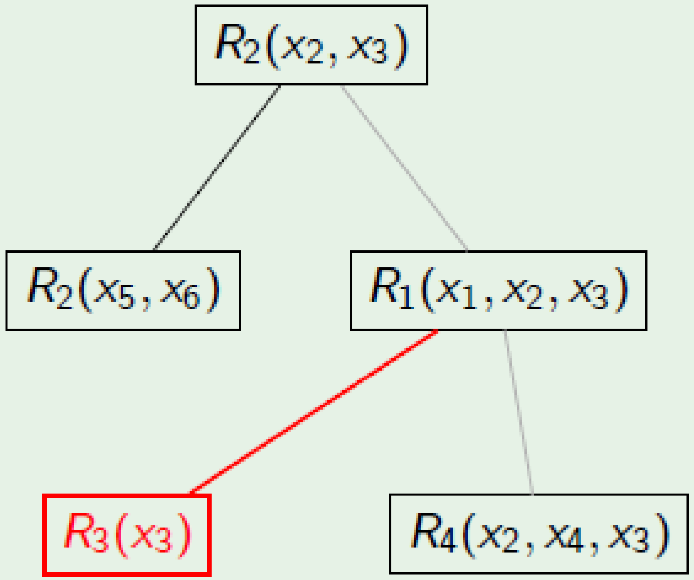
Example

Consider again $Q(x_1, x_2, x_3, x_4, x_5, x_6)$:–

$$\underbrace{R_3(x_3)}_{r_1} \wedge \underbrace{R_4(x_2, x_4, x_3)}_{r_2} \wedge \underbrace{R_1(x_1, x_2, x_3)}_{r_3} \wedge \underbrace{R_2(x_2, x_3)}_{r_4} \wedge \underbrace{R_2(x_5, x_6)}_{r_5}$$

$\mathcal{A}_0 = \{r_1, r_2, r_3, r_4, r_5\}$

$\mathcal{A}_1 = \{r_1, r_2, r_3, r_4\}$



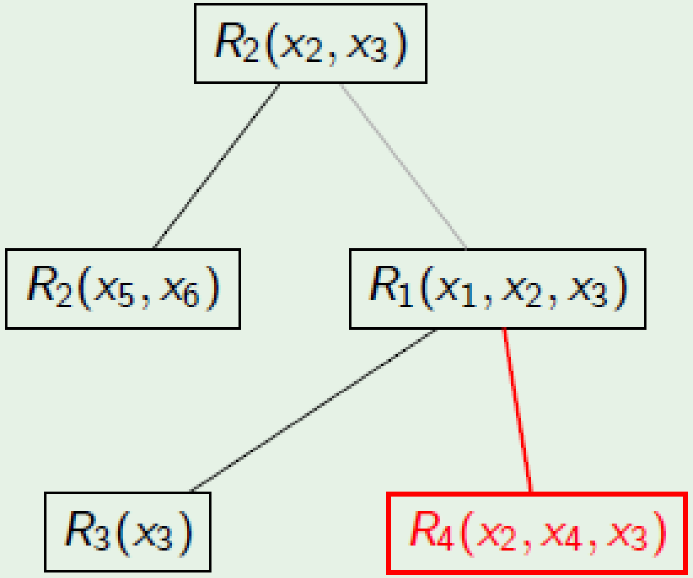
Join Tree Construction

Example

Consider again $Q(x_1, x_2, x_3, x_4, x_5, x_6)$:–

$$\underset{r_1}{R_3(x_3)} \wedge \underset{r_2}{R_4(x_2, x_4, x_3)} \wedge \underset{r_3}{R_1(x_1, x_2, x_3)} \wedge \underset{r_4}{R_2(x_2, x_3)} \wedge \underset{r_5}{R_2(x_5, x_6)}$$

$$\mathcal{A}_0 = \{r_1, r_2, r_3, r_4, r_5\}$$
$$\mathcal{A}_1 = \{r_1, r_2, r_3, r_4\}$$
$$\mathcal{A}_2 = \{r_2, r_3, r_4\}$$



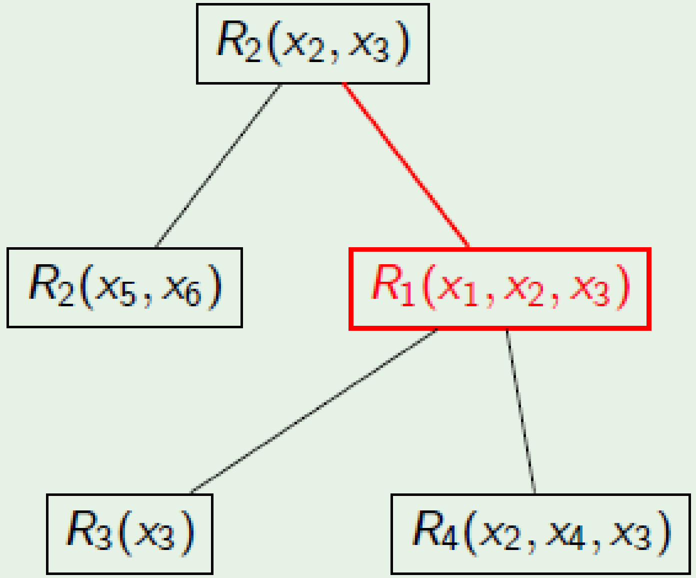
Join Tree Construction

Example

Consider again $Q(x_1, x_2, x_3, x_4, x_5, x_6)$:–

$$\underset{r_1}{R_3(x_3)} \wedge \underset{r_2}{R_4(x_2, x_4, x_3)} \wedge \underset{r_3}{R_1(x_1, x_2, x_3)} \wedge \underset{r_4}{R_2(x_2, x_3)} \wedge \underset{r_5}{R_2(x_5, x_6)}$$

$$\mathcal{A}_0 = \{r_1, r_2, r_3, r_4, r_5\}$$
$$\mathcal{A}_1 = \{r_1, r_2, r_3, r_4\}$$
$$\mathcal{A}_2 = \{r_2, r_3, r_4\}$$
$$\mathcal{A}_3 = \{r_3, r_4\}$$



Join Tree Construction

Example

Consider again $Q(x_1, x_2, x_3, x_4, x_5, x_6)$:–

$$R_3(x_3) \wedge R_4(x_2, x_4, x_3) \wedge R_1(x_1, x_2, x_3) \wedge R_2(x_2, x_3) \wedge R_2(x_5, x_6)$$

r_1 r_2 r_3 r_4 r_5

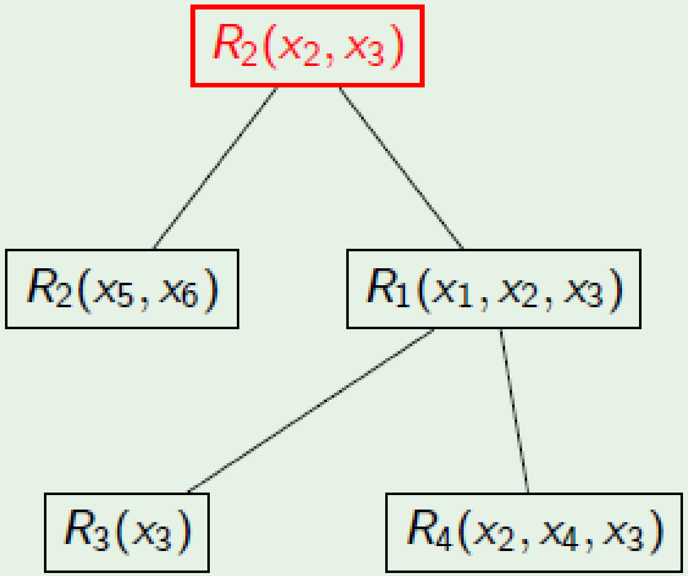
$\mathcal{A}_0 = \{r_1, r_2, r_3, r_4, r_5\}$

$\mathcal{A}_1 = \{r_1, r_2, r_3, r_4\}$

$\mathcal{A}_2 = \{r_2, r_3, r_4\}$

$\mathcal{A}_3 = \{r_3, r_4\}$

$\mathcal{A}_4 = \{r_4\}$



Yannakakis' Algorithm

Label each node t in the join tree with the actual relation R_t

Boolean ACQ:

- Semi-joins in a bottom-up traversal of the join tree

Non-Boolean ACQ:

- Semi-joins in a top-down traversal of the join tree
- Joins in another bottom-up traversal of the join tree

Correctness of Yannakakis' Algorithm

Correctness of the algorithm follows from the following propositions: Given join tree T, for $t \in V(T)$ let T_t be the subtree of T rooted at t, and let R_t be the relation at node t; moreover, let R'_t / R''_t / R'''_t denote the result of the first / second / third traversal of the join tree:

1. After the 1^{st} bottom-up traversal:
$$R'_t = \pi_{vars(t)}(\bowtie_{v \in V(T_t)} R_v) \text{ for each } t \in T$$
2. After the top-down traversal:
$$R''_t = \pi_{vars(t)}(\bowtie_{v \in V(T)} R_v) \text{ for each } t \in T$$
3. After the 2^{nd} bottom-up traversal:
$$R'''_t = \pi_{vars(T_t)}(\bowtie_{v \in V(T)} R_v) \text{ for each } t \in T$$
$\Rightarrow R'''_r$ at root r contains all results

How to generalize query acyclicity?

Generalizations of acyclicity come with some notion of width expressing the degree of cyclicity.

Desiderata for a "good" generalization:

- Generalization of Acyclicity:
  Queries of width k≥1 include all acyclic CQs
- Tractable Recognizability:
  Width k queries can be recognized efficiently
- Tractable Query Answering:
  Width k queries can be answered efficiently

Roadmap

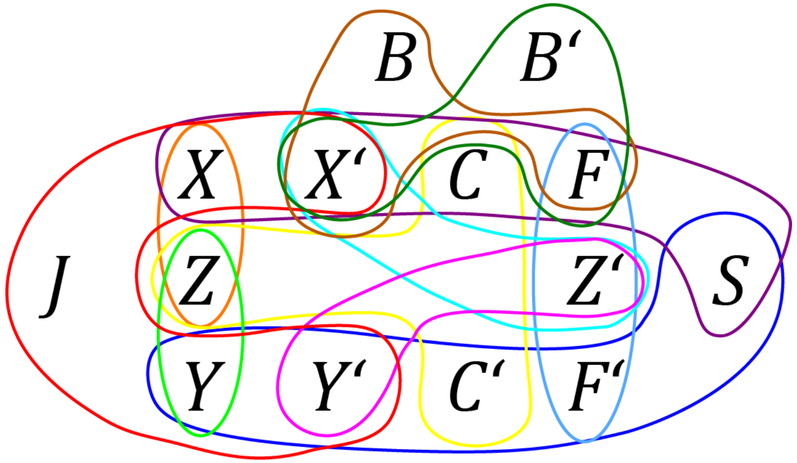
- 3 Problems: HOM, CSP, BCQ
- Hypergraphs and acyclicity
- Well-known width notions: tw, hw, ghw, fhw
- hw vs. ghw
- Tractable cases of ghw (and fhw) computation
- NP-hardness of the Check-problem for ghw and fhw
- A glimpse beyond fhw

Tree Decomposition

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

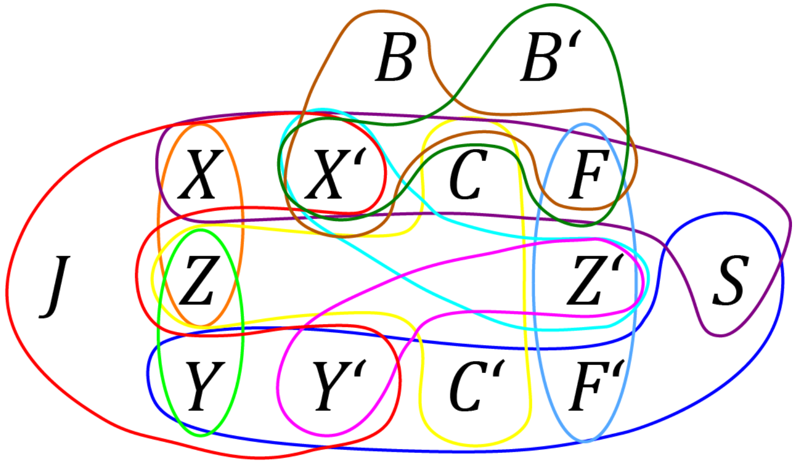
Tree Decomposition

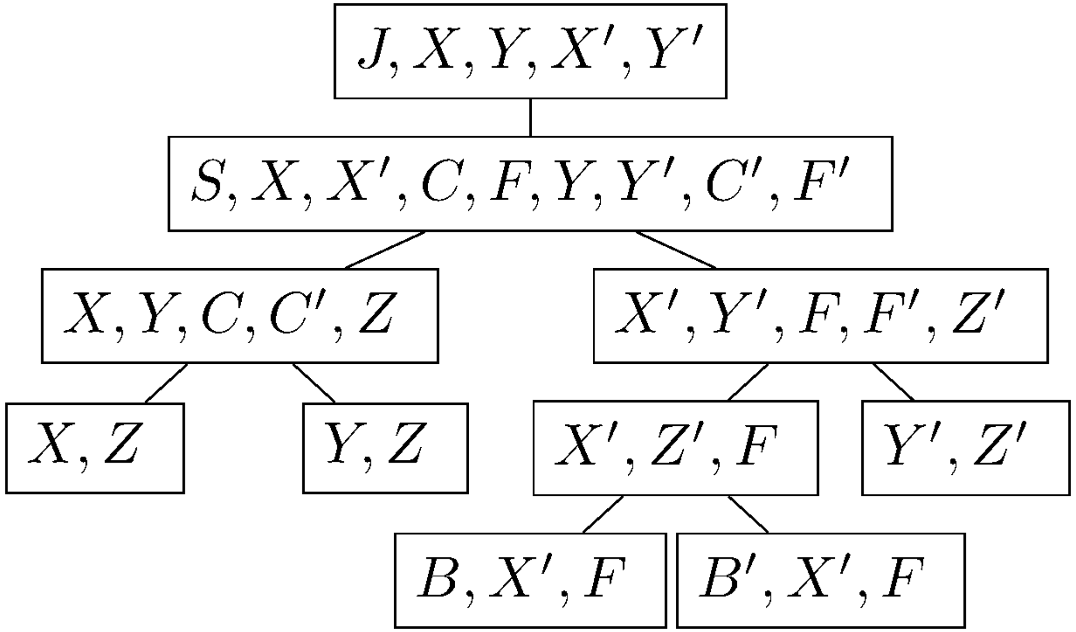
$$ans \leftarrow a(S,X,X',C,F) \wedge b(S,Y,Y',C',F') \wedge c(C,C',Z) \wedge d(X,Z) \wedge e(Y,Z) \wedge$$
$$f(F,F',Z') \wedge g(X',Z') \wedge h(Y',Z') \wedge j(J,X,Y,X',Y') \wedge p(B,X',F) \wedge q(B',X',F)$$



Tree Decomposition

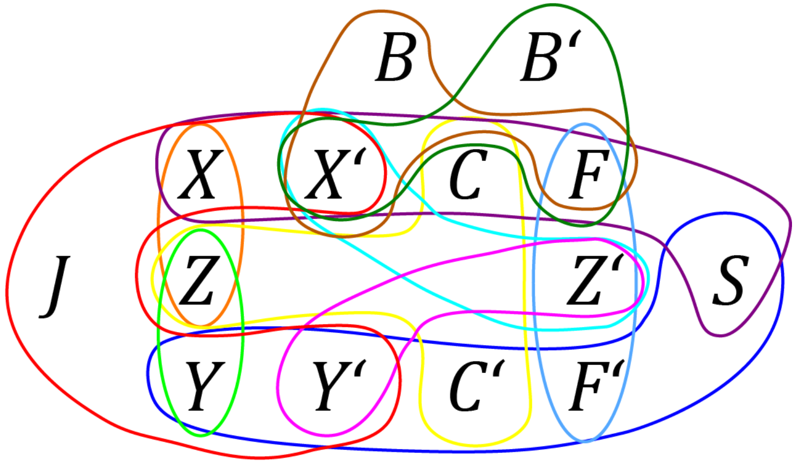
$$ans \leftarrow a(S,X,X',C,F) \wedge b(S,Y,Y',C',F') \wedge c(C,C',Z) \wedge d(X,Z) \wedge e(Y,Z) \wedge f(F,F',Z') \wedge g(X',Z') \wedge h(Y',Z') \wedge j(J,X,Y,X',Y') \wedge p(B,X',F) \wedge q(B',X',F)$$

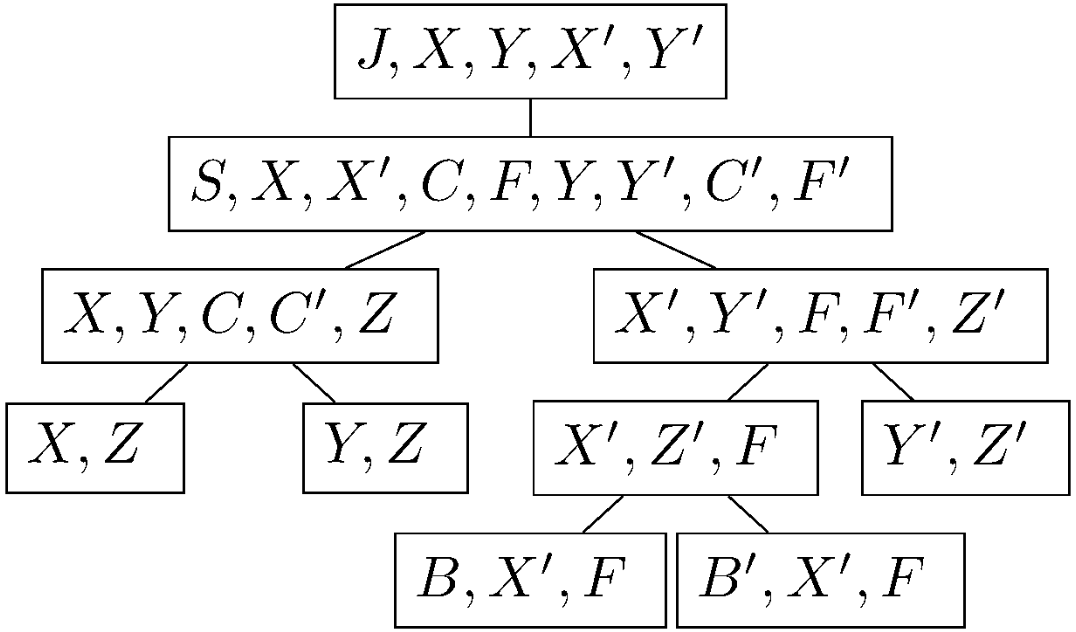




Tree Decomposition

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

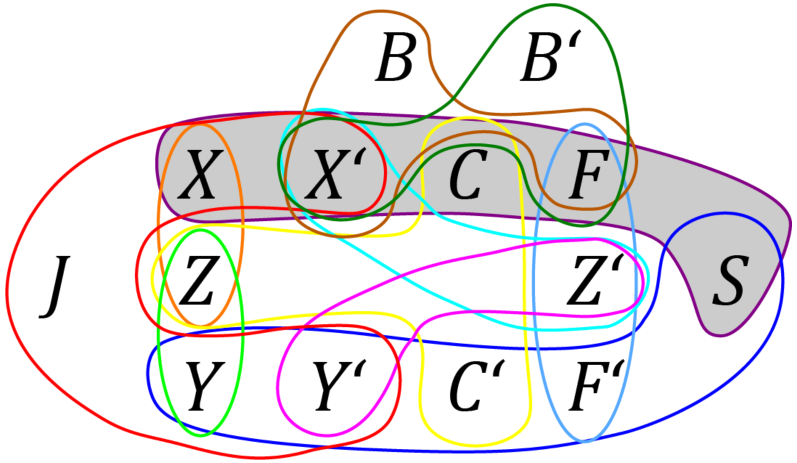


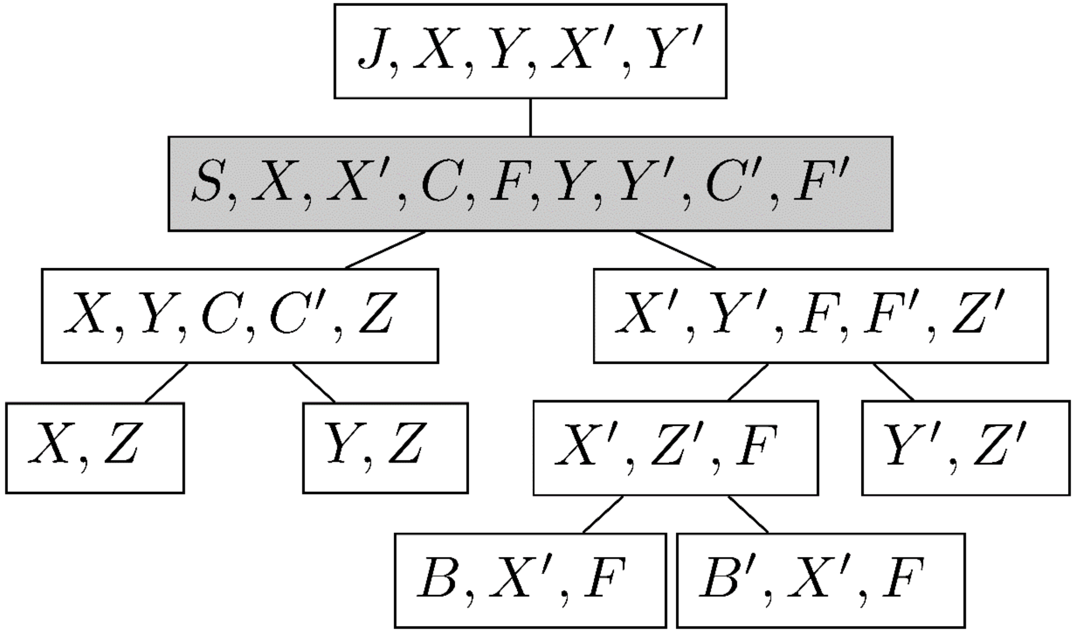


- Variables of each atom covered by some node

Tree Decomposition

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

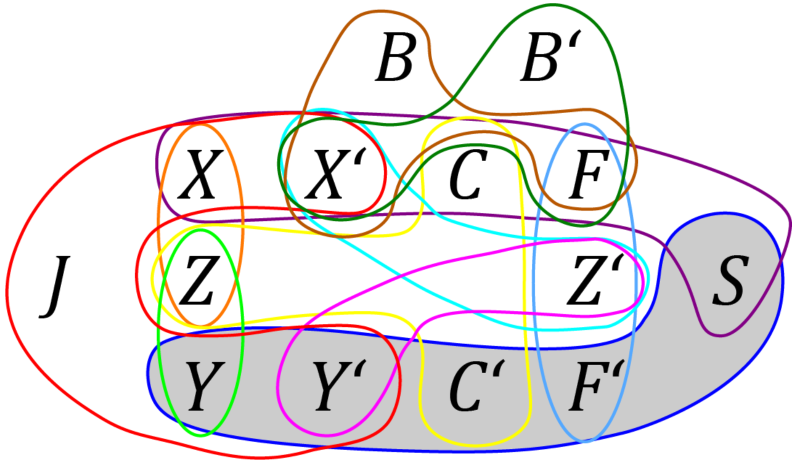


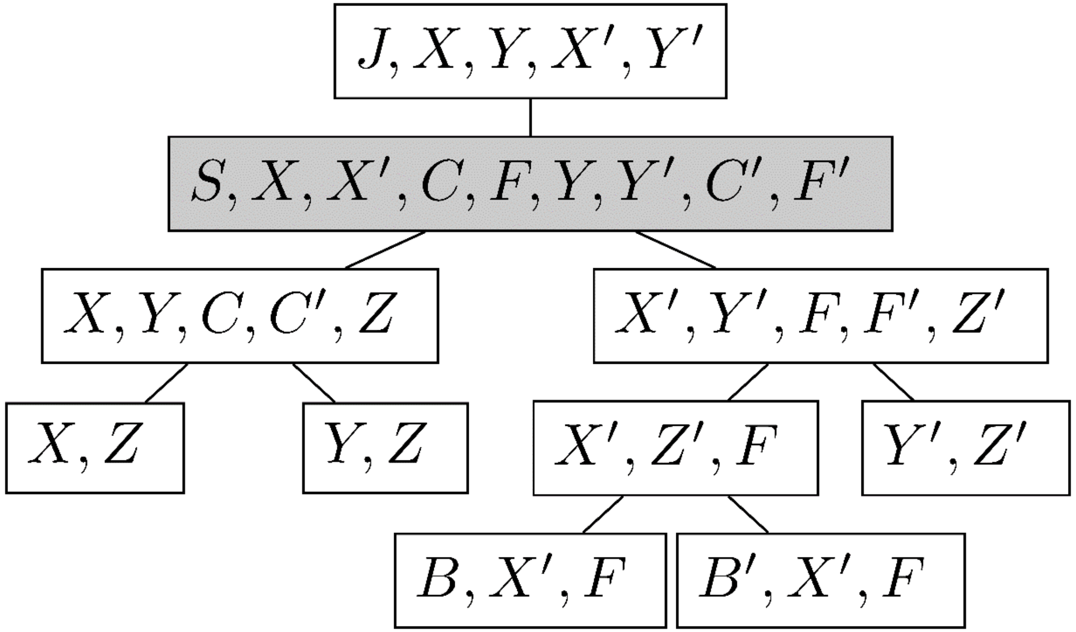


- Variables of each atom covered by some node

Tree Decomposition

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

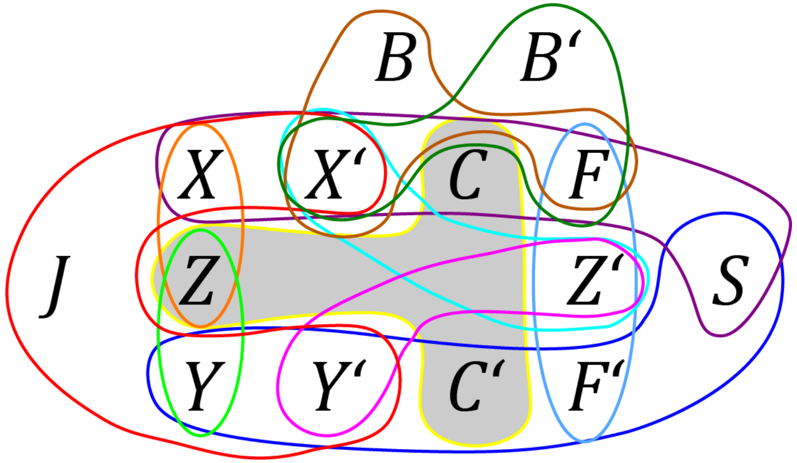


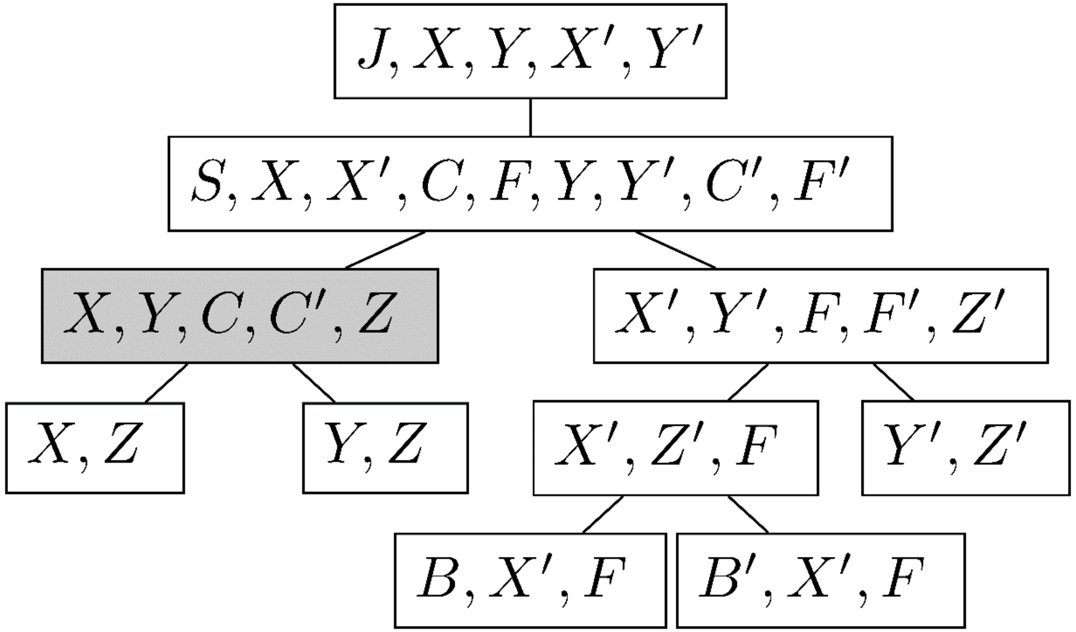


- Variables of each atom covered by some node

Tree Decomposition

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

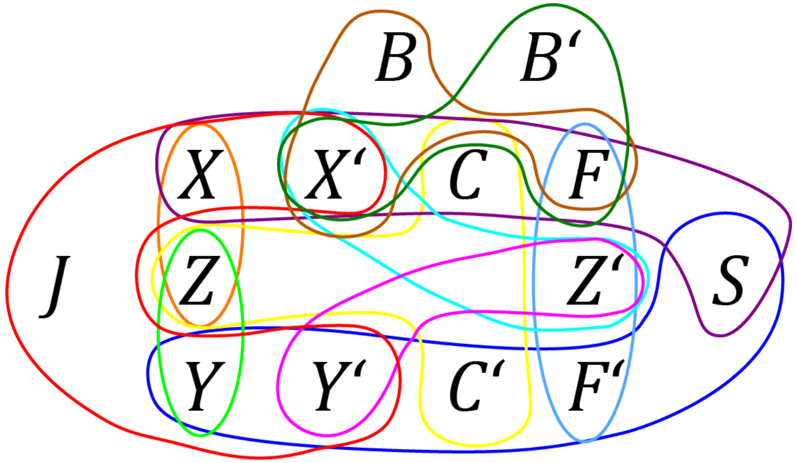


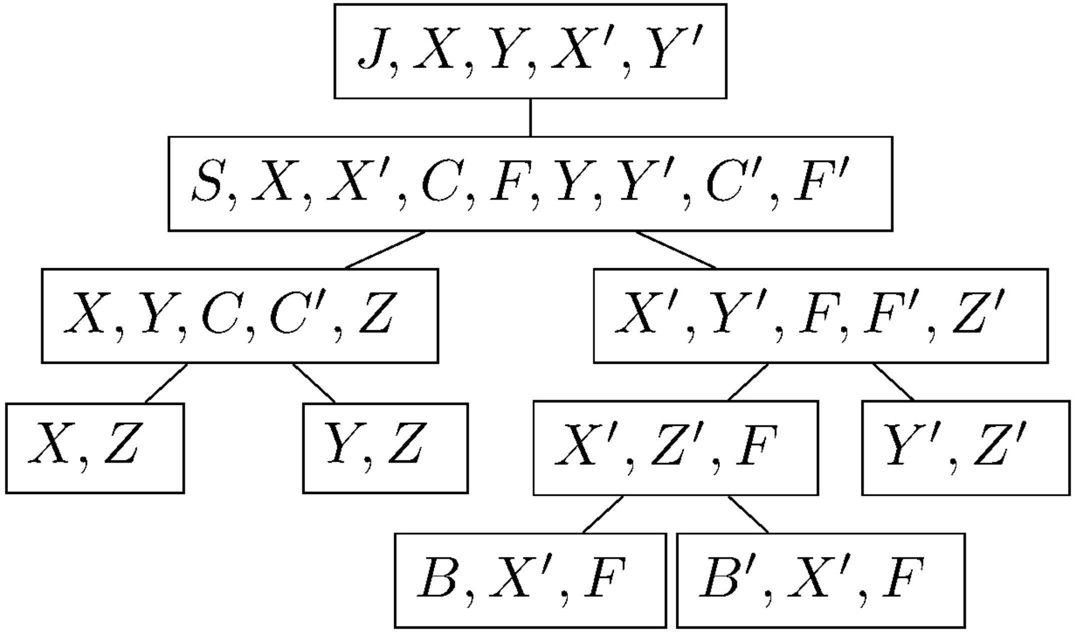


- Variables of each atom covered by some node

Tree Decomposition

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

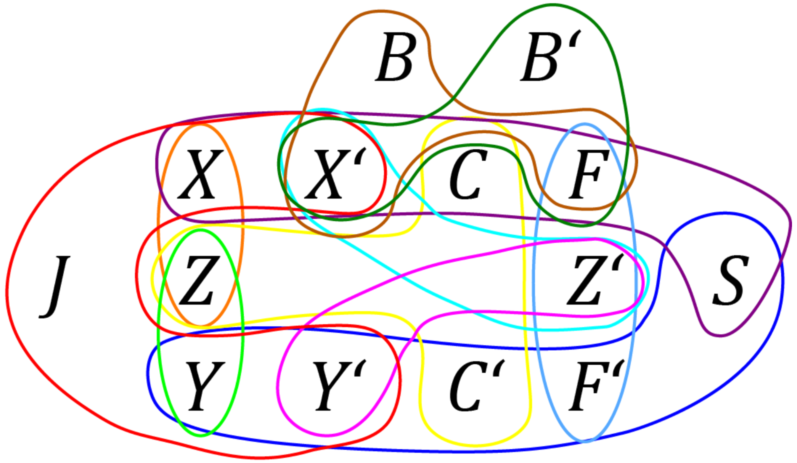


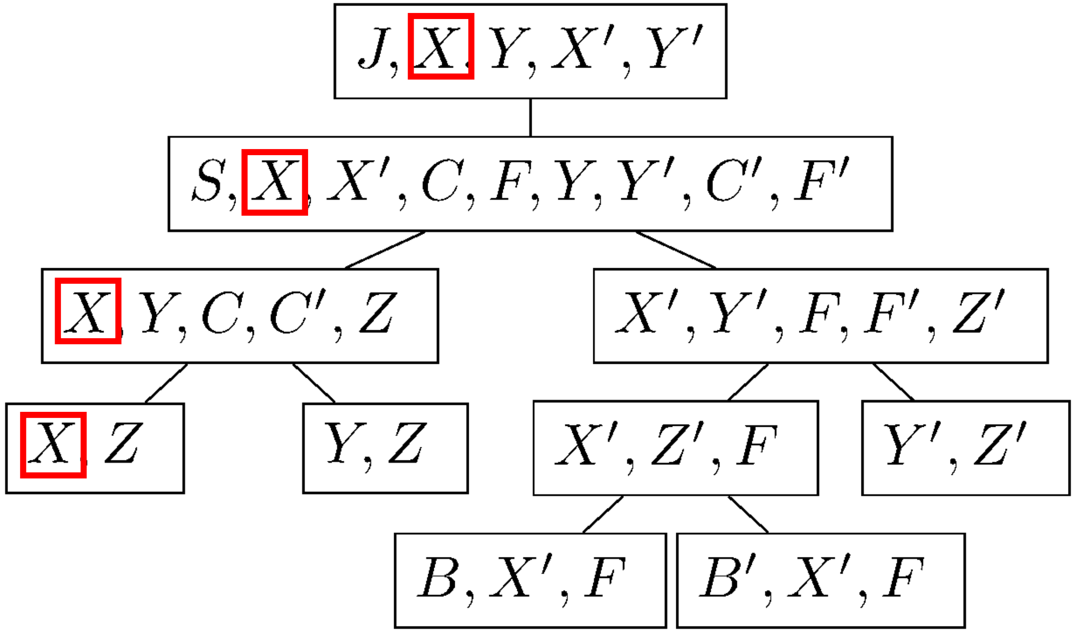


- Variables of each atom covered by some node
- Connectedness Condition

Tree Decomposition

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

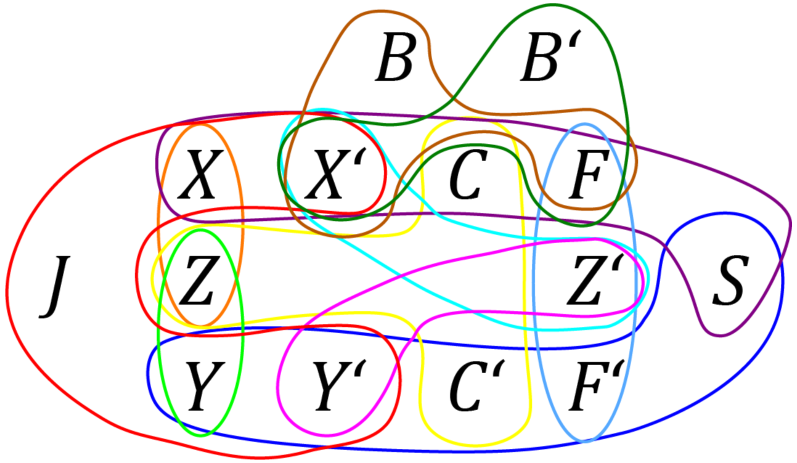


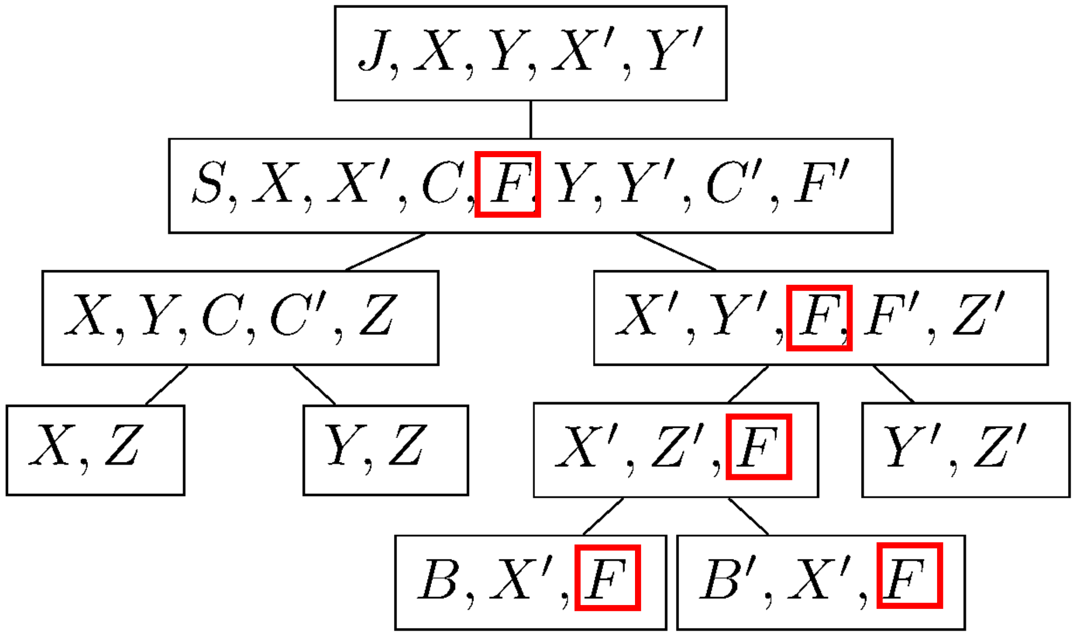


- Variables of each atom covered by some node
- Connectedness Condition

Tree Decomposition

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

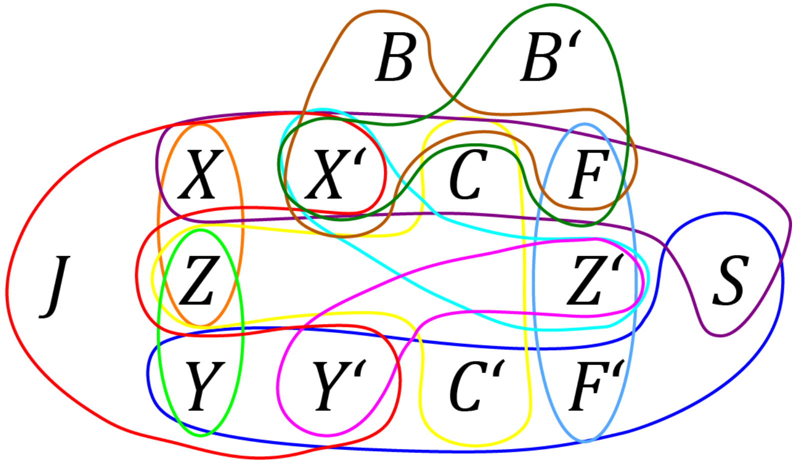


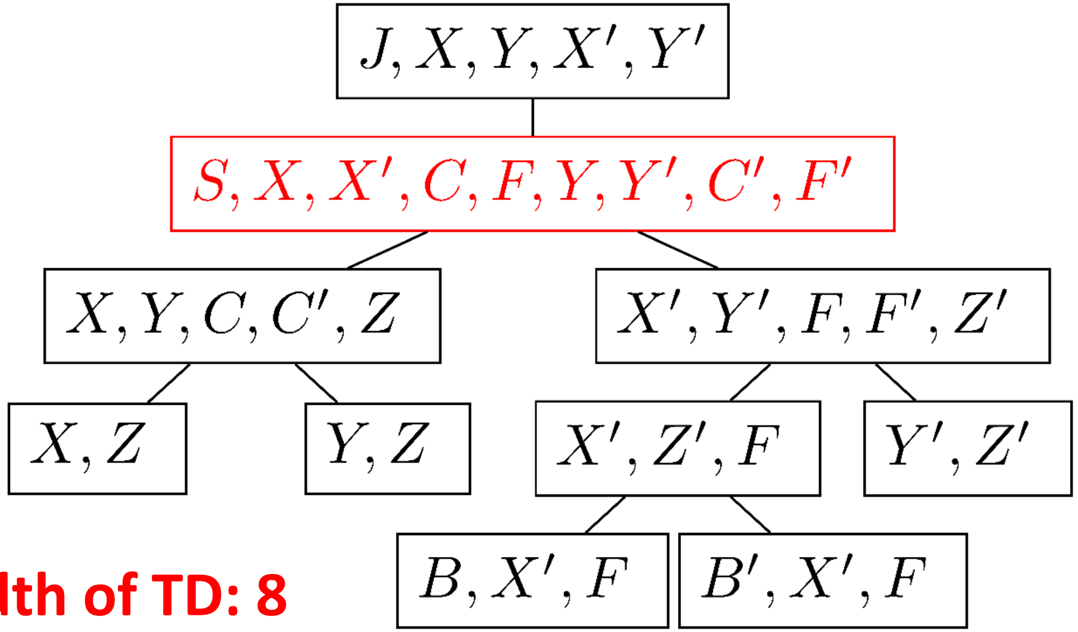


- Variables of each atom covered by some node
- Connectedness Condition

Tree Decomposition

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



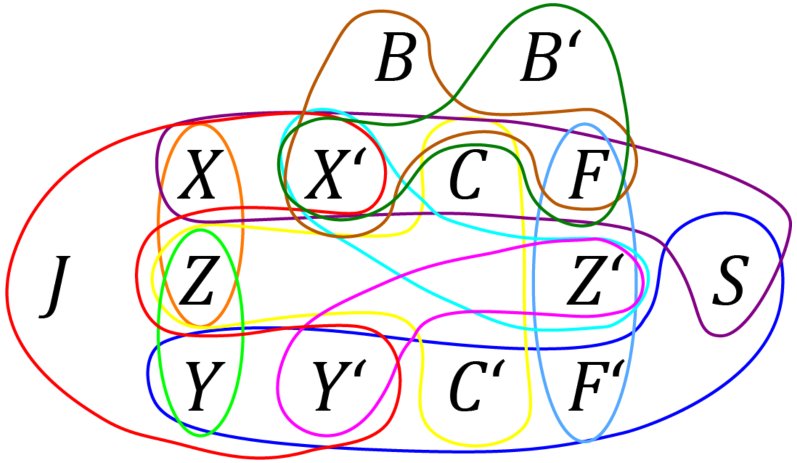


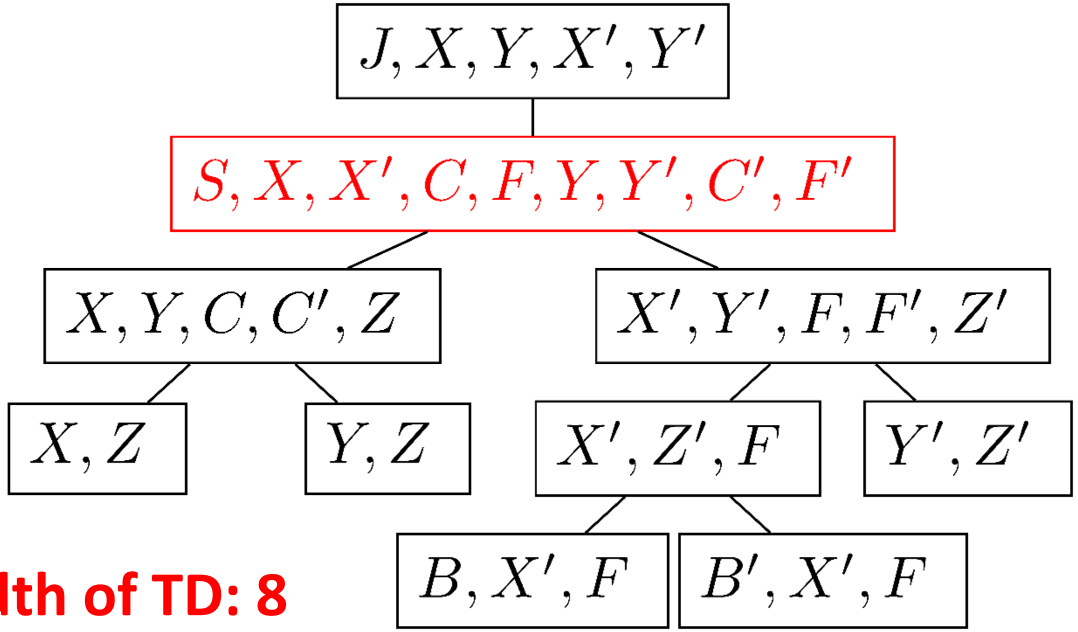
Width of TD: 8

- Variables of each atom covered by some node
- Connectedness Condition

Tree Decomposition

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





- Variables of each atom covered by some node

Width of TD: 8

- Connectedness Condition

tw(Q) = minimum width over all TDs of Q

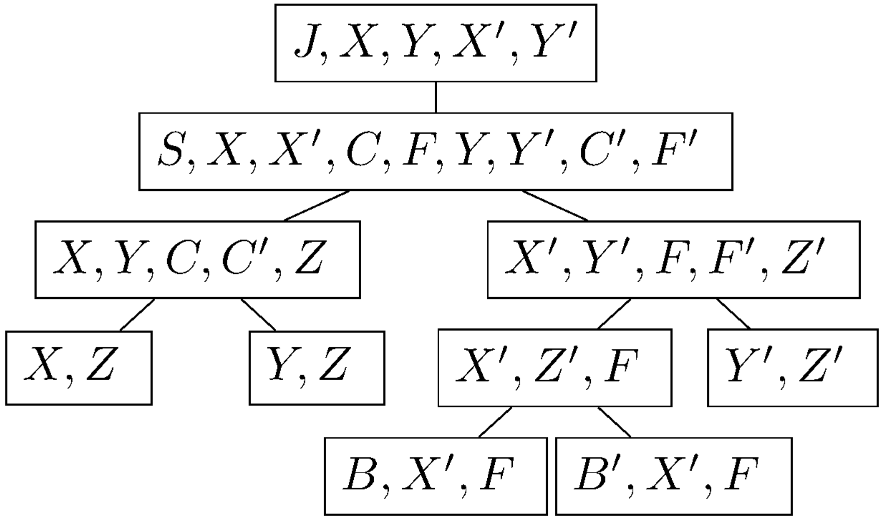
Edge Covers

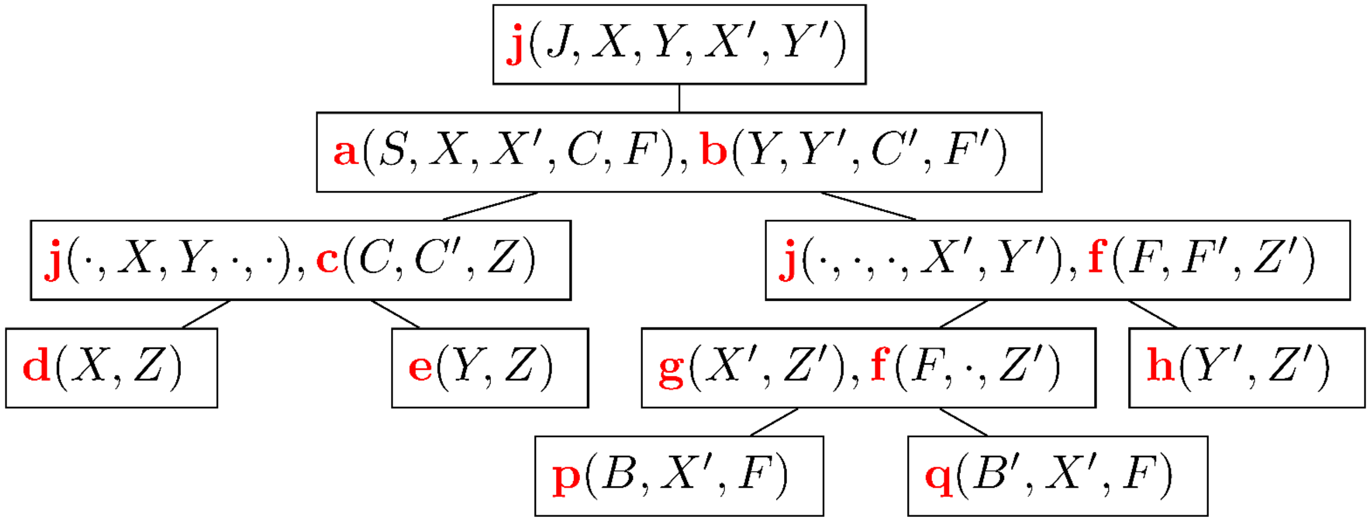
- Consider a set of vertices $V' \subseteq V(H)$
- An edge cover is a set of edges $E' \subseteq E(H)$, s.t. all vertices in V' are "covered" by E', i.e. $V' \subseteq \bigcup_{e \in E'} e$
- Add edge covers to the tree decomposition
- Each node p in the decomposition has two "labels":
  - λ(p): set of edges
  - χ(p): set of vertices

Generalized Hypertree Decompositions

Tree Decompositions + Edge Covers

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





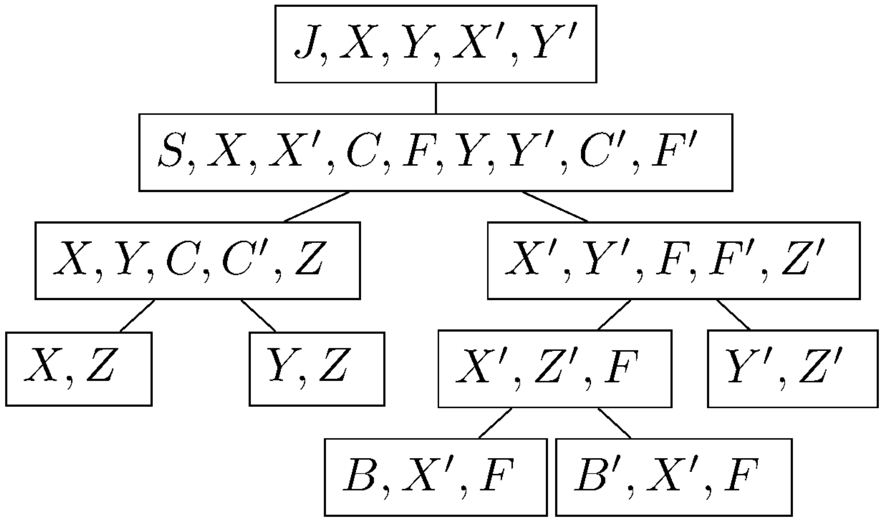
Width of TD: 8

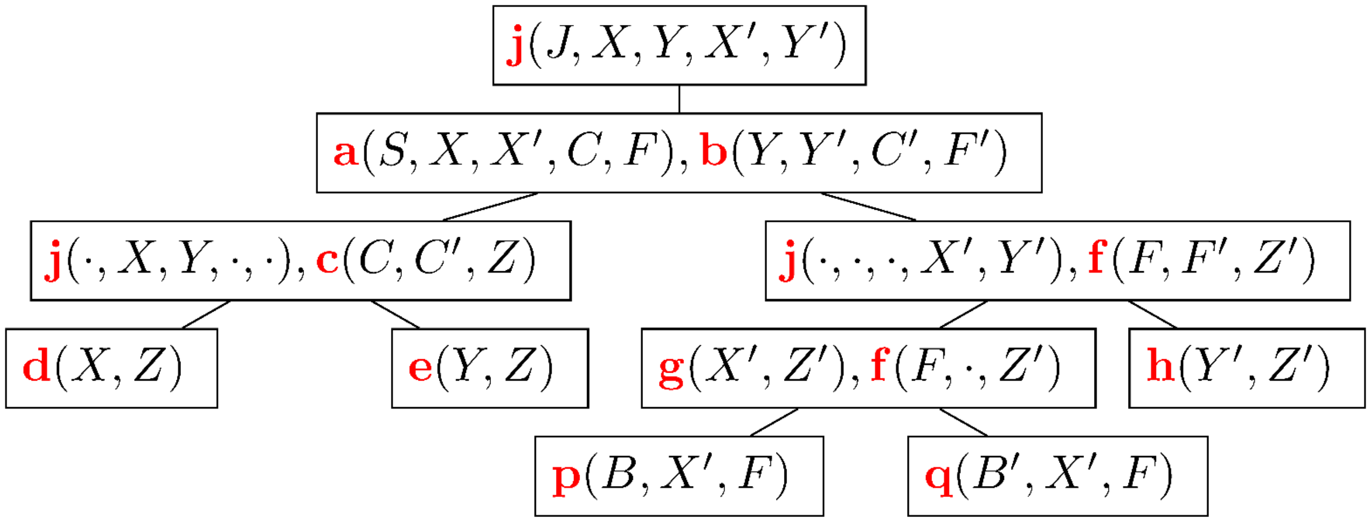
Width of GHD: 2

Generalized Hypertree Decompositions

Tree Decompositions + Edge Covers

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





Width of TD: 8

Width of GHD: 2

ghw(Q) = minimum width over all GHDs of Q

Tractable CQ answering for bounded ghw

- Key Idea: local joins of "edge labels" in the GHD to obtain ACQ
- for ghw = k: joins of up to k relations
- Time complexity of CQ answering if ghw(Q) ≤ k:
  $O(|Q| \cdot |N|^k + |Output|)$

Tractable CQ answering for bounded ghw

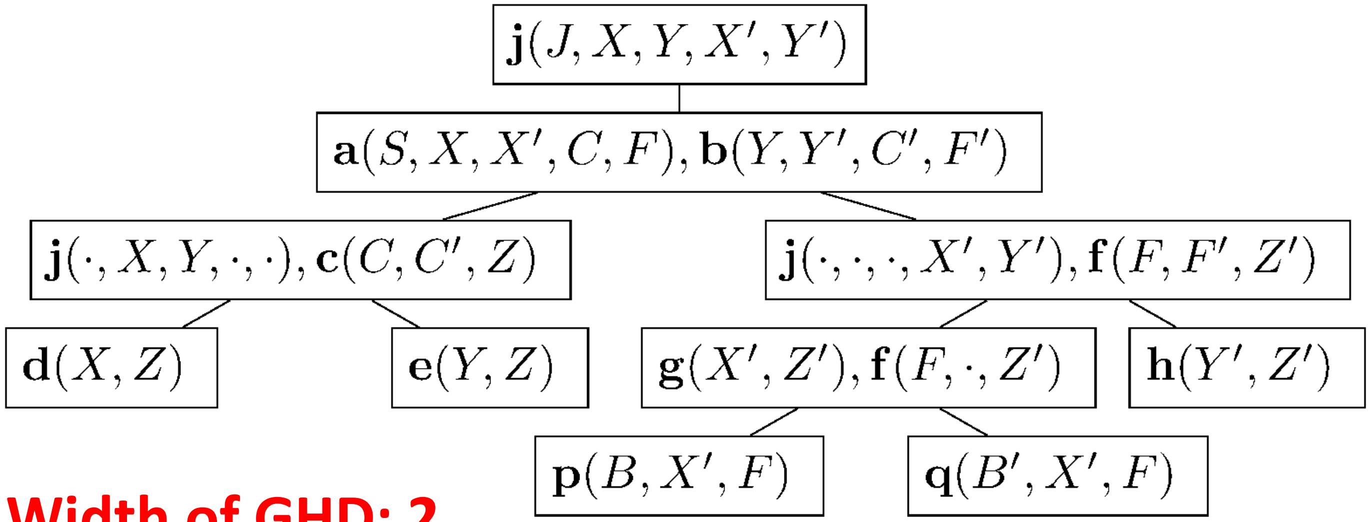
- Key Idea: local joins of "edge labels" in the GHD to obtain ACQ
- for ghw = k: joins of up to k relations
- Time complexity of CQ answering if ghw(Q) ≤ k:
  $O(|Q| \cdot |N|^k + |Output|)$

Compare this with bounded tree width:

- for tw = k: views of up to k+1 variables
- Time complexity of CQ answering if tw(Q) ≤ k:
  $O(|Q| \cdot |adom|^{k+1} + |Output|)$

Hypertree Decompositions

GHD + Special Condition

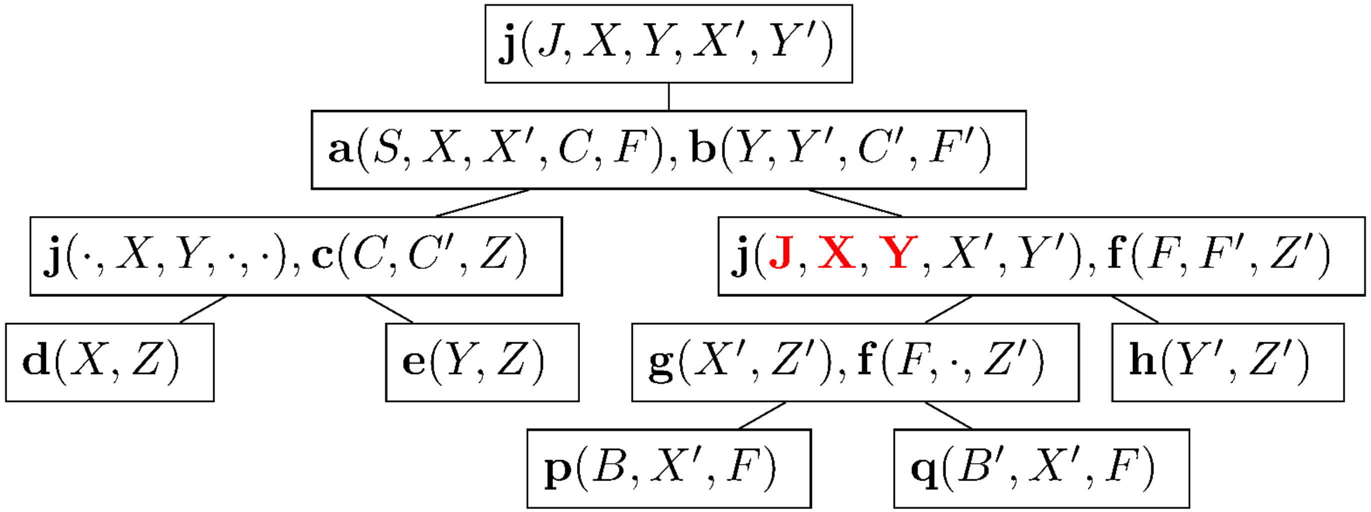


Width of GHD: 2

Hypertree Decompositions

GHD + Special Condition

Each variable that **disappeared** at some node **n**

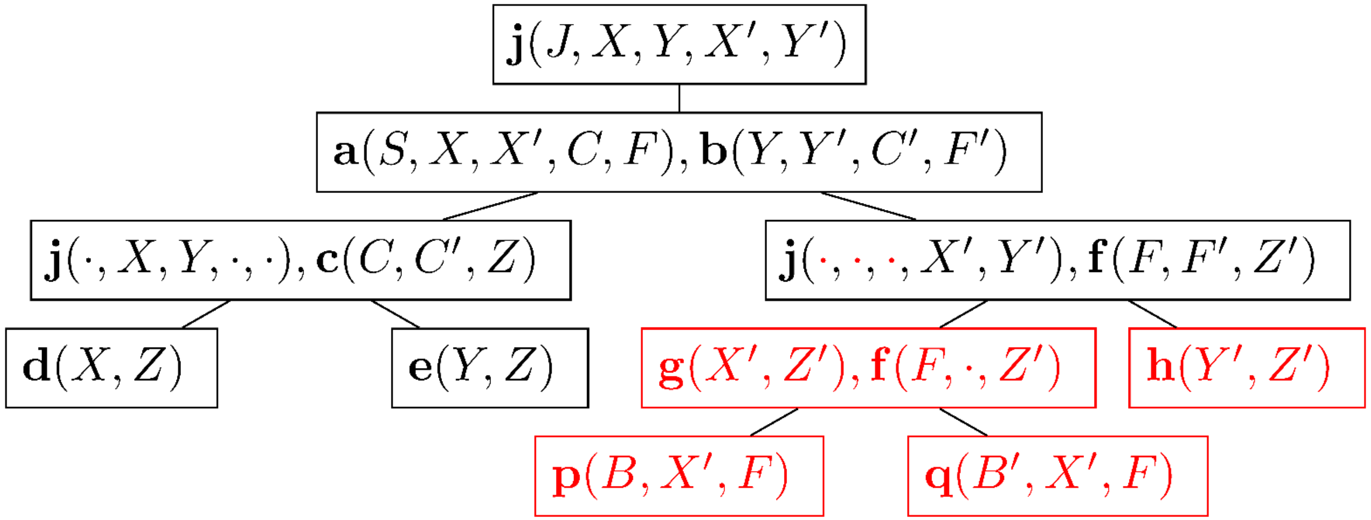


Hypertree Decompositions

GHD + Special Condition

Each variable that **disappeared** at some node **n**

does **not reappear** in the subtree rooted at **n**

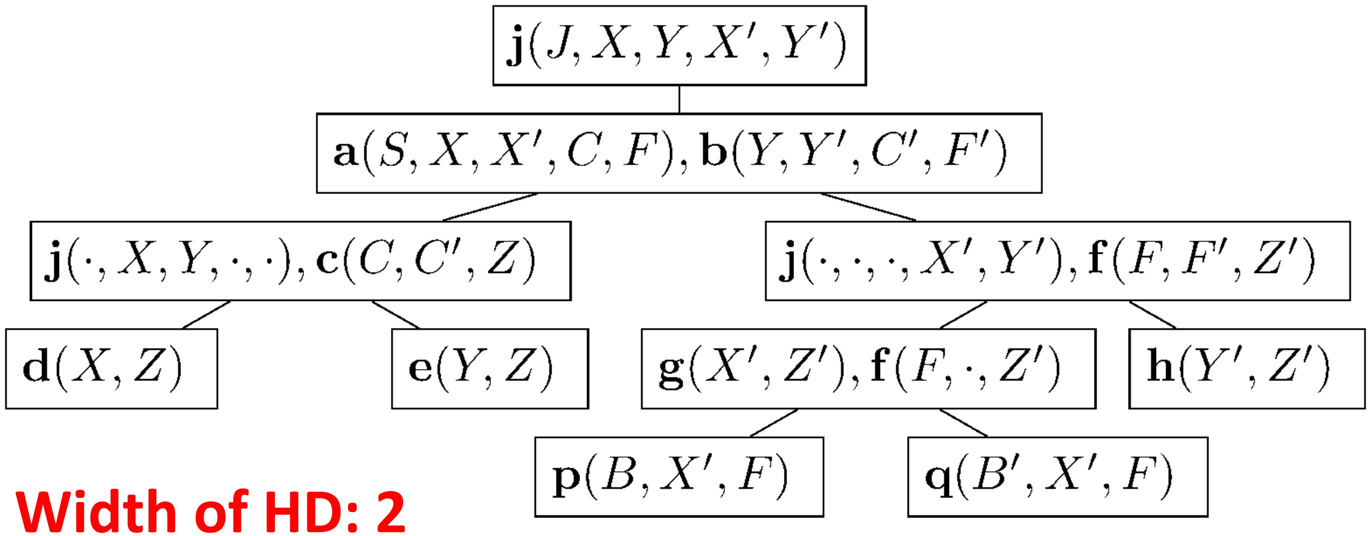


Hypertree Decompositions

GHD + Special Condition

Each variable that **disappeared** at some node **n**

does **not reappear** in the subtree rooted at **n**



Width of HD: 2

hw(Q) = minimum width over all HDs of Q

Integral vs. Fractional Edge Covers

Integral Edge Covers

Let λ be a function: $E(H) \rightarrow \{\mathbf{0}, \mathbf{1}\}$ then

$$B(\lambda) = \{v \in V(H) | \sum_{e \in E(H), v \in e} \lambda(e) \geq 1\}.$$

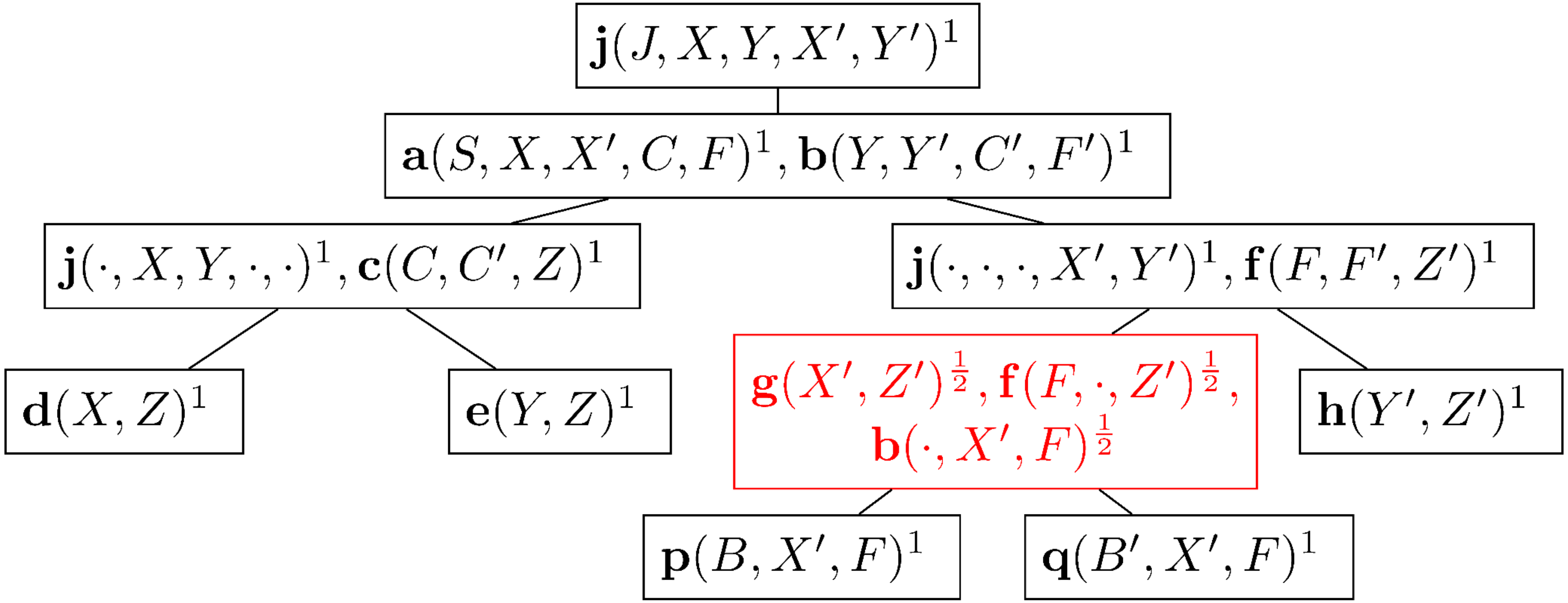
Fractional Edge Covers

Let γ be a function: $E(H) \rightarrow [\mathbf{0}, \mathbf{1}]$ then

$$B(\gamma) = \{v \in V(H) | \sum_{e \in E(H), v \in e} \gamma(e) \geq 1\}.$$

Fractional Hypertree Decompositions

Tree Decompositions + Fractional Edge Covers



Width of FHD: 2

fhw(Q) = minimum width over all FHDs of Q

Tractable CQ Answering for Bounded Width

Proposition

For every hypergraph H: $fhw(H) \leq ghw(H) \leq hw(H) \leq tw(H) + 1$.

Theorem

Answering CQs is tractable for classes of CQs with bounded

- tw [Chekuri and Rajaraman 1997, Kolaitis and Vardi 1998];
- hw, ghw [Gottlob, Leone, and Scarcello 1999], [Adler, Gottlob, and Grohe 2007]
- fhw [Grohe and Marx 2006], [Marx 2010].

Checking Low Width

CHECK($tw|hw|ghw|fhw$) for fixed $k \geq 1$	
input	hypergraph H
output	„yes“ if $tw(H)$ | $hw(H)$ | $ghw(H)$ | $fhw(H) \leq k$ (and output decomposition of width $\leq k$) „no“ otherwise

Checking Low Width

CHECK($tw|hw|ghw|fhw$) for fixed $\boldsymbol{k} \geq 1$	
input	hypergraph H
output	„yes" if $tw(H)$ | $hw(H)$ | $ghw(H)$ | $fhw(H) \leq k$ (and output decomposition of width $\leq k$) „no" otherwise

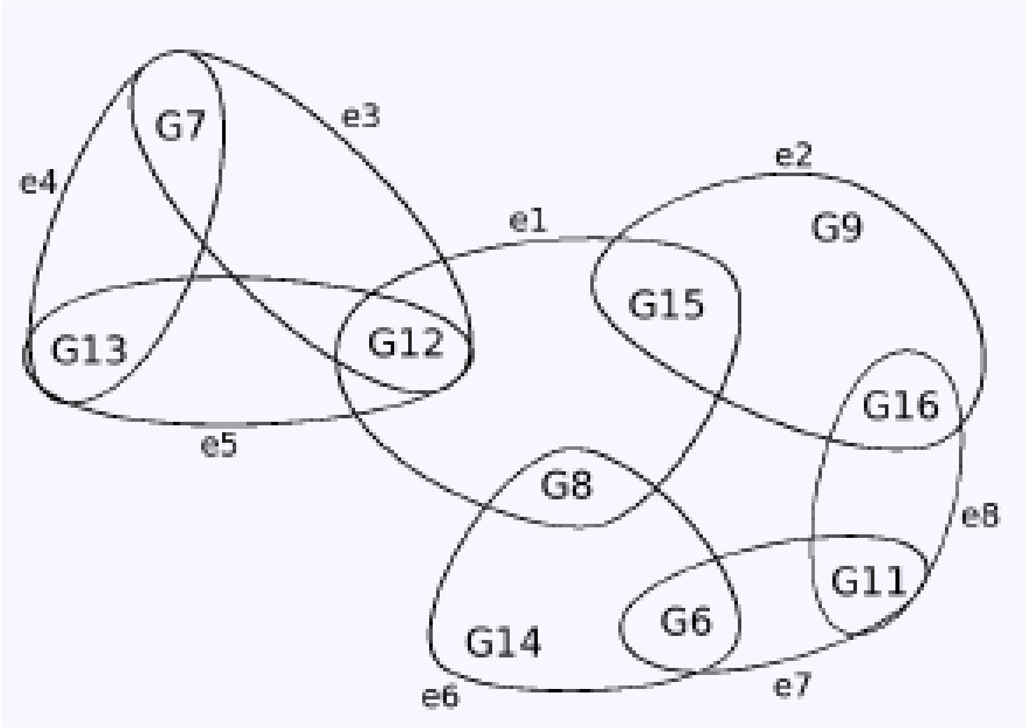
Complexity of the CHECK-Problem

- tw: **tractable** (even FPL in k) [Freuder 1990], [Bodlaender 1993]
- hw: **tractable** [Gottlob, Leone, and Scarcello 1999]
- ghw: **NP-complete** for $\boldsymbol{k} \geq \mathbf{3}$ [Gottlob, Miklos, and Schwentick 2007]
- fhw: **NP-complete** for $\boldsymbol{k} \geq \mathbf{2}$ [Fischl, Gottlob, and P. , 2018]

Roadmap

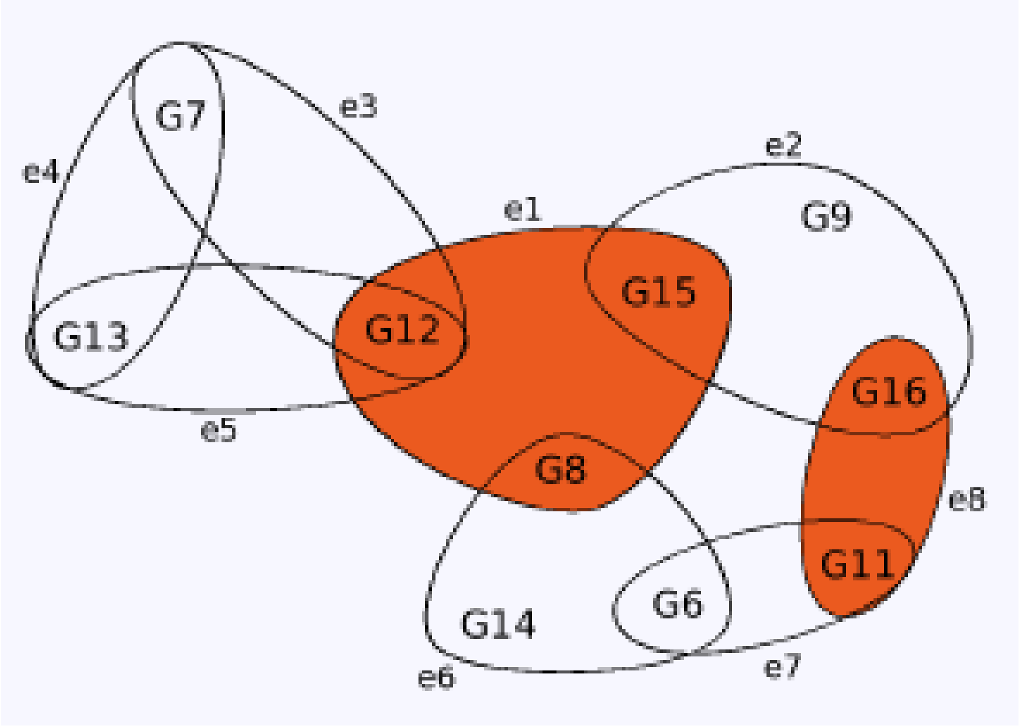
- 3 Problems: HOM, CSP, BCQ
- Hypergraphs and acyclicity
- Well-known width notions: tw, hw, ghw, fhw
- hw vs. ghw
- Tractable cases of ghw (and fhw) computation
- NP-hardness of the Check-problem for ghw and fhw
- A glimpse beyond fhw

Decomposition



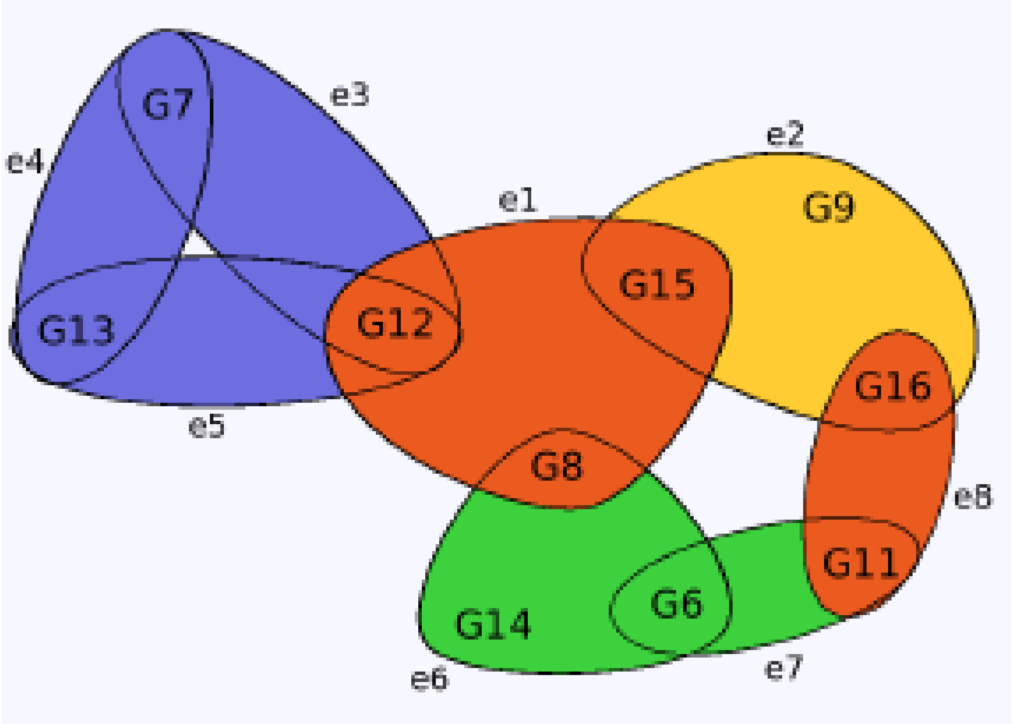
Decomposition

e1, e8



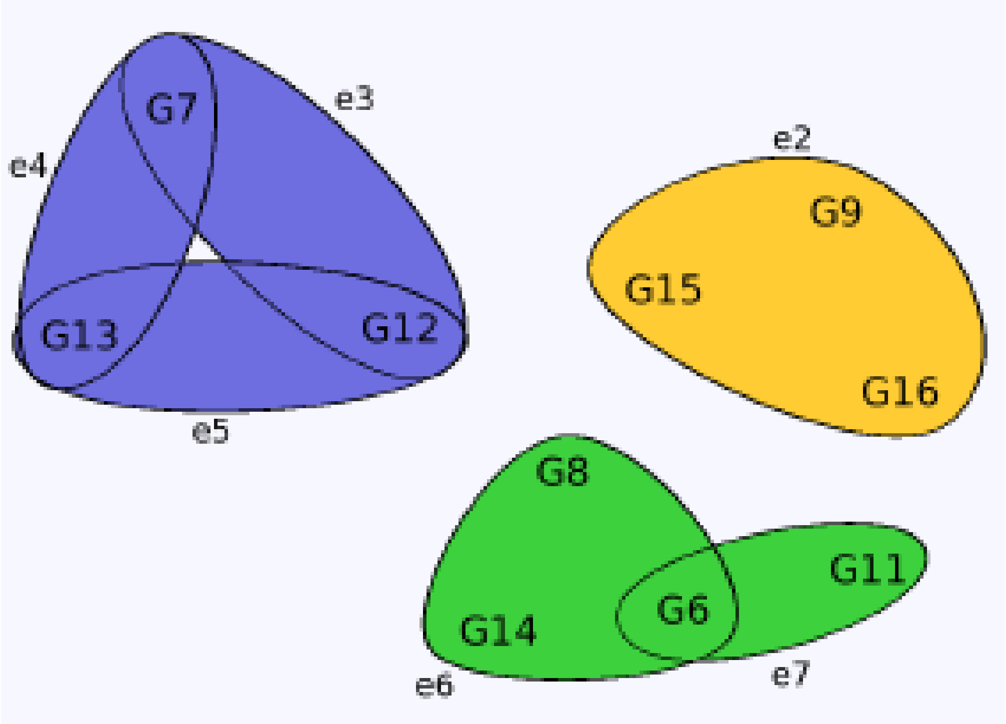
Decomposition

e1, e8

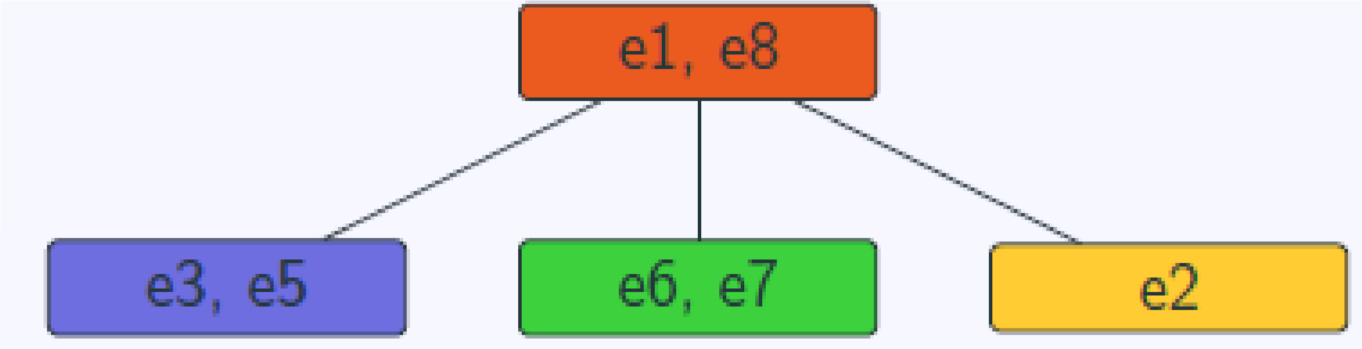


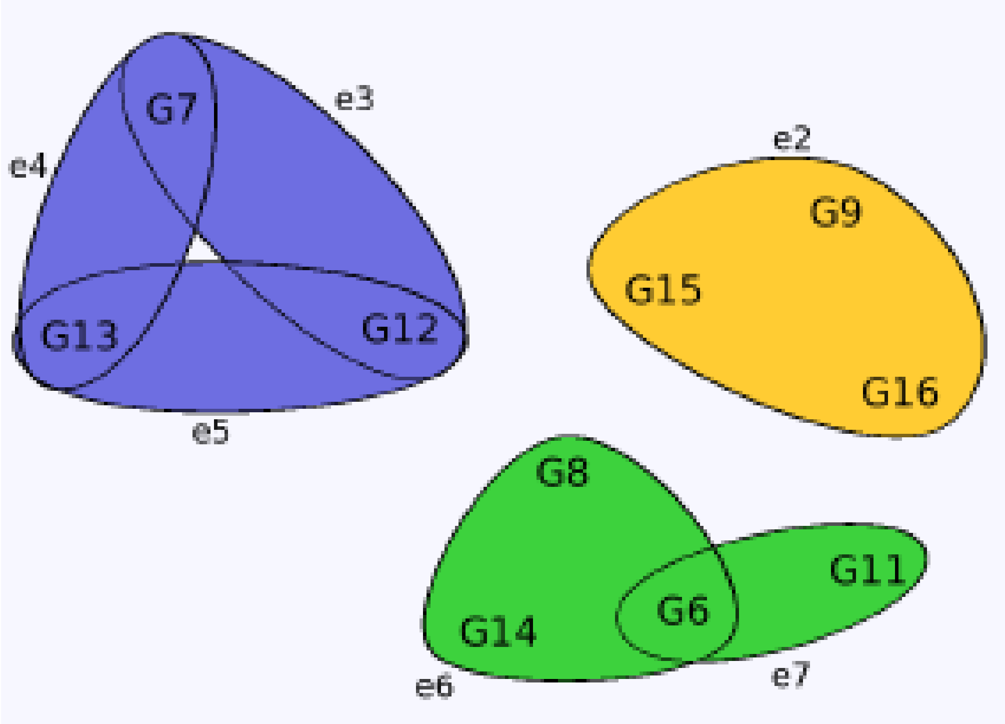
Decomposition

e1, e8



Decomposition





Components

[U]-Compnents

Consider hypergraph H and subset U of the vertices of H:

- Two edges e_1, e_2 are [U]-adjacent, if $(e_1 \cap e_2) \setminus U \neq \emptyset$.
- Define [U]-connectedness as transitive closure of [U]-adjacency.
- A [U]-component of H is a maximally [U]-connected subset of E(H).

Components

[U]-Compnents

Consider hypergraph H and subset U of the vertices of H:

- Two edges e_1, e_2 are [U]-adjacent, if $(e_1 \cap e_2) \setminus U \neq \emptyset$.
- Define [U]-connectedness as transitive closure of [U]-adjacency.
- A [U]-component of H is a maximally [U]-connected subset of E(H).

Observation [Gottlob, Leone, and Scarcello 1999]

Given an HD (likewise a GHD) T of width k, we can transform T into an HD (resp. GHD) T' of width ≤ k, such that for every node p in T' we have: each subtree rooted at a child node of p covers exactly one [χ(p)]-component of H.

Tractable Computation of an HD

Idea

Recursive procedure:

Input: a component C of the hypergraph H

the bag at the parent node p in the HD

{ guess an edge cover $\lambda(c)$ of size ≤ k at the child c of p;

determine the bag $\chi(c)$; // subset of $\bigcup\lambda(c)$ to ensure connectedness of

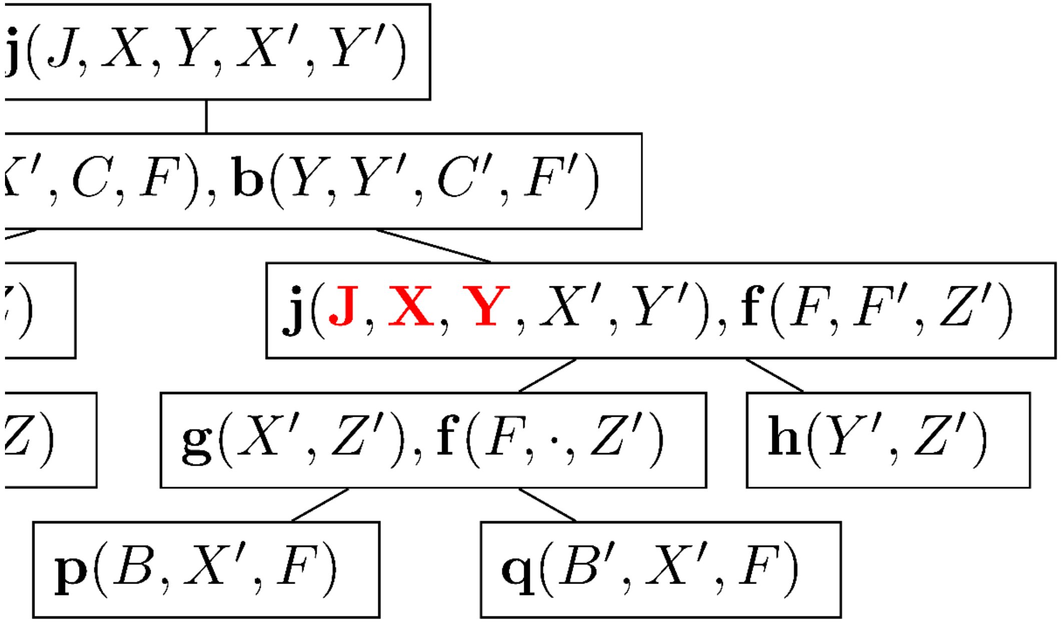
// $\chi(c)$ with C and all vertices in $C \cap \bigcup\lambda(c)$

determine the [$\chi(c)$]-components of H inside C;

recursively call the procedure for every such component }

Difficulty of Checking Low ghw

Hypertree Decomposition Computation

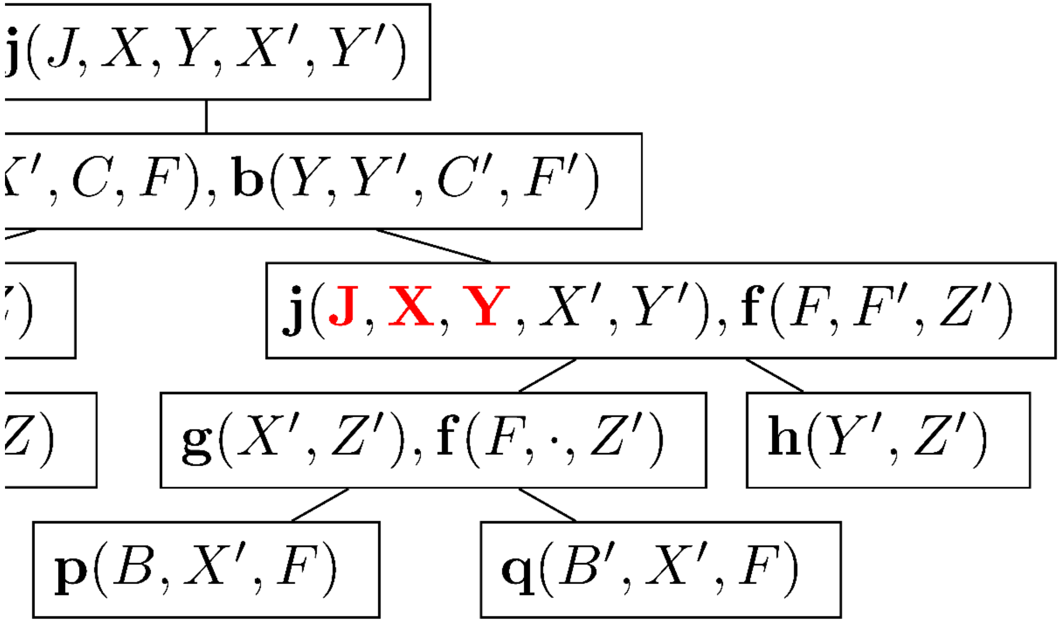


Vertices **disappearing**
may **never appear** below

- **Top down** construction of decomposition
- **Guess** $\leq k$ edges
- Bag of nodes **fully determined**

Difficulty of Checking Low *ghw*

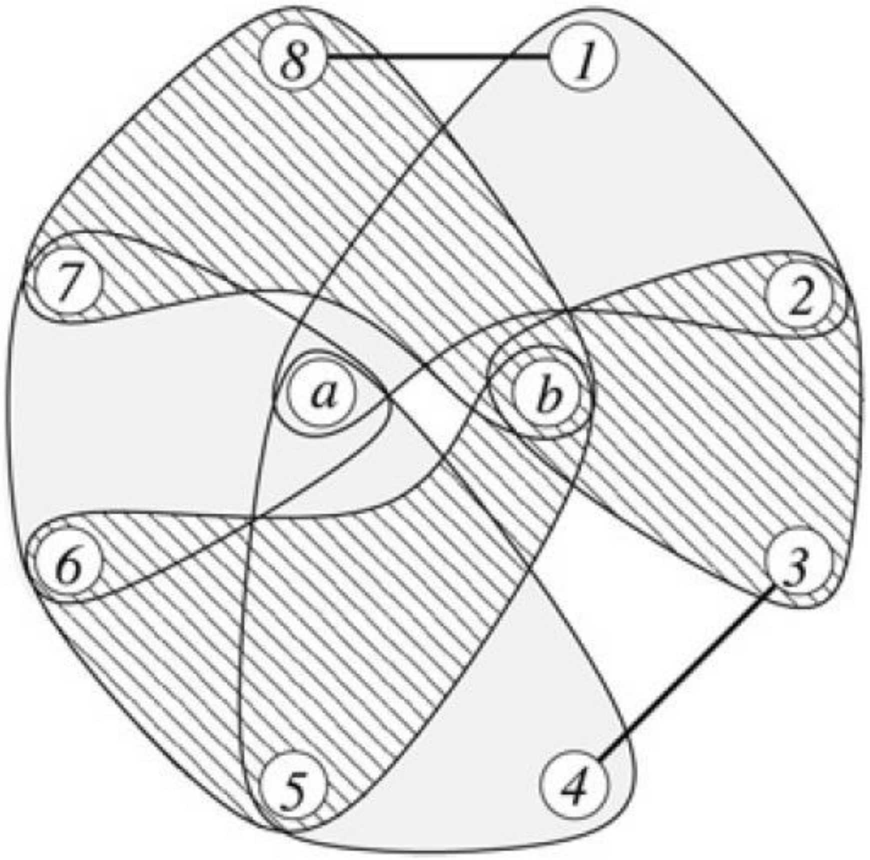
Generalized Hypertree Decomposition Computation



Vertices **disappearing** may **appear** below

- **Top down** construction of decomposition
- **Guess** $\leq k$ edges
- **Question**:
  How to determine bag of variables?
- **Problem**: (for unbounded arity)
  There are **exponentially many** possible subsets of the edge cover

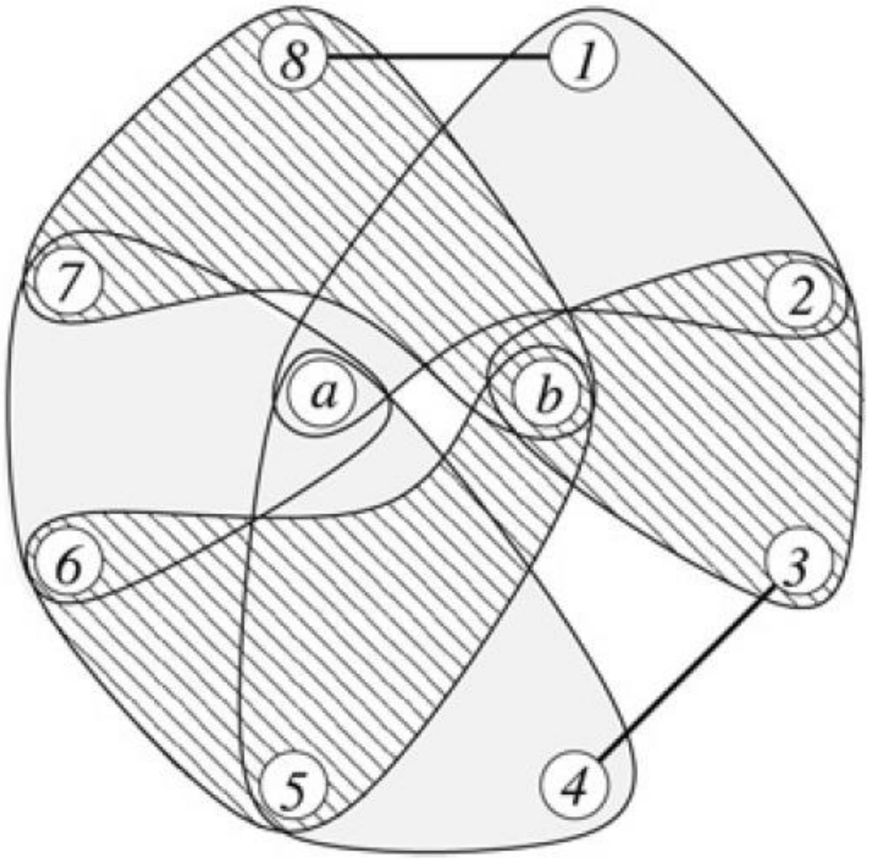
HW vs. GHW



$1,2,7,8,a,b$	$\{1,2,a\},\{7,8,b\}$
$2,6,7,a,b$	$\{2,3,b\},\{6,7,a\}$
$2,5,6,a,b$	$\{1,2,a\},\{5,6,b\}$
$2,3,4,5,a,b$	$\{2,3,b\},\{4,5,a\}$

GHD of width 2 [Adler, Gottlob, and Grohe 2007]

HW vs. GHW



$1,2,7,8,a,b$	$\{1,2,a\},\{7,8,b\}$

↓

$2,6,7,a,b$	$\{2,3,b\},\{6,7,a\}$

↓

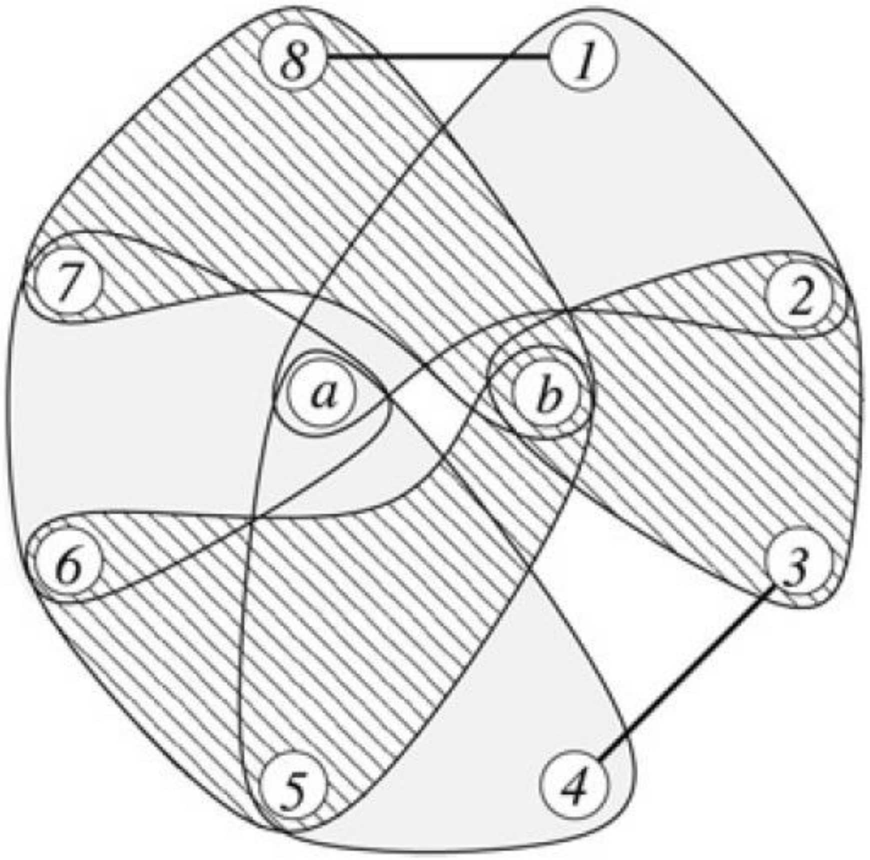
$2,5,6,a,b$	$\{1,2,a\},\{5,6,b\}$

↓

$2,3,4,5,a,b$	$\{2,3,b\},\{4,5,a\}$

violation of the special condition!

HW vs. GHW



$1,2,7,8,a,b$	$\{1,2,a\},\{7,8,b\}$
↓	
$2,3,6,7,a,b$	$\{2,3,b\},\{6,7,a\}$
↓	
$2,3,5,6,a,b$	$\{1,2,a\},\{2,3,b\},\{5,6,b\}$
↓	
$2,3,4,5,a,b$	$\{2,3,b\},\{4,5,a\}$

HD of width 3 [Adler, Gottlob, and Grohe 2007]

HW vs. GHW

Theorem [Adler, Gottlob, and Grohe 2007]

For every hypergraph H, we have $\mathrm{hw}(H) \leq 3 \cdot ghw(H) + 1$.
Hence, a class of hypergraphs has bounded hw iff it has bounded ghw.

HW vs. GHW

Theorem [Adler, Gottlob, and Grohe 2007]

For every hypergraph H, we have $\text{hw}(H) \leq 3 \cdot ghw(H) + 1$.
Hence, a class of hypergraphs has bounded hw iff it has bounded ghw.

Empirical observation: the difference between hw and ghw is much smaller in practice.

[Fischl, Gottlob, Longo, and P. 2019]

$hw \rightarrow ghw$	yes	no	timeout
$3 \rightarrow 2$	0	309 (10)	1
$4 \rightarrow 3$	0	262 (57)	124
$5 \rightarrow 4$	0	148 (13)	279
$6 \rightarrow 5$	18 (129)	180 (288)	261

GHW of instances with average runtime in s

Roadmap

- 3 Problems: HOM, CSP, BCQ
- Hypergraphs and acyclicity
- Well-known width notions: tw, hw, ghw, fhw
- hw vs. ghw
- Tractable cases of ghw (and fhw) computation
- NP-hardness of the Check-problem for ghw and fhw
- A glimpse beyond fhw

Tractable Classes

We were looking for restrictions giving large classes for which computing ghw and fhw is tractable, or fhw PTIME approximable (better than k^3)

Such classes should fullfil the following criteria:

Polynomial-time recognizable

Nontrivial: They should not guarantee tractability of CQ Answering by themselves (e.g. acyclic queries).

Realistic: A large proportion of the existing real-life benchmarks is covered, or some important classes (e.g. bounded arity).

Restrictions for Tractability

A class C of hypergraphs enjoys:

BIP (bounded intersection property): $\exists i\ \forall H \in C,\ \forall e_1,e_2 \in E(H),\ |e_1 \cap e_2| \leq i.$

BMIP (bd. multi-intersection prop.): $\exists i\ \exists c\ \forall H \in C,\ \forall e_1 \ldots e_c \in E(H),\ |e_1 \cap \ldots \cap e_c| \leq i.$

BR (bounded rank): $\exists r\ \forall H \in C\ \forall e \in E(H),\ |e| \leq i.$

BD (bounded degree): $\exists d\ \forall H \in C\ \forall v \in V(H),\ |\{e \in E(H)\ |\ v \in e\}| \leq d.$

BVC (bounded vc-dimension): $\exists \delta\ \forall H \in C\ \ vc(H) \leq \delta$

Restrictions for Tractability

A class C of hypergraphs enjoys:

BIP (bounded intersection property): $\exists i \, \forall H \in C, \forall e_1, e_2 \in E(H), \; |e_1 \cap e_2| \leq i.$

BMIP (bd. multi-intersection prop.): $\exists i \, \exists c \, \forall H \in C, \forall e_1 \ldots e_c \in E(H), \, |e_1 \cap \ldots \cap e_c| \leq i.$

BR (bounded rank): $\exists r \, \forall H \in C \, \forall e \in E(H), \; |e| \leq i.$

BD (bounded degree): $\exists d \, \forall H \in C \, \forall v \in V(H), \; |\{e \in E(H) \mid v \in e\}| \leq d.$

BVC (bounded vc-dimension): $\exists \delta \, \forall H \in C \;\; vc(H) \leq \delta$

Note: BR→BIP→BMIP→BVC; BD→BMIP; none of the implications reversible.

Results

Problem / Property	GHW=k	FHW=k
BIP	tractable	tractable (*)
BMIP	tractable	tractable (**)
BR	tractable	tractable
BD	tractable	tractable
BVC	NP-complete	PTIME Approx: O(k log k)

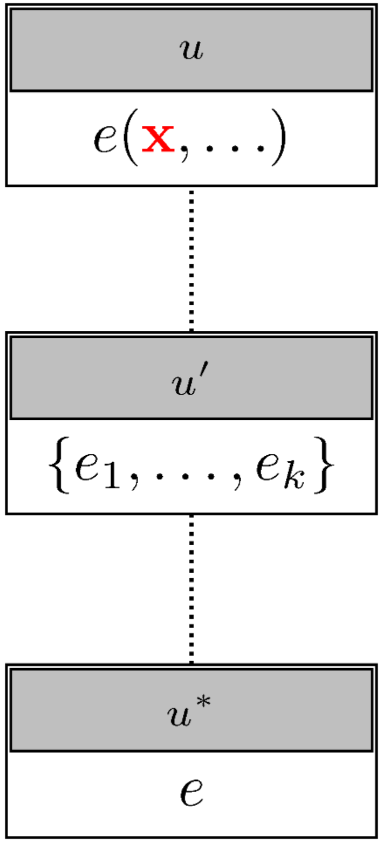
GHW Computation for Bounded Intersection

Goal: add polynomially many subedges to H so that $\chi(p) = \bigcup \lambda(p)$ at each node p of a GHD.

$$f(H,k) = \bigcup_{e \in \mathrm{E}(H)} \left(\bigcup_{e_1,\ldots,e_j \in (E(H) \setminus \{e\}), j \leq k} 2^{\left(e \cap \left(e_1 \cup \cdots \cup e_j\right)\right)} \right)$$

GHW Computation for Bounded Intersection

Goal: add polynomially many subedges to H so that $\chi(p) = \bigcup\lambda(p)$ at each node p of a GHD.



$$f(H,k) = \bigcup_{e \in E(H)} \left(\bigcup_{e_1,\ldots,e_j \in (E(H)\setminus\{e\}), j \le k} 2^{(e \cap (e_1 \cup \cdots \cup e_j))} \right)$$

- e must be fully covered at some node u^*
- **Case 1**: e is used in every cover along the path $u \leftrightarrow u^*$:
  simply add all vertices of e to all bags on this path.
- **Case 2**: e does not appear at some node u' on path $u \leftrightarrow u^*$:
  let $\lambda_{u'} = \{e_1, \ldots, e_k\}$
  by connectedness condition:
  $e \cap bag(u) \subseteq e \cap (e_1 \cup \cdots \cup e_k)$

Realistic Properties?

CQ Application

i	Deg	BIP	3-BMIP	4-BMIP	VC-dim
0	0	0	118	173	10
1	2	421	348	302	393
2	176	85	59	50	132
3	137	7	5	5	0
4	87	5	5	5	0
5	35	17	0	0	0
6	98	0	0	0	0

CSP Application & Other

i	Deg	BIP	3-BMIP	4-BMIP	VC-dim
0	0	0	597	603	0
1	0	1037	495	525	0
2	597	95	57	23	1115
3	6	29	21	21	52
4	20	10	2	0	0
5	6	0	0	0	0
>5	543	1	0	0	0

[Fischl, Gottlob, Longo, and P. 2019]

Roadmap

- 3 Problems: HOM, CSP, BCQ
- Hypergraphs and acyclicity
- Well-known width notions: tw, hw, ghw, fhw
- hw vs. ghw
- Tractable cases of ghw (and fhw) computation
- NP-hardness of the Check-problem for ghw and fhw
- A glimpse beyond fhw

Intractability of Checking Low Width

Theorem [Gottlob,Miklos, Schwentick 2007]

Checking whether a hypergraph H has $ghw(H) \leq 3$ is NP-complete.

Theorem [Fischl, Gottlob, P 2018]

Checking whether a hypergraph H has $fhw(H) \leq 2$ is NP-complete.

and as a side result:

Theorem [Fischl, Gottlob, P 2018]

Checking whether a hypergraph H has $ghw(H) \leq 2$ is NP-complete.

NP-Hardness Proof by Reduction from 3-SAT

- From propositional formula φ construct hypergraph H, s.t.

$$\varphi \text{ is satisfiable} \Leftrightarrow fhw(H) \leq 2 \text{ and } ghw(H) \leq 2$$

- Easy Part: φ is satisfiable $\Rightarrow fhw(H) \leq 2$ and $ghw(H) \leq 2$

NP-Hardness Proof by Reduction from 3-SAT

- From propositional formula φ construct hypergraph H, s.t.

$$\varphi \text{ is satisfiable} \Leftrightarrow fhw(H) \leq 2 \text{ and } ghw(H) \leq 2$$

- Easy Part: φ is satisfiable $\Rightarrow fhw(H) \leq 2$ and $ghw(H) \leq 2$
  - Intended form of decomposition: "long path"

NP-Hardness Proof by Reduction from 3-SAT

- From propositional formula φ construct hypergraph H, s.t.

$$\varphi \text{ is satisfiable} \Leftrightarrow fhw(H) \leq 2 \text{ and } ghw(H) \leq 2$$

- Easy Part: φ is satisfiable $\Rightarrow fhw(H) \leq 2$ and $ghw(H) \leq 2$
  - Intended form of decomposition: "long path"

- Hard Part: $fhw(H) \leq 2$ and $ghw(H) \leq 2 \Rightarrow \varphi$ is satisfiable

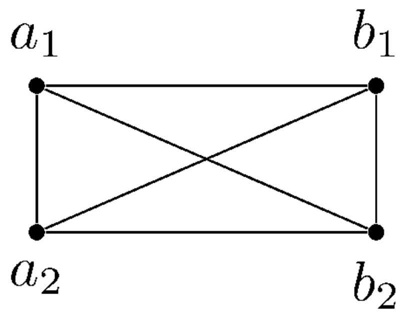
NP-Hardness Proof by Reduction from 3-SAT

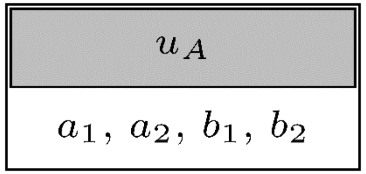
- From propositional formula φ construct hypergraph H, s.t.

$$\varphi \text{ is satisfiable} \Leftrightarrow fhw(H) \leq 2 \text{ and } ghw(H) \leq 2$$

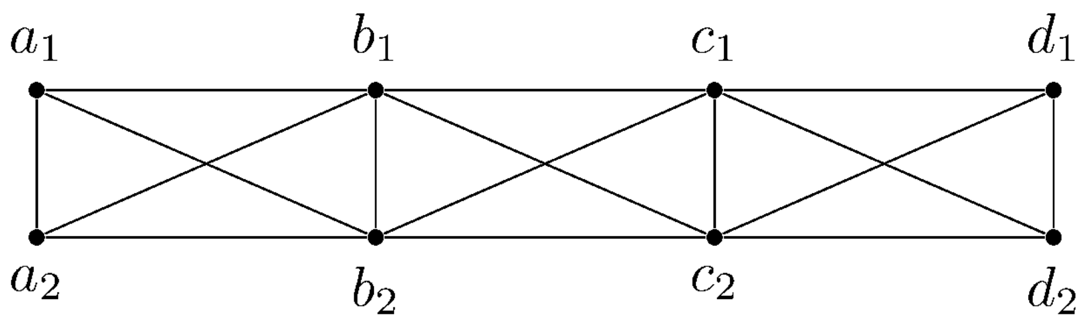
- Easy Part: φ is satisfiable $\Rightarrow fhw(H) \leq 2$ and $ghw(H) \leq 2$
  - Intended form of decomposition: "long path"
- Hard Part: $fhw(H) \leq 2$ and $ghw(H) \leq 2 \Rightarrow \varphi$ is satisfiable
  - Use gadgets to enforce intended form of decomposition
  - "Read off" truth assignment on "long path"

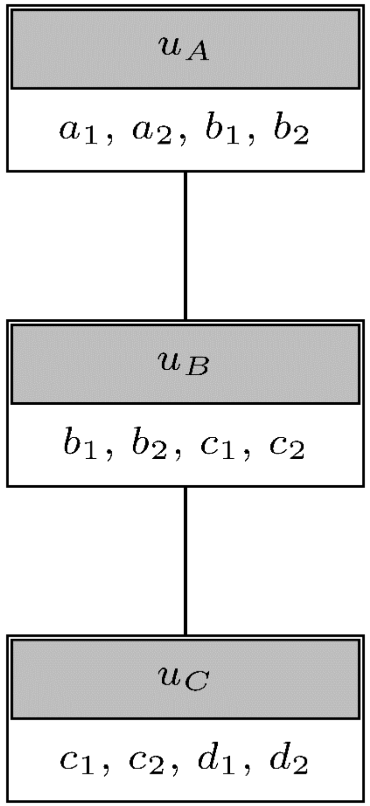
Gadgets H_0, H_0'



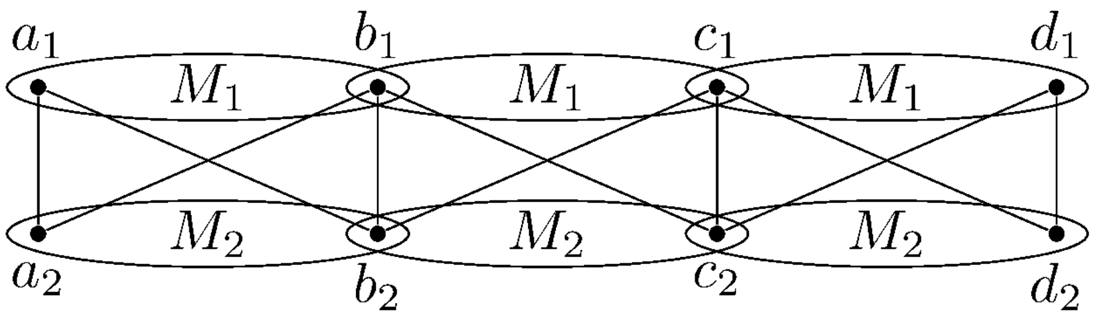


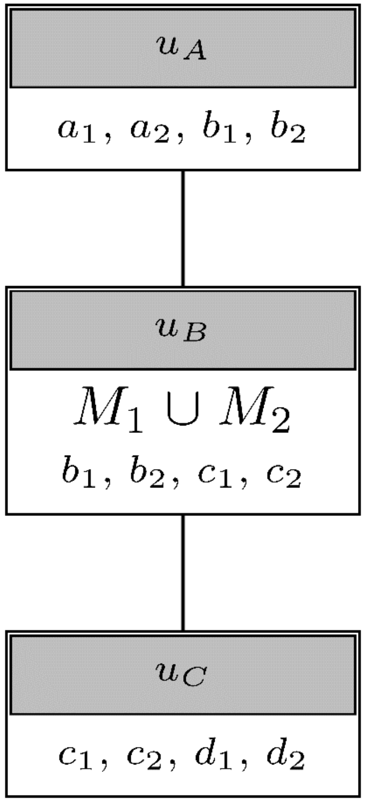
Gadgets H_0, H_0'



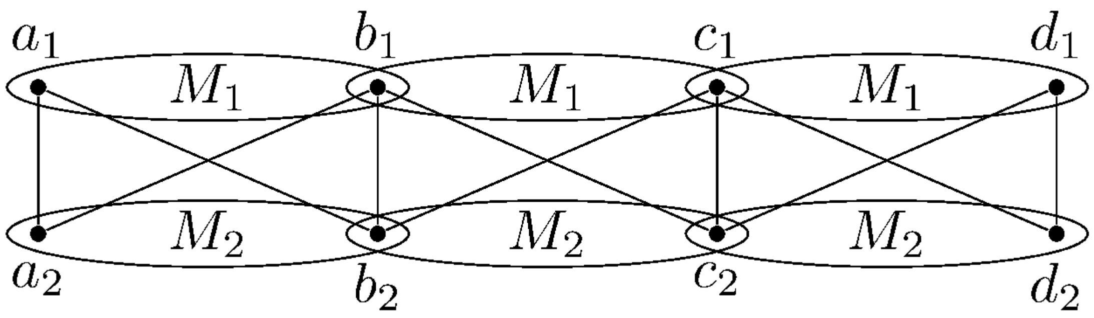


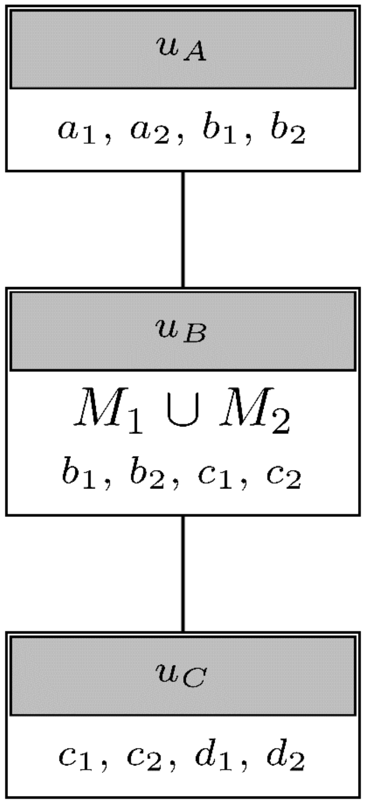
Gadgets H_0, H_0'





Gadgets H_0, H_0'





Variables in $\boldsymbol{\varphi}$: $\{\boldsymbol{x_1}, \dots, \boldsymbol{x_n}\}$

in H_0: $a_1, a_2, \dots, d_1, d_2$
$M_1 \cup M_2$: large set S and $Y = \{y_1, \dots, y_n\}$

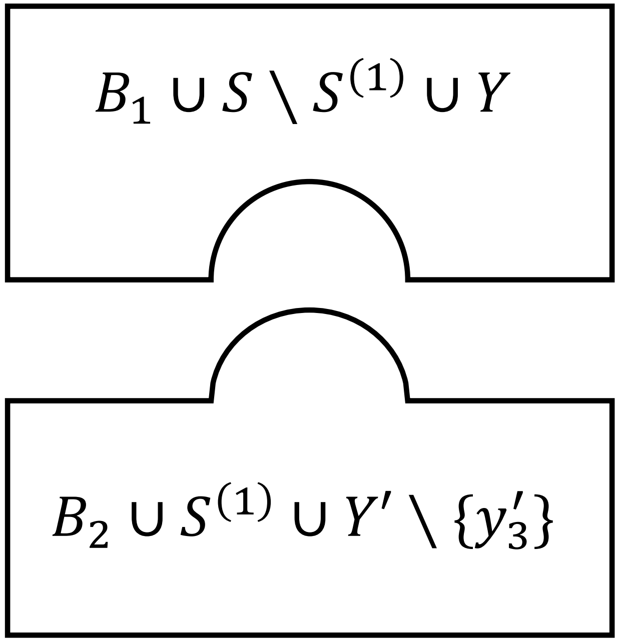
in H_0': $a_1', a_2', \dots, d_1', d_2'$
$M_1' \cup M_2'$: large set S and $Y' = \{y_1', \dots, y_n'\}$

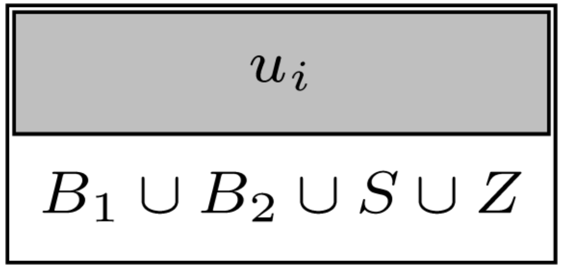
Encoding the clauses of φ

$$c = x_3 \lor \neg x_5 \lor x_8$$

$e_1^{(1)}$:

$e_2^{(1)}$:

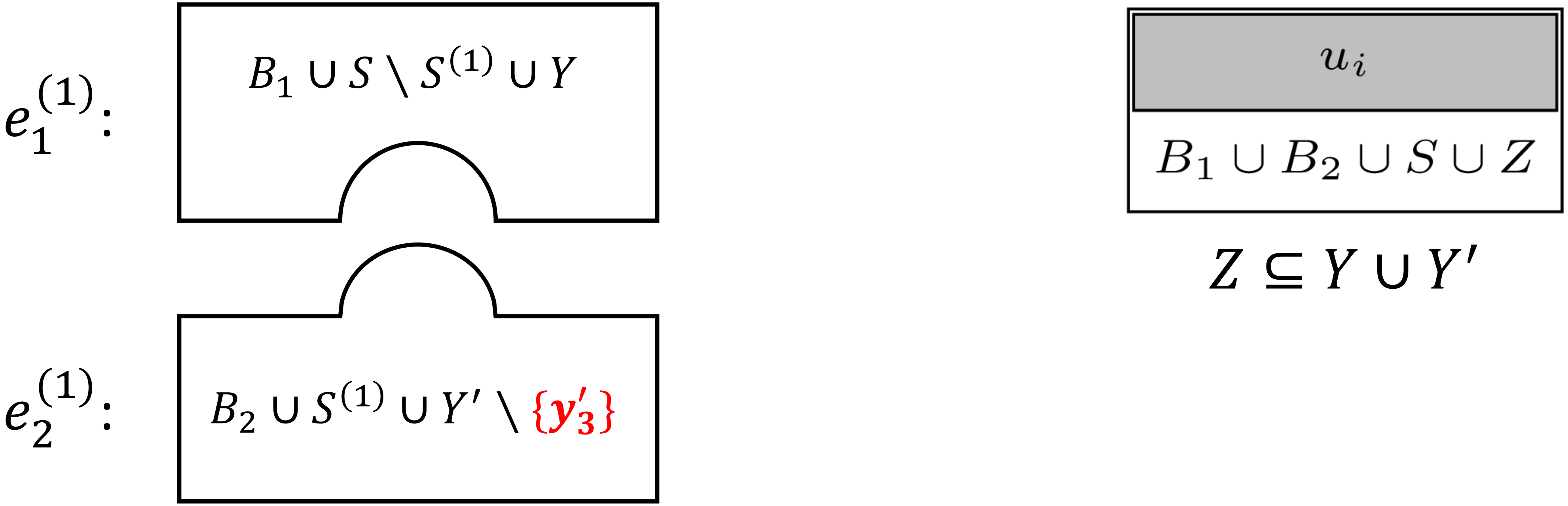




$$Z \subseteq Y \cup Y'$$

Encoding the clauses of φ

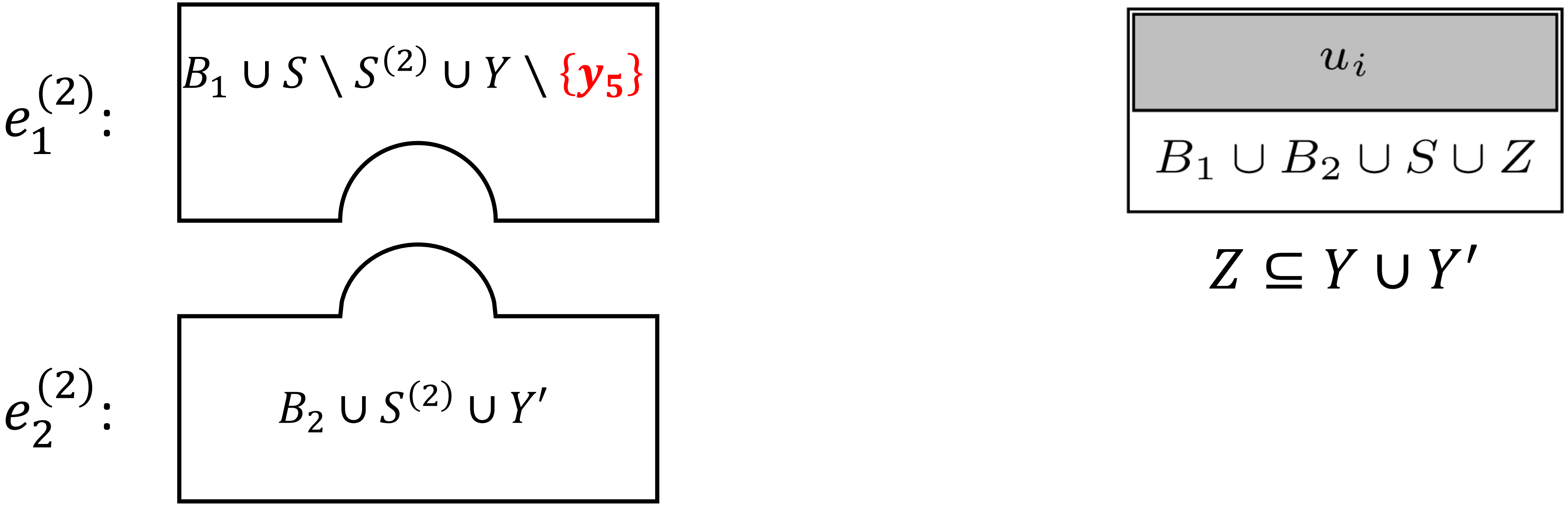
$$c = \boldsymbol{x_3} \lor \neg x_5 \lor x_8$$



set $\boldsymbol{x_3}$ **to true**: Z does not contain y'_3

Encoding the clauses of φ

$$c = x_3 \lor \neg x_5 \lor x_8$$

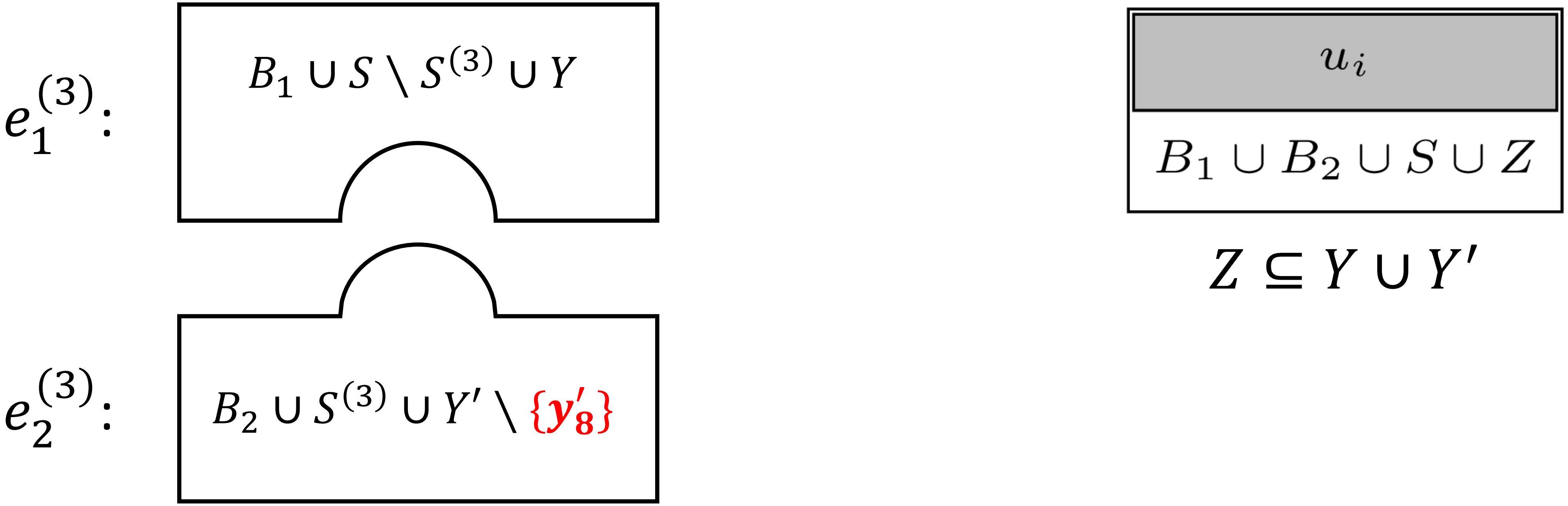


$Z \subseteq Y \cup Y'$

set **x_5 to false**: Z does not contain y_5

Encoding the clauses of φ

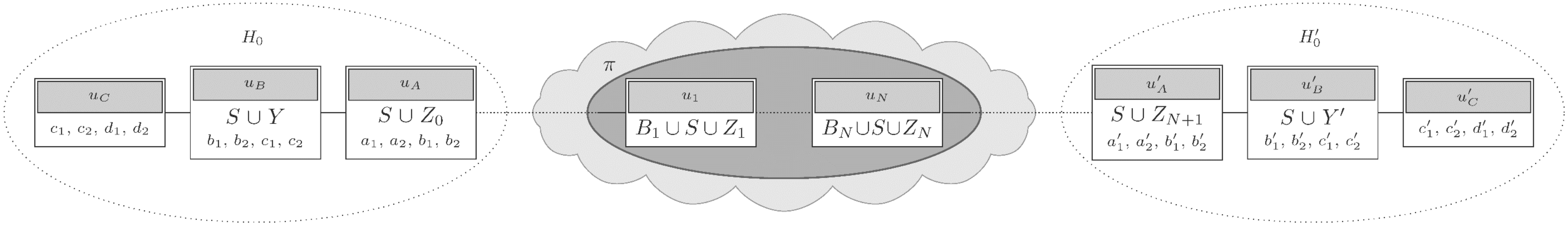
$$c = x_3 \vee \neg x_5 \vee \boldsymbol{x_8}$$



$Z \subseteq Y \cup Y'$

set $\boldsymbol{x_8}$ **to true**: Z does not contain y_8'

Intended Decomposition



- $Z_i \subseteq Y \cup Y'$

- Left to right:
  $\boldsymbol{Z_i \cap Y}$ monotonically **decreasing**
  $\boldsymbol{Z_i \cap Y'}$ monotonically **increasing**

- N big enough, s.t. $\boldsymbol{Z_i = Z_{i+1}}$ for some i
  $\Rightarrow$ read off **truth assignment at $\boldsymbol{Z_i}$**

Roadmap

- 3 Problems: HOM, CSP, BCQ
- Hypergraphs and acyclicity
- Well-known width notions: tw, hw, ghw, fhw
- hw vs. ghw
- Tractable cases of ghw (and fhw) computation
- NP-hardness of the Check-problem for ghw and fhw
- A glimpse beyond fhw

Motivation

Best algorithm based on fhw (likewise ghw, hw, tw):

- Choose ***optimal tree*** T for Q
- Compute full CQ Q_t for all t ∈Nodes(T)
- Run Yannakakis algorithm on the join tree

Motivation

Best algorithm based on fhw (likewise ghw, hw, tw):

- Choose ***optimal tree*** T for Q
- Compute full CQ Q_t for all t $\in$Nodes(T)
- Run Yannakakis algorithm on the join tree

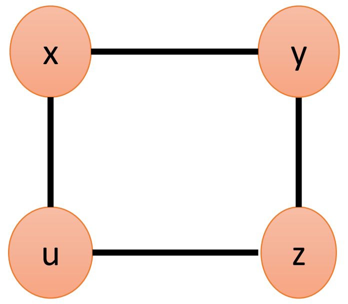
Total time: $= O(|Q| * N^{fhw(Q)} + |Output|)$,

with N = max-size of relations

However, this is not optimal!

The 4-Cycle Query

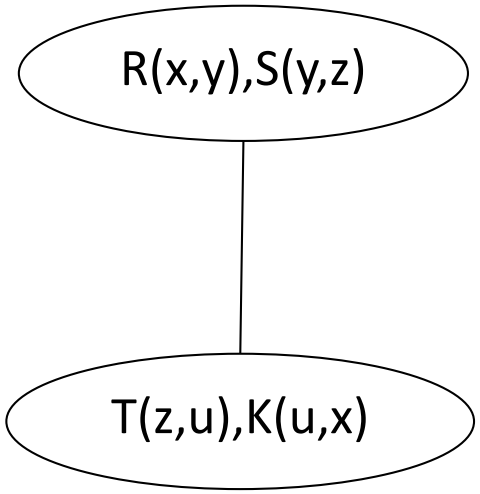
Q() = R(x,y),S(y,z),T(z,u),K(u,x)



The 4-Cycle Query

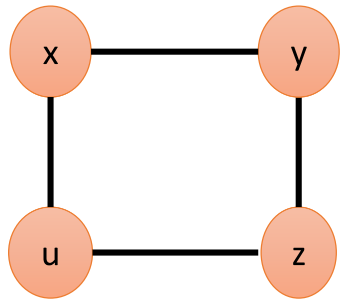
$Q() = R(x,y),S(y,z),T(z,u),K(u,x)$

Tree T1=



$\rho^*=2$

$\rho^*=2$

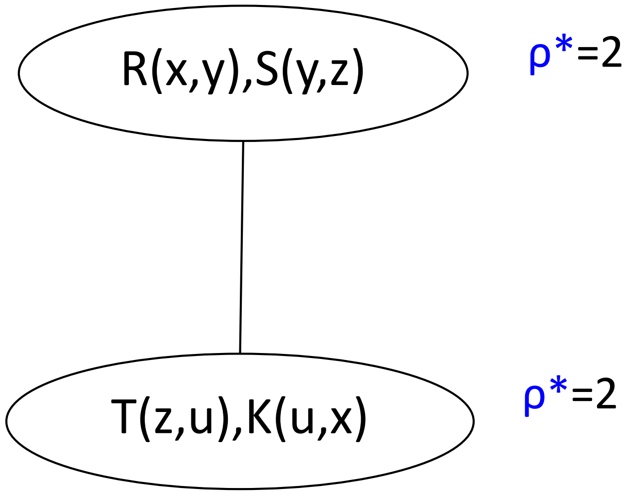


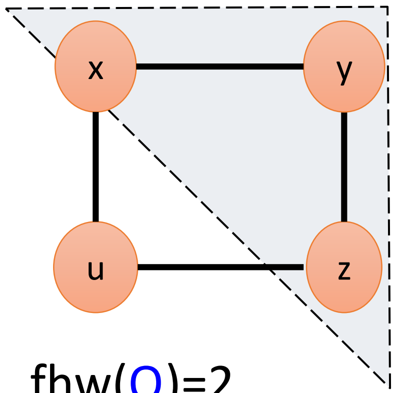
fhw(Q)=2

The 4-Cycle Query

Q() = R(x,y),S(y,z),T(z,u),K(u,x)

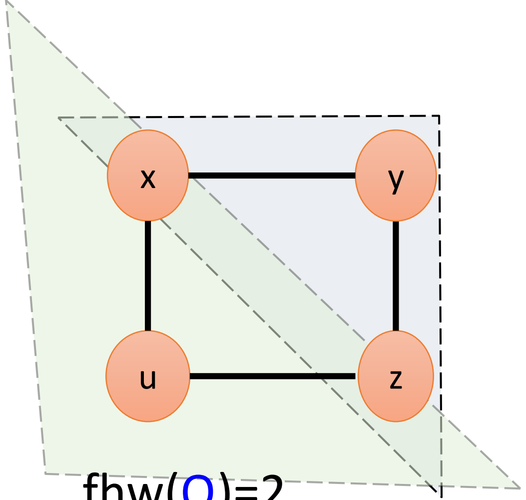
Tree T1=





fhw(Q)=2

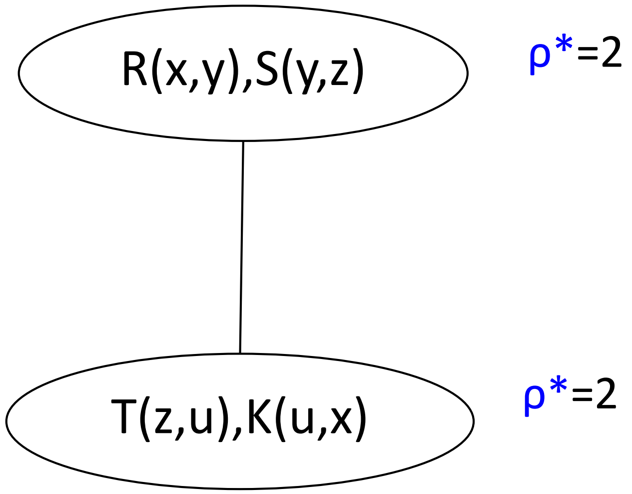
The 4-Cycle Query



fhw(Q)=2

Q() = R(x,y),S(y,z),T(z,u),K(u,x)

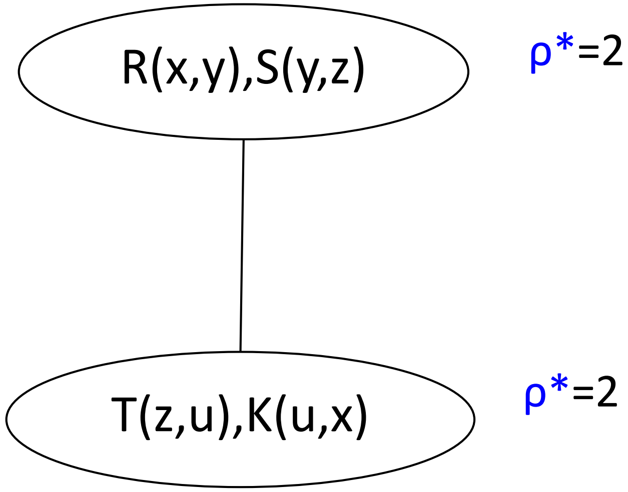
Tree T1=



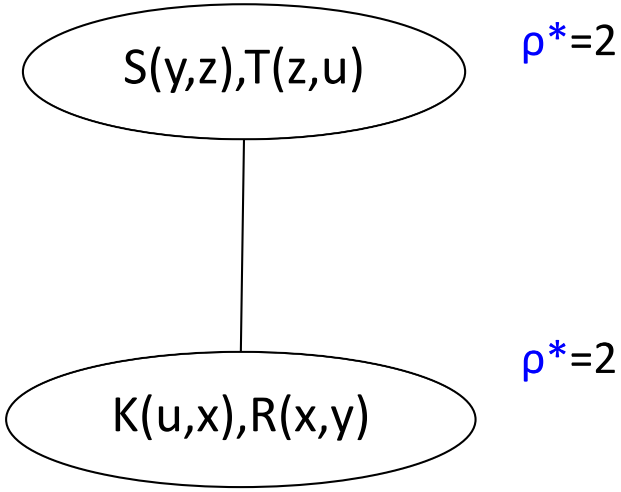
The 4-Cycle Query

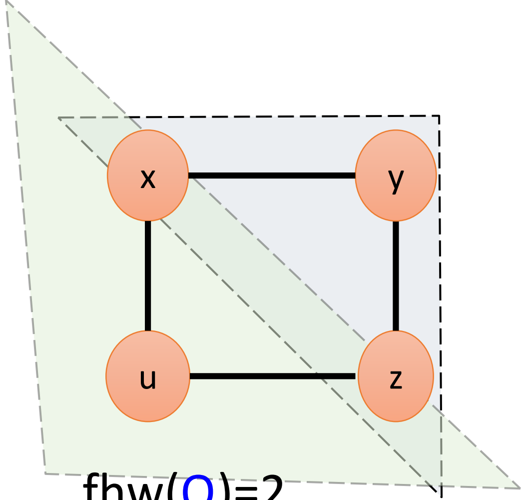
Q() = R(x,y),S(y,z),T(z,u),K(u,x)

Tree T1=



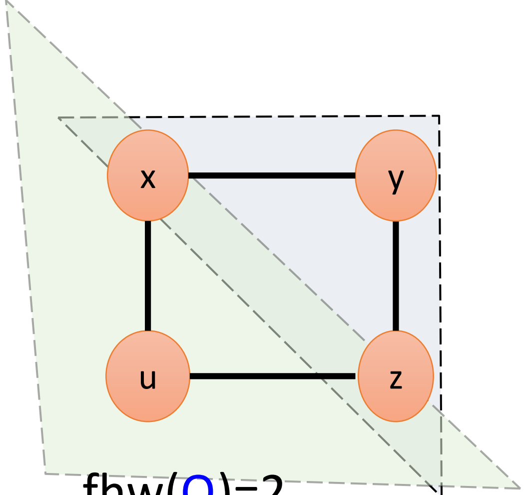
Tree T2=





fhw(Q)=2

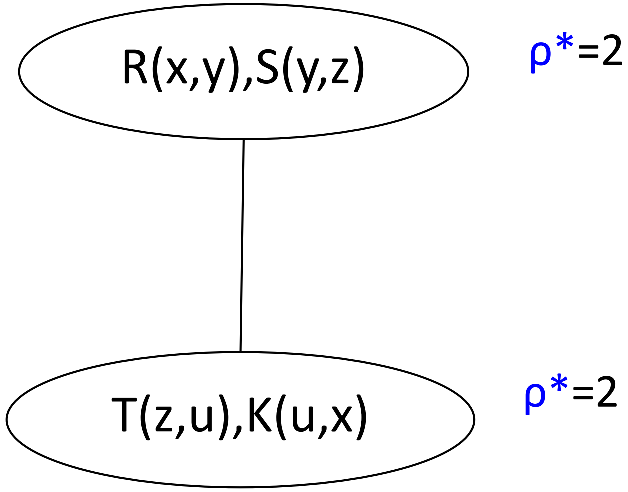
The 4-Cycle Query



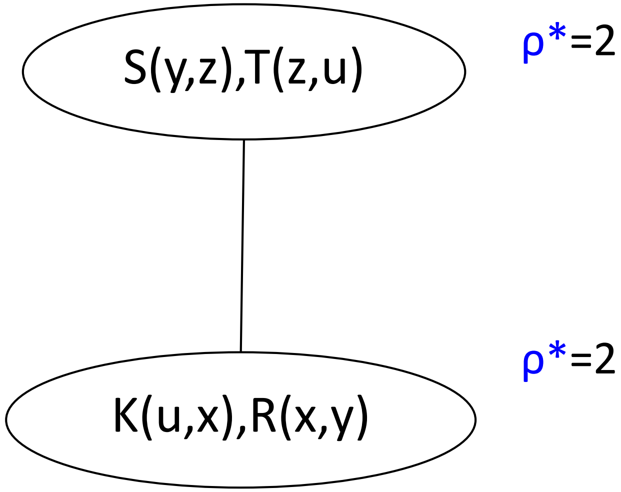
fhw(Q)=2

Q() = R(x,y),S(y,z),T(z,u),K(u,x)

Tree T1=

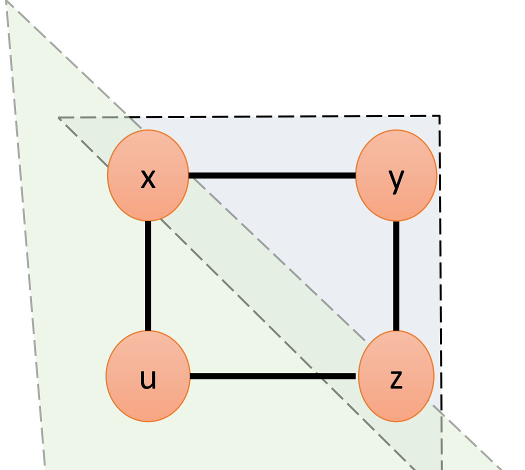


Tree T2=



If we choose T1, then time = $\Omega(N^2)$ on R=T=[N]×[1], S=K=[1]×[N]

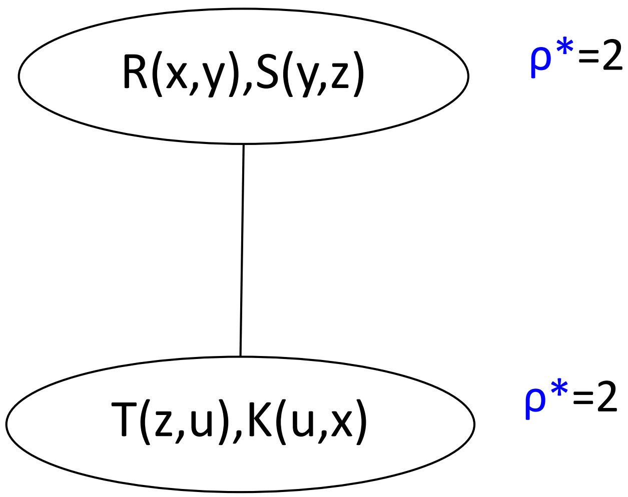
The 4-Cycle Query



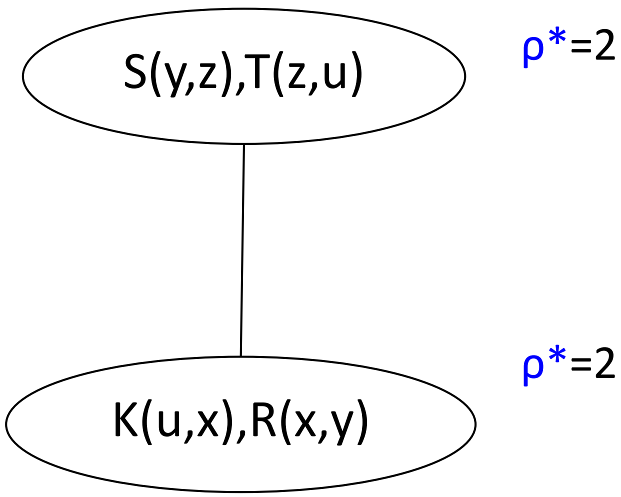
fhw(Q)=2

Q() = R(x,y),S(y,z),T(z,u),K(u,x)

Tree T1=



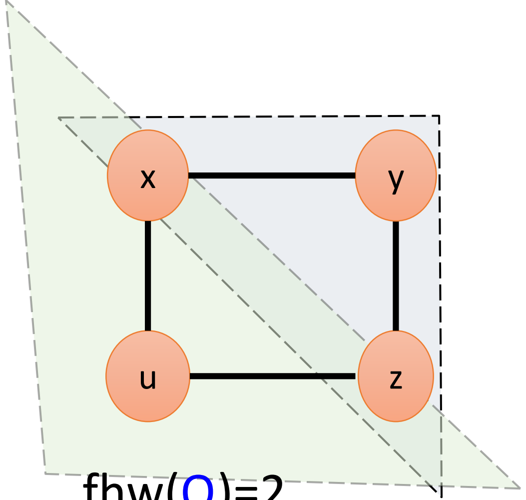
Tree T2=



If we choose T1, then time = $\Omega(N^2)$ on R=T=[N]×[1], S=K=[1]×[N]

If we choose T2, then time = $\Omega(N^2)$ on R=T=[1]×[N], S=K=[N]×[1]

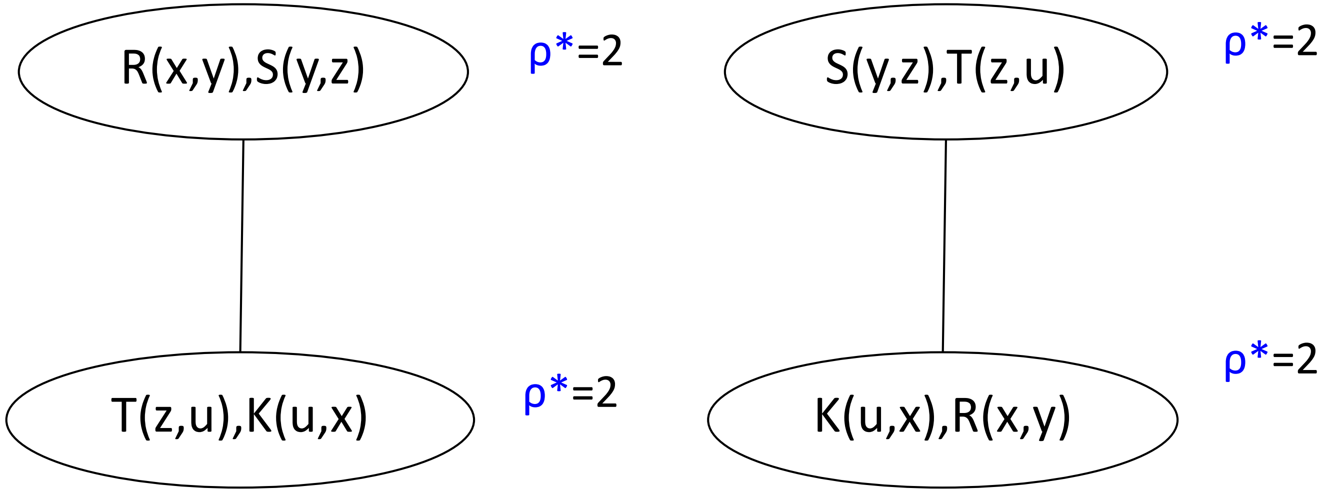
The 4-Cycle Query



fhw(Q)=2

Q() = R(x,y),S(y,z),T(z,u),K(u,x)

Tree T1= Tree T2=



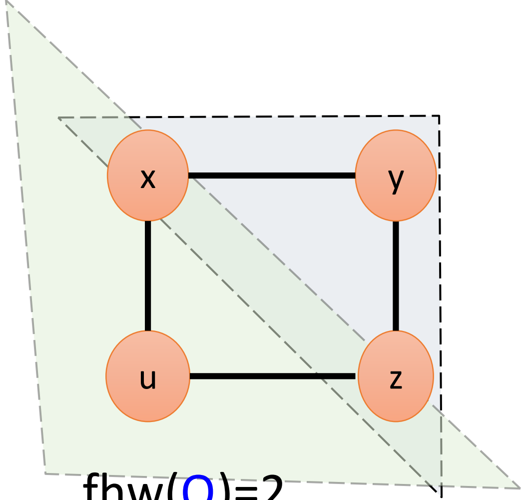
If we choose T1, then time = $\Omega(N^2)$ on R=T=[N]×[1], S=K=[1]×[N]

If we choose T2, then time = $\Omega(N^2)$ on R=T=[1]×[N], S=K=[N]×[1]

Best runtime using traditional tree decompositions = $O(N^2)$

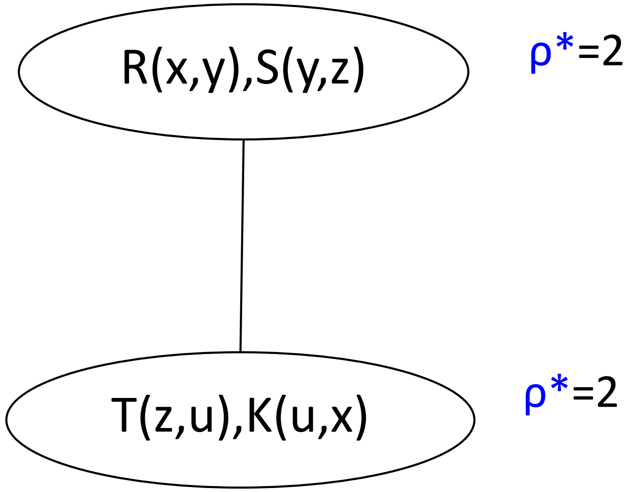
The 4-Cycle Query

$Q() = R(x,y),S(y,z),T(z,u),K(u,x)$

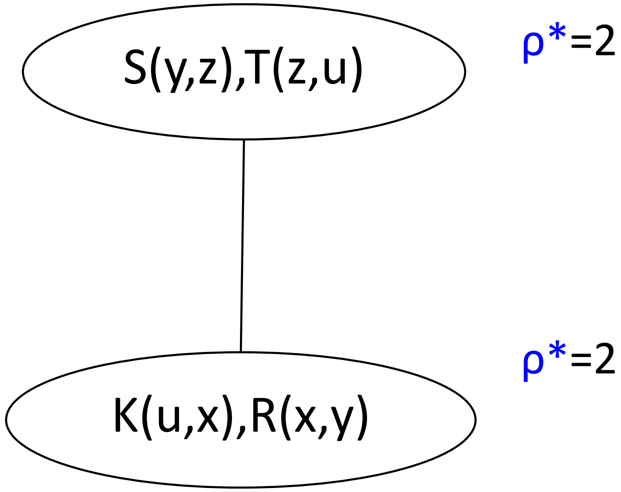


fhw(Q)=2

Tree T1=



Tree T2=



If we choose T1, then time = $\Omega(N^2)$ on R=T=[N]×[1], S=K=[1]×[N]

If we choose T2, then time = $\Omega(N^2)$ on R=T=[1]×[N], S=K=[N]×[1]

Best runtime using traditional tree decompositions = $O(N^2)$

[Alon,Yuster,Zwick 1997] $O(N^{3/2})$ algorithm for detecting a 4-cycle

General Framework for Defining Width

f-width:

- let $f: 2^V \rightarrow R^+$
- f-width of a tree decomposition T: $\max(\{f(B_t) | t \in V(T)\})$
- f-width of a hypergraph H: =
  minimum f-width over all tree decompositions of H

General Framework for Defining Width

f-width:

- let $f: 2^V \rightarrow R^+$
- f-width of a tree decomposition T: $\max(\{f(B_t) | t \in V(T)\})$
- f-width of a hypergraph H: = minimum f-width over all tree decompositions of H

$tw(H) \quad = s\text{-}width(H)$ with $s(B) = |B| - 1$
$ghw(H) = \rho_H\text{-}width(H)$ with $\rho_H(B) =$ edge cover number of B
$fhw(H) = \rho_H^*\text{-}width(H)$ with $\rho_H^*(B) =$ fractional edge cover number of B

Remark: hw with the special condition is outside this framework

Duality of Linear Programs

- The dual of covering is independence.
- $X \subseteq V(H)$ is independent, if $|e \cap X| \leq 1$ for every $e \in E(H)$
- $\Phi\colon V(H) \to [0,1]$ is a fractional independent set (FIS), if $\sum_{\{v \in \mathrm{e}\}} \Phi(v) \leq 1$ for every $e \in E(H)$

Duality of Linear Programs

- The dual of covering is independence.
- $X \subseteq V(H)$ is independent, if $|e \cap X| \leq 1$ for every $e \in E(H)$
- $\Phi\colon V(H) \to [0,1]$ is a fractional independent set (FIS), if $\sum_{\{v \in \mathrm{e}\}} \Phi(v) \leq 1$ for every $e \in E(H)$
- The fractional independent set number α_H^* of H is the maximum of $\sum_{\{v \in V(H)\}} \Phi(v)$ over all fractional independent sets Φ of H.
- By duality of Linear Programs, we have $\rho_H^* = \alpha_H^*$

Adaptive Width (adw)

F-width:

- let F be a set of functions $f: 2^V \rightarrow R^+$
- *F-width* of a tree decomposition T: $\sup(\{f\text{-width}(H) \mid f \in F\})$

Adaptive Width (adw)

F-width:

- let F be a set of functions $f: 2^V \to R^+$
- *F-width* of a tree decomposition T: $\sup(\{f\text{-width}(H) \mid f \in F\})$

Adaptive width (adw) [Marx 2011]:

$adw(H) =$ *F-width*(*H*), where F = set of all FIS of *H*.

Adaptive Width (adw)

F-width:

- let F be a set of functions $f: 2^V \to R^+$
- *F-width* of a tree decomposition T: $\sup(\{f\text{-width}(H) \mid f \in F\})$

Adaptive width (adw) [Marx 2011]:

$adw(H) =$ *F-width*(*H*), where F = set of all FIS of *H*.

BCQ_{tt}-problem (= BCQ with truth tables)
BCQ answering problem, where the database is given as truth tables (i.e., relation of arity k is given as set of all k-tuples over the domain with values true/false.

Adaptive Width (adw)

F-width:

- let F be a set of functions $f : 2^V \rightarrow R^+$
- *F-width* of a tree decomposition T: $\sup(\{f\text{-width}(H) \mid f \in F\})$

Adaptive width (adw) [Marx 2011]:

$adw(H) =$ *F-width*(*H*), where F = set of all FIS of *H*.

BCQ_{tt}-problem (= BCQ with truth tables)

BCQ answering problem, where the database is given as truth tables (i.e., relation of arity k is given as set of all k-tuples over the domain with values true/false.

Theorem [Marx 2011]

Let C be a class of BCQs of bounded adaptive width.
Then the BCQ_{tt}-problem for C is in PTIME.

fhw vs. adw

Let F = set of all FIS of *H* and let G = set of fractional edge covers:

- $fhw(H) = min_T max_t \underbrace{min_{\Psi \in G} \Psi(B_t) =}$

$$\rho_H^*(B_t) = \alpha_H^*(B_t)$$

$$min_T max_t \overbrace{max_{\{\Phi \in F\}} \Phi(B_t)}$$

fhw vs. adw

Let F = set of all FIS of *H* and let G = set of fractional edge covers:

- $fhw(H) = min_T max_t \underbrace{min_{\Psi \in G} \Psi(B_t)}_{\rho_H^*(B_t) \,=\, \alpha_H^*(B_t)} =$

$$min_T max_t \overbrace{max_{\{\Phi \in F\}} \Phi(B_t)}$$

Big difference! In adw, we are allowed to choose T ***after*** we see Φ

- $adw(H) = max_{\{\Phi \in F\}} min_T max_t \Phi(B_t)$

- Easy to check: $adw(H) \leq fhw(H)$
- Fact: bounded fhw does not imply bounded adw.

Towards Submodular-Width (subw)

Properties of fractional independent sets:

- non-negative: $f: 2^V \rightarrow R^+$
- edge-dominated: $f(e) \leq 1$ for every e $\in E(H)$
- modular: $f(X) + f(Y) = f(X \cup Y) + f(X \cap Y)$ for every $\mathrm{X}, \mathrm{Y} \subseteq V(H)$
- $f(\emptyset) = 0$
- (and therefore monotone)

Towards Submodular-Width (subw)

Properties of fractional independent sets:

- non-negative: $f: 2^V \to R^+$
- edge-dominated: $f(e) \leq 1$ for every e $\in E(H)$
- modular: $f(X) + f(Y) = f(X \cup Y) + f(X \cap Y)$ for every X, Y $\subseteq V(H)$
- $f(\emptyset) = 0$
- (and therefore monotone)

Relaxation:

- non-negative: $f: 2^V \to R^+$
- edge-dominated: $f(e) \leq 1$ for every e $\in E(H)$
- submodular: $f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y)$ for every X, Y $\subseteq V(H)$
- monotone: X $\subseteq Y \Rightarrow f(X) \leq f(Y)$
- $f(\emptyset) = 0$

Submodular-Width (subw)

Submodular-width (subw) [Marx 2013]

$subw(H) = F\text{-}width(H)$, where F is the set of non-negative, monotone, edge-dominated, submodular functions with $f(\emptyset) = 0$.

Submodular-Width (subw)

Submodular-width (subw) [Marx 2013]

$subw(H) = F\text{-}width(H)$, where F is the set of non-negative, monotone, edge-dominated, submodular functions with $f(\emptyset) = 0$.

BCQ (H) for a class H of hypergraphs.

BCQ-answering problem restricted to BCQs whose hypergraphs are in class H.

Theorem [Marx 2013]

Let C be a recursively enumerable class of hypergraphs. Then, assuming the Exponential Time Hypothesis, BCQ(C) is fixed-parameter tractable with query Q as parameter, if and only if C has bounded submodular-width.

subw vs. adw vs. fhw

Easy observation

For every hypergraph *H*: $adw(H) \leq subw(H)$.
Immediate from the fact that subw is max over a bigger set than adw.

subw vs. adw vs. fhw

Easy observation

For every hypergraph *H*: $adw(H) \leq subw(H)$.
Immediate from the fact that subw is max over a bigger set than adw.

Lemma [Marx 2013]

For every hypergraph *H*: $subw(H) \leq fhw(H)$.

Theorem [Marx 2013]

For every hypergraph H, we have $subw(H) = O(adw(H)^4)$.
Hence, a class C of hypergraphs has bounded subw iff it has bounded adw.

Current State of Affairs

- Relationship between various notions of width is well understood
- Boundary of tractability of the Check-problem
- Progress with computation of HDs, GHDs, and FHDs
- Precise characterization of FPT CQ-Answering
- Precise characterization of PTIME CQ-Answering for bounded arity

Future Work

- Precise characterization of PTIME CQ-Answering for unbounded arity
- Further improvement of HD, GHD, and FHD computation
- Decomposition-based query answering vs. cost-based optimization

References: Classics

- [Bodlaender, 1993] Hans L. Bodlaender: A linear time algorithm for finding tree-decompositions of small treewidth. STOC 1993: 226-234
- [Chandra and Merlin 1977] Ashok K. Chandra and Philip M. Merlin: Optimal implementation of conjunctive queries in relational data bases. STOC 1977: 77–90.
- [Feder and Vardi 1993] Tomás Feder, Moshe Y. Vardi: Monotone monadic SNP and constraint satisfaction. STOC 1993: 612-622
- [Freuder, 1990] Eugene C. Freuder: Complexity of K-Tree Structured Constraint Satisfaction Problems. AAAI 1990: 4-9
- [Kolaitis and Vardi, 1998]: Phokion G. Kolaitis, Moshe Y. Vardi: Conjunctive-Query Containment and Constraint Satisfaction. PODS 1998: 205-213
- [Yannakakis 1981] Mihalis Yannakakis: Algorithms for Acyclic Database Schemes. VLDB 1981: 82-94

References: Width Notions

- [Abo Khamis, Ngo, Suciu 2017] Mahmoud Abo Khamis, Hung Q. Ngo, Dan Suciu: What Do Shannon-type Inequalities, Submodular Width, and Disjunctive Datalog Have to Do with One Another? PODS 2017: 429-444
- [Adler, Gottlob, Grohe, 2007] Isolde Adler, Georg Gottlob, Martin Grohe: Hypertree width and related hypergraph invariants. Eur. J. Comb. 28(8): 2167-2181 (2007)
- [Chekuri and Rajaraman 1997] Chandra Chekuri, Anand Rajaraman: Conjunctive Query Containment Revisited. ICDT 1997: 56-70
- [Gottlob, Leone, Scarcello 1999] Georg Gottlob, Nicola Leone, Francesco Scarcello: Hypertree Decompositions and Tractable Queries. PODS 1999: 21-32
- [Grohe 2007] Martin Grohe: The complexity of homomorphism and constraint satisfaction problems seen from the other side. J. ACM 54(1): 1:1-1:24 (2007)

References: Width Notions (cont.)

- [Gottlob, Miklos, Schwentick 2007] Georg Gottlob, Zoltán Miklós, Thomas Schwentick: Generalized hypertree decompositions: NP-hardness and tractable variants. PODS 2007: 13-22
- [Grohe and Marx 2006] Martin Grohe, Dániel Marx: Constraint solving via fractional edge covers. SODA 2006: 289-298
- [Marx, 2010] Dániel Marx: Approximating fractional hypertree width. ACM Trans. Algorithms 6(2): 29:1-29:17 (2010)
- [Marx, 2011] Dániel Marx: Tractable Structures for Constraint Satisfaction with Truth Tables. Theory Comput. Syst. 48(3): 444-464 (2011)
- [Marx 2013] Dániel Marx: Tractable Hypergraph Properties for Constraint Satisfaction and Conjunctive Queries. J. ACM 60(6): 42:1-42:51 (2013)

References: Our Recent Works

- [Fischl, Gottlob, P., 2018] W.Fischl, G.Gottlob, R.Pichler: General and Fractional Hypertree Decompositions: Hard and Easy Cases. PODS 2018: 17-32 (extended version in J. ACM, 2021)
- [Fischl, Gottlob, Longo, P., 2019] W.Fischl, G.Gottlob, D.Longo, R.Pichler: HyperBench: A Benchmark and Tool for Hypergraphs and Empirical Findings. PODS 2019: 464-480 (full version in ACM J. Exp. Alg., 2021)
- [Gottlob, Lanzinger, P., Razgon, 2020] G.Gottlob, M.Lanzinger, R.Pichler, I.Razgon: Fractional Covers of Hypergraphs with Bounded Multi-Intersection. MFCS 2020: 41:1-41:14
- [Gottlob, Okulmus, P., 2020] G.Gottlob, C.Okulmus, R.Pichler: Fast and Parallel Decomposition of Constraint Satisfaction Problems. IJCAI 2020: 1155-1162.
- [Gottlob, Lanzinger, Okulmus, P., 2021] G.Gottlob, M.Lanzinger, C.Okulmus, R.Pichler: Fast Parallel Hypertree Decompositions in Logarithmic Recursion Depth. PODS 2022: 325-336