A Faster Algorithm for Propositional Model Counting Parameterized by Incidence Treewidth

Friedrich Slivovsky and Stefan Szeider

Propositional Model Counting (#SAT)

Propositional Model Counting (#SAT)

SAT

Does φ have a satisfying assignment?

Propositional Model Counting (#SAT)

SAT

Does φ have a satisfying assignment?

#SAT

Count the satisfying assignments of φ.

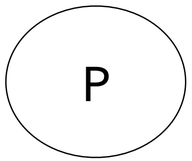
Propositional Model Counting (#SAT)

SAT

Does φ have a satisfying assignment?

#SAT

Count the satisfying assignments of φ.



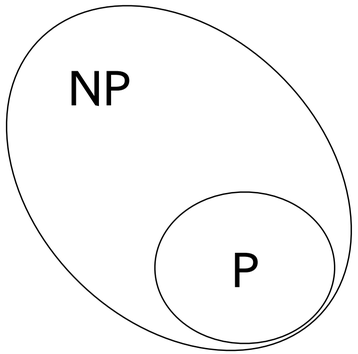
Propositional Model Counting (#SAT)

SAT

Does φ have a satisfying assignment?

#SAT

Count the satisfying assignments of φ.



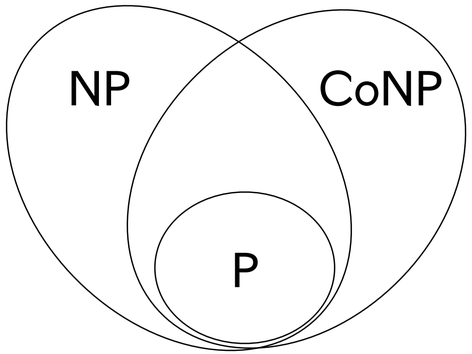
Propositional Model Counting (#SAT)

SAT

Does φ have a satisfying assignment?

#SAT

Count the satisfying assignments of φ.



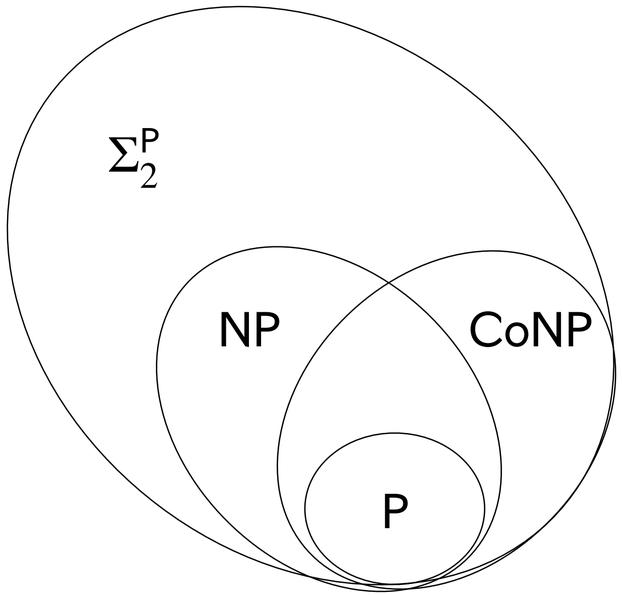
Propositional Model Counting (#SAT)

SAT

Does φ have a satisfying assignment?

#SAT

Count the satisfying assignments of φ.



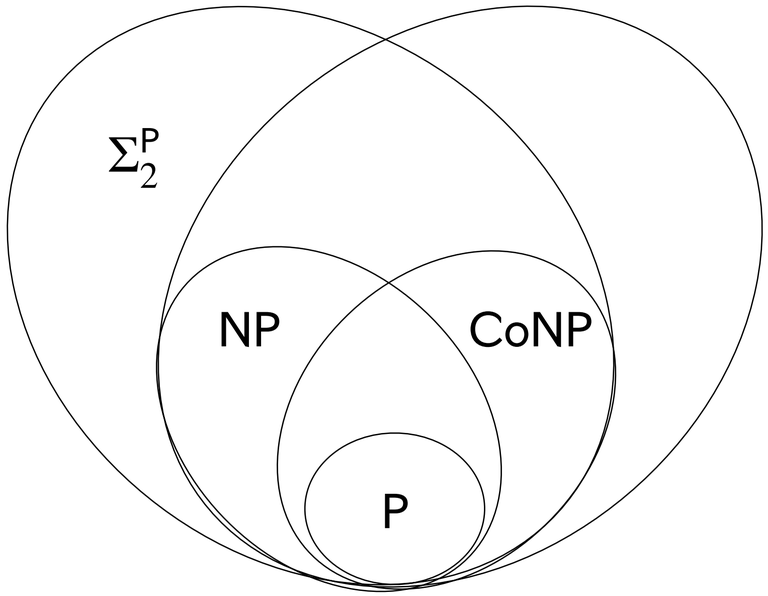
Propositional Model Counting (#SAT)

SAT

Does φ have a satisfying assignment?

#SAT

Count the satisfying assignments of φ.



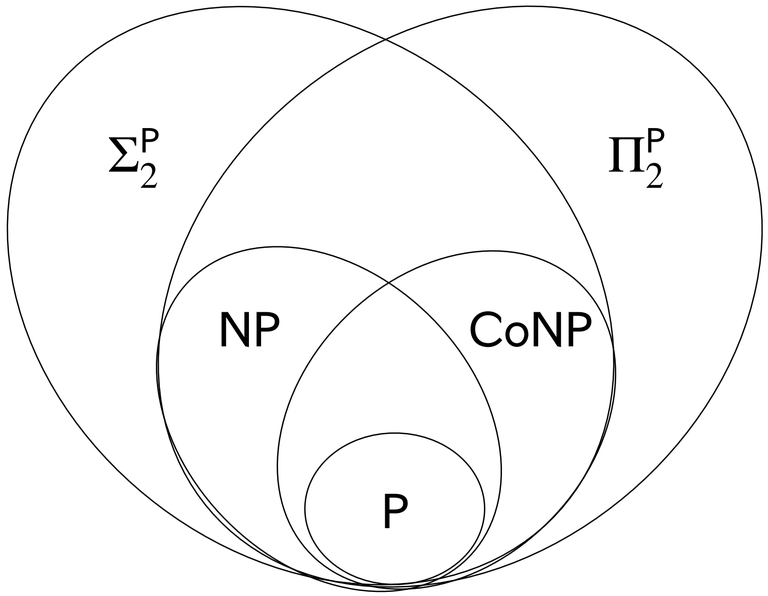
Propositional Model Counting (#SAT)

SAT

Does φ have a satisfying assignment?

#SAT

Count the satisfying assignments of φ.



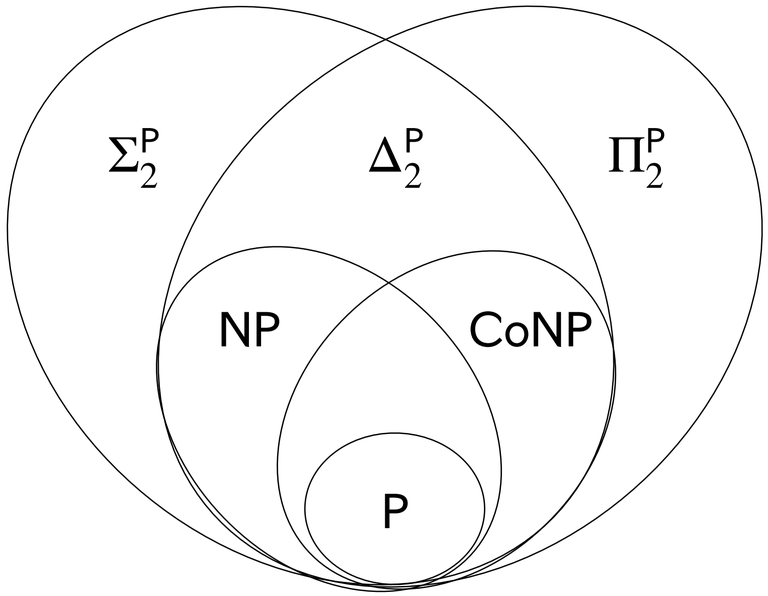
Propositional Model Counting (#SAT)

SAT

Does φ have a satisfying assignment?

#SAT

Count the satisfying assignments of φ.



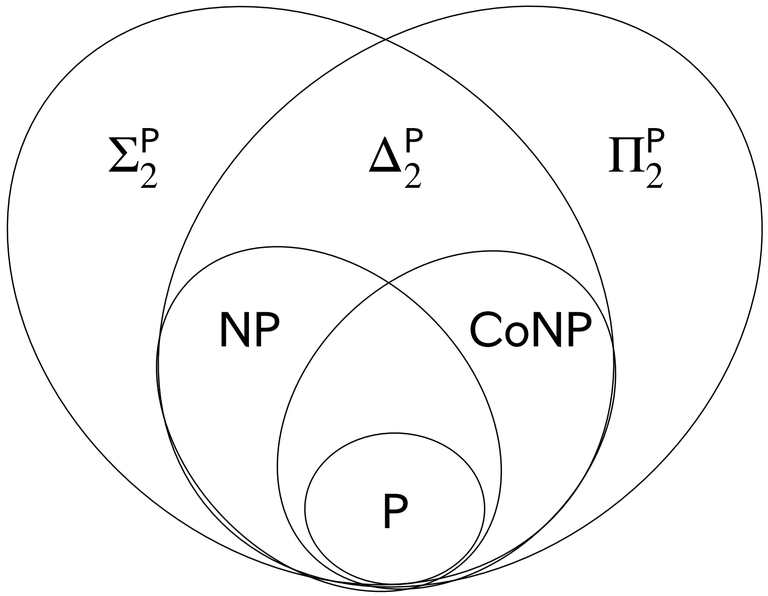
Propositional Model Counting (#SAT)

SAT

Does φ have a satisfying assignment?

#SAT

Count the satisfying assignments of φ.



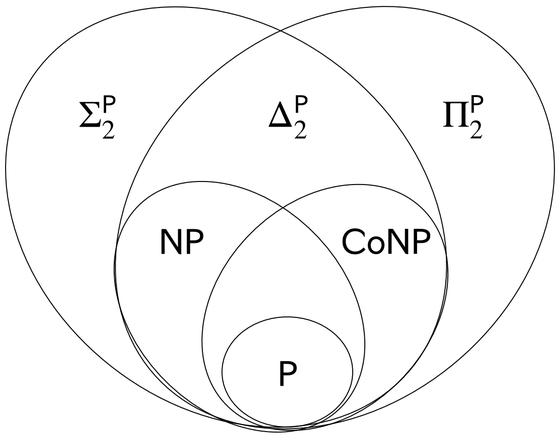
Propositional Model Counting (#SAT)

SAT

Does φ have a satisfying assignment?

#SAT

Count the satisfying assignments of φ.



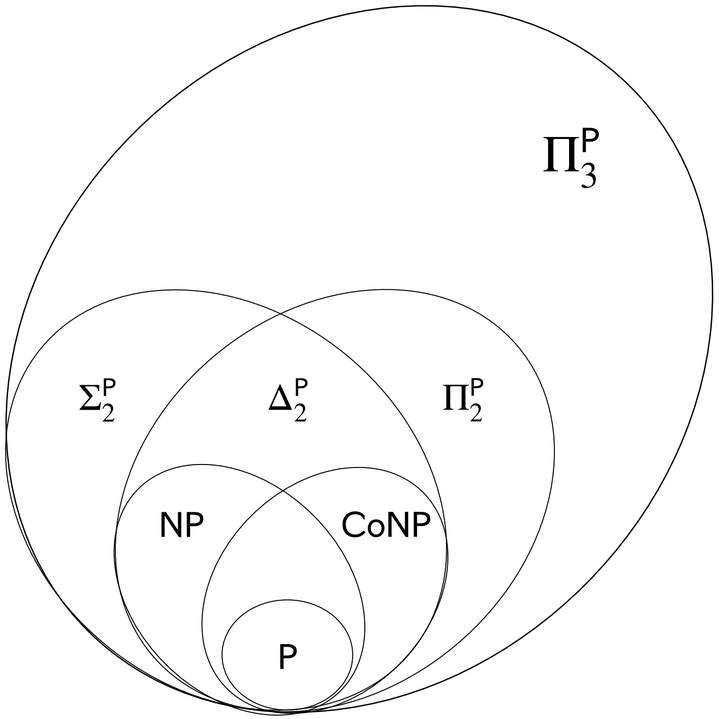
Propositional Model Counting (#SAT)

SAT

Does φ have a satisfying assignment?

#SAT

Count the satisfying assignments of φ.



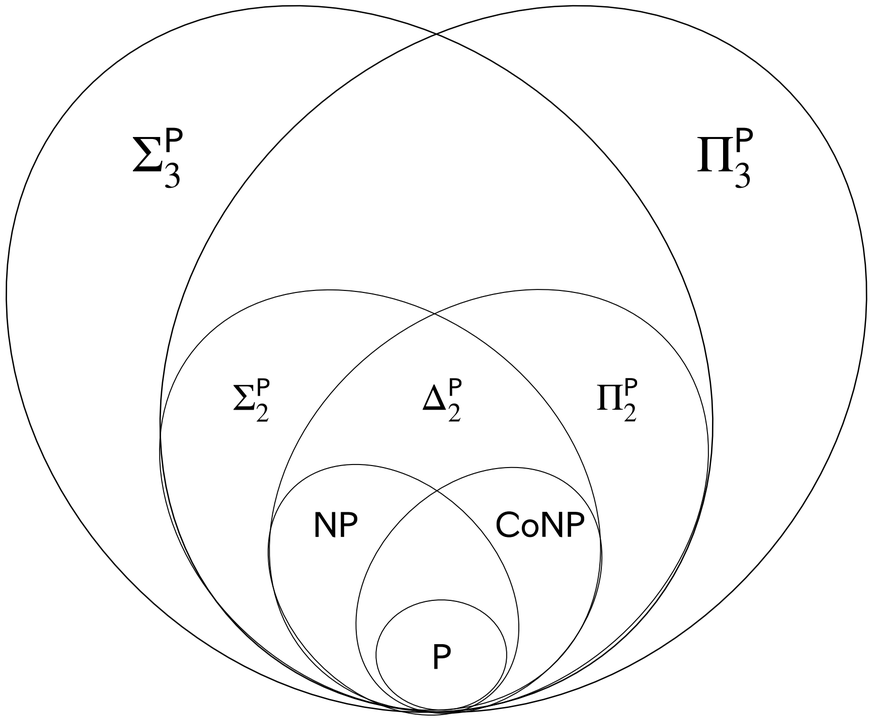
Propositional Model Counting (#SAT)

SAT

Does φ have a satisfying assignment?

#SAT

Count the satisfying assignments of φ.



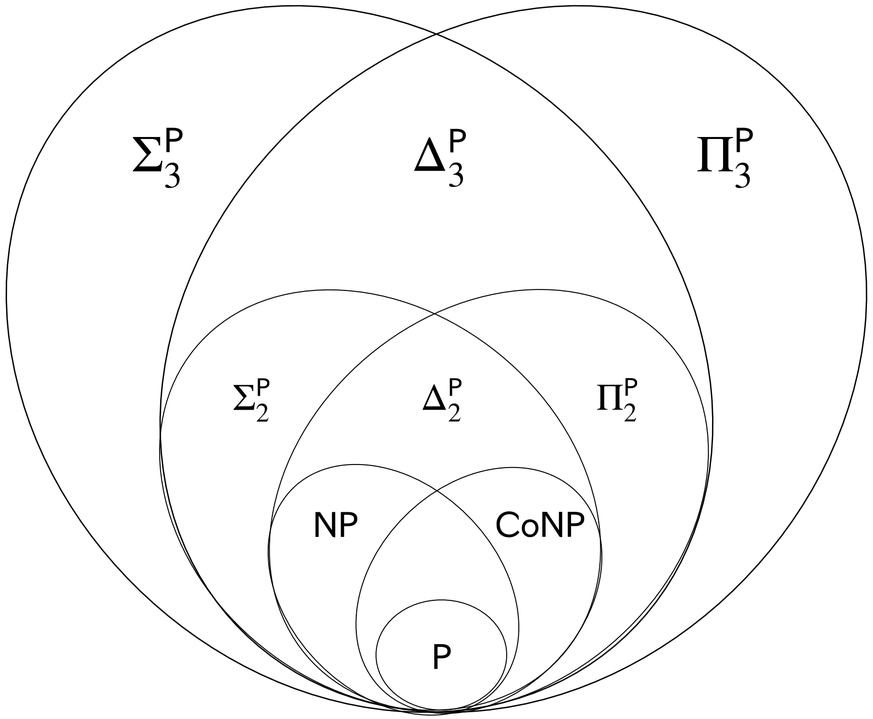
Propositional Model Counting (#SAT)

SAT

Does φ have a satisfying assignment?

#SAT

Count the satisfying assignments of φ.



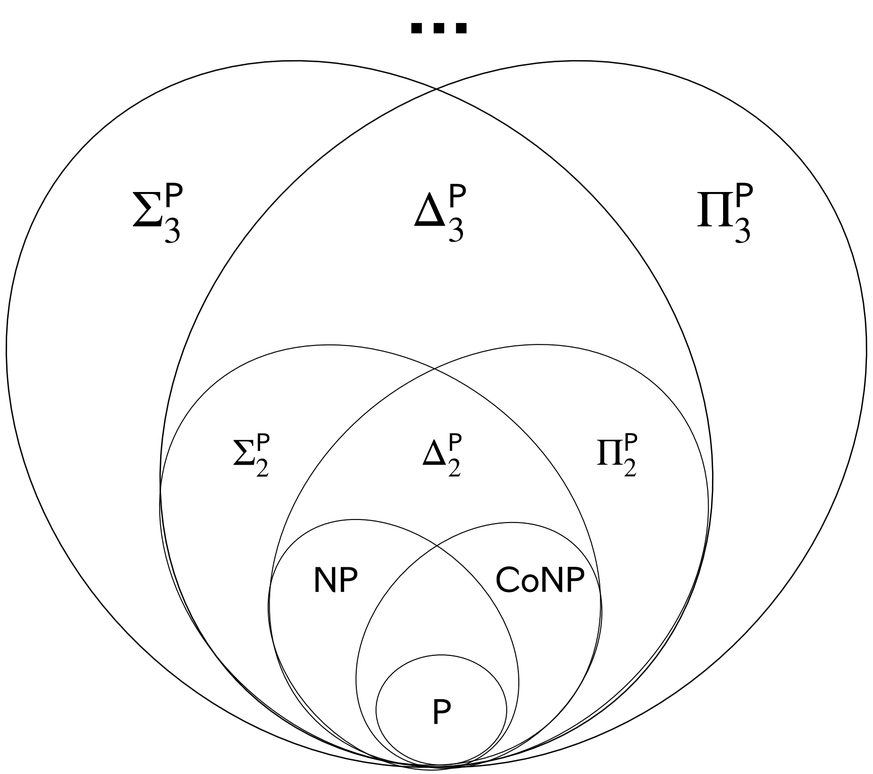
Propositional Model Counting (#SAT)

SAT

Does φ have a satisfying assignment?

#SAT

Count the satisfying assignments of φ.



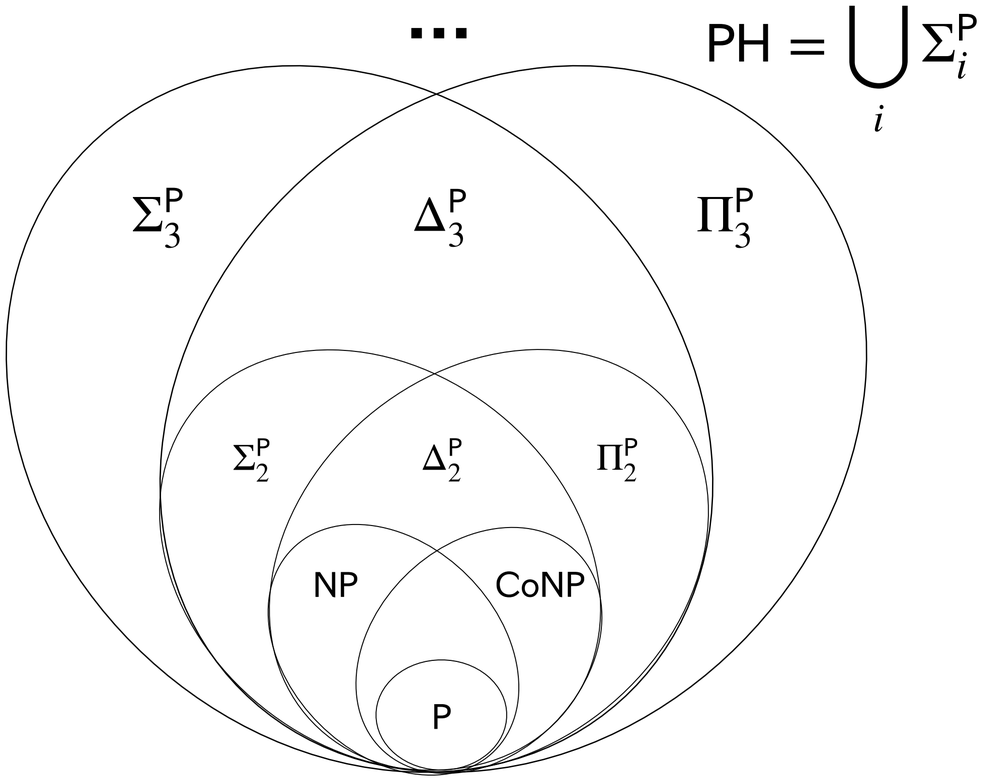
Propositional Model Counting (#SAT)

SAT

Does φ have a satisfying assignment?

#SAT

Count the satisfying assignments of φ.



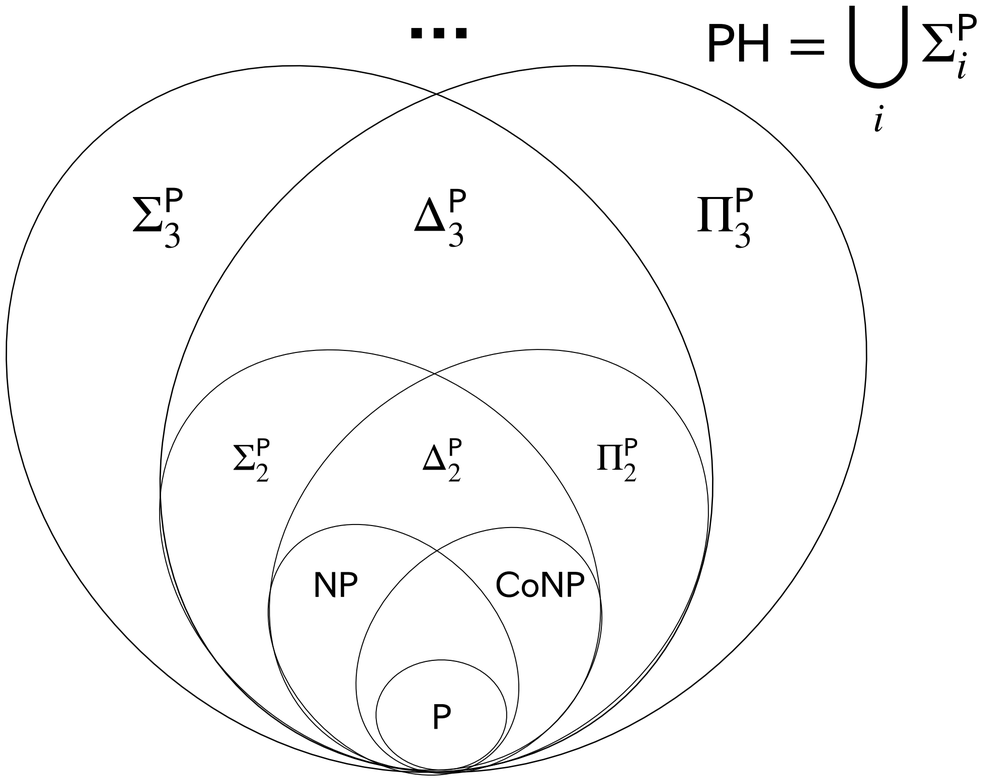
Propositional Model Counting (#SAT)

SAT

Does φ have a satisfying assignment?

#SAT

Count the satisfying assignments of φ.



Toda's Theorem

Every problem in the polynomial hierarchy can be solved in **polynomial time** using a #SAT oracle.

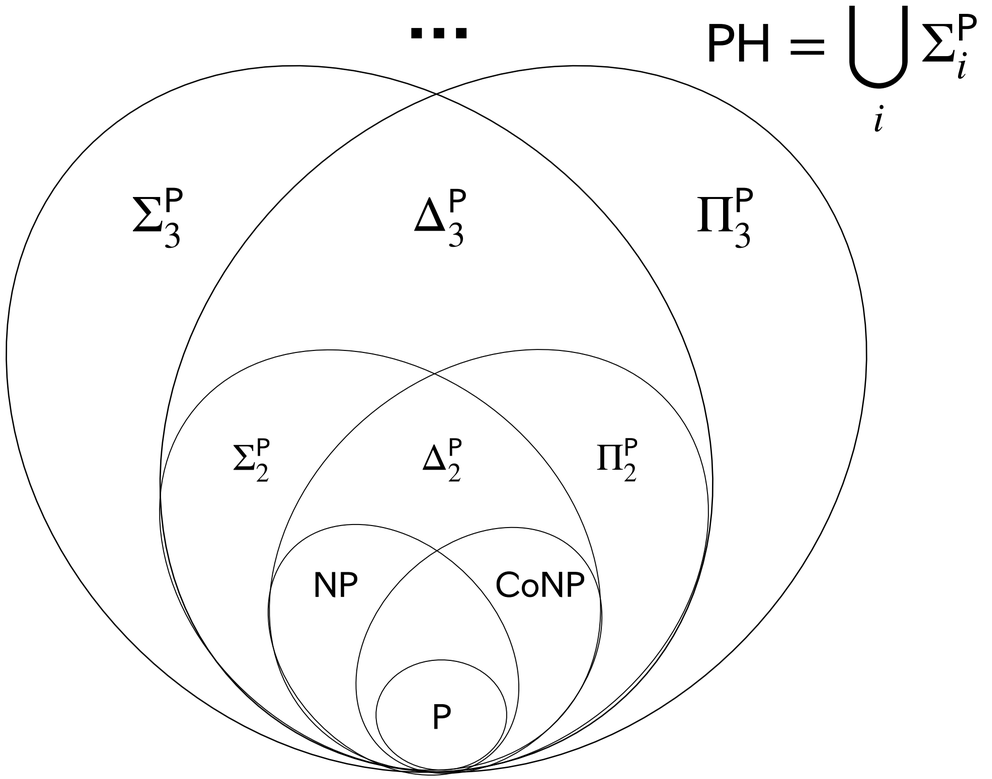
Propositional Model Counting (#SAT)

SAT

Does φ have a satisfying assignment?

#SAT

Count the satisfying assignments of φ.



Toda's Theorem

Every problem in the polynomial hierarchy can be solved in **polynomial time** using a #SAT oracle.

#SAT **#P-complete** even for

- monotone 2CNF and
- Horn formulas.

Applications and Solvers

Applications and Solvers

Probabilistic Inference

Applications and Solvers

Probabilistic Inference

$$Pr(A \mid B) = \frac{Pr(A \wedge B)}{Pr(B)}$$

Applications and Solvers

Probabilistic Inference

$$Pr(A \mid B) = \frac{Pr(A \wedge B)}{Pr(B)}$$

$$\frac{\#(A \wedge B)}{\#B}$$

Applications and Solvers

Probabilistic Inference

$$Pr(A \mid B) = \frac{Pr(A \wedge B)}{Pr(B)}$$

$$\frac{\#(A \wedge B)}{\#B}$$

Circuit Design

Applications and Solvers

Probabilistic Inference

$$Pr(A \mid B) = \frac{Pr(A \wedge B)}{Pr(B)}$$

$$\frac{\#(A \wedge B)}{\#B}$$

Circuit Design

Product Configuration

Applications and Solvers

Probabilistic Inference

$$Pr(A \mid B) = \frac{Pr(A \wedge B)}{Pr(B)}$$

$$\frac{\#(A \wedge B)}{\#B}$$

Model Counters

Circuit Design

Product Configuration

Applications and Solvers

Probabilistic Inference

$$Pr(A \mid B) = \frac{Pr(A \wedge B)}{Pr(B)}$$

$$\frac{\#(A \wedge B)}{\#B}$$

Circuit Design

Product Configuration

Model Counters

Exact

Applications and Solvers

Probabilistic Inference

$$Pr(A \mid B) = \frac{Pr(A \wedge B)}{Pr(B)}$$

$$\frac{\#(A \wedge B)}{\#B}$$

Circuit Design

Product Configuration

Model Counters

Exact

Exhaustive DPLL

Applications and Solvers

Probabilistic Inference

$$Pr(A \mid B) = \frac{Pr(A \wedge B)}{Pr(B)}$$

$$\frac{\#(A \wedge B)}{\#B}$$

Circuit Design

Product Configuration

Model Counters

Exact

Exhaustive DPLL

Knowledge Compilation

Applications and Solvers

Probabilistic Inference

$$Pr(A \mid B) = \frac{Pr(A \wedge B)}{Pr(B)}$$

$$\frac{\#(A \wedge B)}{\#B}$$

Circuit Design

Product Configuration

Model Counters

Exact

Exhaustive DPLL

Knowledge Compilation

Dynamic Programming

Applications and Solvers

Probabilistic Inference

$$Pr(A \mid B) = \frac{Pr(A \wedge B)}{Pr(B)}$$

$$\frac{\#(A \wedge B)}{\#B}$$

Circuit Design

Product Configuration

Model Counters

Exact

Exhaustive DPLL

Knowledge Compilation

Dynamic Programming

Approximate

Applications and Solvers

Probabilistic Inference

$$Pr(A \mid B) = \frac{Pr(A \wedge B)}{Pr(B)}$$

$$\frac{\#(A \wedge B)}{\#B}$$

Circuit Design

Product Configuration

Model Counters

Exact

Exhaustive DPLL

Knowledge Compilation

Dynamic Programming

Approximate

Hashing-Based

Applications and Solvers

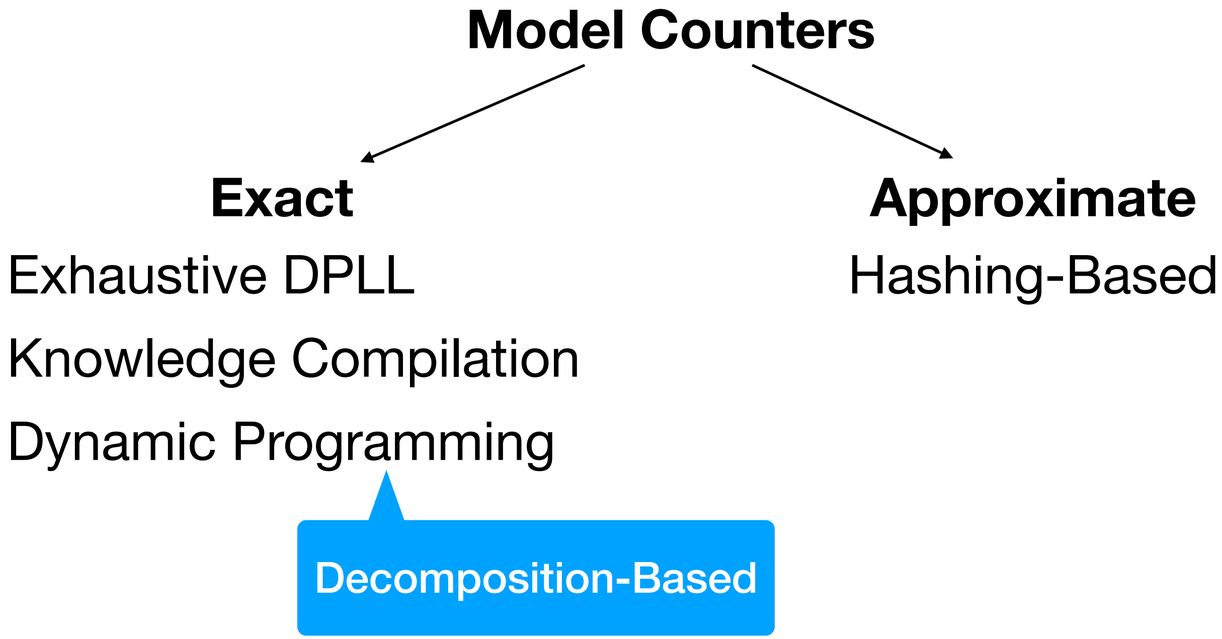
Probabilistic Inference

$$Pr(A \mid B) = \frac{Pr(A \wedge B)}{Pr(B)}$$

$$\frac{\#(A \wedge B)}{\#B}$$

Circuit Design

Product Configuration



Treewidth of Formulas

Treewidth of Formulas

treewidth measures the “tree-likeness” of a graph

Treewidth of Formulas

treewidth measures the “tree-likeness” of a graph

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4)$$

Treewidth of Formulas

treewidth measures the “tree-likeness” of a graph

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4)$$

primal graph

Treewidth of Formulas

treewidth measures the “tree-likeness” of a graph

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4)$$

primal graph

Treewidth of Formulas

treewidth measures the “tree-likeness” of a graph

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4)$$

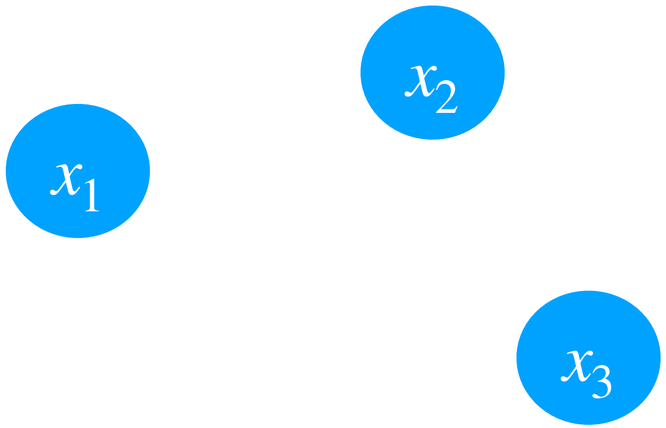
primal graph

Treewidth of Formulas

treewidth measures the “tree-likeness” of a graph

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4)$$

primal graph

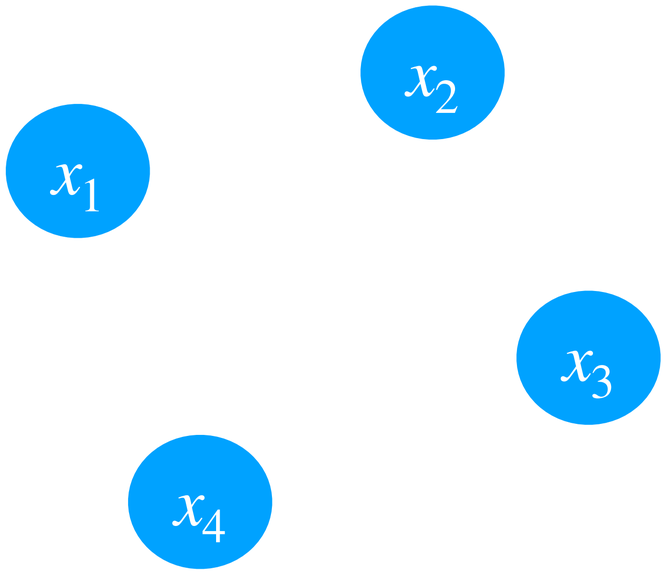


Treewidth of Formulas

treewidth measures the “tree-likeness” of a graph

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4)$$

primal graph

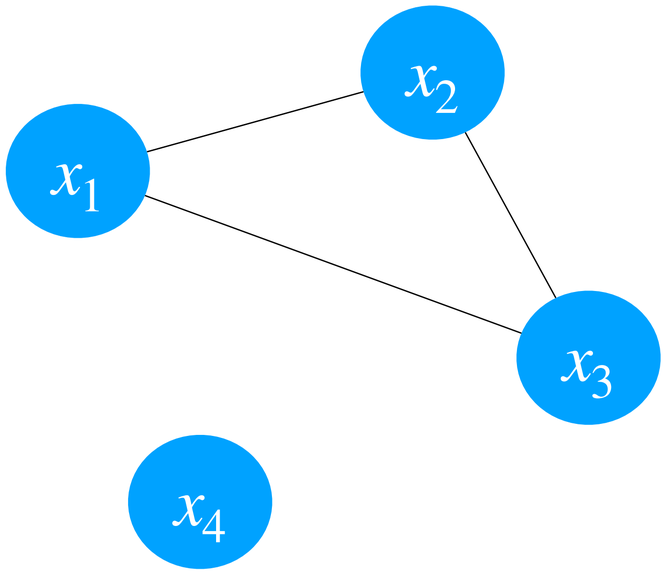


Treewidth of Formulas

treewidth measures the “tree-likeness” of a graph

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4)$$

primal graph

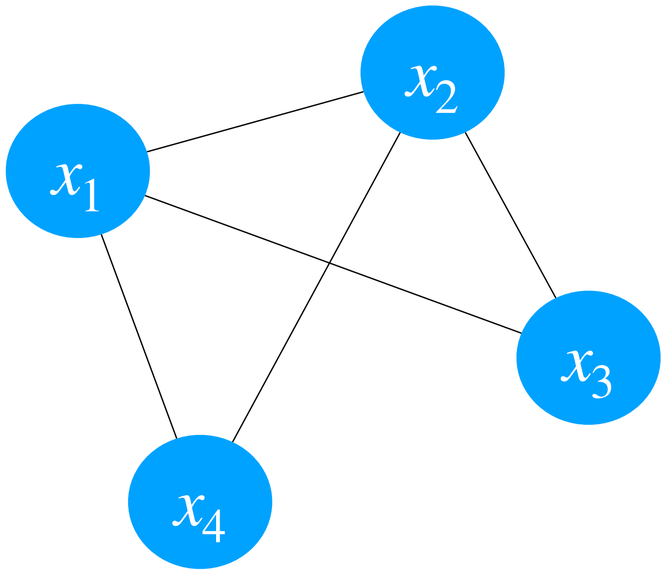


Treewidth of Formulas

treewidth measures the “tree-likeness” of a graph

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4)$$

primal graph

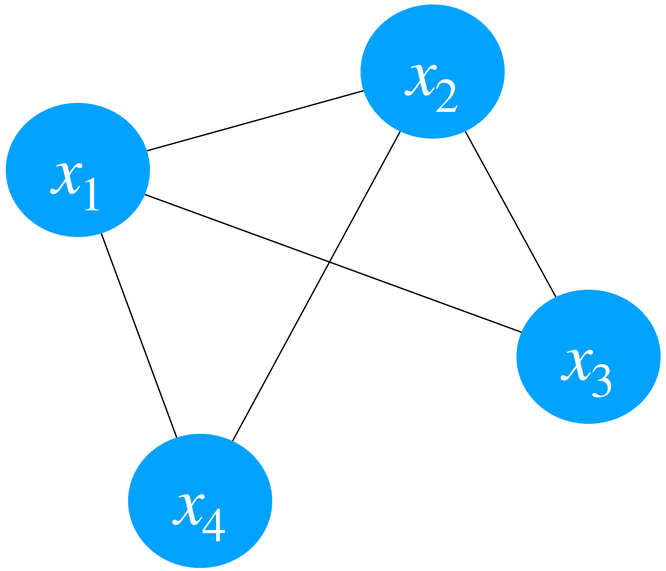


Treewidth of Formulas

treewidth measures the “tree-likeness” of a graph

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4)$$

primal graph



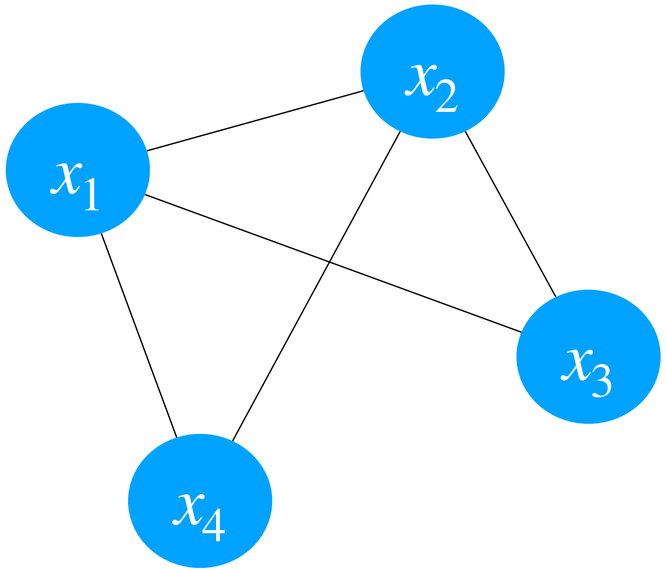
incidence graph

Treewidth of Formulas

treewidth measures the “tree-likeness” of a graph

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4)$$

primal graph



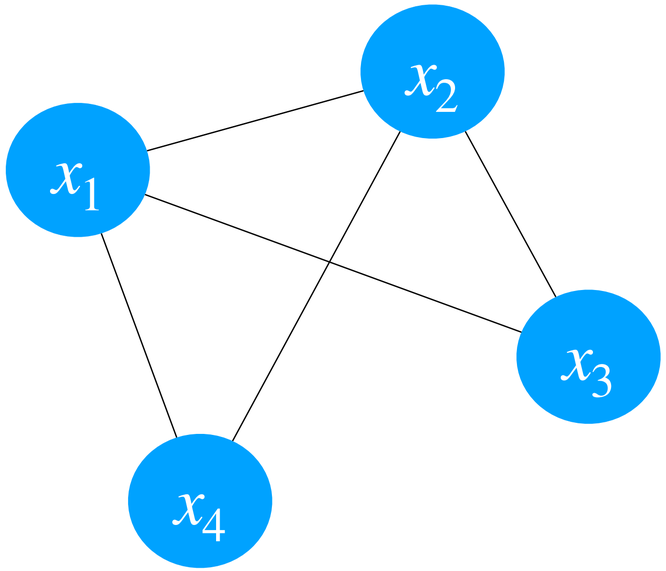
incidence graph

Treewidth of Formulas

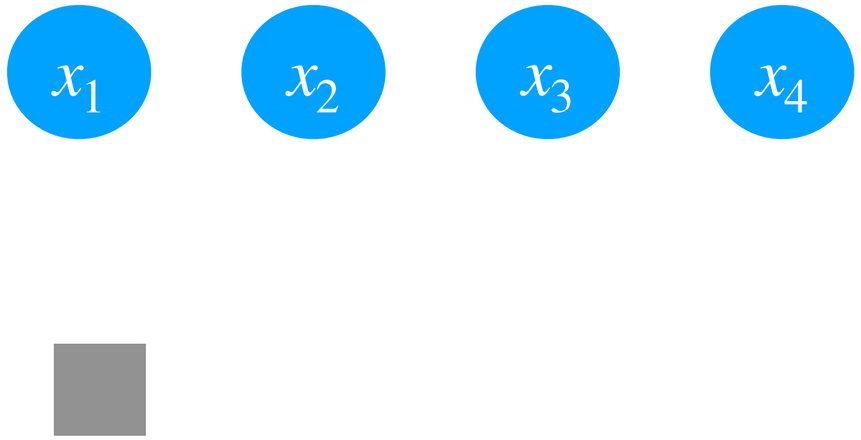
treewidth measures the “tree-likeness” of a graph

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4)$$

primal graph



incidence graph

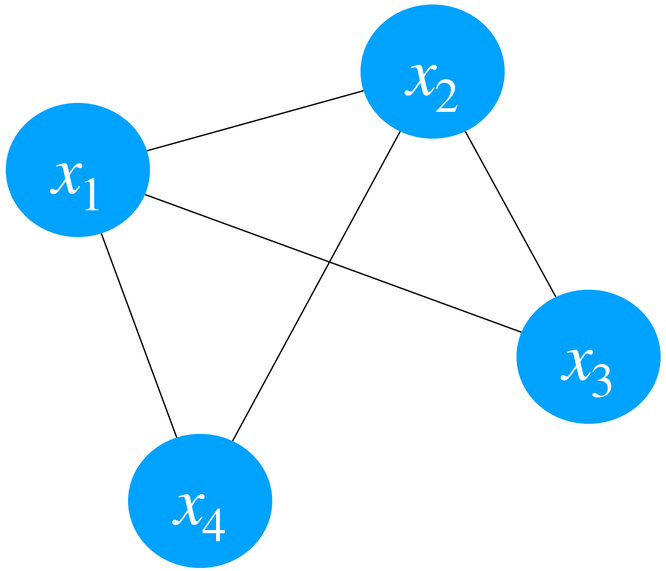


Treewidth of Formulas

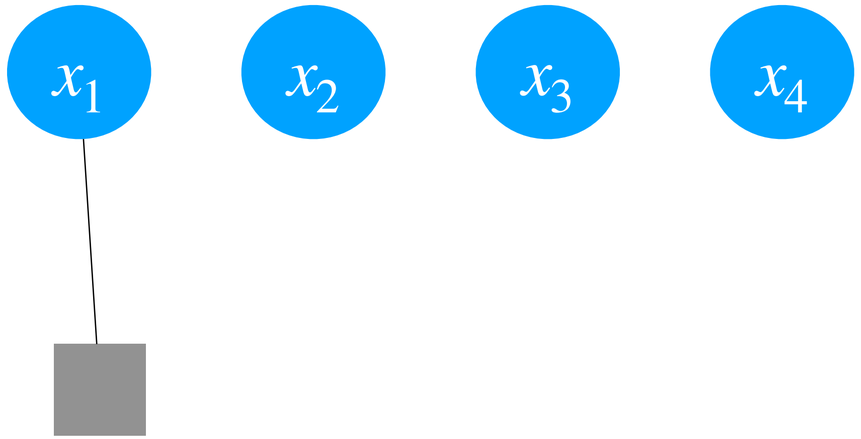
treewidth measures the “tree-likeness” of a graph

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4)$$

primal graph



incidence graph

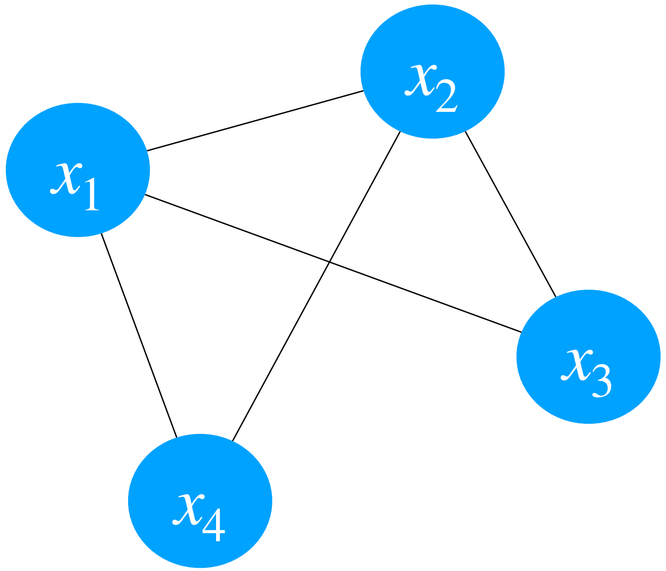


Treewidth of Formulas

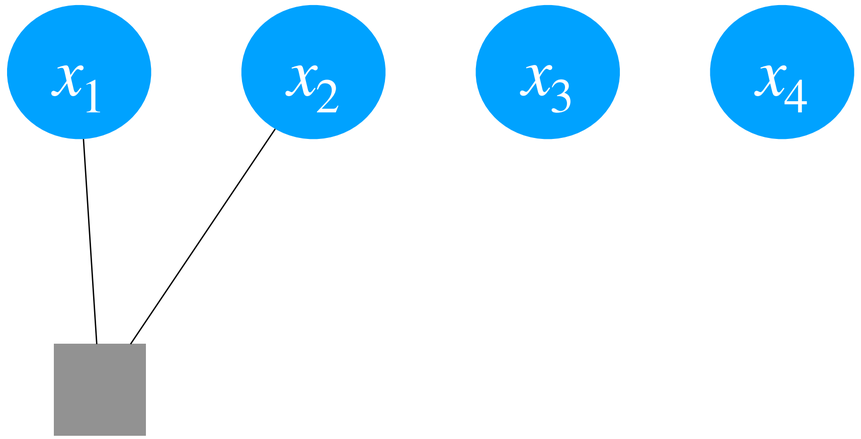
treewidth measures the “tree-likeness” of a graph

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4)$$

primal graph



incidence graph

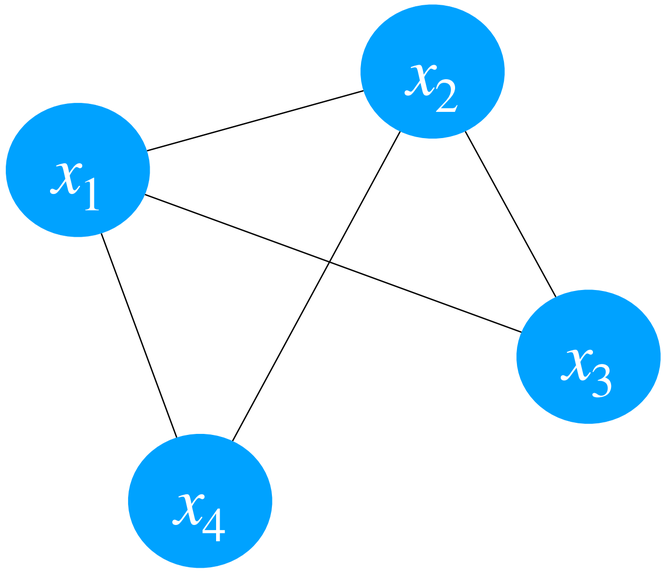


Treewidth of Formulas

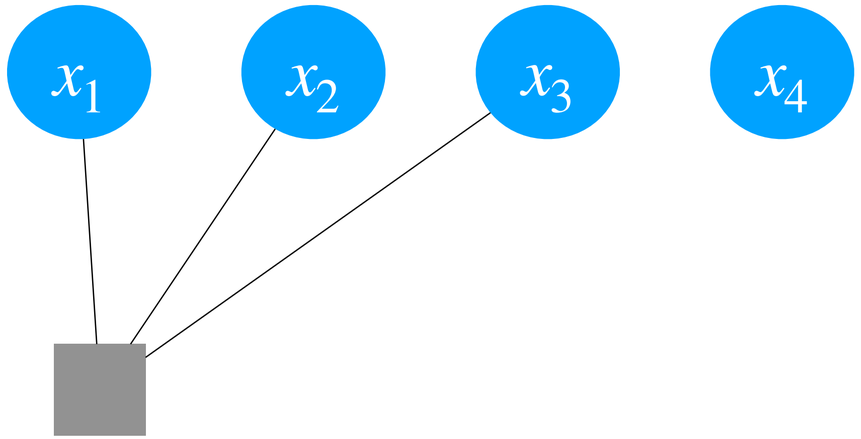
treewidth measures the “tree-likeness” of a graph

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4)$$

primal graph



incidence graph

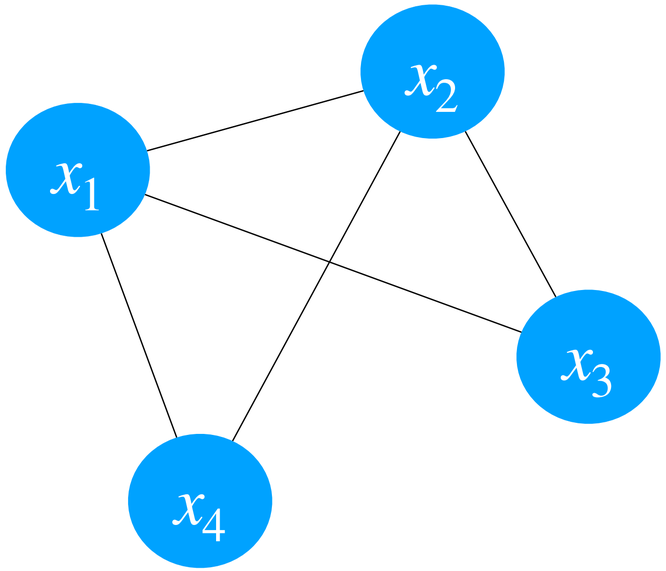


Treewidth of Formulas

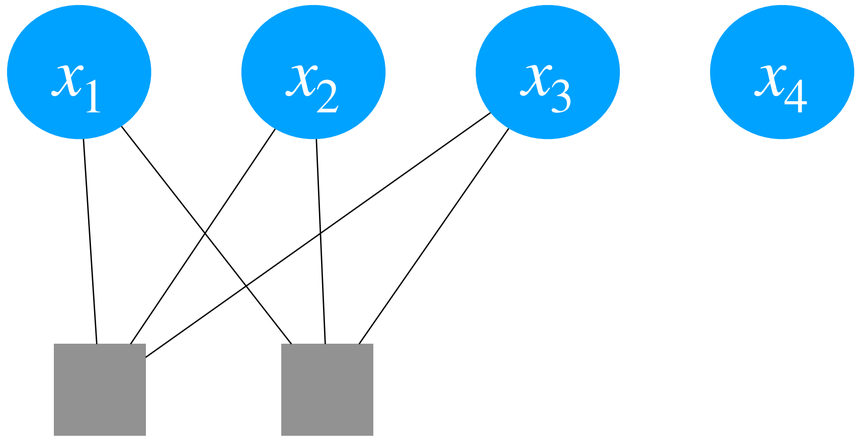
treewidth measures the “tree-likeness” of a graph

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4)$$

primal graph



incidence graph

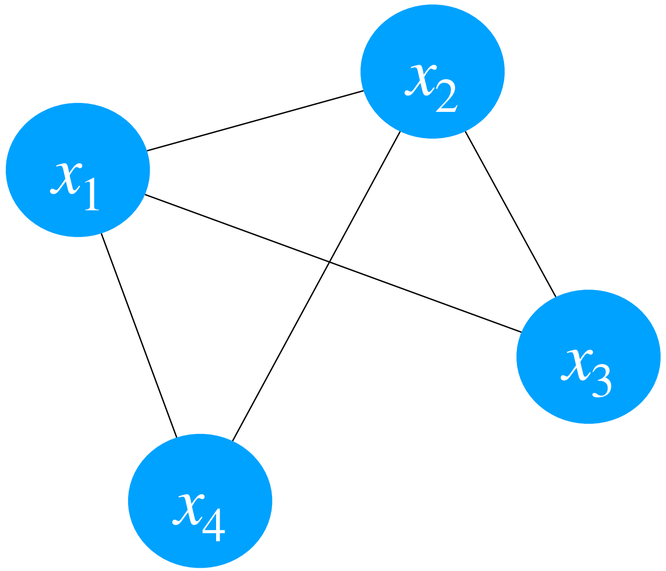


Treewidth of Formulas

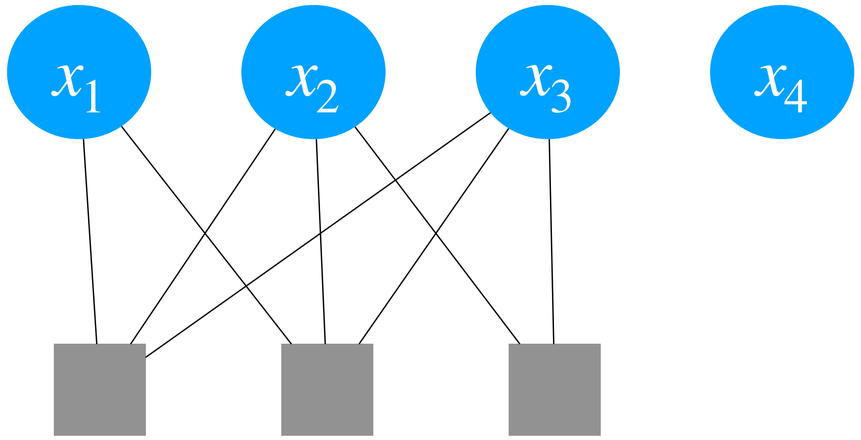
treewidth measures the “tree-likeness” of a graph

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4)$$

primal graph



incidence graph

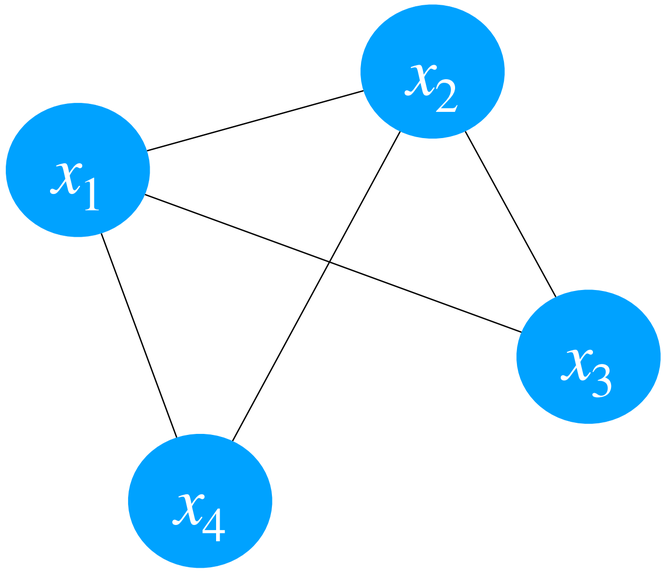


Treewidth of Formulas

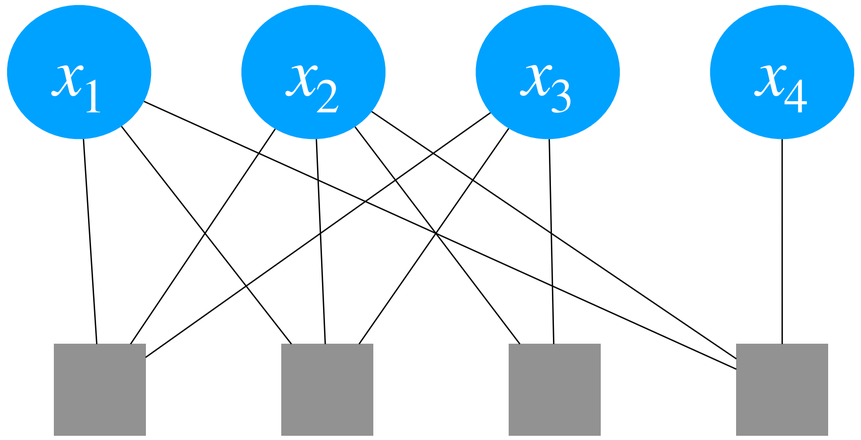
treewidth measures the “tree-likeness” of a graph

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4)$$

primal graph



incidence graph



#SAT and Treewidth

#SAT and Treewidth

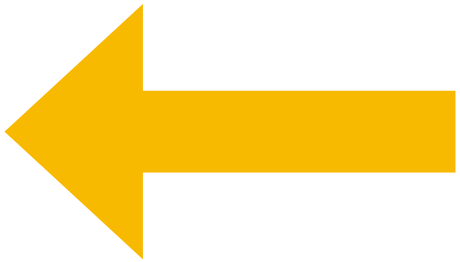
primal treewidth k

#SAT and Treewidth

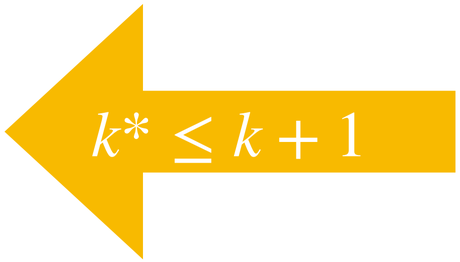
incidence treewidth k^*

primal treewidth k

#SAT and Treewidth

incidence treewidth k^*  **primal treewidth k**

#SAT and Treewidth

incidence treewidth k^*  **primal treewidth** k

#SAT and Treewidth

incidence treewidth k^*

$k^* \leq k + 1$

primal treewidth k

$f(k) \cdot p(F)$

#SAT and Treewidth

incidence treewidth k^* ← $k^* \leq k + 1$ → **primal treewidth k**

$f(k^*) \cdot p(F)$ — Courcelle, Makowsky, Rotics 2001

$f(k) \cdot p(F)$

#SAT and Treewidth

incidence treewidth k^*

$k^* \leq k + 1$

primal treewidth k

$f(k^*) \cdot p(F)$

Courcelle, Makowsky, Rotics 2001

$f(k) \cdot p(F)$

$2^{O(k)} \cdot p(F)$

#SAT and Treewidth

incidence treewidth k^*

$k^* \leq k + 1$

primal treewidth k

$f(k^*) \cdot p(F)$ — Courcelle, Makowsky, Rotics 2001 — $f(k) \cdot p(F)$

Bacchus, Dalmao, Pitassi 2003 — $2^{O(k)} \cdot p(F)$

#SAT and Treewidth

incidence treewidth k^*

$k^* \leq k + 1$

primal treewidth k

$f(k^*) \cdot p(F)$ — Courcelle, Makowsky, Rotics 2001

$f(k) \cdot p(F)$

Bacchus, Dalmao, Pitassi 2003 — $2^{O(k)} \cdot p(F)$

$4^{k^*} \cdot p(F)$

#SAT and Treewidth

incidence treewidth k^*

$k^* \leq k + 1$

primal treewidth k

$f(k^*) \cdot p(F)$ — Courcelle, Makowsky, Rotics 2001

$f(k) \cdot p(F)$

$2^{O(k)} \cdot p(F)$ — Bacchus, Dalmao, Pitassi 2003

$4^{k^*} \cdot p(F)$ — Fischer, Makowsky, Ravve 2006; Samer and Szeider 2007

#SAT and Treewidth

incidence treewidth k^*

$k^* \leq k + 1$

primal treewidth k

$f(k^*) \cdot p(F)$ — Courcelle, Makowsky, Rotics 2001 — $f(k) \cdot p(F)$

Bacchus, Dalmao, Pitassi 2003 — $2^{O(k)} \cdot p(F)$

$4^{k^*} \cdot p(F)$ — Fischer, Makowsky, Ravve 2006 — $4^k \cdot p(F)$

Fischer, Makowsky, Ravve 2006
Samer and Szeider 2007

#SAT and Treewidth

incidence treewidth k^*

$k^* \leq k + 1$

primal treewidth k

$f(k^*) \cdot p(F)$ — Courcelle, Makowsky, Rotics 2001 — $f(k) \cdot p(F)$

Bacchus, Dalmao, Pitassi 2003 — $2^{O(k)} \cdot p(F)$

$4^{k^*} \cdot p(F)$ — Fischer, Makowsky, Ravve 2006 — $4^k \cdot p(F)$

Fischer, Makowsky, Ravve 2006
Samer and Szeider 2007

$2^k \cdot p(F)$

#SAT and Treewidth

incidence treewidth k^*

$k^* \leq k + 1$

primal treewidth k

$f(k^*) \cdot p(F)$ — Courcelle, Makowsky, Rotics 2001

$f(k) \cdot p(F)$

$2^{O(k)} \cdot p(F)$ — Bacchus, Dalmao, Pitassi 2003

$4^{k^*} \cdot p(F)$ — Fischer, Makowsky, Ravve 2006

$4^k \cdot p(F)$

Fischer, Makowsky, Ravve 2006
Samer and Szeider 2007

Samer and Szeider 2007

$2^k \cdot p(F)$

#SAT and Treewidth

incidence treewidth k^*

$k^* \leq k+1$

primal treewidth k

$f(k^*) \cdot p(F)$ — Courcelle, Makowsky, Rotics 2001

$f(k) \cdot p(F)$

$2^{O(k)} \cdot p(F)$ — Bacchus, Dalmao, Pitassi 2003

$4^{k^*} \cdot p(F)$ — Fischer, Makowsky, Ravve 2006

$4^k \cdot p(F)$ — Fischer, Makowsky, Ravve 2006

Samer and Szeider 2007

$2^{k^*} \cdot p(F)$

Samer and Szeider 2007

$2^k \cdot p(F)$

#SAT and Treewidth

incidence treewidth k^*

$k^* \leq k + 1$

primal treewidth k

$f(k^*) \cdot p(F)$ — Courcelle, Makowsky, Rotics 2001

$f(k) \cdot p(F)$

$2^{O(k)} \cdot p(F)$ — Bacchus, Dalmao, Pitassi 2003

$4^{k^*} \cdot p(F)$ — Fischer, Makowsky, Ravve 2006

$4^k \cdot p(F)$ — Fischer, Makowsky, Ravve 2006

Samer and Szeider 2007

Our result

$2^{k^*} \cdot p(F)$

Samer and Szeider 2007

$2^k \cdot p(F)$

#SAT and Treewidth

incidence treewidth k^*

$k^* \leq k+1$

primal treewidth k

$f(k^*) \cdot p(F)$ — Courcelle, Makowsky, Rotics 2001

$f(k) \cdot p(F)$

Bacchus, Dalmao, Pitassi 2003 — $2^{O(k)} \cdot p(F)$

$4^{k^*} \cdot p(F)$ — Fischer, Makowsky, Ravve 2006 — $4^k \cdot p(F)$

Fischer, Makowsky, Ravve 2006
Samer and Szeider 2007

Our result

$2^{k^*} \cdot p(F)$

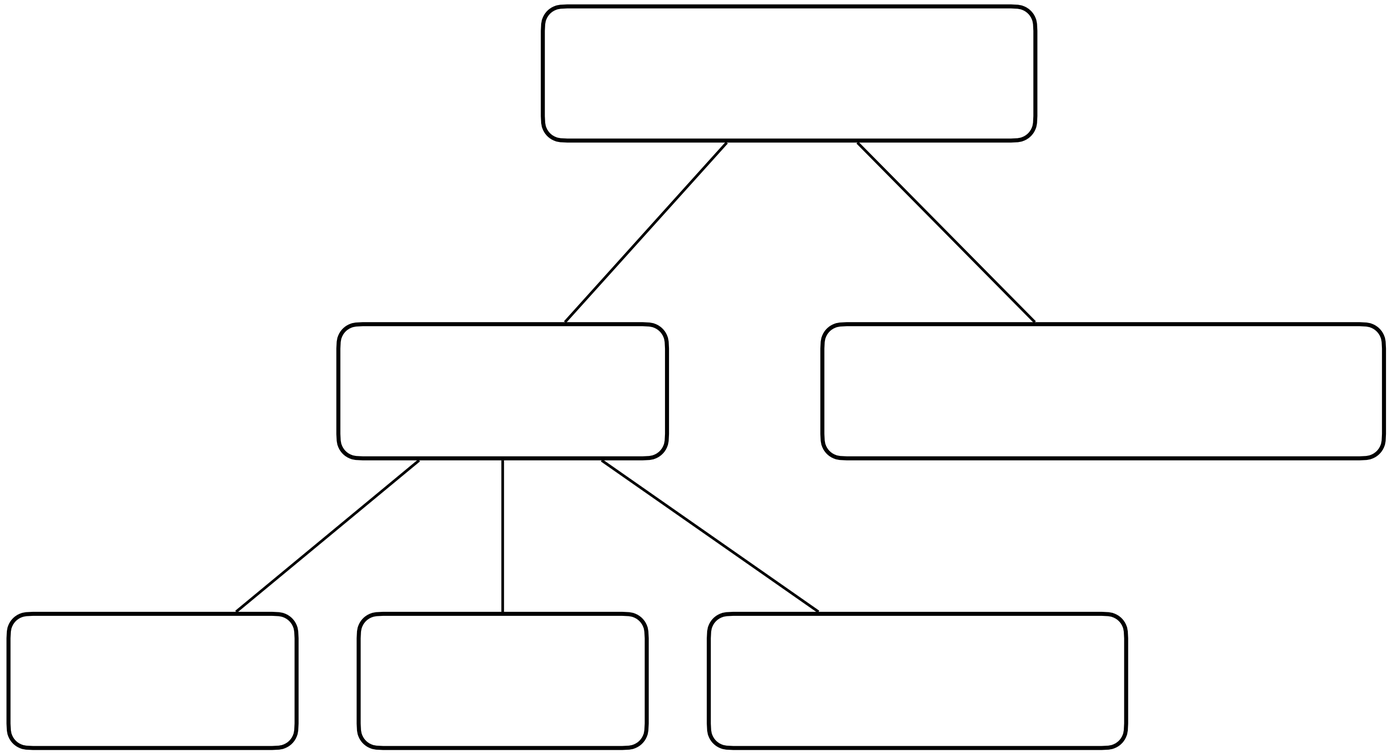
Optimal under ETH

Samer and Szeider 2007

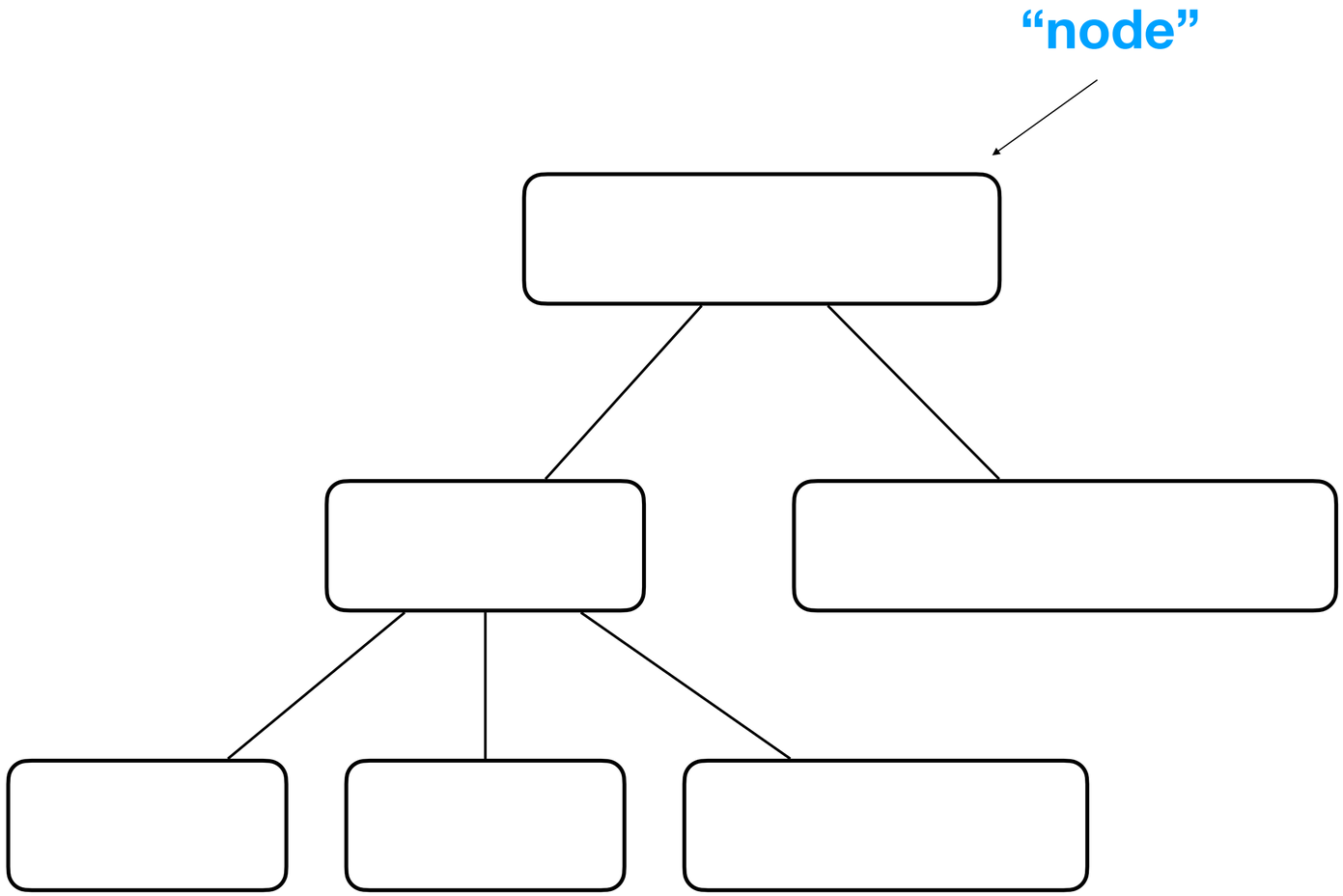
$2^k \cdot p(F)$

Tree Decompositions

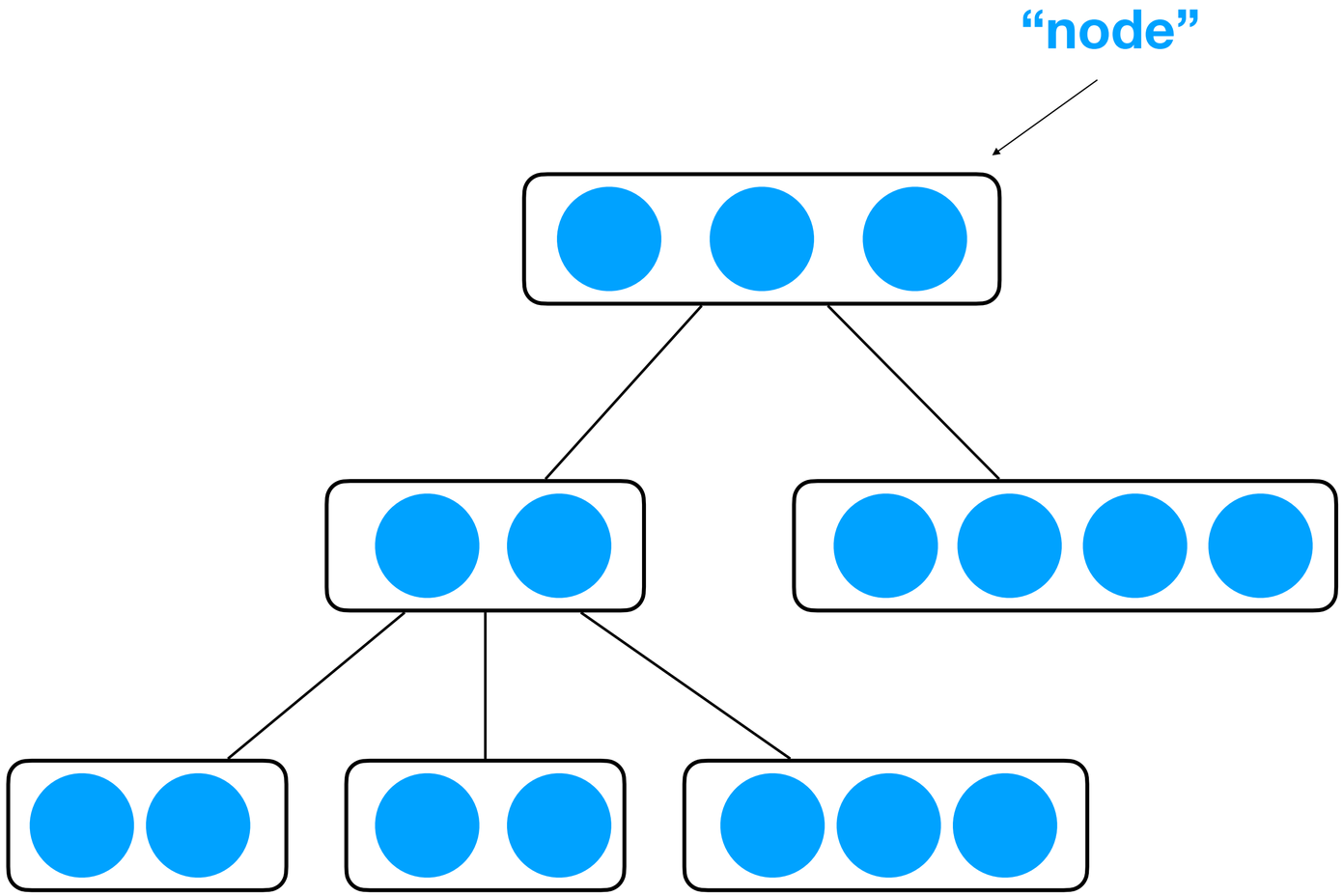
Tree Decompositions



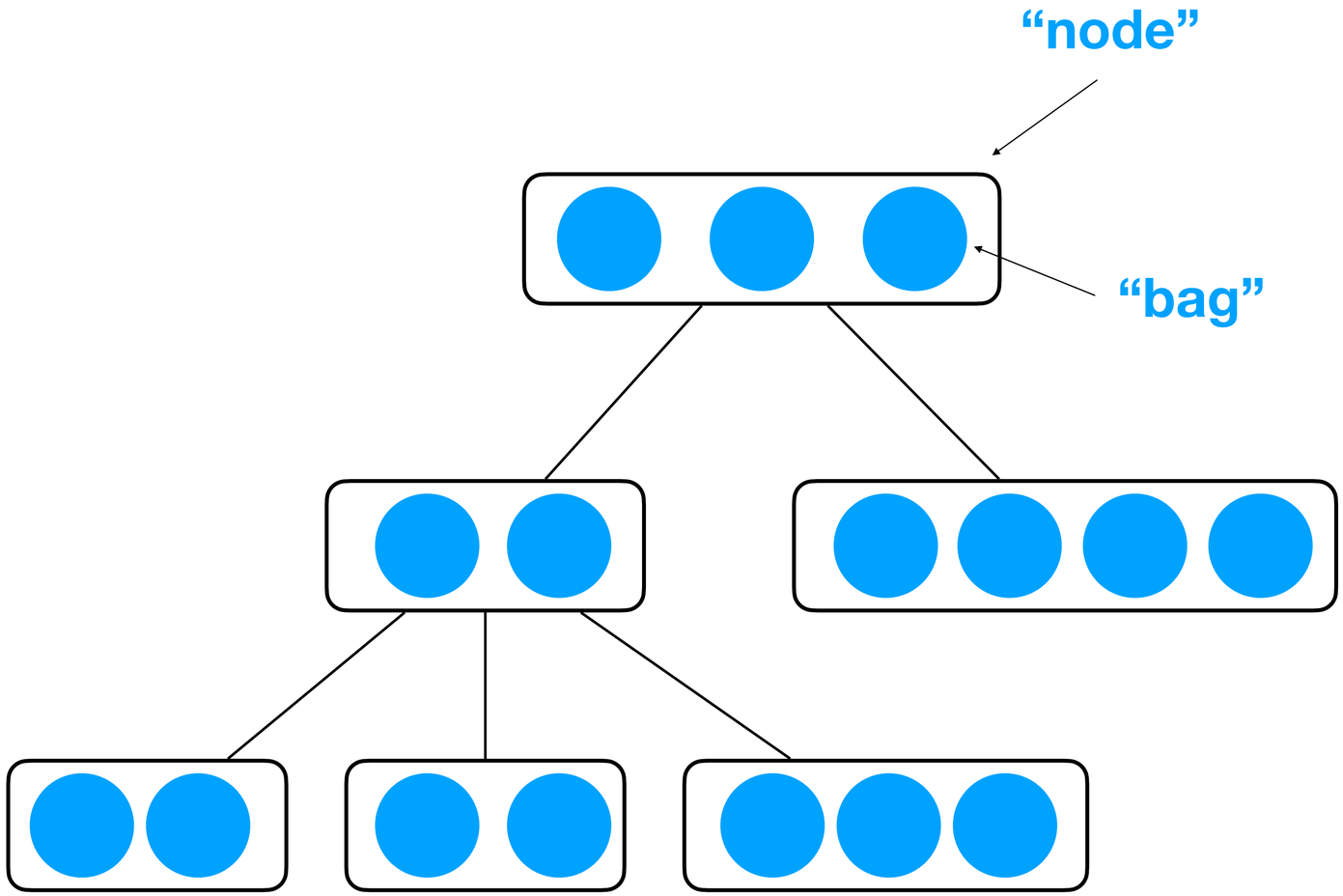
Tree Decompositions



Tree Decompositions

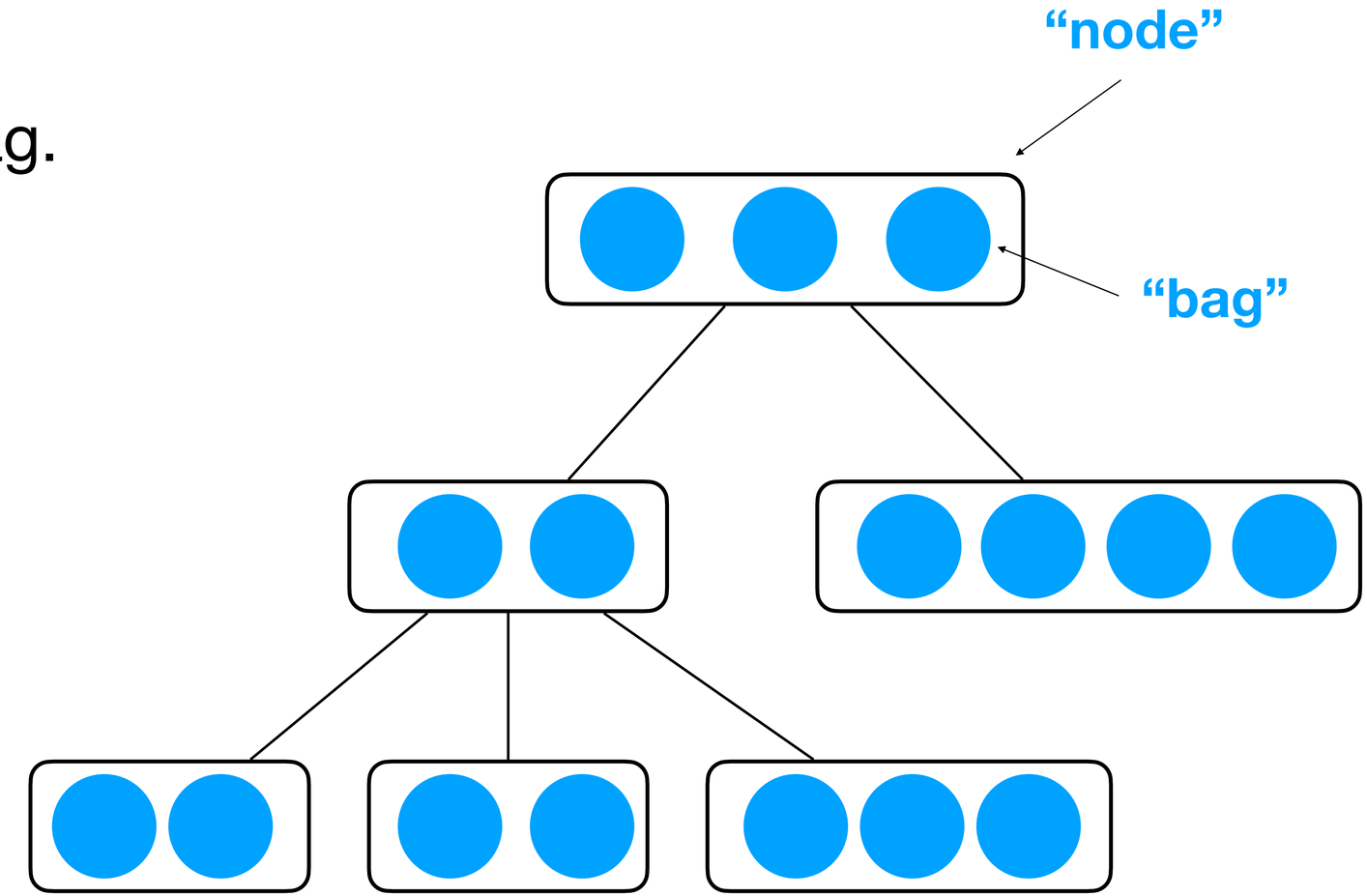


Tree Decompositions



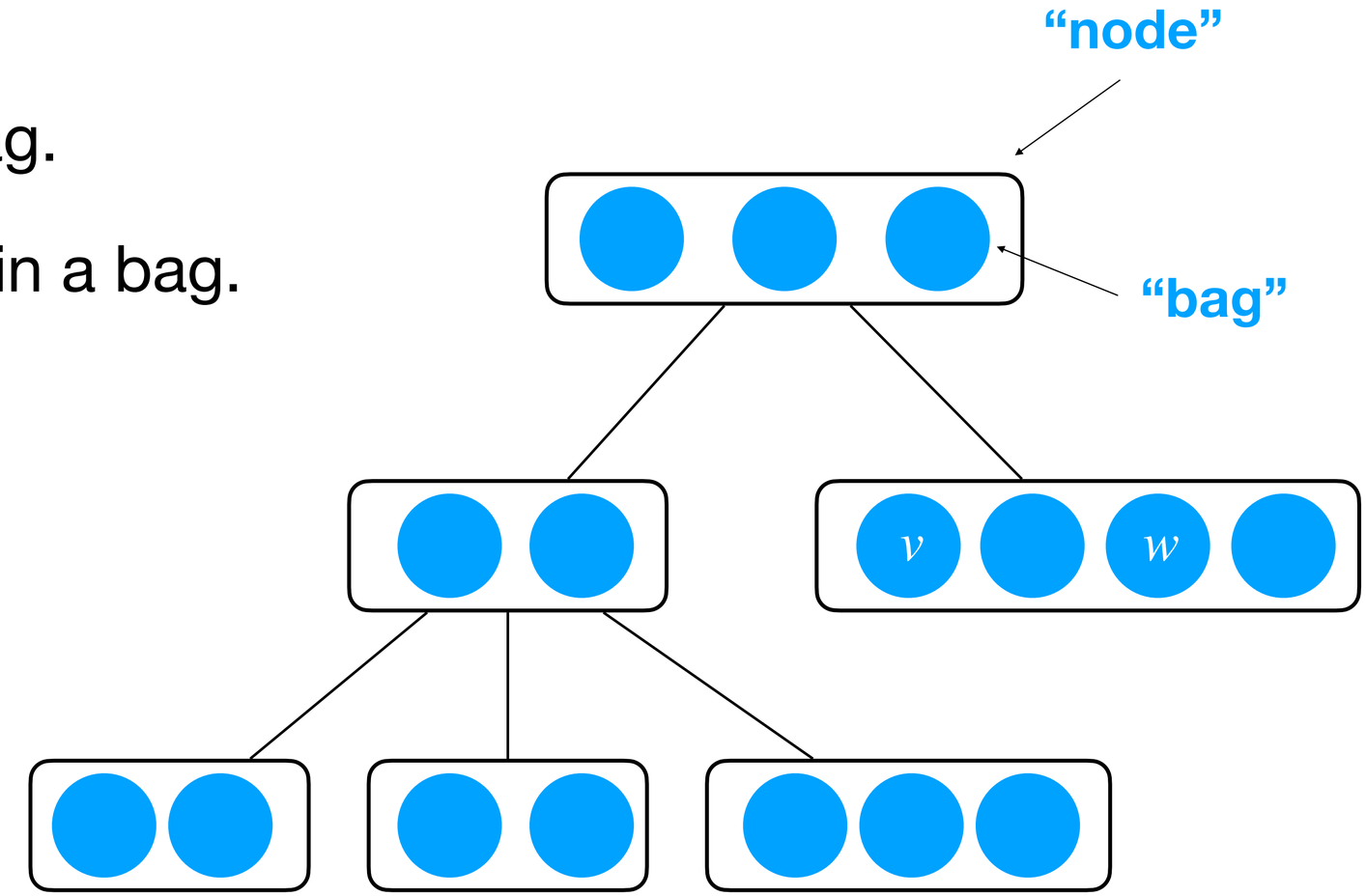
Tree Decompositions

1. Each **vertex** appears in a bag.



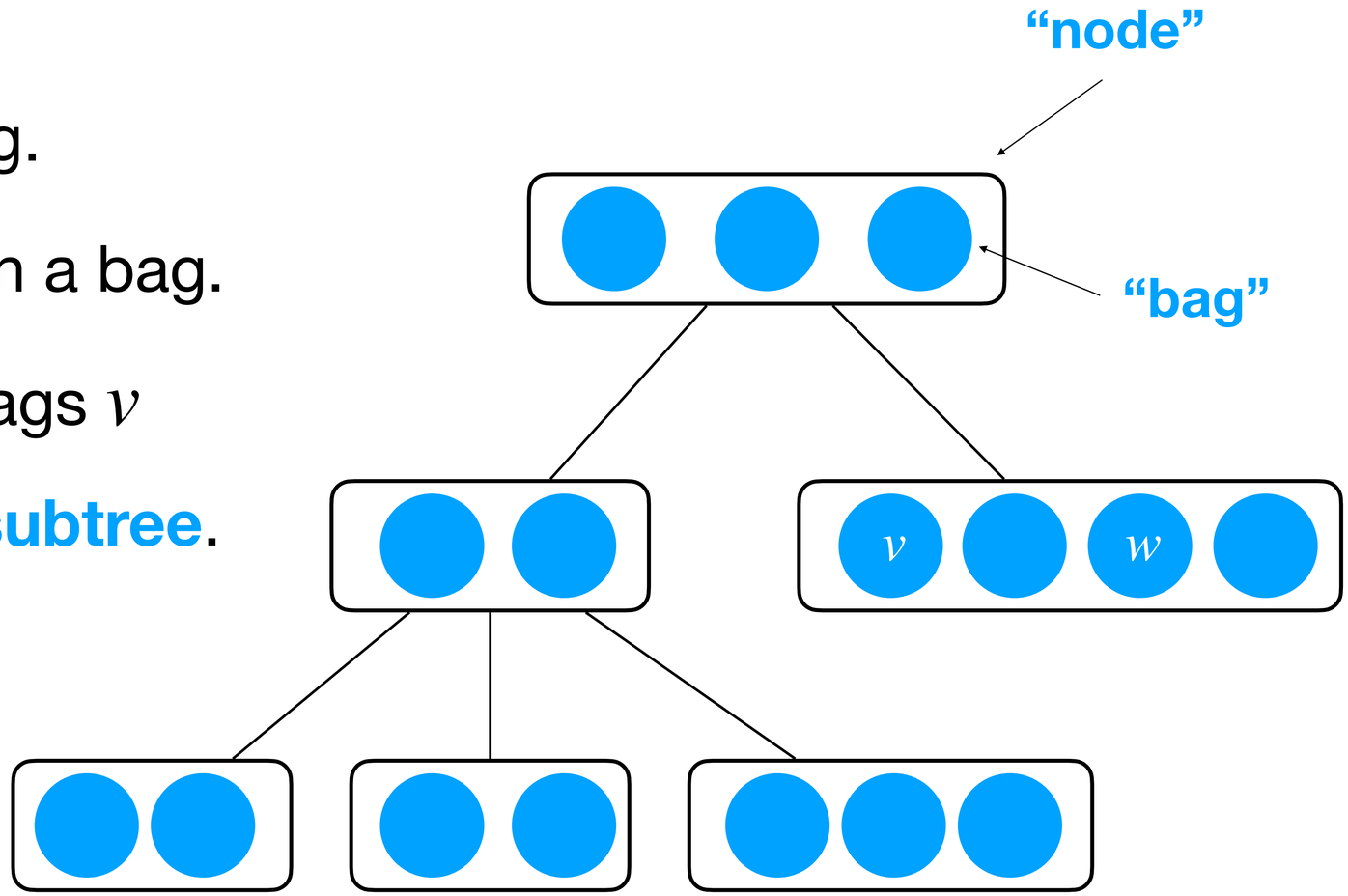
Tree Decompositions

1. Each **vertex** appears in a bag.
2. Each **edge** vw is contained in a bag.



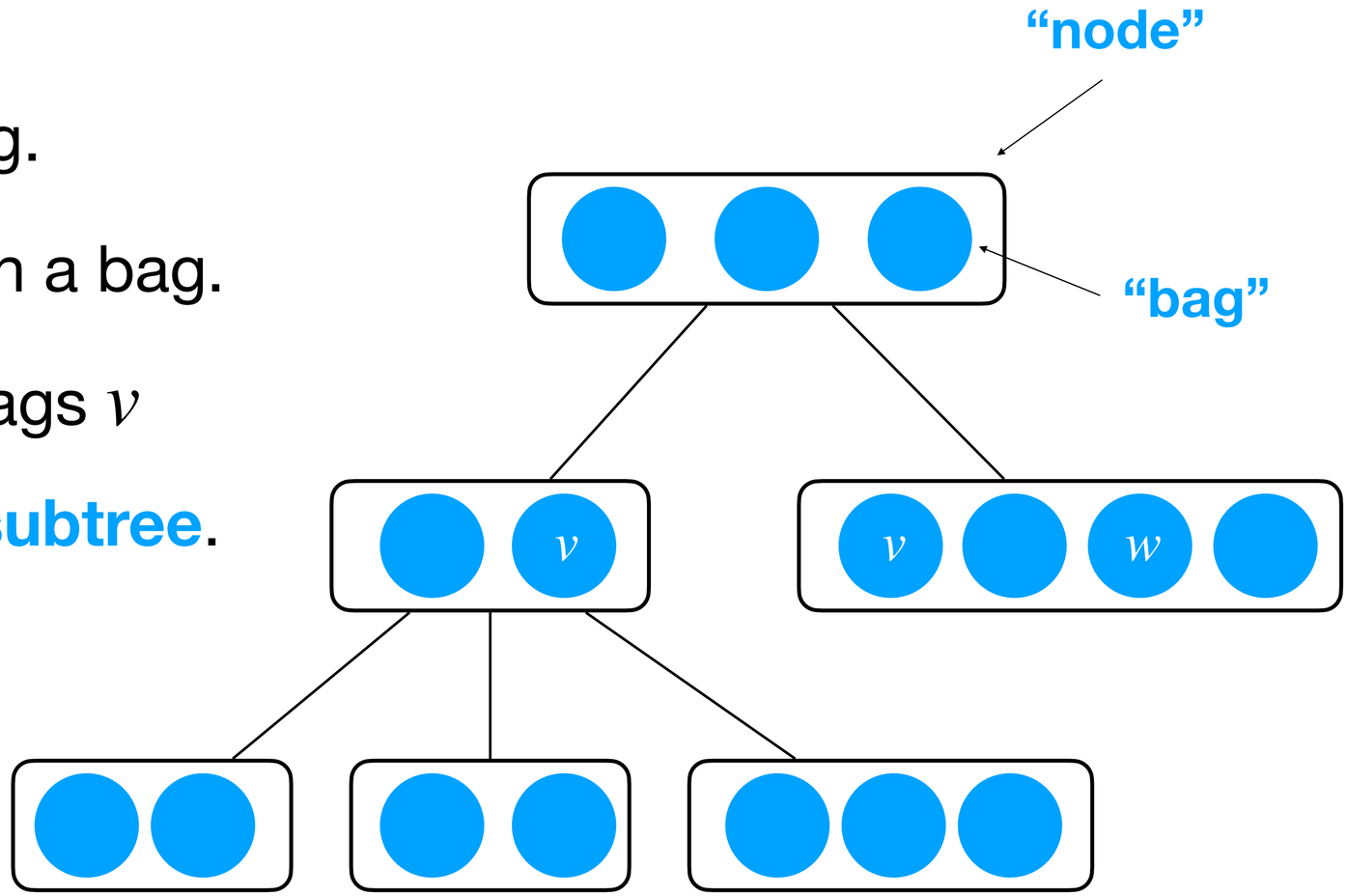
Tree Decompositions

1. Each **vertex** appears in a bag.
2. Each **edge** vw is contained in a bag.
3. The set of nodes in whose bags v appears form a **connected subtree**.



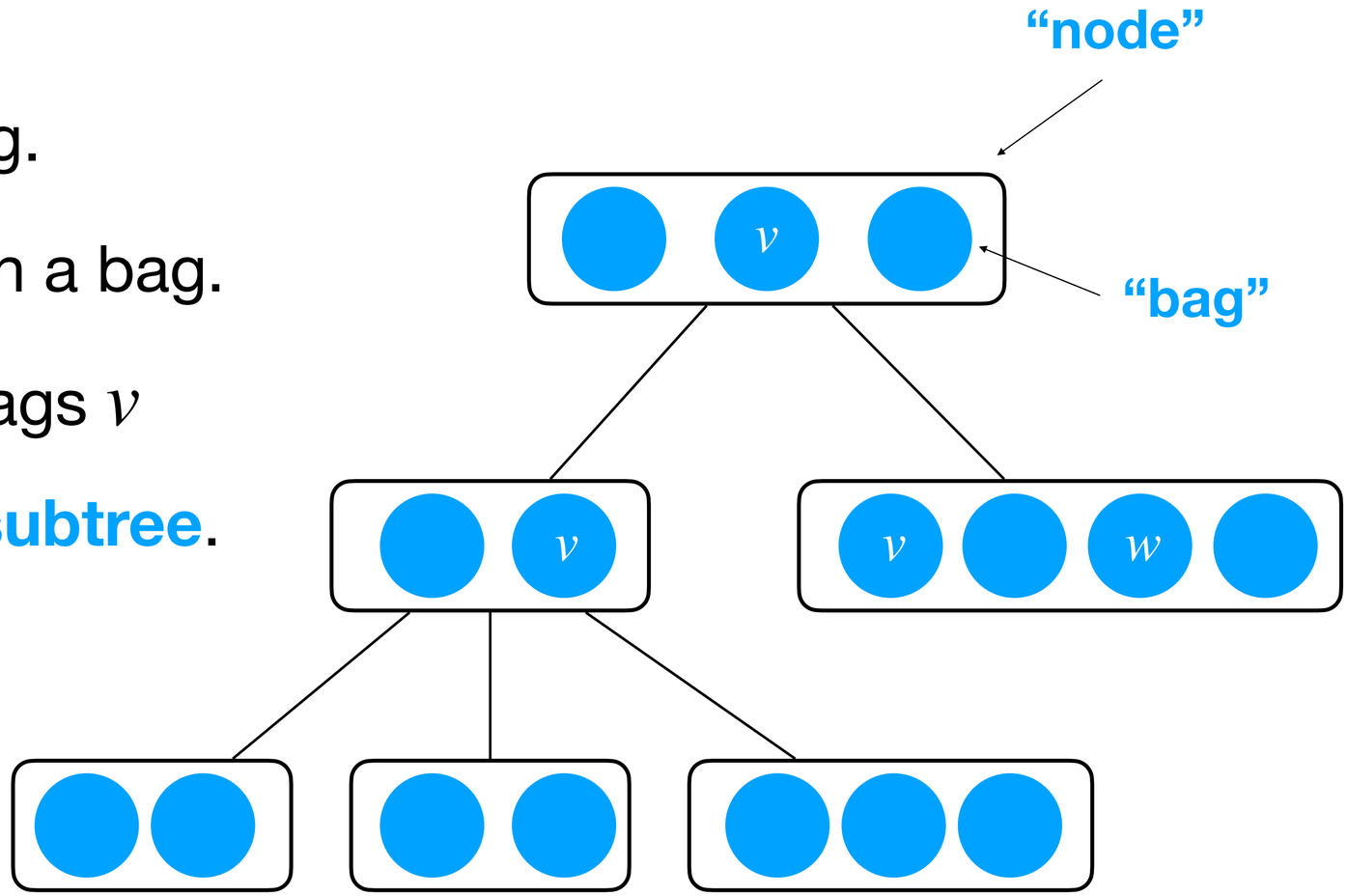
Tree Decompositions

1. Each **vertex** appears in a bag.
2. Each **edge** vw is contained in a bag.
3. The set of nodes in whose bags v appears form a **connected subtree**.



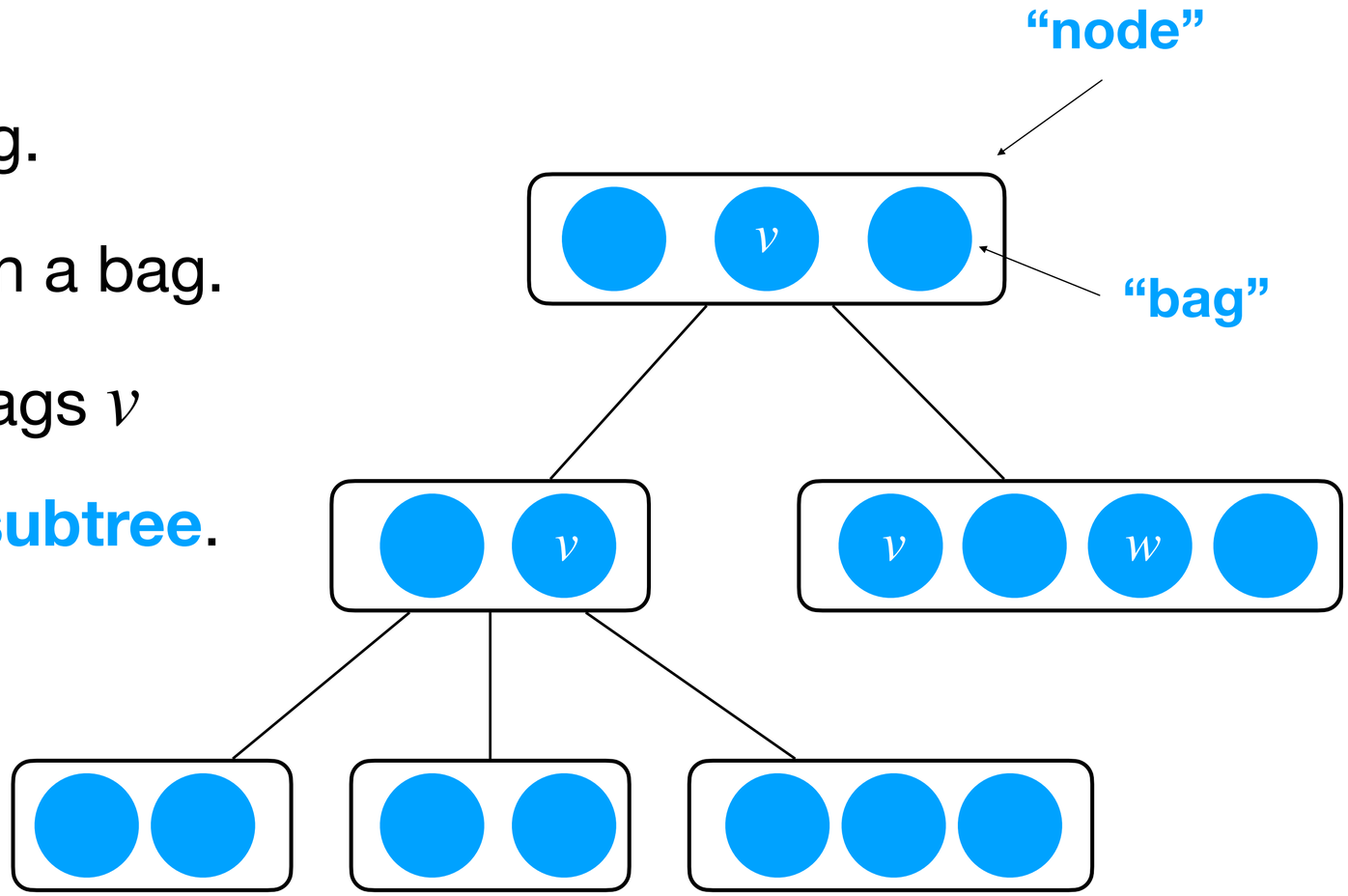
Tree Decompositions

1. Each **vertex** appears in a bag.
2. Each **edge** vw is contained in a bag.
3. The set of nodes in whose bags v appears form a **connected subtree**.



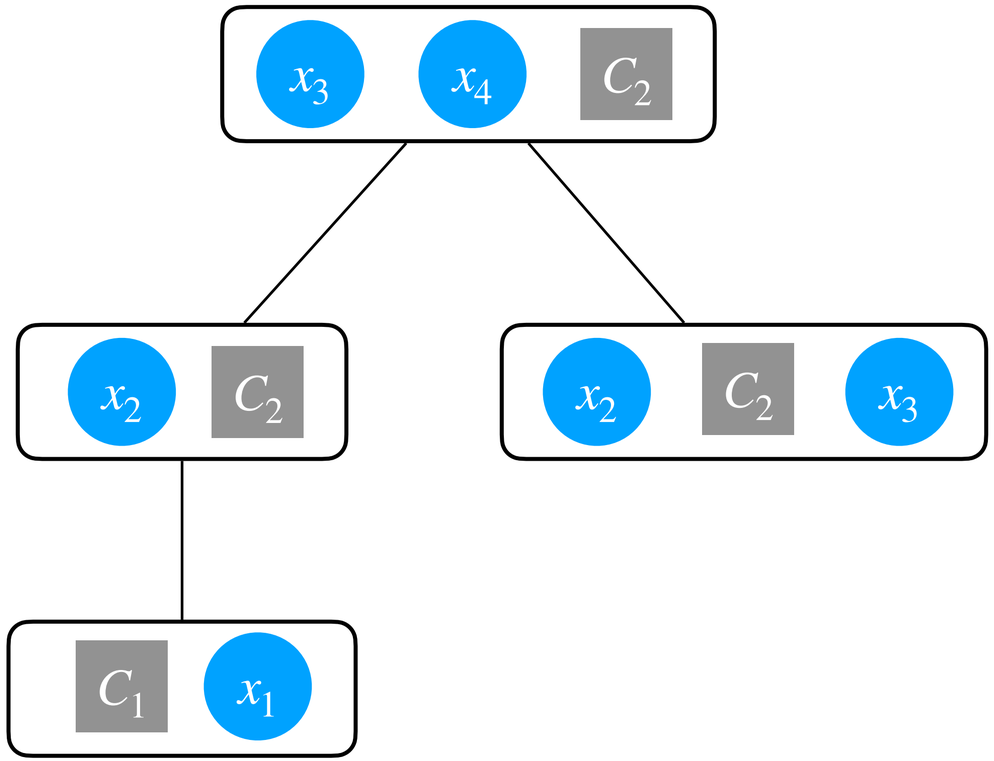
Tree Decompositions

1. Each **vertex** appears in a bag.
2. Each **edge** vw is contained in a bag.
3. The set of nodes in whose bags v appears form a **connected subtree**.

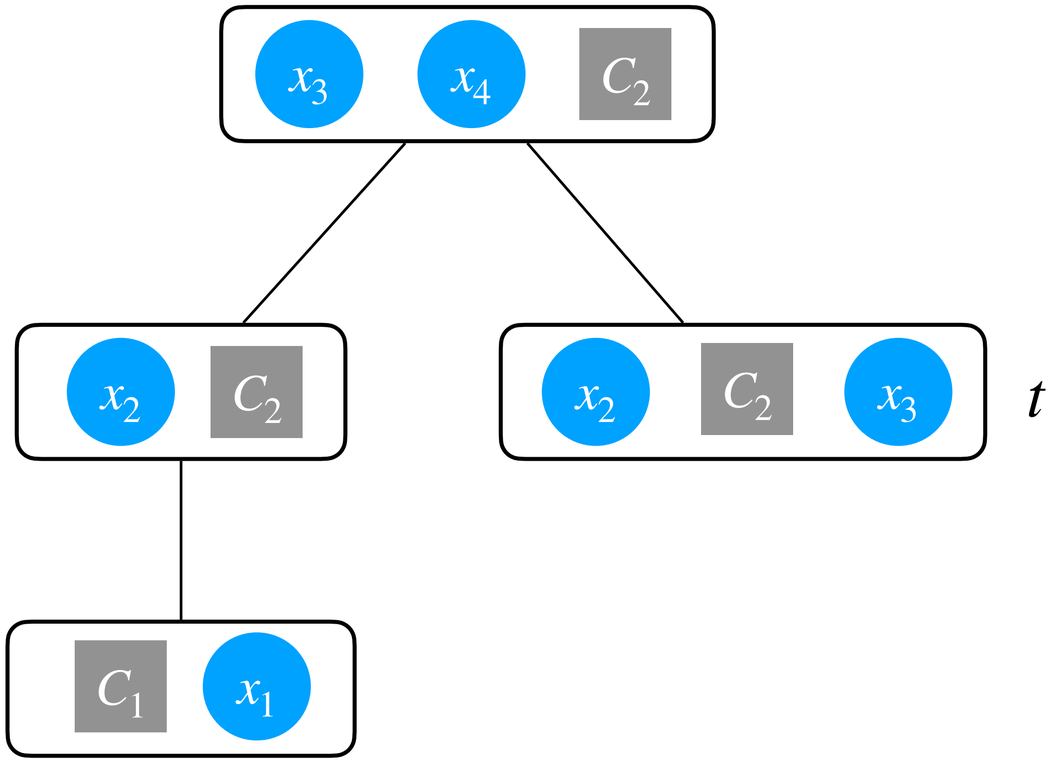


The **width** of a tree decomposition is the size of its largest bag - 1.

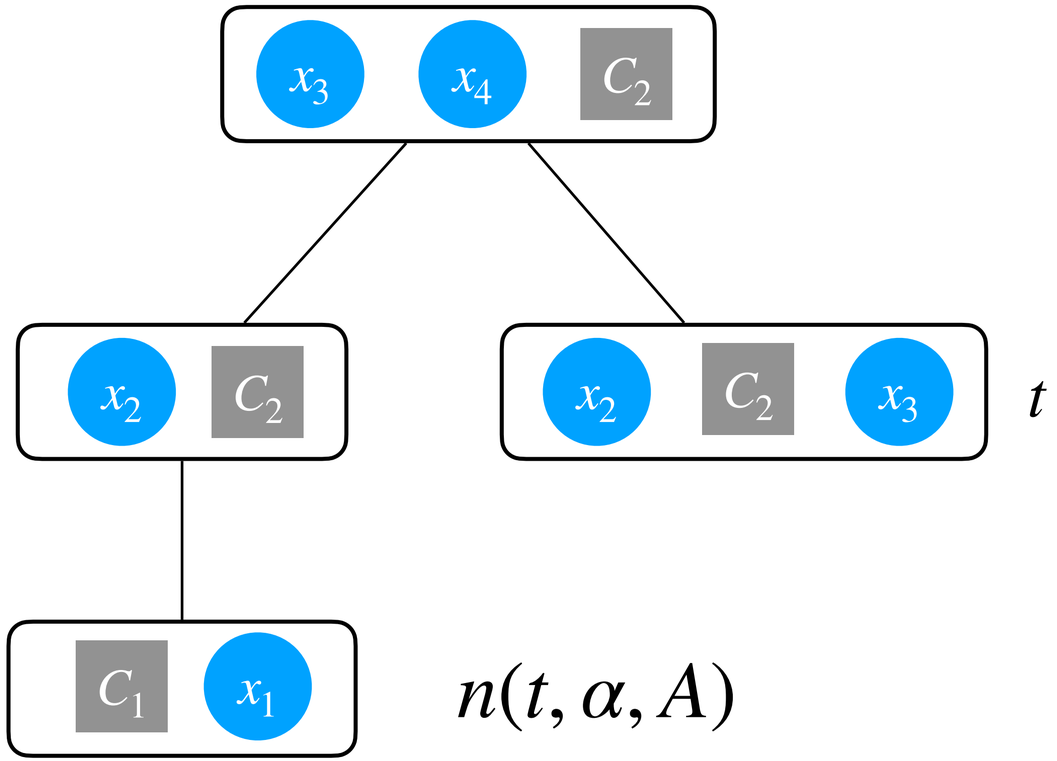
Dynamic Programming for #SAT



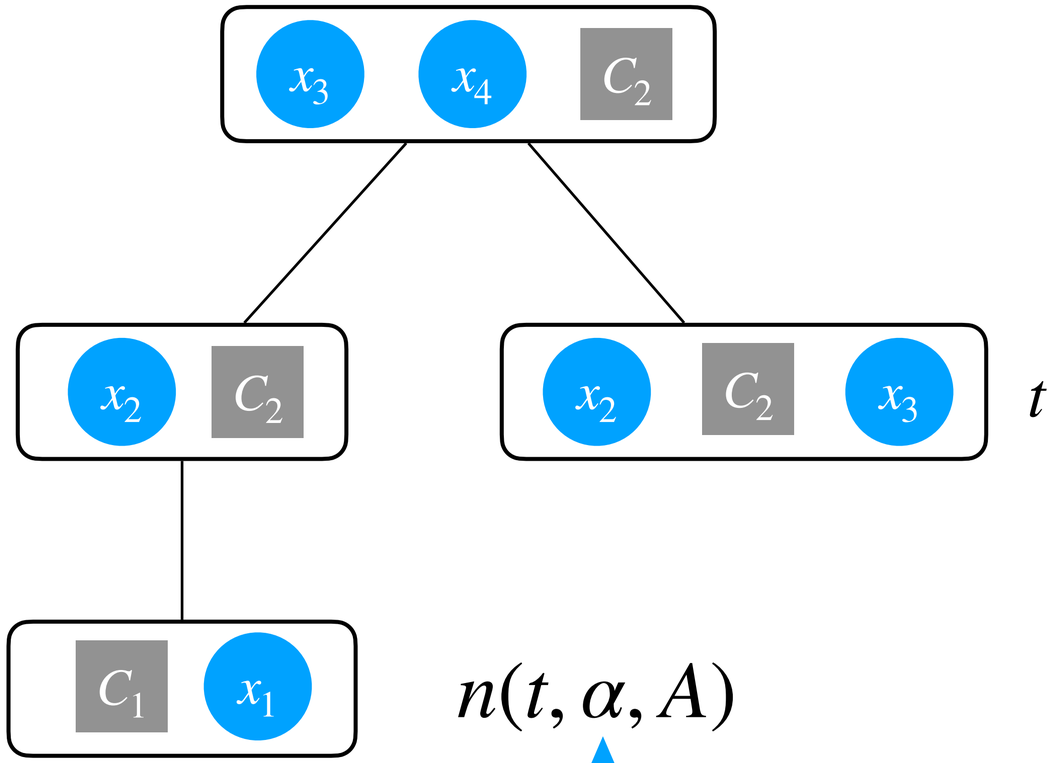
Dynamic Programming for #SAT



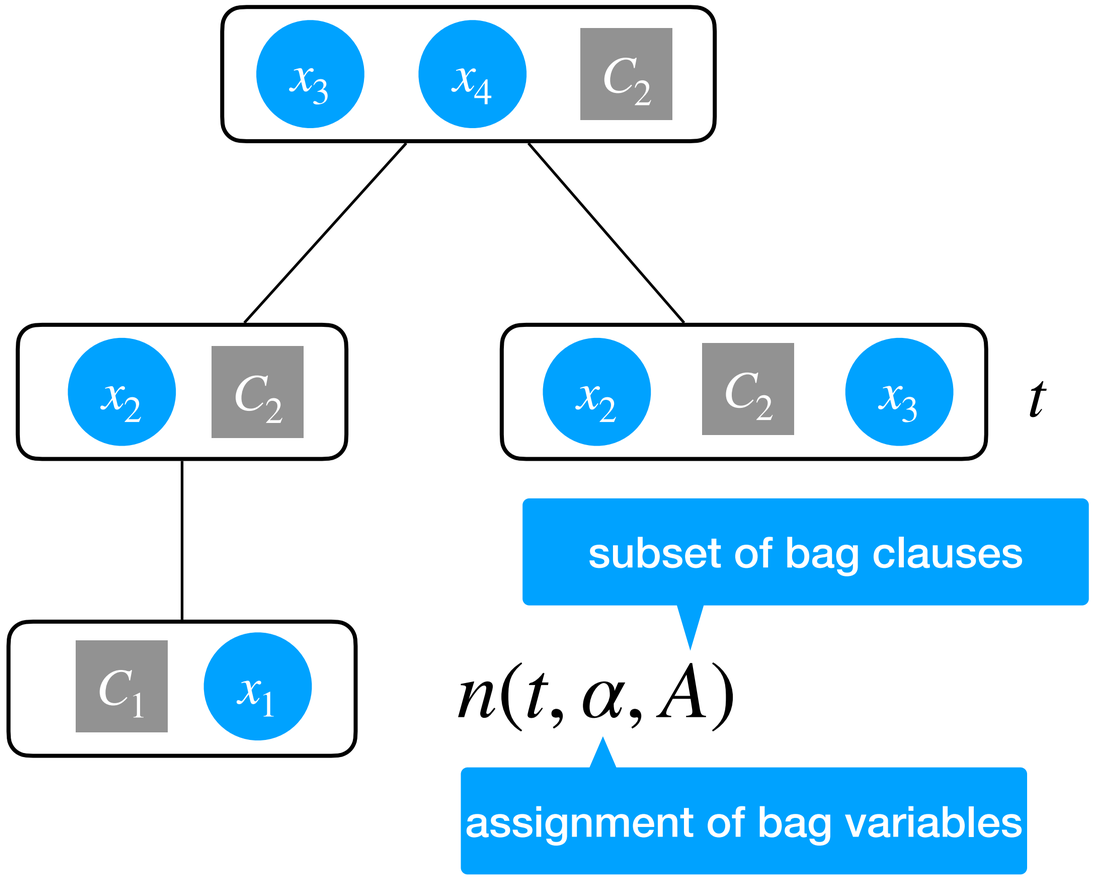
Dynamic Programming for #SAT



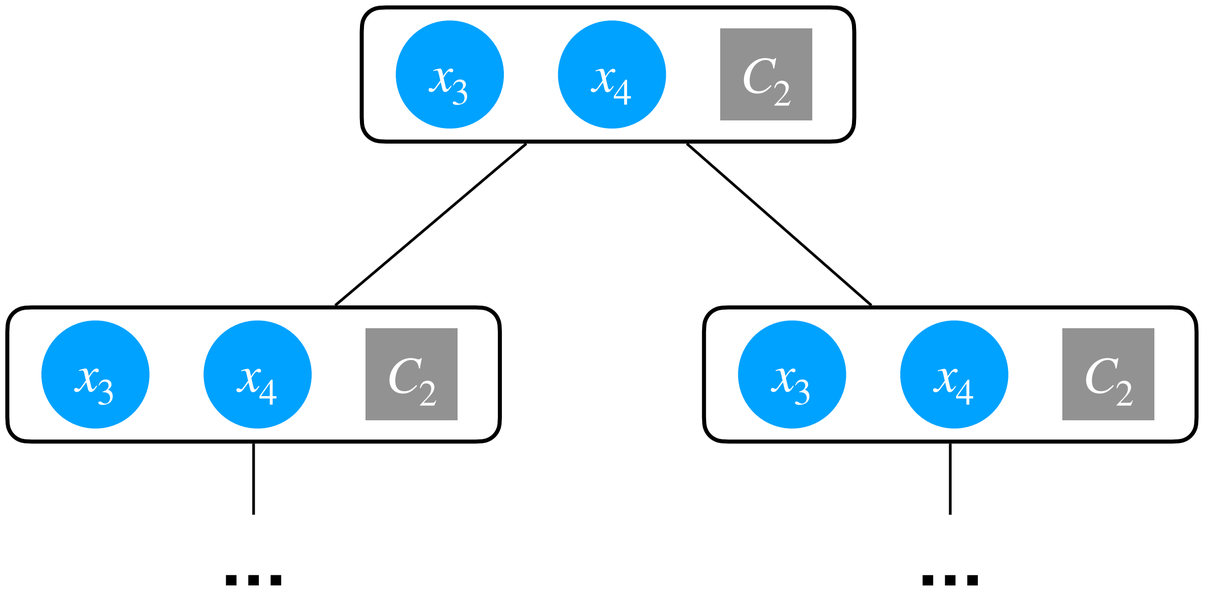
Dynamic Programming for #SAT



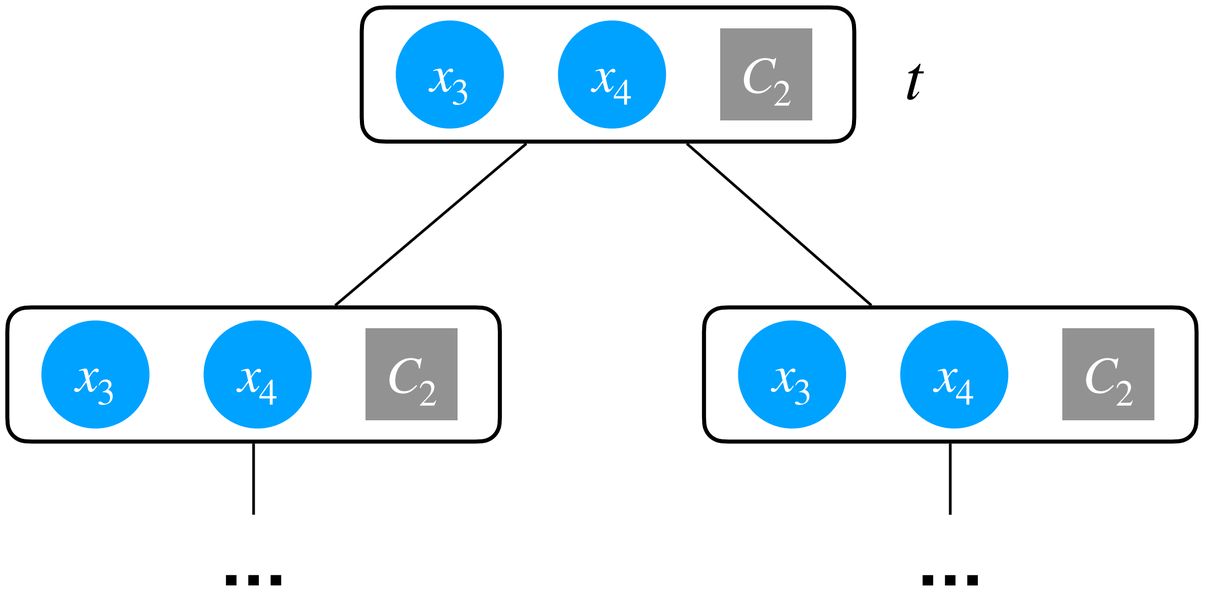
Dynamic Programming for #SAT



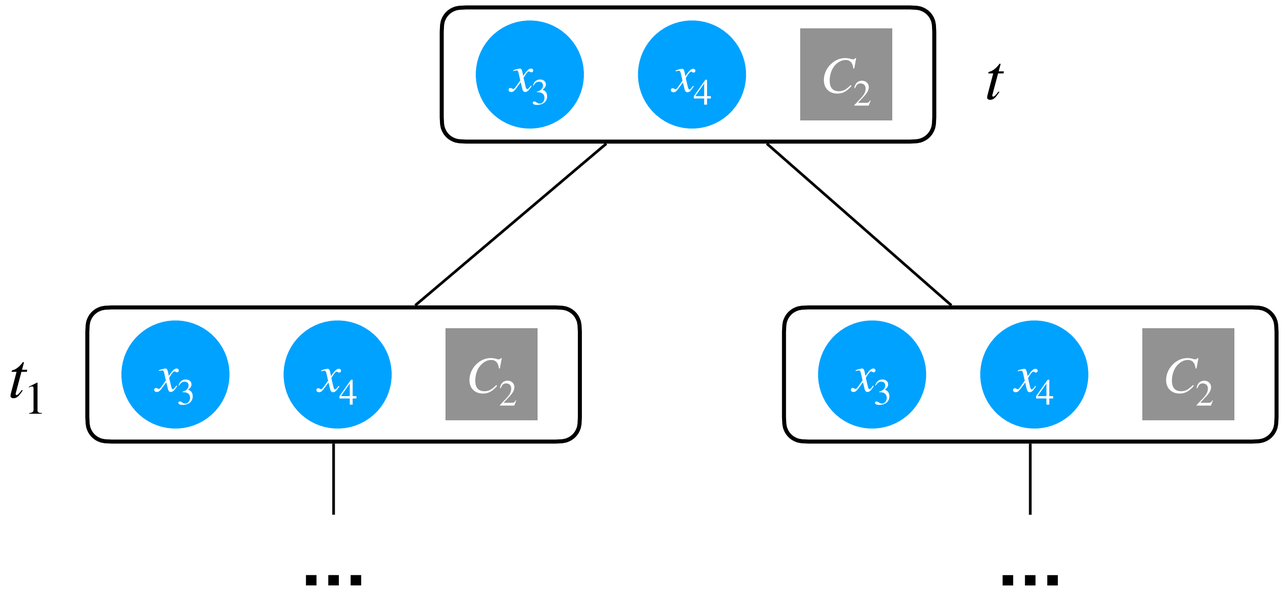
Join Nodes



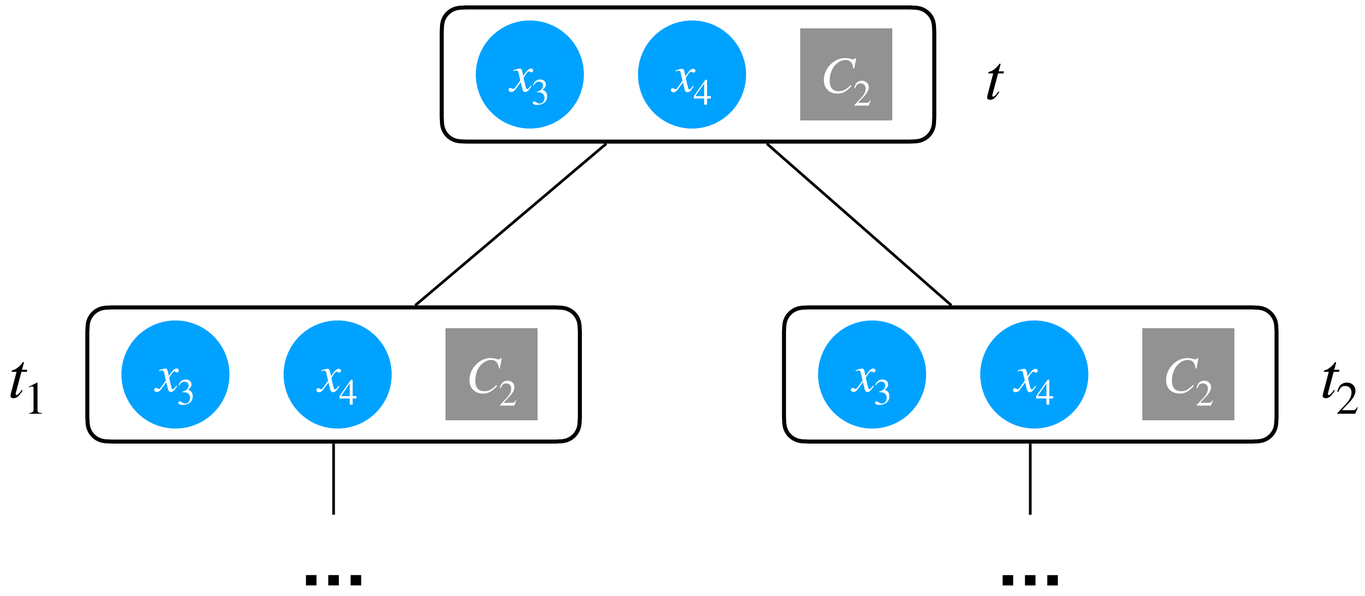
Join Nodes



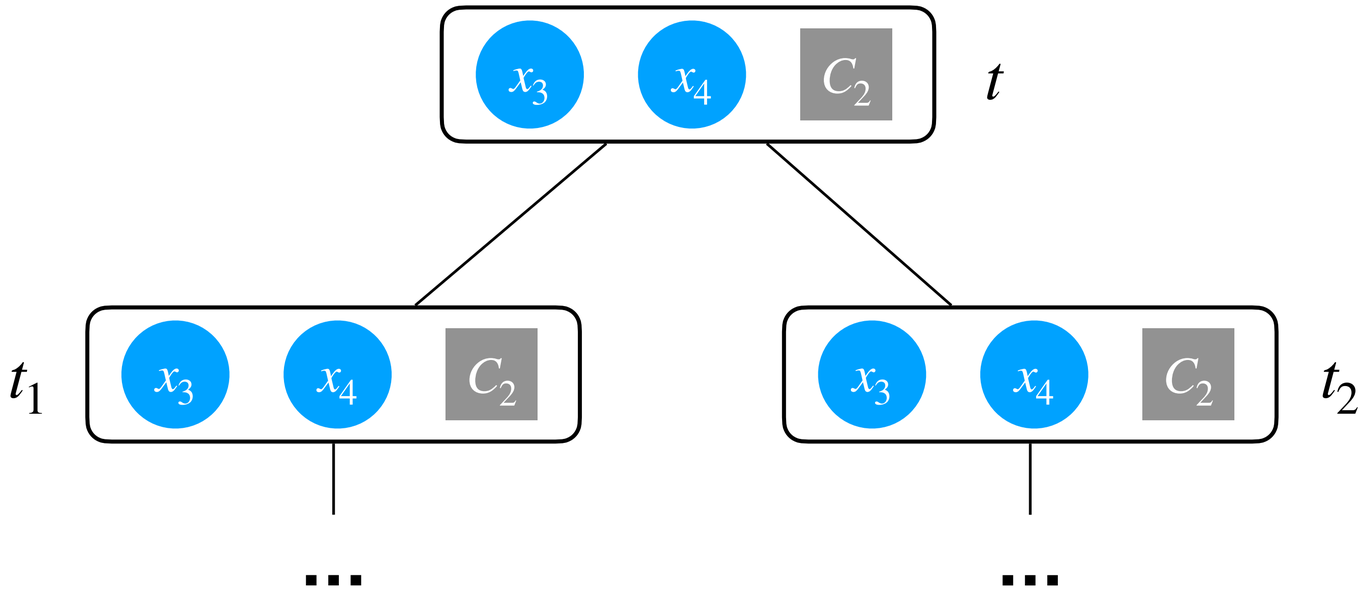
Join Nodes



Join Nodes



Join Nodes



$$n(t, \alpha, A) = \sum_{\substack{A_1, A_2 \subseteq \chi_c(t) \\ A_1 \cap A_2 = A}} n(t_1, \alpha, A_1)\, n(t_2, \alpha, A_2)$$

Covering Product

$$n(t, \alpha, A) = \sum_{\substack{A_1, A_2 \subseteq \chi_c(t) \\ A_1 \cap A_2 = A}} n(t_1, \alpha, A_1)\, n(t_2, \alpha, A_2)$$

Covering Product

$f, g : 2^S \to \mathbb{Z}$

$$n(t, \alpha, A) = \sum_{\substack{A_1, A_2 \subseteq \chi_c(t) \\ A_1 \cap A_2 = A}} n(t_1, \alpha, A_1)\, n(t_2, \alpha, A_2)$$

Covering Product

$f, g : 2^S \to \mathbb{Z}$

$|S| = k$

$$n(t, \alpha, A) = \sum_{\substack{A_1, A_2 \subseteq \chi_c(t) \\ A_1 \cap A_2 = A}} n(t_1, \alpha, A_1)\, n(t_2, \alpha, A_2)$$

Covering Product

$$f, g : 2^S \to \mathbb{Z}$$

$|S| = k$

$$(f *_c g)(Y) = \sum_{A \cup B = Y} f(A)\, g(B)$$

$$n(t, \alpha, A) = \sum_{\substack{A_1, A_2 \subseteq \chi_c(t) \\ A_1 \cap A_2 = A}} n(t_1, \alpha, A_1)\, n(t_2, \alpha, A_2)$$

Covering Product

$Y \subseteq S$

$$f, g : 2^S \to \mathbb{Z} \qquad (f *_c g)(Y) = \sum_{A \cup B = Y} f(A)\, g(B)$$

$|S| = k$

$$n(t, \alpha, A) = \sum_{\substack{A_1, A_2 \subseteq \chi_c(t) \\ A_1 \cap A_2 = A}} n(t_1, \alpha, A_1)\, n(t_2, \alpha, A_2)$$

Covering Product

$Y \subseteq S$

$$f, g : 2^S \to \mathbb{Z} \qquad (f *_c g)(Y) = \sum_{A \cup B = Y} f(A)\, g(B)$$

$|S| = k$

Theorem (Björklund, Husfeldt, Kaski, Koivisto 2007)

All values of the covering product $f *_c g$ can be computed with $O(2^k k)$ arithmetic operations.

$$n(t, \alpha, A) = \sum_{\substack{A_1, A_2 \subseteq \chi_c(t) \\ A_1 \cap A_2 = A}} n(t_1, \alpha, A_1)\, n(t_2, \alpha, A_2)$$

Summary

#SAT with incidence treewidth k^*

$$4^{k^*} \cdot p(F)$$

Summary

#SAT with incidence treewidth k^*

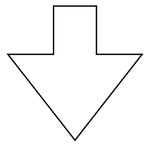
$$4^{k^*} \cdot p(F)$$

Fast Zeta- and Möbius Transform
Covering Product

Summary

#SAT with incidence treewidth k^*

$$4^{k^*} \cdot p(F)$$

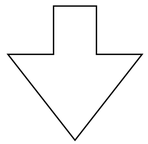


Fast Zeta- and Möbius Transform
Covering Product

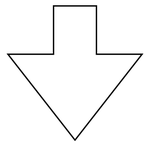
Summary

#SAT with incidence treewidth k^*

$$4^{k^*} \cdot p(F)$$



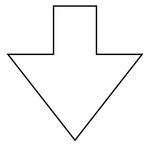
Fast Zeta- and Möbius Transform
Covering Product



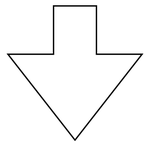
Summary

#SAT with incidence treewidth k^*

$4^{k^*} \cdot p(F)$



Fast Zeta- and Möbius Transform
Covering Product



$2^{k^*} \cdot p(F)$