Structure-Guided Automated Reasoning

Max Bannach, Markus Hecher

European Space Agency, Advanced Concepts Team, Netherlands
Univ. Artois, CNRS, Centre de Recherche en Informatique de Lens (CRIL), France

STACS 2025, Jena, 06.03.2025

Structure?

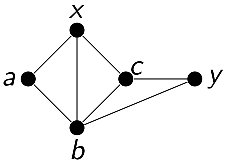
- Many problems are (computationally) hard, but simpler on trees
- There is a way to capture how *"tree-like"* a structure is – the so-called treewidth, which can be defined in terms of tree decompositions (TDs)

Why tree decompositions? Efficient Solving!

$F = (\neg a \vee b \vee x) \wedge (a \vee b) \wedge (c \vee \neg x) \wedge (b \vee \neg c) \wedge (\neg b \vee \neg c \vee \neg y)$

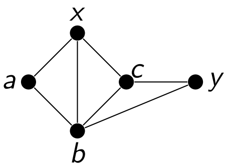
Why tree decompositions? Efficient Solving!

$F = (\neg a \vee b \vee x) \wedge (a \vee b) \wedge (c \vee \neg x) \wedge (b \vee \neg c) \wedge (\neg b \vee \neg c \vee \neg y)$

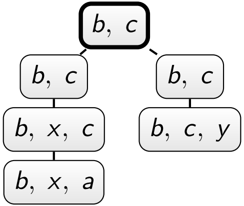


Why tree decompositions? Efficient Solving!

$F = (\neg a \vee b \vee x) \wedge (a \vee b) \wedge (c \vee \neg x) \wedge (b \vee \neg c) \wedge (\neg b \vee \neg c \vee \neg y)$

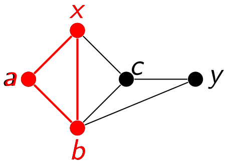


1. Decompose structure of P_F

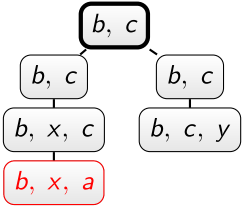


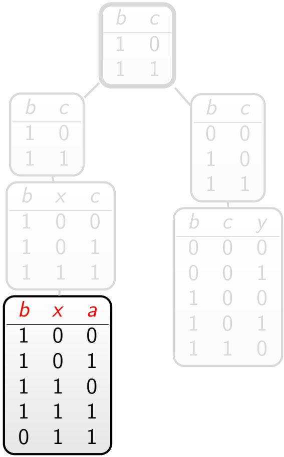
Why tree decompositions? Efficient Solving!

$F = (\neg a \vee b \vee x) \wedge (a \vee b) \wedge (c \vee \neg x) \wedge (b \vee \neg c) \wedge (\neg b \vee \neg c \vee \neg y)$



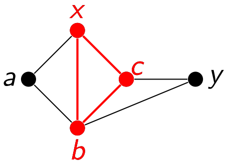
1. Decompose structure of P_F
2. Solve subproblems



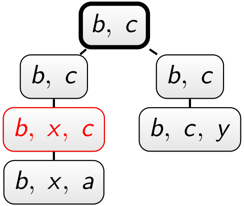


Why tree decompositions? Efficient Solving!

$F = (\neg a \vee b \vee x) \wedge (a \vee b) \wedge (c \vee \neg x) \wedge (b \vee \neg c) \wedge (\neg b \vee \neg c \vee \neg y)$



1. Decompose structure of P_F
2. Solve subproblems



b	c
1	0
1	1

b	c
1	0
1	1

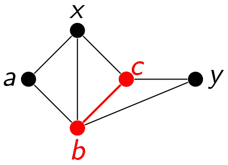
b	c
0	0
1	0
1	1

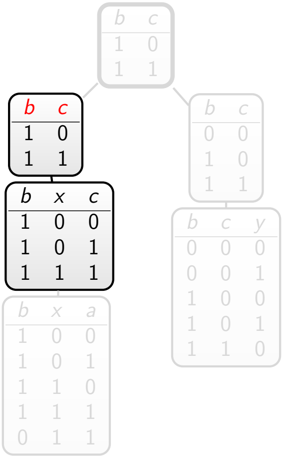
b	x	c
1	0	0
1	0	1
1	1	1

b	c	y
0	0	0
0	0	1
1	0	0
1	0	1
1	1	0

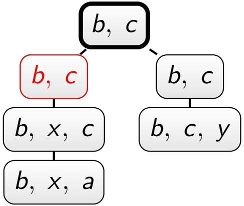
b	x	a
1	0	0
1	0	1
1	1	0
1	1	1
0	1	1

$F = (\neg a \vee b \vee x) \wedge (a \vee b) \wedge (c \vee \neg x) \wedge (b \vee \neg c) \wedge (\neg b \vee \neg c \vee \neg y)$



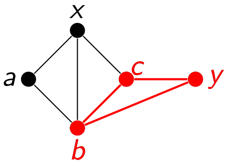


1. Decompose structure of P_F
2. Solve subproblems

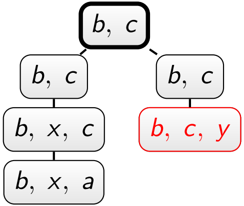


Why tree decompositions? Efficient Solving!

$F = (\neg a \lor b \lor x) \land (a \lor b) \land (c \lor \neg x) \land (b \lor \neg c) \land (\neg b \lor \neg c \lor \neg y)$



1. Decompose structure of P_F
2. Solve subproblems



b	c
1	0
1	1

b	c
1	0
1	1

b	x	c
1	0	0
1	0	1
1	1	1

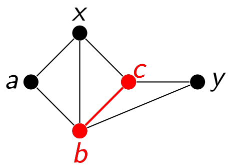
b	x	a
1	0	0
1	0	1
1	1	0
1	1	1
0	1	1

b	c
0	0
1	0
1	1

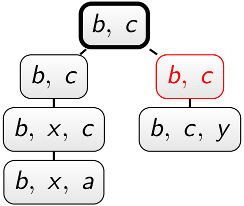
b	c	y
0	0	0
0	0	1
1	0	0
1	0	1
1	1	0

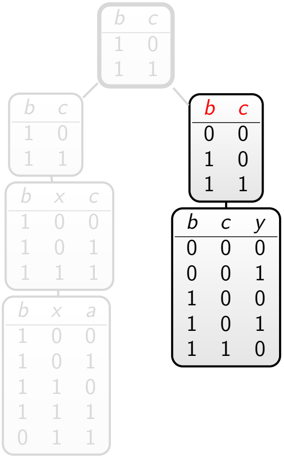
Why tree decompositions? Efficient Solving!

$F = (\neg a \vee b \vee x) \wedge (a \vee b) \wedge (c \vee \neg x) \wedge (b \vee \neg c) \wedge (\neg b \vee \neg c \vee \neg y)$

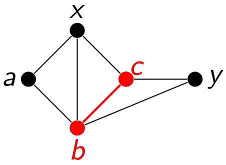


1. Decompose structure of P_F
2. Solve subproblems

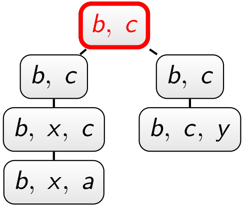


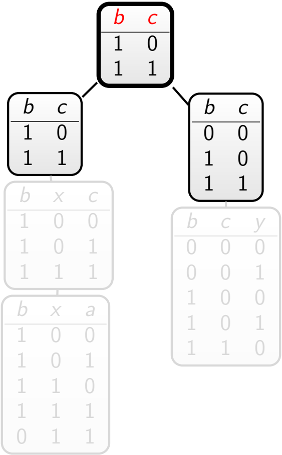


$F = (\neg a \vee b \vee x) \wedge (a \vee b) \wedge (c \vee \neg x) \wedge (b \vee \neg c) \wedge (\neg b \vee \neg c \vee \neg y)$



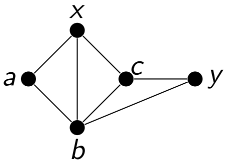
1. Decompose structure of P_F
2. Solve subproblems



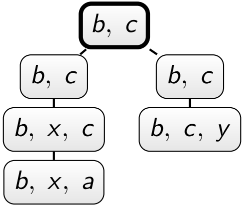


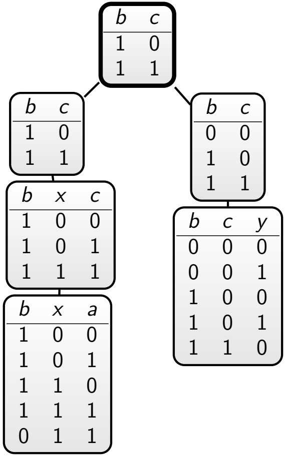
Why tree decompositions? Efficient Solving!

$F = (\neg a \vee b \vee x) \wedge (a \vee b) \wedge (c \vee \neg x) \wedge (b \vee \neg c) \wedge (\neg b \vee \neg c \vee \neg y)$



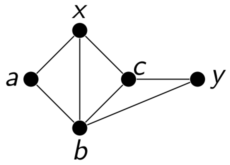
1. Decompose structure of P_F
2. Solve subproblems



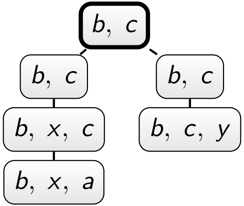


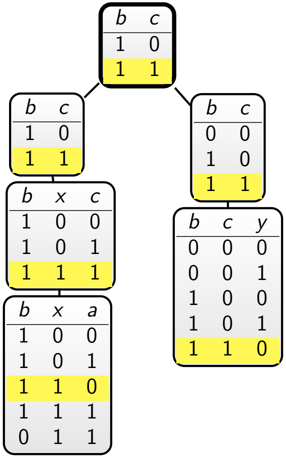
Why tree decompositions? Efficient Solving!

$F = (\neg a \vee b \vee x) \wedge (a \vee b) \wedge (c \vee \neg x) \wedge (b \vee \neg c) \wedge (\neg b \vee \neg c \vee \neg y)$



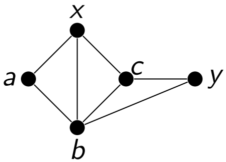
1. Decompose structure of P_F
2. Solve subproblems
3. Combine solutions



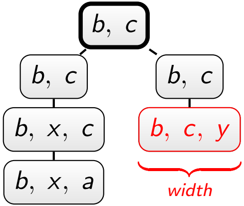


Why tree decompositions? Efficient Solving!

$F = (\neg a \vee b \vee x) \wedge (a \vee b) \wedge (c \vee \neg x) \wedge (b \vee \neg c) \wedge (\neg b \vee \neg c \vee \neg y)$



1. Decompose structure of P_F
2. Solve subproblems
3. Combine solutions



b	c
1	0
1	1

b	c
1	0
1	1

b	x	c
1	0	0
1	0	1
1	1	1

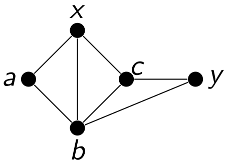
b	x	a
1	0	0
1	0	1
1	1	0
1	1	1
0	1	1

b	c
0	0
1	0
1	1

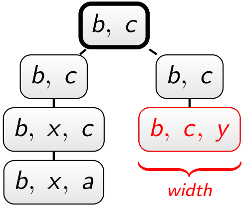
b	c	y
0	0	0
0	0	1
1	0	0
1	0	1
1	1	0

Why tree decompositions? Efficient Solving!

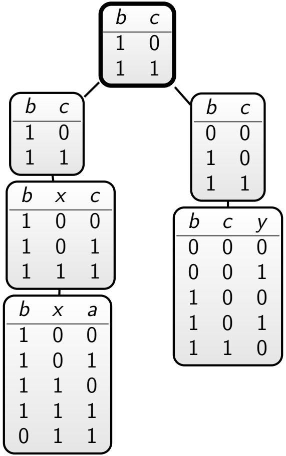
$F = (\neg a \vee b \vee x) \wedge (a \vee b) \wedge (c \vee \neg x) \wedge (b \vee \neg c) \wedge (\neg b \vee \neg c \vee \neg y)$



1. Decompose structure of P_F
2. Solve subproblems
3. Combine solutions

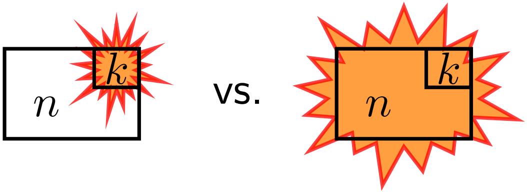


$\rightarrow 2^{width} \cdot n$



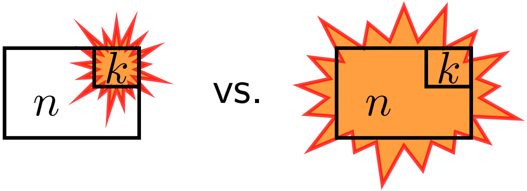
Treewidth k [Robertson & Seymour 83], [Bertele & Brioschi 69]

- renders large variety of NP-hard problems *fixed-parameter tractable (FPT)*



Treewidth k [Robertson & Seymour 83], [Bertele & Brioschi 69]

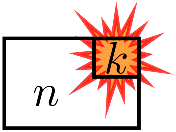
- renders large variety of NP-hard problems *fixed-parameter tractable (FPT)*



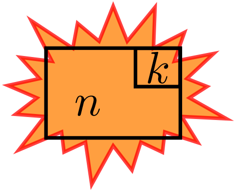
$f(k) \cdot n$ vs. $g(n)$

Treewidth k [Robertson & Seymour 83], [Bertele & Brioschi 69]

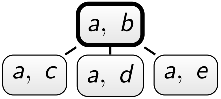
- renders large variety of NP-hard problems *fixed-parameter tractable (FPT)*



vs.



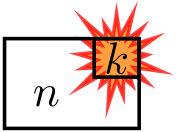
$f(k) \cdot n$ vs. $g(n)$



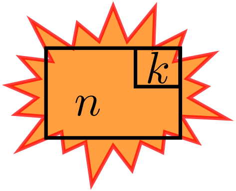
vs.

Treewidth k [Robertson & Seymour 83], [Bertele & Brioschi 69]

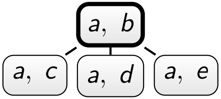
- renders large variety of NP-hard problems *fixed-parameter tractable (FPT)*



vs.



$f(k) \cdot n$ vs. $g(n)$



vs.

⇝ How do these $f(k)$ look like? Can we do better?

What's the Issue?

$\rightsquigarrow$ Bad worst case: $f(k) \gg k$ for large treewidth k

- polynomial $f(k)$ probably not possible

What's the Issue?

⇝ Bad worst case: $f(k) \gg k$ for large treewidth k

- polynomial $f(k)$ probably not possible

⇝ Not much hope to improve $f(k)$ in the worst case

- neither for SAT extensions [Lampis & Mitsou 17], nor QSAT [Fichte, H & Pfandler 20]

What's the Issue?

- Bad worst case: $f(k) \gg k$ for large treewidth k
  - polynomial $f(k)$ probably not possible
- Not much hope to improve $f(k)$ in the worst case
  - neither for SAT extensions [Lampis & Mitsou 17], nor QSAT [Fichte, H & Pfandler 20]

Problem	Bounds $f(k)$ Upper	Bounds $f(k)$ Lower (ETH)	Complexity
SAT, #SAT, MAXSAT	$2^{\mathcal{O}(k)}$	$2^{o(k)}$	NP, #P, optP

What's the Issue?

⇝ Bad worst case: $f(k) \gg k$ for large treewidth k

- polynomial $f(k)$ probably not possible

⇝ Not much hope to improve $f(k)$ in the worst case

- neither for SAT extensions [Lampis & Mitsou 17], nor QSAT [Fichte, H & Pfandler 20]

Problem	Bounds $f(k)$ Upper	Bounds $f(k)$ Lower (ETH)	Complexity
SAT, #SAT, MAXSAT	$2^{\mathcal{O}(k)}$	$2^{o(k)}$	NP, #P, optP
#∃SAT	$2^{2^{\mathcal{O}(k)}}$	$2^{2^{o(k)}}$	$\# \cdot$ NP

What's the Issue?

⇝ Bad worst case: $f(k) \gg k$ for large treewidth k

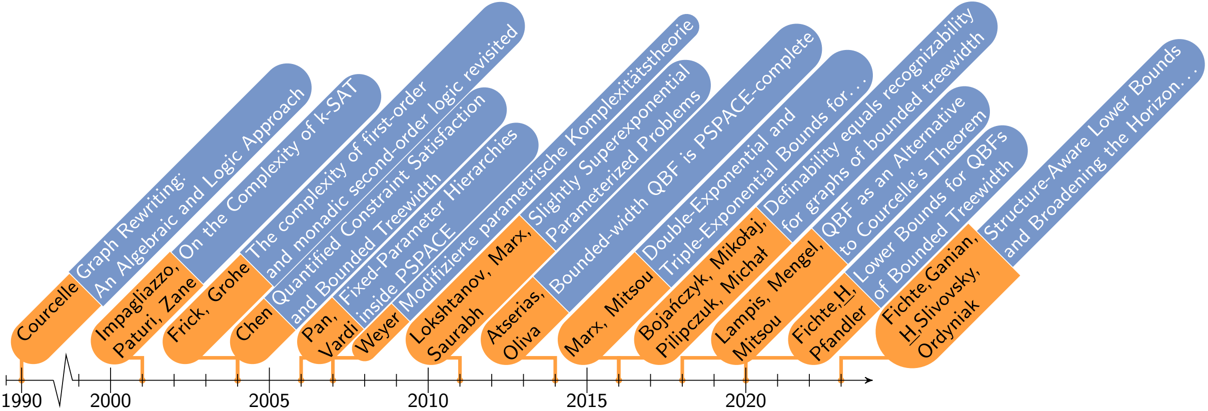
- polynomial $f(k)$ probably not possible

⇝ Not much hope to improve $f(k)$ in the worst case

- neither for SAT extensions [Lampis & Mitsou 17], nor QSAT [Fichte, H & Pfandler 20]

Problem	Bounds $f(k)$ Upper	Bounds $f(k)$ Lower (ETH)	Complexity
SAT, #SAT, MAXSAT	$2^{\mathcal{O}(k)}$	$2^{o(k)}$	NP, #P, optP
#∃SAT	$2^{2^{\mathcal{O}(k)}}$	$2^{2^{o(k)}}$	$\# \cdot \mathrm{NP}$
QSAT_ℓ	$\mathrm{tower}(\ell, \mathcal{O}(k))$	$\mathrm{tower}(\ell, o(k))$	Σ_ℓ^{P} / Π_ℓ^{P}

Biased Excerpt of Related Work



Thanks to Arne Meier (Uni Hannover) for the timeline latex package.

FPT Sufficient? Courcelle's Theorem!

⇝ Well-known meta-result *Courcelle's theorem* [Courcelle 90]

FPT Sufficient? Courcelle's Theorem!

⇝ Well-known meta-result *Courcelle's theorem* [Courcelle 90]

Example: 3-Coloring

Can we color a graph with 3 vertex colors s.t. adjacent vertices are colored differently?

$$\varphi_{\mathrm{3col}} = \exists R, G, B\, \forall x, y\, .\, (Rx \vee Gx \vee Bx) \quad \wedge$$

FPT Sufficient? Courcelle's Theorem!

$\rightsquigarrow$ Well-known meta-result *Courcelle's theorem* [Courcelle 90]

Example: 3-Coloring

Can we color a graph with 3 vertex colors s.t. adjacent vertices are colored differently?

$$\varphi_{\mathrm{3col}} = \exists R, G, B\, \forall x, y\, .\, (Rx \vee Gx \vee Bx) \quad \wedge$$
$$[Exy \rightarrow \neg((Rx \wedge Ry) \vee (Gx \wedge Gy) \vee (Bx \wedge By))]$$

FPT Sufficient? Courcelle's Theorem!

⇝ Well-known meta-result *Courcelle's theorem* [Courcelle 90]

Example: 3-Coloring

Can we color a graph with 3 vertex colors s.t. adjacent vertices are colored differently?

$$\varphi_{\mathrm{3col}} = \exists R, G, B\, \forall x, y\, .\, (Rx \vee Gx \vee Bx) \quad \wedge$$
$$[Exy \rightarrow \neg((Rx \wedge Ry) \vee (Gx \wedge Gy) \vee (Bx \wedge By))]$$

We have  $\models \varphi_{\mathrm{3col}}$ and  $\not\models \varphi_{\mathrm{3col}}$

Can We Efficiently Translate To SAT?

MSO To SAT?

- Can we derive SAT formula ψ with $tw(\psi) \leqslant g\big(tw(\mathcal{S})\big)$?
  ($\mathcal{S} \models \varphi$ iff $\psi \in$ SAT)

MSO To SAT?

- Can we derive SAT formula ψ with $tw(\psi) \leqslant g\big(tw(\mathcal{S})\big)$?
  ($\mathcal{S} \models \varphi$ iff $\psi \in$ SAT)

$\rightsquigarrow$ Naive Approach: $\forall$ translates to $\bigwedge$, $\exists$ translates to $\bigvee$

MSO To SAT?

- Can we derive SAT formula ψ with $tw(\psi) \leqslant g(tw(\mathcal{S}))$?
  ($\mathcal{S} \models \varphi$ iff $\psi \in$ SAT)

⇝ Naive Approach: $\forall$ translates to $\bigwedge$, $\exists$ translates to $\bigvee$

Example: 3-Coloring

$$\psi_{3\text{col}} = \overbrace{\bigwedge_{u \in V(G)} \bigwedge_{v \in V(G)}}^{\forall x \forall y} \overbrace{(R_u \vee G_u \vee B_u)}^{Rx \vee Gx \vee Bx} \wedge \overbrace{\bigwedge_{\{u,v\} \in E(G)}}^{Exy \rightarrow} \neg((R_u \wedge R_v) \vee (G_u \wedge G_v) \vee (B_u \wedge B_v)).$$

propositional variables

MSO To Sat?

- Can we derive Sat formula ψ with $tw(\psi) \leqslant g(tw(\mathcal{S}))$?
  $(\mathcal{S} \models \varphi$ iff $\psi \in$ Sat$)$

⇝ Naive Approach: $\forall$ translates to $\bigwedge$, $\exists$ translates to $\bigvee$

Example: 3-Coloring

$\psi_{3col} =$

$$\overbrace{\bigwedge_{u \in V(G)} \bigwedge_{v \in V(G)}}^{\forall x \forall y} \overbrace{(R_u \vee G_u \vee B_u)}^{Rx \vee Gx \vee Bx} \wedge \overbrace{\bigwedge_{\{u,v\} \in E(G)}}^{Exy \rightarrow} \neg((R_u \wedge R_v) \vee (G_u \wedge G_v) \vee (B_u \wedge B_v)).$$

propositional variables

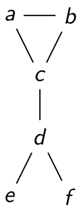
What about the treewidth?

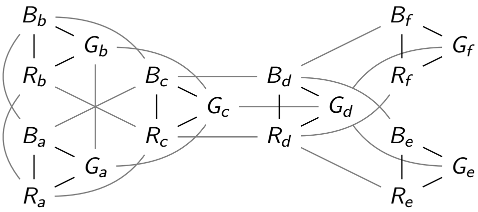
MSO To SAT?

- Can we derive SAT formula ψ with $tw(\psi) \leqslant g\big(tw(\mathcal{S})\big)$?
  $(\mathcal{S} \models \varphi$ iff $\psi \in$ SAT$)$

$\rightsquigarrow$ Naive Approach: $\forall$ translates to $\bigwedge$, $\exists$ translates to $\bigvee$

Example: 3-Coloring



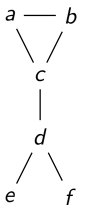


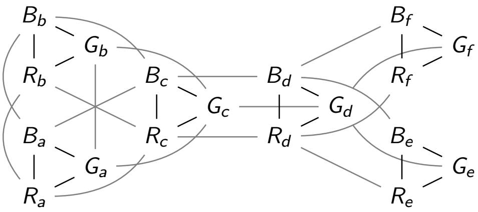
MSO To Sat?

- Can we derive Sat formula ψ with $tw(\psi) \leqslant g\big(tw(\mathcal{S})\big)$?
  ($\mathcal{S} \models \varphi$ iff $\psi \in$ Sat)

$\rightsquigarrow$ Naive Approach: $\forall$ translates to $\bigwedge$, $\exists$ translates to $\bigvee$

Example: 3-Coloring





Factor 3 overhead: $\rightsquigarrow$ 3-Coloring with $f(k) = 2^{3k}$

MSO To Sat?

- Does this always work?

MSO To Sat?

- Does this always work?

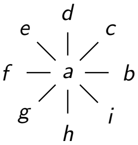
Example: Dominating Set

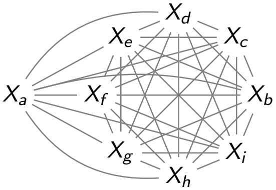
$$\varphi_{\mathrm{ds}}(X) = \forall x \exists y \, . \, Xx \vee (Exy \wedge Xy)$$

- Does this always work? NO!

Example: Dominating Set

$$\varphi_{\mathrm{ds}}(X) = \forall x \exists y \,.\, Xx \lor (Exy \land Xy)$$





Contributions

1. Faster Structure-Guided Reasoning . . .
   - . . . for $\textsc{QSat}_\ell$: $f(k) = \mathsf{tower}(\ell, k + 3.92)$
   - . . . for $\#\exists\textsc{Sat}$: $f(k) = 2^{2^{k+3.59}}$

Contributions

1. Faster Structure-Guided Reasoning …
   - … for $\textsc{QSat}_\ell$: $f(k) = \text{tower}(\ell, k + 3.92)$
   - … for $\#\exists\textsc{Sat}$: $f(k) = 2^{2^{k+3.59}}$
2. A $\textsc{Sat}$ version of Courcelle's Theorem
   - $\text{MC}(\text{MSO}_\ell)$ to $\textsc{Sat}$ of size: $\text{tower}(\ell - 1, (k + 9)|\varphi| + 3.92)$
   - $\text{FD}(\text{MSO}_\ell)$ to $\textsc{MaxSat}$ of size: $\text{tower}(\ell, (k + 9)|\varphi| + 3.92)$
   - $\#\text{FD}(\text{MSO}_\ell)$ to $\#\textsc{Sat}$ of size: $\text{tower}(\ell, (k + 9)|\varphi| + 3.92)$

Contributions

1. Faster Structure-Guided Reasoning ...
   - ... for $\textsc{QSat}_\ell$: $f(k) = \text{tower}(\ell, k + 3.92)$
   - ... for $\#\exists\textsc{Sat}$: $f(k) = 2^{2^{k+3.59}}$
2. A $\textsc{Sat}$ version of Courcelle's Theorem
   - $\text{MC}(\text{MSO}_\ell)$ to $\textsc{Sat}$ of size: $\text{tower}(\ell - 1, (k + 9)|\varphi| + 3.92)$
   - $\text{FD}(\text{MSO}_\ell)$ to $\textsc{MaxSat}$ of size: $\text{tower}(\ell, (k + 9)|\varphi| + 3.92)$
   - $\#\text{FD}(\text{MSO}_\ell)$ to $\#\textsc{Sat}$ of size: $\text{tower}(\ell, (k + 9)|\varphi| + 3.92)$

$\rightsquigarrow$ Implies translations to ILP

Contributions

1. Faster Structure-Guided Reasoning ...
   - ... for QSAT_ℓ: $f(k) = \text{tower}(\ell, k + 3.92)$
   - ... for $\#\exists\text{SAT}$: $f(k) = 2^{2^{k+3.59}}$

2. A SAT version of Courcelle's Theorem
   - $\text{MC}(\text{MSO}_\ell)$ to SAT of size: $\text{tower}(\ell - 1, (k + 9)|\varphi| + 3.92)$
   - $\text{FD}(\text{MSO}_\ell)$ to MAXSAT of size: $\text{tower}(\ell, (k + 9)|\varphi| + 3.92)$
   - $\#\text{FD}(\text{MSO}_\ell)$ to #SAT of size: $\text{tower}(\ell, (k + 9)|\varphi| + 3.92)$

   $\rightsquigarrow$ Implies translations to ILP

3. ETH Lower Bounds on Encoding Size ...
   - excludes SAT translations of size: $\text{tower}(\ell - 3, o(k))$

Contributions

1. Faster Structure-Guided Reasoning ...
   - ... for $\textsc{QSat}_\ell$: $f(k) = \text{tower}(\ell, k + 3.92)$
   - ... for $\#\exists\textsc{Sat}$: $f(k) = 2^{2^{k+3.59}}$

2. A $\textsc{Sat}$ version of Courcelle's Theorem
   - $\text{MC}(\text{MSO}_\ell)$ to $\textsc{Sat}$ of size: $\text{tower}(\ell - 1, (k + 9)|\varphi| + 3.92)$
   - $\text{FD}(\text{MSO}_\ell)$ to $\textsc{MaxSat}$ of size: $\text{tower}(\ell, (k + 9)|\varphi| + 3.92)$
   - $\#\text{FD}(\text{MSO}_\ell)$ to $\#\textsc{Sat}$ of size: $\text{tower}(\ell, (k + 9)|\varphi| + 3.92)$

   $\rightsquigarrow$ Implies translations to ILP

3. ETH Lower Bounds on Encoding Size ...
   - excludes $\textsc{Sat}$ translations of size: $\text{tower}(\ell - 3, o(k))$
   - (Trade-Off:) excludes $\textsc{Sat}$ translations of size: $\text{tower}(\ell - 3, o(k \cdot \text{bs}(\varphi)))$

Contributions

1. Faster Structure-Guided Reasoning ...
   - ... for $\textsc{QSat}_\ell$: $f(k) = \text{tower}(\ell, k + 3.92)$
   - ... for $\#\exists\textsc{Sat}$: $f(k) = 2^{2^{k+3.59}}$

2. A $\textsc{Sat}$ version of Courcelle's Theorem
   - $\text{MC}(\text{MSO}_\ell)$ to $\textsc{Sat}$ of size: $\text{tower}(\ell - 1, (k + 9)|\varphi| + 3.92)$
   - $\text{FD}(\text{MSO}_\ell)$ to $\textsc{MaxSat}$ of size: $\text{tower}(\ell, (k + 9)|\varphi| + 3.92)$
   - $\#\text{FD}(\text{MSO}_\ell)$ to $\#\textsc{Sat}$ of size: $\text{tower}(\ell, (k + 9)|\varphi| + 3.92)$

   $\rightsquigarrow$ Implies translations to ILP

3. ETH Lower Bounds on Encoding Size ...
   - excludes $\textsc{Sat}$ translations of size: $\text{tower}(\ell - 3, o(k))$
   - (Trade-Off:) excludes $\textsc{Sat}$ translations of size: $\text{tower}(\ell - 3, o(k \cdot \text{bs}(\varphi)))$

     $\rightsquigarrow$ compress treewidth k to $\frac{k}{c}$; costing block size increase to $c \cdot \text{bs}(\varphi)$

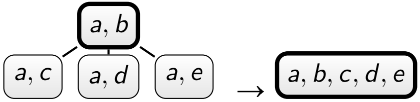
How Do We Translate To SAT?

Warmup: Eliminate QBF Quantifiers

Standard Translations	Structure-Aware Encoding
$(\textsc{QSat}, k)$ $\downarrow$ $(\textsc{Sat}, ???)$	

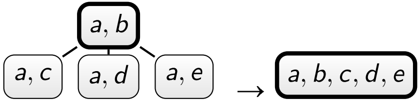
Warmup: Eliminate QBF Quantifiers

Standard Translations	Structure-Aware Encoding
$(\textsc{QSat}, k)$ $\downarrow$ $(\textsc{Sat}, ???)$	



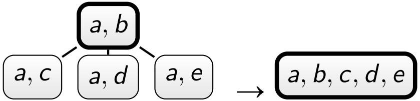
Warmup: Eliminate QBF Quantifiers

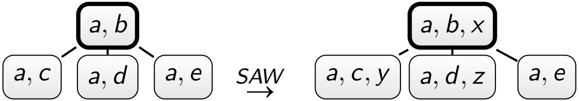
Standard Translations	Structure-Aware Encoding
$(\textsc{QSat}, k)$	$(\textsc{QSat}_\ell, k)$
$\downarrow$	$\downarrow_{SAW}$
$(\textsc{Sat}, ???)$	$(\textsc{QSat}_{\ell-1}, 2^k)$



Warmup: Eliminate QBF Quantifiers

Standard Translations	Structure-Aware Encoding
(QSAT, k) $\downarrow$ $(\text{SAT}, ???)$	(QSAT_{ℓ}, k) $\downarrow_{SAW}$ $(\text{QSAT}_{\ell-1}, 2^k)$





Structure-Aware Quantifier Block Elimination

Given: $\exists V_1 \; \forall V_2 \; \ldots \forall V_\ell \,.\, \psi$ (DNF), TD $\mathcal{T}$ of P_ψ,

Structure-Aware Quantifier Block Elimination

Given: $\exists V_1 \, \forall V_2 \, \ldots \forall V_\ell \, . \, \psi$ (DNF), TD $\mathcal{T}$ of P_ψ, labeling δ from ψ to nodes $\mathcal{T}$

Structure-Aware Quantifier Block Elimination

Given: $\exists V_1 \; \forall V_2 \; \ldots \forall V_\ell \, . \, \psi$ (DNF), TD $\mathcal{T}$ of P_ψ, labeling δ from ψ to nodes $\mathcal{T}$

$\rightsquigarrow$ Construct $\exists V_1 \forall V_2 \ldots \exists V_{\ell-1} \cup VSat \, . \, \varphi$ (CNF):

Structure-Aware Quantifier Block Elimination

Given: $\exists V_1 \; \forall V_2 \; \ldots \forall V_\ell \,.\, \psi$ (DNF), TD $\mathcal{T}$ of P_ψ, labeling δ from ψ to nodes $\mathcal{T}$

$\rightsquigarrow$ Construct $\exists V_1 \forall V_2 \ldots \exists V_{\ell-1} \cup VSat \,.\, \varphi$ (CNF):

$$\bigwedge_{d \in \psi}$$

Structure-Aware Quantifier Block Elimination

Given: $\exists V_1 \; \forall V_2 \; \ldots \forall V_\ell \, . \, \psi$ (DNF), TD $\mathcal{T}$ of P_ψ, labeling δ from ψ to nodes $\mathcal{T}$

⇝ Construct $\exists V_1 \forall V_2 \ldots \exists V_{\ell-1} \cup VSat \, . \, \varphi$ (CNF):

$$\bigwedge_{d \in \psi} \; \bigwedge_{\substack{\alpha \in 2^{\chi(\delta(d)) \cap V_\ell} \\ (d|\alpha) = \emptyset}}$$

Structure-Aware Quantifier Block Elimination

Given: $\exists V_1 \; \forall V_2 \; \ldots \forall V_\ell \, . \, \psi$ (DNF), TD $\mathcal{T}$ of P_ψ, labeling δ from ψ to nodes $\mathcal{T}$

$\rightsquigarrow$ Construct $\exists V_1 \forall V_2 \ldots \exists V_{\ell-1} \cup VSat \, . \, \varphi$ (CNF):

$$\bigwedge_{d \in \psi} \; \bigwedge_{\substack{\alpha \in 2^{\chi(\delta(d)) \cap V_\ell} \\ (d|\alpha) = \emptyset}} \left[\neg \mathrm{sat}_d^\alpha \right] \qquad (\alpha \text{ falsifies } d)$$

Structure-Aware Quantifier Block Elimination

Given: $\exists V_1 \; \forall V_2 \; \ldots \forall V_\ell \, . \, \psi$ (DNF), TD $\mathcal{T}$ of P_ψ, labeling δ from ψ to nodes $\mathcal{T}$

$\rightsquigarrow$ Construct $\exists V_1 \forall V_2 \ldots \exists V_{\ell-1} \cup VSat \, . \, \varphi$ (CNF):

$$\bigwedge_{d \in \psi} \; \bigwedge_{\substack{\alpha \in 2^{\chi(\delta(d)) \cap V_\ell} \\ (d|\alpha) = \emptyset}} \left[\neg \mathrm{sat}_d^\alpha \right] \qquad (\alpha \text{ falsifies } d)$$

$$\bigwedge_{d \in \psi} \; \bigwedge_{\substack{\alpha \in 2^{\chi(\delta(d)) \cap V_\ell} \\ (d|\alpha) \neq \emptyset}} \left[\mathrm{sat}_d^\alpha \leftrightarrow \bigwedge_{d' \in (d|\alpha), l \in d'} l \right] \qquad (\alpha \text{ may satisfy } d)$$

Structure-Aware Quantifier Block Elimination

Given: $\exists V_1 \; \forall V_2 \; \ldots \forall V_\ell \,.\, \psi$ (DNF), TD $\mathcal{T}$ of P_ψ, labeling δ from ψ to nodes $\mathcal{T}$

$\rightsquigarrow$ Construct $\exists V_1 \forall V_2 \ldots \exists V_{\ell-1} \cup VSat \,.\, \varphi$ (CNF):

$$\bigwedge_{d \in \psi} \; \bigwedge_{\substack{\alpha \in 2^{\chi(\delta(d)) \cap V_\ell} \\ (d|\alpha) = \emptyset}} \left[\neg \mathrm{sat}^\alpha_d \right] \qquad (\alpha \text{ falsifies } d)$$

$$\bigwedge_{d \in \psi} \; \bigwedge_{\substack{\alpha \in 2^{\chi(\delta(d)) \cap V_\ell} \\ (d|\alpha) \neq \emptyset}} \left[\mathrm{sat}^\alpha_d \leftrightarrow \bigwedge_{d' \in (d|\alpha), l \in d'} l \right] \qquad (\alpha \text{ may satisfy } d)$$

$$\bigwedge_{t \in \mathcal{T}}$$

Structure-Aware Quantifier Block Elimination

Given: $\exists V_1 \, \forall V_2 \, \ldots \forall V_\ell \, . \, \psi$ (DNF), TD $\mathcal{T}$ of P_ψ, labeling δ from ψ to nodes $\mathcal{T}$

$\leadsto$ Construct $\exists V_1 \forall V_2 \ldots \exists V_{\ell-1} \cup VSat \, . \, \varphi$ (CNF):

$$\bigwedge_{d \in \psi} \; \bigwedge_{\substack{\alpha \in 2^{\chi(\delta(d)) \cap V_\ell} \\ (d|\alpha) = \emptyset}} \left[\neg \mathrm{sat}_d^\alpha \right] \qquad (\alpha \text{ falsifies } d)$$

$$\bigwedge_{d \in \psi} \; \bigwedge_{\substack{\alpha \in 2^{\chi(\delta(d)) \cap V_\ell} \\ (d|\alpha) \neq \emptyset}} \left[\mathrm{sat}_d^\alpha \leftrightarrow \bigwedge_{d' \in (d|\alpha), l \in d'} l \right] \qquad (\alpha \text{ may satisfy } d)$$

$$\bigwedge_{t \in \mathcal{T}} \; \bigwedge_{\alpha \in 2^{\chi(t) \cap V_\ell}}$$

Structure-Aware Quantifier Block Elimination

Given: $\exists V_1\ \forall V_2\ \ldots \forall V_\ell\,.\,\psi$ (DNF), TD $\mathcal{T}$ of P_ψ, labeling δ from ψ to nodes $\mathcal{T}$

$\rightsquigarrow$ Construct $\exists V_1 \forall V_2 \ldots \exists V_{\ell-1} \cup VSat\,.\,\varphi$ (CNF):

$$\bigwedge_{d \in \psi} \bigwedge_{\substack{\alpha \in 2^{\chi(\delta(d)) \cap V_\ell} \\ (d|\alpha) = \emptyset}} \left[\neg \mathrm{sat}_d^\alpha\right] \qquad (\alpha \text{ falsifies } d)$$

$$\bigwedge_{d \in \psi} \bigwedge_{\substack{\alpha \in 2^{\chi(\delta(d)) \cap V_\ell} \\ (d|\alpha) \neq \emptyset}} \left[\mathrm{sat}_d^\alpha \leftrightarrow \bigwedge_{d' \in (d|\alpha), l \in d'} l\right] \qquad (\alpha \text{ may satisfy } d)$$

$$\bigwedge_{t \in \mathcal{T}} \bigwedge_{\alpha \in 2^{\chi(t) \cap V_\ell}} \left[\mathrm{sat}_{\leqslant t}^\alpha \leftrightarrow \bigvee_{d \in \delta^{-1}(t)} \mathrm{sat}_d^\alpha \vee \bigvee_{t' \in \mathrm{children}(t)} \mathrm{sat}_{\leqslant t', t}^\alpha\right] \qquad (\text{either satisfied term or propagate})$$

Structure-Aware Quantifier Block Elimination

Given: $\exists V_1 \ \forall V_2 \ \ldots \forall V_\ell \,.\, \psi$ (DNF), TD $\mathcal{T}$ of P_ψ, labeling δ from ψ to nodes $\mathcal{T}$

$\rightsquigarrow$ Construct $\exists V_1 \forall V_2 \ldots \exists V_{\ell-1} \cup VSat \,.\, \varphi$ (CNF):

$$\bigwedge_{d \in \psi} \bigwedge_{\substack{\alpha \in 2^{\chi(\delta(d)) \cap V_\ell} \\ (d|\alpha) = \emptyset}} \left[\neg \mathrm{sat}_d^\alpha \right] \qquad (\alpha \text{ falsifies } d)$$

$$\bigwedge_{d \in \psi} \bigwedge_{\substack{\alpha \in 2^{\chi(\delta(d)) \cap V_\ell} \\ (d|\alpha) \neq \emptyset}} \left[\mathrm{sat}_d^\alpha \leftrightarrow \bigwedge_{d' \in (d|\alpha), l \in d'} l \right] \qquad (\alpha \text{ may satisfy } d)$$

$$\bigwedge_{t \in \mathcal{T}} \bigwedge_{\alpha \in 2^{\chi(t) \cap V_\ell}} \left[\mathrm{sat}_{\leqslant t}^\alpha \leftrightarrow \bigvee_{d \in \delta^{-1}(t)} \mathrm{sat}_d^\alpha \vee \bigvee_{t' \in \mathrm{children}(t)} \mathrm{sat}_{\leqslant t', t}^\alpha \right] \qquad (\text{either satisfied term or propagate})$$

$$\bigwedge_{t \in \mathcal{T}} \bigwedge_{\alpha \in 2^{\chi(t) \cap V_\ell}} \bigwedge_{t' \in \mathrm{children}(t)}$$

Structure-Aware Quantifier Block Elimination

Given: $\exists V_1 \; \forall V_2 \; \ldots \forall V_\ell \,.\, \psi$ (DNF), TD $\mathcal{T}$ of P_ψ, labeling δ from ψ to nodes $\mathcal{T}$

$\rightsquigarrow$ Construct $\exists V_1 \forall V_2 \ldots \exists V_{\ell-1} \cup VSat \,.\, \varphi$ (CNF):

$$\bigwedge_{d \in \psi} \; \bigwedge_{\substack{\alpha \in 2^{\chi(\delta(d)) \cap V_\ell} \\ (d|\alpha) = \emptyset}} \left[\neg \mathrm{sat}_d^\alpha \right] \qquad (\alpha \text{ falsifies } d)$$

$$\bigwedge_{d \in \psi} \; \bigwedge_{\substack{\alpha \in 2^{\chi(\delta(d)) \cap V_\ell} \\ (d|\alpha) \neq \emptyset}} \left[\mathrm{sat}_d^\alpha \leftrightarrow \bigwedge_{d' \in (d|\alpha), l \in d'} l \right] \qquad (\alpha \text{ may satisfy } d)$$

$$\bigwedge_{t \in \mathcal{T}} \; \bigwedge_{\alpha \in 2^{\chi(t) \cap V_\ell}} \left[\mathrm{sat}_{\leqslant t}^\alpha \leftrightarrow \bigvee_{d \in \delta^{-1}(t)} \mathrm{sat}_d^\alpha \vee \bigvee_{t' \in \mathrm{children}(t)} \mathrm{sat}_{\leqslant t', t}^\alpha \right] \qquad (\text{either satisfied term or propagate})$$

$$\bigwedge_{t \in \mathcal{T}} \; \bigwedge_{\alpha \in 2^{\chi(t) \cap V_\ell}} \; \bigwedge_{t' \in \mathrm{children}(t)} \left[\mathrm{sat}_{\leqslant t', t}^\alpha \leftrightarrow \bigwedge_{\substack{\beta \in 2^{\chi(t') \cap V_\ell} \\ \beta \cap \chi(t) = \alpha \cap \chi(t')}} \mathrm{sat}_{\leqslant t'}^\beta \right] \qquad (\text{propagate compatible satisfiability})$$

Structure-Aware Quantifier Block Elimination

Given: $\exists V_1\ \forall V_2\ \ldots \forall V_\ell\,.\,\psi$ (DNF), TD $\mathcal{T}$ of P_ψ, labeling δ from ψ to nodes $\mathcal{T}$

$\rightsquigarrow$ Construct $\exists V_1 \forall V_2 \ldots \exists V_{\ell-1} \cup VSat\,.\,\varphi$ (CNF):

$$\bigwedge_{d \in \psi} \bigwedge_{\substack{\alpha \in 2^{\chi(\delta(d)) \cap V_\ell} \\ (d|\alpha) = \emptyset}} \left[\neg \mathrm{sat}_d^\alpha \right]$$ (α falsifies d)

$$\bigwedge_{d \in \psi} \bigwedge_{\substack{\alpha \in 2^{\chi(\delta(d)) \cap V_\ell} \\ (d|\alpha) \neq \emptyset}} \left[\mathrm{sat}_d^\alpha \leftrightarrow \bigwedge_{d' \in (d|\alpha), l \in d'} l \right]$$ (α may satisfy d)

$$\bigwedge_{t \in \mathcal{T}} \bigwedge_{\alpha \in 2^{\chi(t) \cap V_\ell}} \left[\mathrm{sat}_{\leqslant t}^\alpha \leftrightarrow \bigvee_{d \in \delta^{-1}(t)} \mathrm{sat}_d^\alpha \vee \bigvee_{t' \in \mathrm{children}(t)} \mathrm{sat}_{\leqslant t', t}^\alpha \right]$$ (either satisfied term or propagate)

$$\bigwedge_{t \in \mathcal{T}} \bigwedge_{\alpha \in 2^{\chi(t) \cap V_\ell}} \bigwedge_{t' \in \mathrm{children}(t)} \left[\mathrm{sat}_{\leqslant t', t}^\alpha \leftrightarrow \bigwedge_{\substack{\beta \in 2^{\chi(t') \cap V_\ell} \\ \beta \cap \chi(t) = \alpha \cap \chi(t')}} \mathrm{sat}_{\leqslant t'}^\beta \right]$$ (propagate compatible satisfiability)

$$\bigwedge_{\alpha \in 2^{\chi(\mathrm{root}(\mathcal{T})) \cap V_\ell}} \mathrm{sat}_{\leqslant \mathrm{root}(\mathcal{T})}^\alpha$$ (root assignments are satisfied)

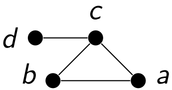
Example: $\text{QS}\textsc{at}_2$

$$\psi = \exists a, b\, \forall c, d \,.\, (\neg a \wedge \neg b) \vee (a \wedge b \wedge \neg c) \vee (c \wedge \neg d)$$

Structure-Aware Quantifier Block Elimination

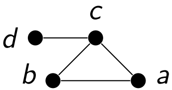
Example: $\textsc{QSat}_2$

$$\psi = \exists a, b\, \forall c, d\,.\, (\neg a \wedge \neg b) \vee (a \wedge b \wedge \neg c) \vee (c \wedge \neg d)$$

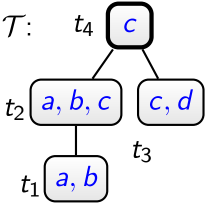


Example: QSAT_2

$$\psi = \exists a, b\ \forall c, d\ .\ (\neg a \wedge \neg b) \vee (a \wedge b \wedge \neg c) \vee (c \wedge \neg d)$$



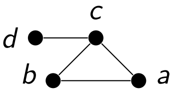
$\mathcal{T}$:

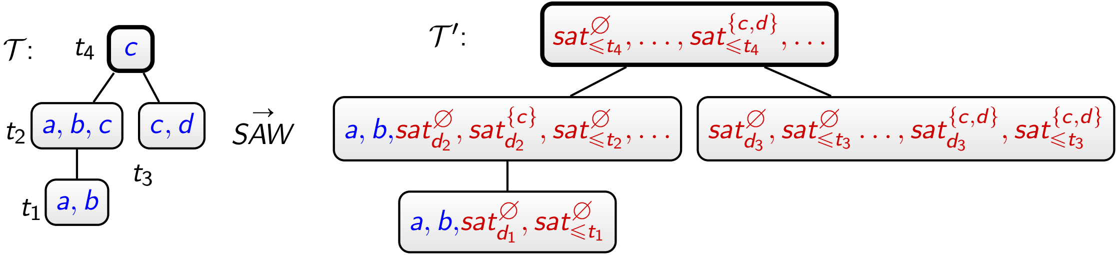


$\overrightarrow{SAW}$

Example: QSAT$_2$

$$\psi = \exists a, b\, \forall c, d\,.\, (\neg a \wedge \neg b) \vee (a \wedge b \wedge \neg c) \vee (c \wedge \neg d)$$





Structure-Aware Quantifier Block Elimination

- works similarly for eliminating $\exists$ block

Structure-Aware Quantifier Block Elimination

- works similarly for eliminating $\exists$ block

$\rightsquigarrow$ direct translation of QSAT to SAT (pure logic)

Structure-Aware Quantifier Block Elimination

- works similarly for eliminating $\exists$ block

$\rightsquigarrow$ direct translation of QSat to Sat (pure logic)

  - perfectly in line with known lower bounds (ETH) [Fichte, H & Pfandler 20]

Structure-Aware Quantifier Block Elimination

- works similarly for eliminating $\exists$ block

$\rightsquigarrow$ direct translation of QSAT to SAT (pure logic)

  - perfectly in line with known lower bounds (ETH) [Fichte, H & Pfandler 20]

- What is missing for reducing MSO to QSAT?
  - Replace SO variables X by X_u for every $u \in U(\mathcal{S})$
  - Replace FO variables x by x_u for every $u \in U(\mathcal{S})$

Structure-Aware Quantifier Block Elimination

- works similarly for eliminating $\exists$ block

⇝ direct translation of QSAT to SAT (pure logic)
  - perfectly in line with known lower bounds (ETH) [Fichte, H & Pfandler 20]

- What is missing for reducing MSO to QSAT?
  - Replace SO variables X by X_u for every $u \in U(\mathcal{S})$
  - Replace FO variables x by x_u for every $u \in U(\mathcal{S})$

  - Remaining issues
    1. FO variables require unique assignment

Structure-Aware Quantifier Block Elimination

- works similarly for eliminating ∃ block

⇝ direct translation of QSAT to SAT (pure logic)
  - perfectly in line with known lower bounds (ETH) [Fichte, H & Pfandler 20]

- What is missing for reducing MSO to QSAT?
  - Replace SO variables X by X_u for every $u \in U(\mathcal{S})$
  - Replace FO variables x by x_u for every $u \in U(\mathcal{S})$
  - Remaining issues
    1. FO variables require unique assignment
    2. We need to evaluate the actual MSO expressions!

② How do we evaluate MSO Expressions?

- Different types of MSO atoms
  - $x = y$ for FO variables x, y
  - X_x for SO variable X, FO variable x
  - $R_{x_1 \ldots x_a}$ for relation R in $\mathcal{S}$, FO variables $x_1, \ldots, x_a$

② How do we evaluate MSO Expressions?

- Different types of MSO atoms
  - $x = y$ for FO variables x, y
  - X_x for SO variable X, FO variable x
  - $R_{x_1 \ldots x_a}$ for relation R in $\mathcal{S}$, FO variables $x_1, \ldots, x_a$
- How do we process atoms in the SAT way?

② How do we evaluate MSO Expressions?

- Different types of MSO atoms
  - $x = y$ for FO variables x, y
  - X_x for SO variable X, FO variable x
  - $R_{x_1 \ldots x_a}$ for relation R in $\mathcal{S}$, FO variables $x_1, \ldots, x_a$
- How do we process atoms in the SAT way?

$\rightsquigarrow$ Replace by fresh propositional variables!

② How do we evaluate MSO Expressions?

- Different types of MSO atoms
  - $x = y$ for FO variables x, y
  - X_x for SO variable X, FO variable x
  - $R_{x_1 \ldots x_a}$ for relation R in $\mathcal{S}$, FO variables $x_1, \ldots, x_a$
- How do we process atoms in the SAT way?

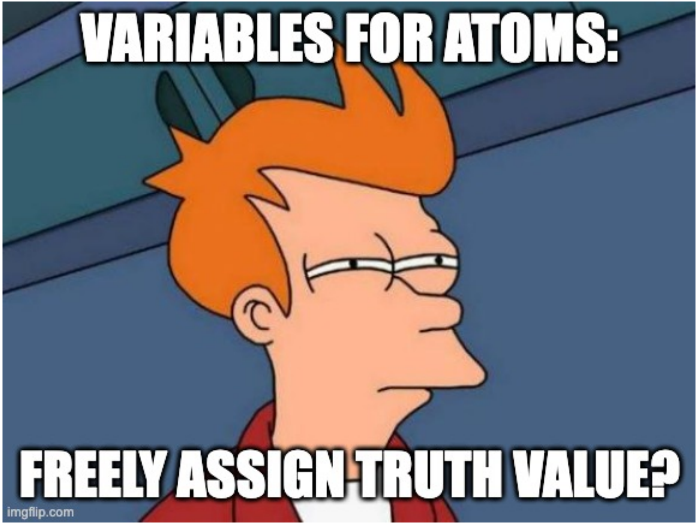
  $\rightsquigarrow$ Replace by fresh propositional variables!
- What if the same atom appears more than once?

② How do we evaluate MSO Expressions?

- Different types of MSO atoms
  - $x = y$ for FO variables x, y
  - X_x for SO variable X, FO variable x
  - $R_{x_1 \ldots x_a}$ for relation R in $\mathcal{S}$, FO variables $x_1, \ldots, x_a$
- How do we process atoms in the SAT way?

  ⇝ Replace by fresh propositional variables!
- What if the same atom appears more than once?

  ⇝ Not an issue; we assume prenex form


VARIABLES FOR ATOMS:
FREELY ASSIGN TRUTH VALUE?
imgflip.com

② How do we evaluate MSO Expressions?

Given: MSO formula φ, structure $\mathcal{S}$, tree decomposition $\mathcal{T}$ of $P_{\mathcal{S}}$

② How do we evaluate MSO Expressions?

Given: MSO formula φ, structure $\mathcal{S}$, tree decomposition $\mathcal{T}$ of $P_{\mathcal{S}}$

$$\bigwedge_{t \in \mathcal{T}} \left[p_t^{x=y} \leftrightarrow \bigvee_{u \in \chi(t)} (x_u \wedge y_u) \right]$$

(provability of equality)

② How do we evaluate MSO Expressions?

Given: MSO formula φ, structure $\mathcal{S}$, tree decomposition $\mathcal{T}$ of $P_{\mathcal{S}}$

$$\bigwedge_{t \in \mathcal{T}} \left[p_t^{x=y} \leftrightarrow \bigvee_{u \in \chi(t)} (x_u \wedge y_u) \right] \quad \text{(provability of equality)}$$

$$\bigwedge_{t \in \mathcal{T}} \left[p_t^{X_x} \leftrightarrow \bigvee_{u \in \chi(t)} (X_u \wedge x_u) \right] \quad \text{(provability of set membership)}$$

② How do we evaluate MSO Expressions?

Given: MSO formula φ, structure $\mathcal{S}$, tree decomposition $\mathcal{T}$ of $P_{\mathcal{S}}$

$$\bigwedge_{t \in \mathcal{T}} \left[p_t^{x=y} \leftrightarrow \bigvee_{u \in \chi(t)} (x_u \wedge y_u) \right]$$ (provability of equality)

$$\bigwedge_{t \in \mathcal{T}} \left[p_t^{X_x} \leftrightarrow \bigvee_{u \in \chi(t)} (X_u \wedge x_u) \right]$$ (provability of set membership)

$$\bigwedge_{t \in \mathcal{T}} \left[p_t^{R_{x_1 \dots x_a}} \leftrightarrow \bigvee_{\substack{u_1, \dots, u_a \in \chi(t) \\ (u_1, \dots, u_a) \in R^{\mathcal{S}}}} \left((x_1)_{u_1} \wedge \dots \wedge (x_a)_{u_a} \right) \right]$$ (provability of relationship)

② How do we evaluate MSO Expressions?

Given: MSO formula φ, structure $\mathcal{S}$, tree decomposition $\mathcal{T}$ of $P_{\mathcal{S}}$

$$\bigwedge_{t \in \mathcal{T}} \Big[p_t^{x=y} \leftrightarrow \bigvee_{u \in \chi(t)} (x_u \wedge y_u) \Big]$$ (provability of equality)

$$\bigwedge_{t \in \mathcal{T}} \Big[p_t^{X_x} \leftrightarrow \bigvee_{u \in \chi(t)} (X_u \wedge x_u) \Big]$$ (provability of set membership)

$$\bigwedge_{t \in \mathcal{T}} \Big[p_t^{R_{x_1 \ldots x_a}} \leftrightarrow \bigvee_{\substack{u_1, \ldots, u_a \in \chi(t) \\ (u_1, \ldots, u_a) \in R^{\mathcal{S}}}} \big((x_1)_{u_1} \wedge \cdots \wedge (x_a)_{u_a}\big) \Big]$$ (provability of relationship)

$$\bigwedge_{t \in \mathcal{T}} \bigwedge_{\iota \in \mathrm{atoms}(\varphi)} \Big[p_{\leqslant t}^{\iota} \leftrightarrow (p_t^{\iota} \vee \bigvee_{t' \in \mathrm{children}(t)} p_{\leqslant t'}^{\iota}) \Big]$$ (propagate provability of atoms)

② How do we evaluate MSO Expressions?

Given: MSO formula φ, structure $\mathcal{S}$, tree decomposition $\mathcal{T}$ of $P_{\mathcal{S}}$

$$\bigwedge_{t \in \mathcal{T}} \Big[p_t^{x=y} \leftrightarrow \bigvee_{u \in \chi(t)} (x_u \wedge y_u) \Big]$$ (provability of equality)

$$\bigwedge_{t \in \mathcal{T}} \Big[p_t^{X_x} \leftrightarrow \bigvee_{u \in \chi(t)} (X_u \wedge x_u) \Big]$$ (provability of set membership)

$$\bigwedge_{t \in \mathcal{T}} \Big[p_t^{R_{x_1 \ldots x_a}} \leftrightarrow \bigvee_{\substack{u_1,\ldots,u_a \in \chi(t) \\ (u_1,\ldots,u_a) \in R^{\mathcal{S}}}} \big((x_1)_{u_1} \wedge \cdots \wedge (x_a)_{u_a}\big) \Big]$$ (provability of relationship)

$$\bigwedge_{t \in \mathcal{T}} \bigwedge_{\iota \in \mathrm{atoms}(\varphi)} \Big[p_{\leqslant t}^{\iota} \leftrightarrow (p_t^{\iota} \vee \bigvee_{t' \in \mathrm{children}(t)} p_{\leqslant t'}^{\iota}) \Big]$$ (propagate provability of atoms)

$$\bigwedge_{\iota \in \mathrm{atoms}(\varphi)} \Big[\iota \leftrightarrow p_{\leqslant \mathrm{root}(\mathcal{T})}^{\iota} \Big]$$ (MSO atoms are proven)

② How do we evaluate MSO Expressions?

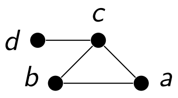
Example: 3-Coloring

$$\varphi_{\mathrm{3col}} = \exists R, G, B\, \forall x, y\,.\,(Rx \lor Gx \lor Bx) \quad \land$$
$$[Exy \to \neg((Rx \land Ry) \lor (Gx \land Gy) \lor (Bx \land By))]$$

② How do we evaluate MSO Expressions?

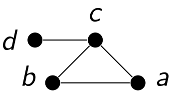
Example: 3-Coloring

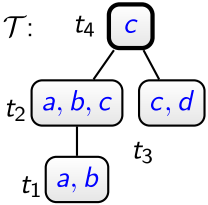
$$\varphi_{\mathrm{3col}} = \exists R, G, B\, \forall x, y\, . \,(Rx \vee Gx \vee Bx) \quad \wedge$$
$$[Exy \to \neg((Rx \wedge Ry) \vee (Gx \wedge Gy) \vee (Bx \wedge By))]$$



Example: 3-Coloring

$$\varphi_{3\text{col}} = \exists R, G, B\, \forall x, y\, .\, (Rx \vee Gx \vee Bx) \quad \wedge$$
$$[Exy \rightarrow \neg((Rx \wedge Ry) \vee (Gx \wedge Gy) \vee (Bx \wedge By))]$$



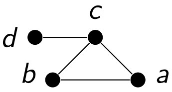


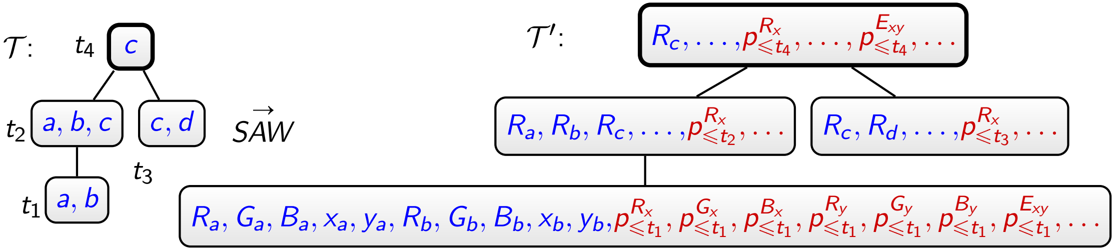
$\overrightarrow{SAW}$

② How do we evaluate MSO Expressions?

Example: 3-Coloring

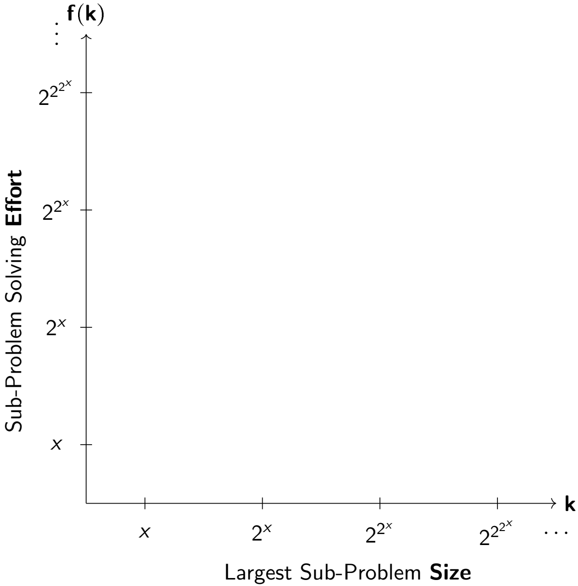
$$\varphi_{\mathrm{3col}} = \exists R, G, B\, \forall x, y\, .\, (Rx \vee Gx \vee Bx) \quad \wedge$$
$$[Exy \rightarrow \neg((Rx \wedge Ry) \vee (Gx \wedge Gy) \vee (Bx \wedge By))]$$



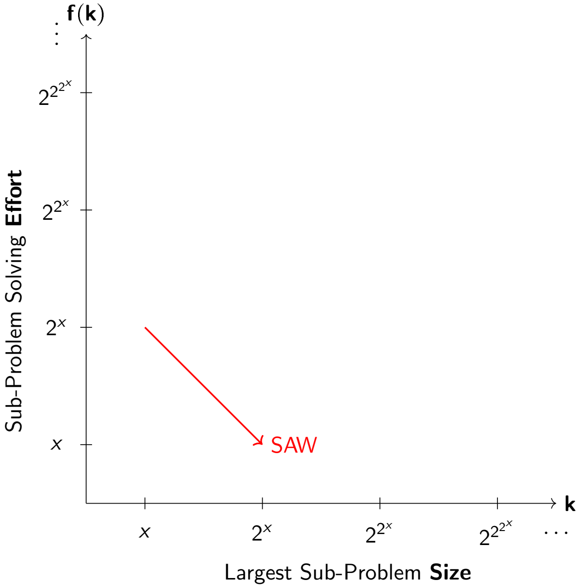


Outlook and Conclusion

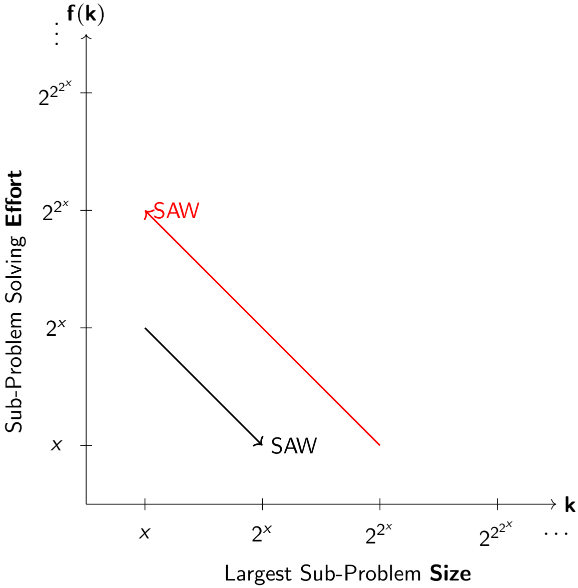
Time vs. Space, you say?



Time vs. Space, you say?



Time vs. Space, you say?



Conclusion & Future Work

Research Insights

- Precise bounds for encoding MSO to SAT

Conclusion & Future Work

Research Insights

- Precise bounds for encoding MSO to SAT
  - ⇝ Translation via SAW to QSAT: ① encoding FO cardinality and ② FO atom evaluation; then reducing to SAT

Conclusion & Future Work

Research Insights

- Precise bounds for encoding MSO to SAT
  - ⇝ Translation via SAW to QSAT: (1) encoding FO cardinality and (2) FO atom evaluation; then reducing to SAT
  - ⇝ Only depends on SAT algorithm; implies translation to ILP

Conclusion & Future Work

Research Insights

- Precise bounds for encoding MSO to SAT
  - ⇝ Translation via SAW to QSAT: (1) encoding FO cardinality and (2) FO atom evaluation; then reducing to SAT
  - ⇝ Only depends on SAT algorithm; implies translation to ILP
- In the paper: Lower bound via trading treewidth decrease for block size increase

Conclusion & Future Work

Research Insights

- Precise bounds for encoding MSO to SAT
  - ⇝ Translation via SAW to QSAT: (1) encoding FO cardinality and (2) FO atom evaluation; then reducing to SAT
  - ⇝ Only depends on SAT algorithm; implies translation to ILP
- In the paper: Lower bound via trading treewidth decrease for block size increase

Open Problems

- Does it work well in practice?

Conclusion & Future Work

Research Insights

- Precise bounds for encoding MSO to SAT
  - ⇝ Translation via SAW to QSAT: (1) encoding FO cardinality and (2) FO atom evaluation; then reducing to SAT
  - ⇝ Only depends on SAT algorithm; implies translation to ILP
- In the paper: Lower bound via trading treewidth decrease for block size increase

Open Problems

- Does it work well in practice?
- Can we close the gap to the lower bounds? Can we improve the dependency on formula size?

Conclusion & Future Work

Research Insights

- Precise bounds for encoding MSO to SAT
  - ⇝ Translation via SAW to QSAT: (1) encoding FO cardinality and (2) FO atom evaluation; then reducing to SAT
  - ⇝ Only depends on SAT algorithm; implies translation to ILP
- In the paper: Lower bound via trading treewidth decrease for block size increase

Open Problems

- Does it work well in practice?
- Can we close the gap to the lower bounds? Can we improve the dependency on formula size?
- What is the precise role of FO variables?
  - ⇝ Do we get tight bounds via "guarded" formulas?

Unique Assignment for FO Variables!

Given: $Q_1 V_1 Q_{q-1} V_{q-1} \ldots Q_q v_q \ldots Q_\ell v_\ell . \psi$, structure $\mathcal{S}$, TD $\mathcal{T}$ of $P_\mathcal{S}$

Unique Assignment for FO Variables!

Given: $Q_1 V_1 Q_{q-1} V_{q-1} \ldots Q_q v_q \ldots Q_\ell v_\ell \,.\, \psi$, structure $\mathcal{S}$, TD $\mathcal{T}$ of $P_\mathcal{S}$

$$\bigwedge_{t \in \mathcal{T}} \bigwedge_{x \in \{v_q, \ldots, v_\ell\}}$$

Unique Assignment for FO Variables!

Given: $Q_1 V_1 Q_{q-1} V_{q-1} \ldots Q_q v_q \ldots Q_\ell v_\ell \,.\, \psi$, structure $\mathcal{S}$, TD $\mathcal{T}$ of $P_{\mathcal{S}}$

$$\bigwedge_{t \in \mathcal{T}} \bigwedge_{x \in \{v_q, \ldots, v_\ell\}} [c^x_{\leqslant t} \leftrightarrow \bigvee_{u \in \chi(t) \setminus \chi(\text{parent}(t))} x_u \vee \bigvee_{t' \in \text{children}(t)} c^x_{\leqslant t'}]$$ (propagate cardinality $\geqslant 1$)

Unique Assignment for FO Variables!

Given: $Q_1 V_1 Q_{q-1} V_{q-1} \ldots Q_q v_q \ldots Q_\ell v_\ell . \psi$, structure $\mathcal{S}$, TD $\mathcal{T}$ of $P_\mathcal{S}$

$$\bigwedge_{t \in \mathcal{T}} \bigwedge_{x \in \{v_q, \ldots, v_\ell\}} \left[c^x_{\leqslant t} \leftrightarrow \bigvee_{u \in \chi(t) \setminus \chi(\mathrm{parent}(t))} x_u \vee \bigvee_{t' \in \mathrm{children}(t)} c^x_{\leqslant t'}\right]$$ (propagate cardinality $\geqslant 1$)

$$\bigwedge_{x \in \{v_q, \ldots, v_\ell\}} \left[c^x_{\leqslant \mathrm{root}(\mathcal{T})}\right]$$ (at least one)

Unique Assignment for FO Variables!

Given: $Q_1 V_1 Q_{q-1} V_{q-1} \dots Q_q v_q \dots Q_\ell v_\ell \,.\, \psi$, structure $\mathcal{S}$, TD $\mathcal{T}$ of $P_\mathcal{S}$

$$\bigwedge_{t \in \mathcal{T}} \bigwedge_{x \in \{v_q,\dots,v_\ell\}} [c^x_{\leqslant t} \leftrightarrow \bigvee_{u \in \chi(t) \setminus \chi(\text{parent}(t))} x_u \vee \bigvee_{t' \in \text{children}(t)} c^x_{\leqslant t'}] \quad \text{(propagate cardinality} \geqslant 1\text{)}$$

$$\bigwedge_{x \in \{v_q,\dots,v_\ell\}} [c^x_{\leqslant \text{root}(\mathcal{T})}] \quad \text{(at least one)}$$

$$\bigwedge_{t \in \mathcal{T}} \bigwedge_{\substack{u,u' \in \chi(t), u \neq u', \\ x \in \{v_q,\dots,v_\ell\}}} [\neg x_u \vee \neg x_{u'}] \quad \text{(at most one)}$$

Unique Assignment for FO Variables!

Given: $Q_1 V_1 Q_{q-1} V_{q-1} \ldots Q_q v_q \ldots Q_\ell v_\ell \,.\, \psi$, structure $\mathcal{S}$, TD $\mathcal{T}$ of $P_{\mathcal{S}}$

$$\bigwedge_{t \in \mathcal{T}} \bigwedge_{x \in \{v_q,\ldots,v_\ell\}} \left[c^x_{\leqslant t} \leftrightarrow \bigvee_{u \in \chi(t) \setminus \chi(\text{parent}(t))} x_u \vee \bigvee_{t' \in \text{children}(t)} c^x_{\leqslant t'}\right] \quad \text{(propagate cardinality} \geqslant 1\text{)}$$

$$\bigwedge_{x \in \{v_q,\ldots,v_\ell\}} \left[c^x_{\leqslant \text{root}(\mathcal{T})}\right] \quad \text{(at least one)}$$

$$\bigwedge_{t \in \mathcal{T}} \bigwedge_{\substack{u,u' \in \chi(t), u \neq u',\\ x \in \{v_q,\ldots,v_\ell\}}} \left[\neg x_u \vee \neg x_{u'}\right] \quad \text{(at most one)}$$

$$\bigwedge_{t \in \mathcal{T}} \bigwedge_{\substack{t' \in \text{children}(t), u \in \chi(t) \setminus \chi(\text{parent}(t)),\\ x \in \{v_q,\ldots,v_\ell\}}} \left[\neg x_u \vee \neg c^x_{\leqslant t'}\right] \quad \text{(at most one, second case)}$$

Unique Assignment for FO Variables!

Given: $Q_1 V_1 Q_{q-1} V_{q-1} \ldots Q_q v_q \ldots Q_\ell v_\ell \,.\, \psi$, structure $\mathcal{S}$, TD $\mathcal{T}$ of $P_\mathcal{S}$

$$\bigwedge_{t \in \mathcal{T}} \bigwedge_{x \in \{v_q, \ldots, v_\ell\}} \left[c^x_{\leqslant t} \leftrightarrow \bigvee_{u \in \chi(t) \setminus \chi(\text{parent}(t))} x_u \vee \bigvee_{t' \in \text{children}(t)} c^x_{\leqslant t'}\right] \quad \text{(propagate cardinality} \geqslant 1)$$

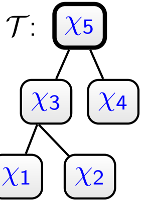
$$\bigwedge_{x \in \{v_q, \ldots, v_\ell\}} \left[c^x_{\leqslant \text{root}(\mathcal{T})}\right] \quad \text{(at least one)}$$

$$\bigwedge_{t \in \mathcal{T}} \bigwedge_{\substack{u, u' \in \chi(t), u \neq u', \\ x \in \{v_q, \ldots, v_\ell\}}} \left[\neg x_u \vee \neg x_{u'}\right] \quad \text{(at most one)}$$

$$\bigwedge_{t \in \mathcal{T}} \bigwedge_{\substack{t' \in \text{children}(t), u \in \chi(t) \setminus \chi(\text{parent}(t)), \\ x \in \{v_q, \ldots, v_\ell\}}} \left[\neg x_u \vee \neg c^x_{\leqslant t'}\right] \quad \text{(at most one, second case)}$$

$$\bigwedge_{t \in \mathcal{T}} \bigwedge_{\substack{t', t'' \in \text{children}(t), t' \neq t'', \\ x \in \{v_q, \ldots, v_\ell\}}} \left[\neg c^x_{\leqslant t'} \vee \neg c^x_{\leqslant t''}\right] \quad \text{(at most one, third case)}$$

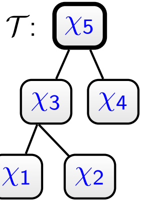
Key: Structure-Aware Encodings



$\overrightarrow{SAW}$

- Reduction to SAT, guided along a tree decomposition $\mathcal{T}$

Key: Structure-Aware Encodings

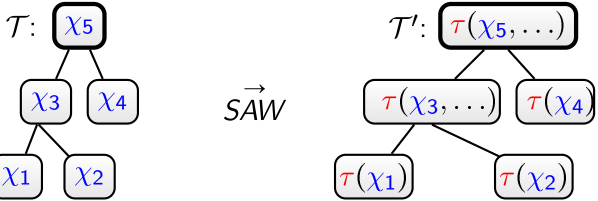


$\overrightarrow{SAW}$

- Reduction to SAT, guided along a tree decomposition $\mathcal{T}$

$\rightsquigarrow$ yields TD $\mathcal{T}'$ of SAT formula that functionally depends on $\mathcal{T}$

Key: Structure-Aware Encodings



- Reduction to SAT, guided along a tree decomposition $\mathcal{T}$

$\rightsquigarrow$ yields TD $\mathcal{T}'$ of SAT formula that functionally depends on $\mathcal{T}$