Proof Complexity using structured circuits

TU Ilmenau

Matthäus Micun

March 02, 2026

Joint work with Christoph Berkholz.

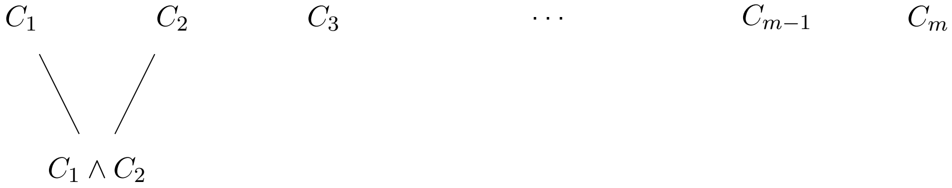
Introduction

Input: CNF $\varphi = \bigwedge_{i \in [m]} C_i$
Goal: Prove that φ is unsatisfiable

$C_1 \quad C_2 \quad C_3 \quad \cdots \quad C_{m-1} \quad C_m$

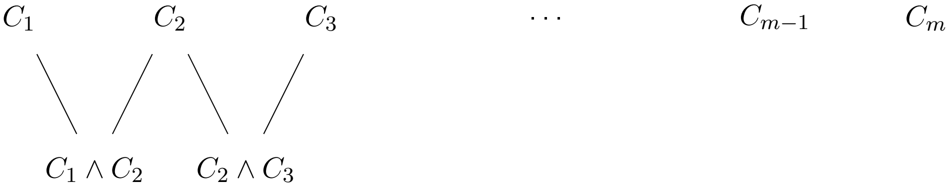
Introduction

Input: CNF $\varphi = \bigwedge_{i \in [m]} C_i$
Goal: Prove that φ is unsatisfiable



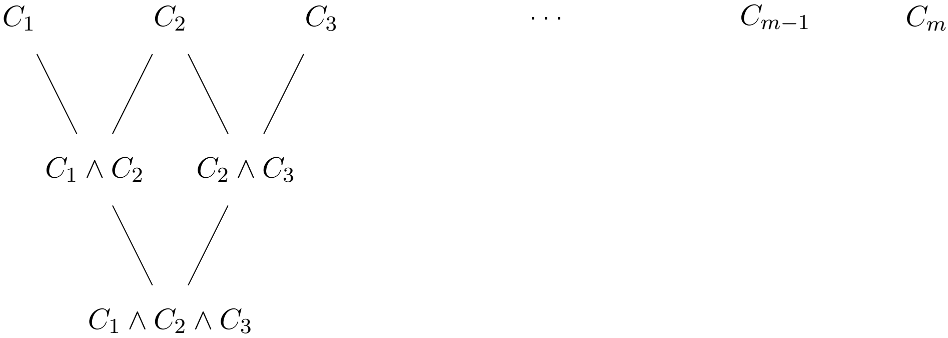
Introduction

Input: CNF $\varphi = \bigwedge_{i\in[m]} C_i$
Goal: Prove that φ is unsatisfiable



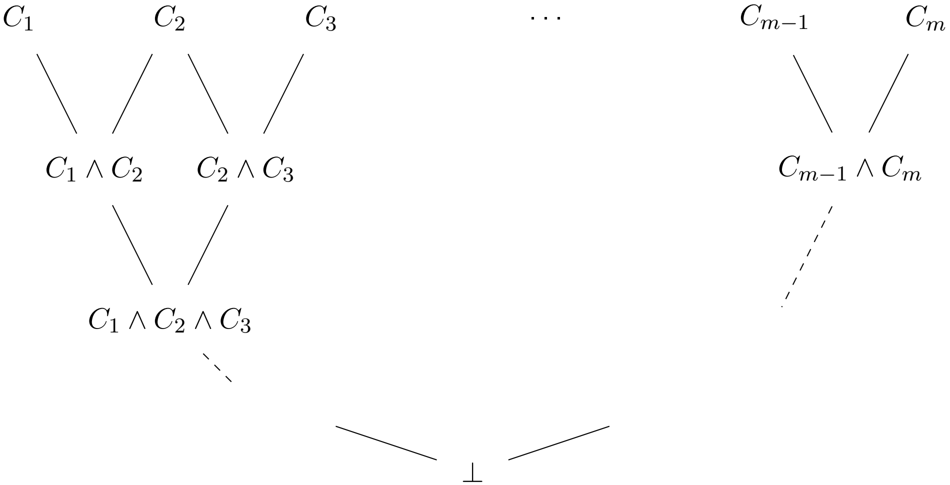
Introduction

Input: CNF $\varphi = \bigwedge_{i \in [m]} C_i$
Goal: Prove that φ is unsatisfiable



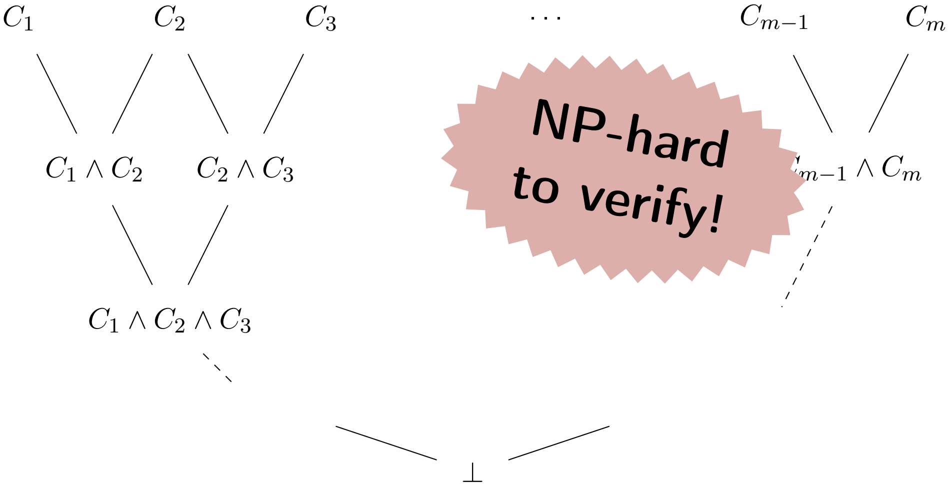
Introduction

Input: CNF $\varphi = \bigwedge_{i \in [m]} C_i$
Goal: Prove that φ is unsatisfiable



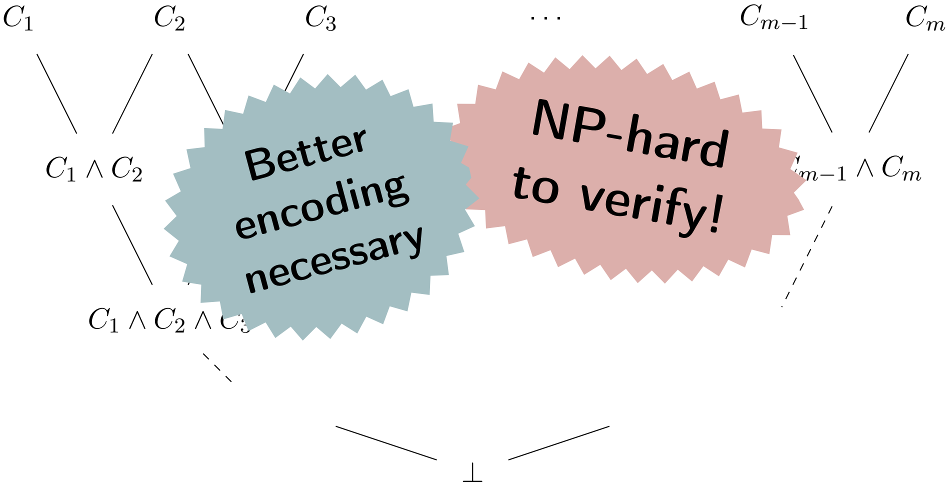
Introduction

Input: CNF $\varphi = \bigwedge_{i \in [m]} C_i$
Goal: Prove that φ is unsatisfiable



Introduction

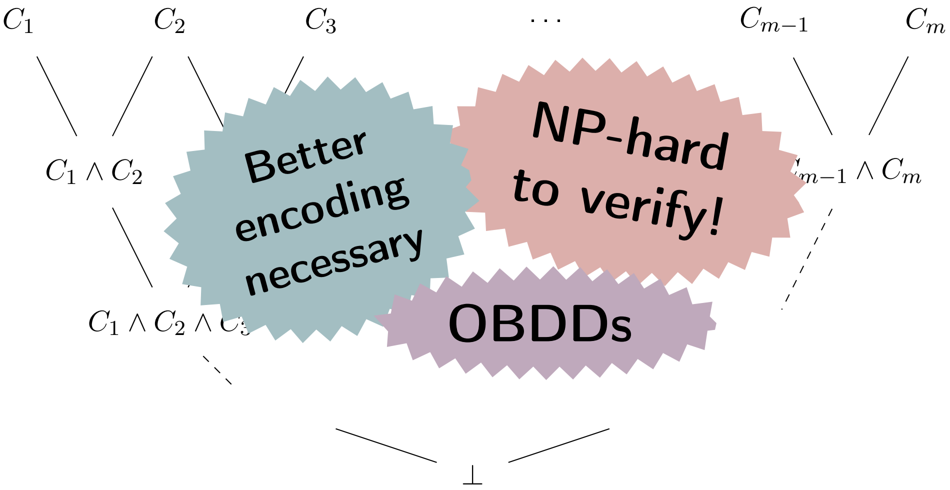
Input: CNF $\varphi = \bigwedge_{i \in [m]} C_i$
Goal: Prove that φ is unsatisfiable



Introduction

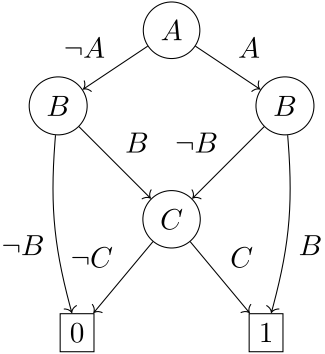
Input: CNF $\varphi = \bigwedge_{i\in[m]} C_i$

Goal: Prove that φ is unsatisfiable



Ordered Binary Decision Diagram

Ordered Binary Decision Diagram (OBDD):



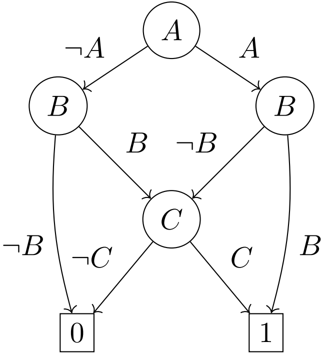
An OBDD for $(A \wedge B) \vee (A \wedge C) \vee (B \wedge C)$

Introduced in Bryant (1986)

Ordered Binary Decision Diagram

Ordered Binary Decision Diagram (OBDD):

- Support polynomial satisfiability checking
- Support polynomial conjunction
  - → size of $D_1 \wedge D_2$ is in $\mathcal{O}(|D_1| \cdot |D_2|)$
  - → both OBDDs must respect the same order
- Support polynomial implication
  - → Given D_1, D_2 does $D_1 \models D_2$ hold?
  - → both OBDDs must respect the same order



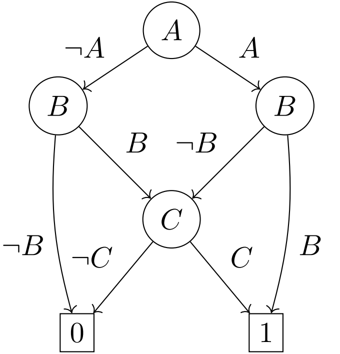
An OBDD for $(A \wedge B) \vee (A \wedge C) \vee (B \wedge C)$

Introduced in Bryant (1986)

Ordered Binary Decision Diagram

Ordered Binary Decision Diagram (OBDD):

- Support polynomial satisfiability checking
- Support polynomial conjunction
  - $\rightarrow$ size of $D_1 \wedge D_2$ is in $\mathcal{O}(|D_1| \cdot |D_2|)$
  - $\rightarrow$ both OBDDs must respect the same order
- Support polynomial implication
  - $\rightarrow$ Given D_1, D_2 does $D_1 \models D_2$ hold?
  - $\rightarrow$ both OBDDs must respect the same order
- Downside: Many Boolean functions do not have small OBDDs



An OBDD for $(A \wedge B) \vee (A \wedge C) \vee (B \wedge C)$

Introduced in Bryant (1986)

OBDD-based proof systems

Proof as a sequence of OBDDs
Let $\varphi := \bigwedge_{i \in [m]} C_i$ be a CNF

- init: construct D_i equivalent to some clause C_j
- Join $(\wedge)$: Construct D from previous $D_1 \wedge D_2$
- Reordering (r): Construct D from previous D' by changing order
- Weakening (w): Construct D from previous D' such that $D' \models D$.

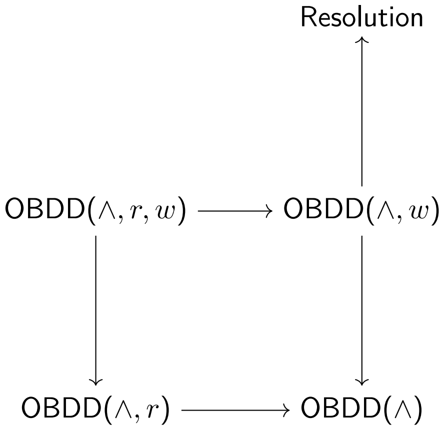
OBDD$(\wedge, w)$ introduced in Atsterias, Kolaitis, Vardi (2004)

OBDD-based proof systems

Proof as a sequence of OBDDs
Let $\varphi := \bigwedge_{i\in[m]} C_i$ be a CNF

- init: construct D_i equivalent to some clause C_j
- Join $(\wedge)$: Construct D from previous $D_1 \wedge D_2$
- Reordering (r): Construct D from previous D' by changing order
- Weakening (w): Construct D from previous D' such that $D' \models D$.

p-simulation: Every Proof in system A of length ℓ can be transformed into a proof in system B of length $p(\ell)$

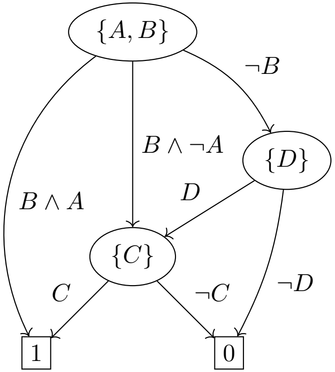


$\text{OBDD}(\wedge, w)$ introduced in Atsterias, Kolaitis, Vardi (2004)

Sentential Decision Diagrams

Sentential Decision Diagram (SDD):

- decide on multiple variables at once
  - decisions have to partition all possible assignments
  - Each decision represented as another SDD



An SDD for
$(A \land B) \lor (B \land C) \lor (C \land D)$

Introduced by Darwiche (2011)

Sentential Decision Diagrams

Sentential Decision Diagram (SDD):

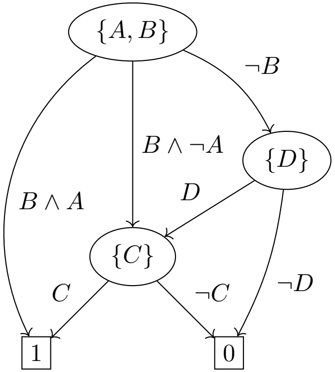
- decide on multiple variables at once
  - decisions have to partition all possible assignments
  - Each decision represented as another SDD

Can also be viewed as a circuit. Decisions take the form

$$\bigvee (f_i \wedge g_i)$$

such that

- each f_i and g_i are again SDDs deciding on disjoint variable sets X_f and X_g
- the models of all f_i form a partition of $\{0,1\}^{X_f}$



An SDD for
$(A \wedge B) \vee (B \wedge C) \vee (C \wedge D)$

Introduced by Darwiche (2011)

Sentential Decision Diagrams

Sentential Decision Diagram (SDD):

- decide on multiple variables at once
  - decisions have to partition all possible assignments
  - Each decision represented as another SDD

Can also be viewed as a circuit. Decisions take the form

$$\bigvee (f_i \wedge g_i)$$

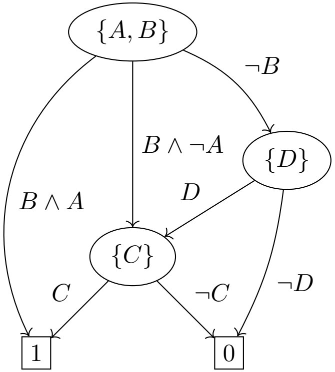
such that

- each f_i and g_i are again SDDs deciding on disjoint variable sets X_f and X_g
- the models of all f_i form a partition of $\{0,1\}^{X_f}$

Linear order not good enough for these decisions

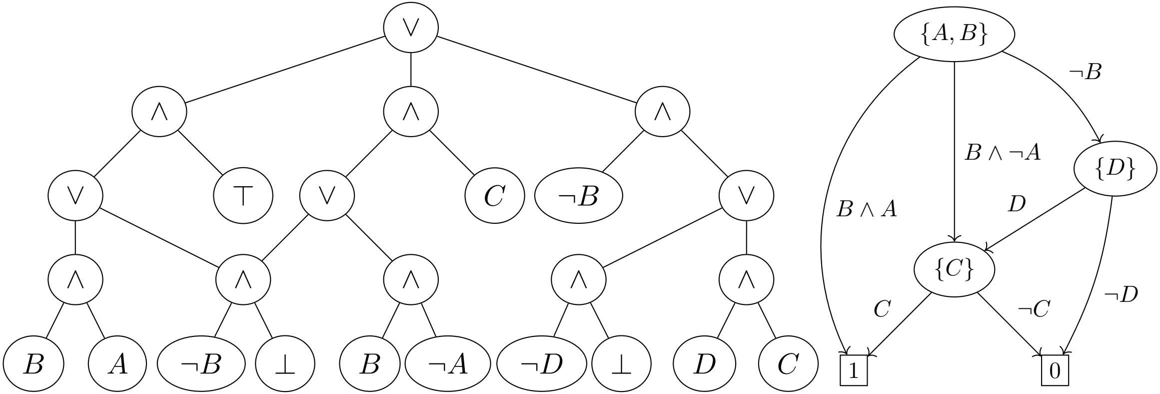
$\rightarrow$ Use tree structure intead!

Introduced by Darwiche (2011)



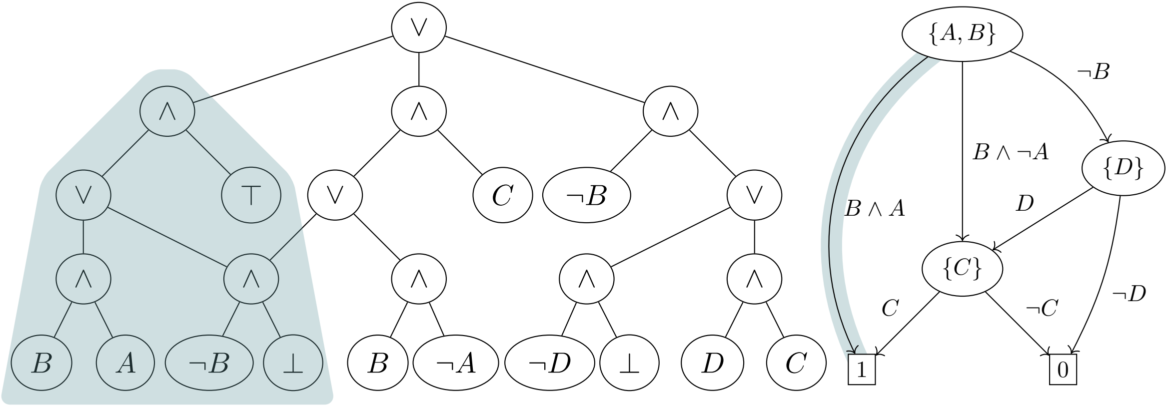
An SDD for
$(A \wedge B) \vee (B \wedge C) \vee (C \wedge D)$

Sentential Decision Diagrams



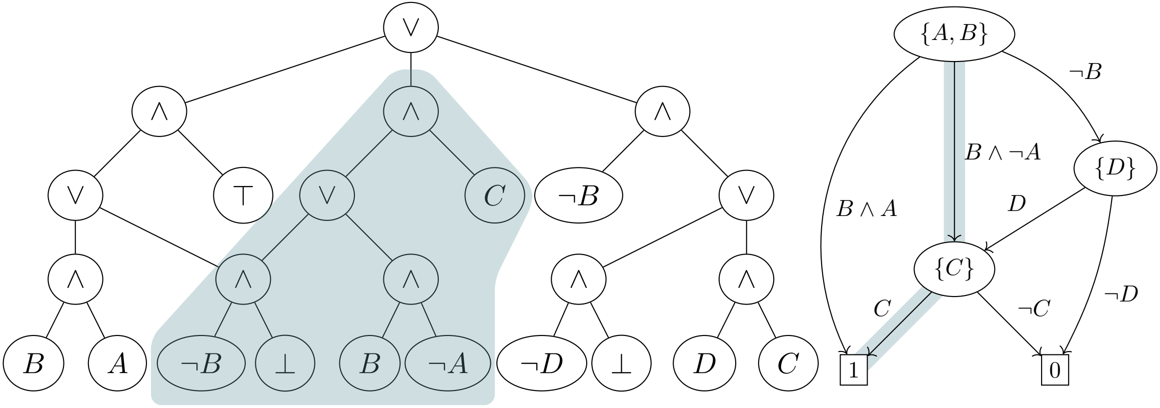
An SDD for $(A \wedge B) \vee (B \wedge C) \vee (C \wedge D)$

Sentential Decision Diagrams



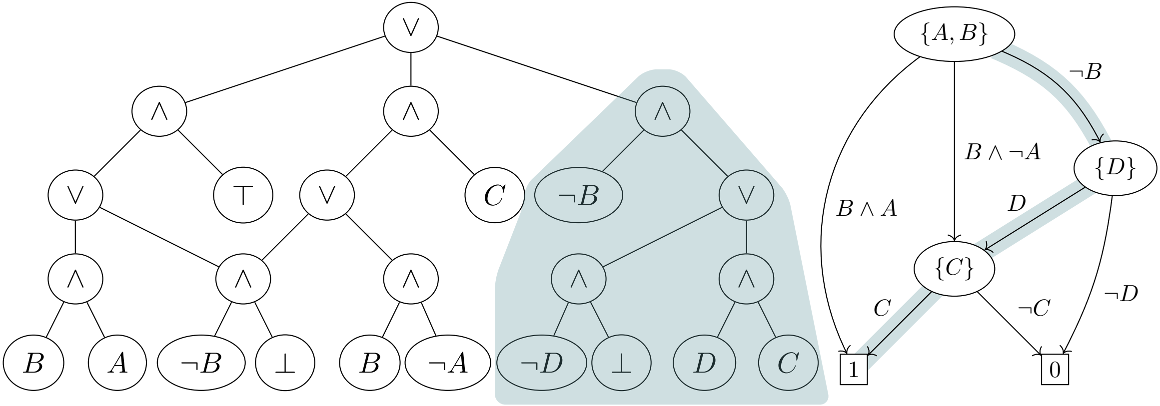
An SDD for $(A \wedge B) \vee (B \wedge C) \vee (C \wedge D)$

Sentential Decision Diagrams



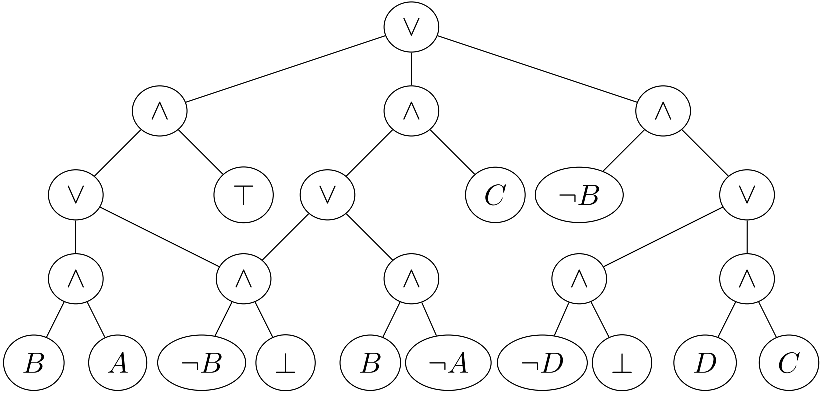
An SDD for $(A \wedge B) \vee (B \wedge C) \vee (C \wedge D)$

Sentential Decision Diagrams



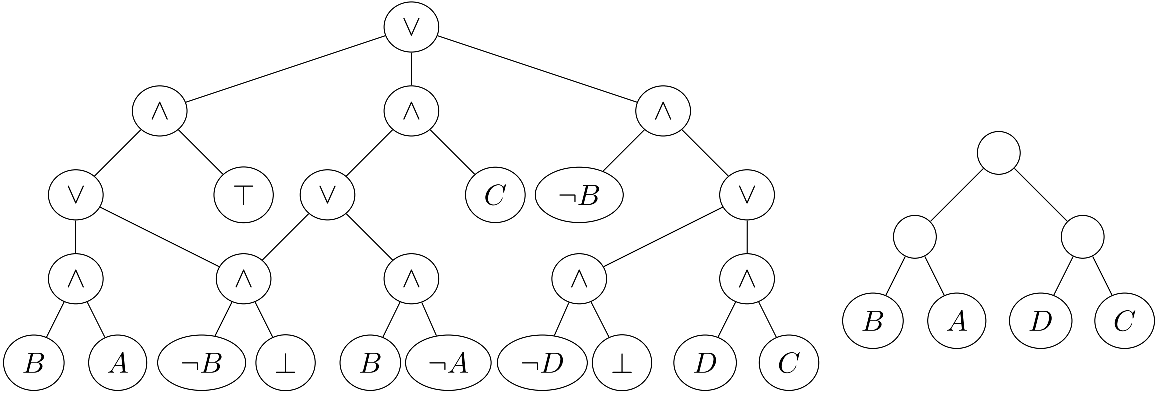
An SDD for $(A \land B) \lor (B \land C) \lor (C \land D)$

Sentential Decision Diagrams



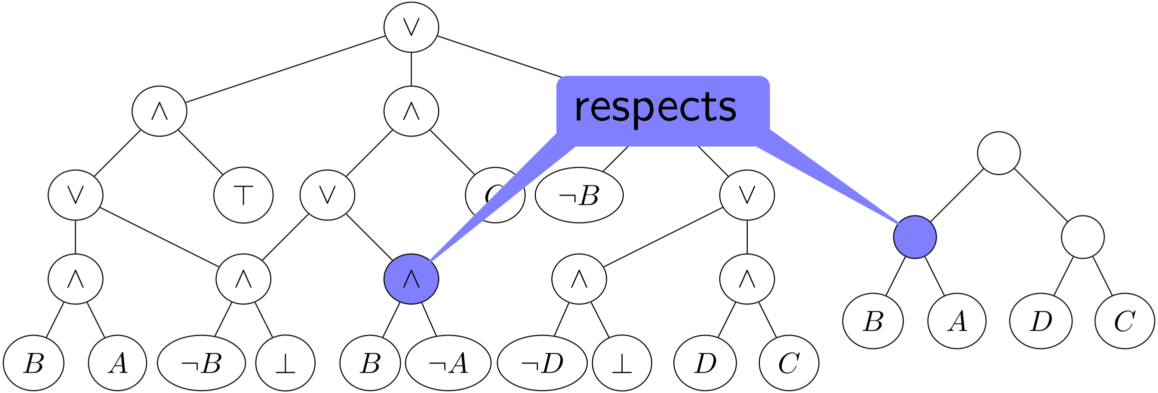
An SDD for $(A \wedge B) \vee (B \wedge C) \vee (C \wedge D)$

Sentential Decision Diagrams



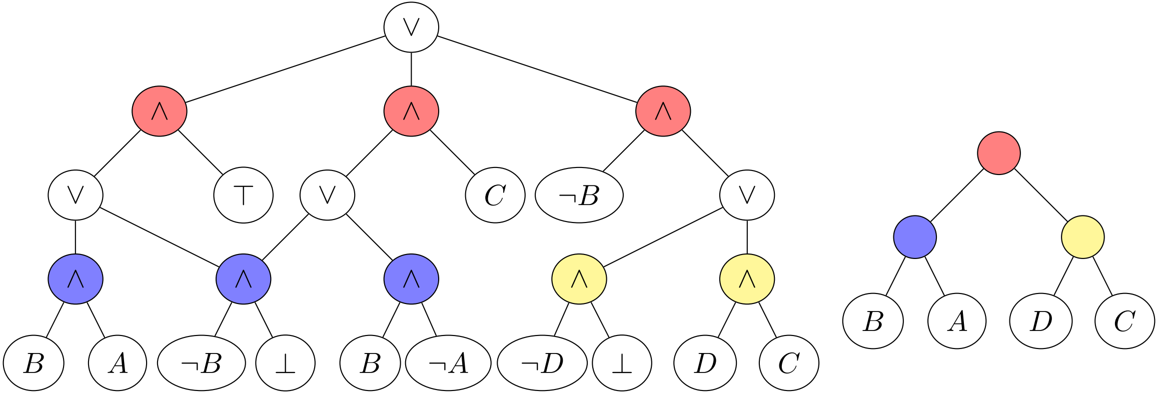
An SDD for $(A \wedge B) \vee (B \wedge C) \vee (C \wedge D)$

Sentential Decision Diagrams



An SDD for $(A \wedge B) \vee (B \wedge C) \vee (C \wedge D)$

Sentential Decision Diagrams

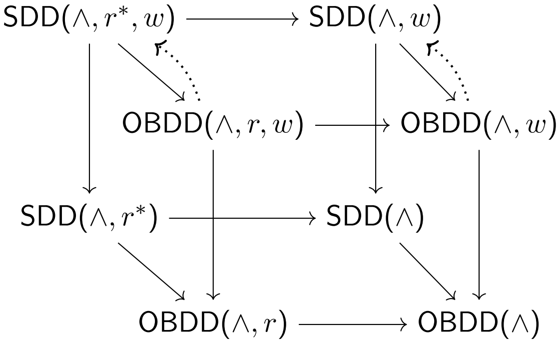


An SDD for $(A \wedge B) \vee (B \wedge C) \vee (C \wedge D)$

Our results

Support the the following polynomial-time operations

- Satisfiability checking
- Conjunction (respecting the same vtree)
- Weakening (respecting the same vtree)



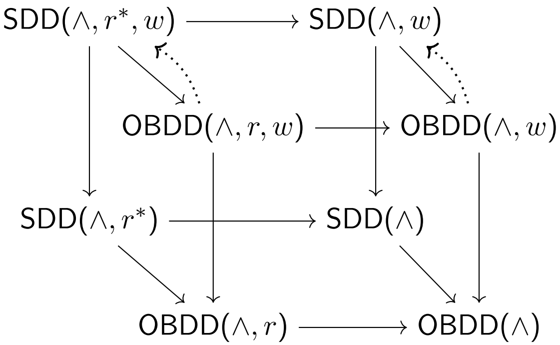
Our results

Support the the following polynomial-time operations

- Satisfiability checking
- Conjunction (respecting the same vtree)
- Weakening (respecting the same vtree)

Further advantages of SDDs

- Exponentially more succinct than OBDDs
- Can represent any CNF φ in size $\mathcal{O}(n \cdot 2^{\mathsf{tw}(\varphi)})$



Our results

Support the the following polynomial-time operations

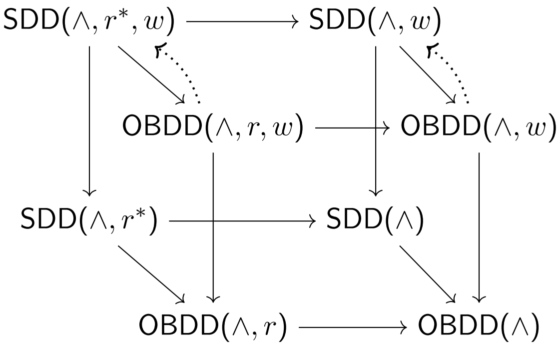
- Satisfiability checking
- Conjunction (respecting the same vtree)
- Weakening (respecting the same vtree)

Further advantages of SDDs

- Exponentially more succinct than OBDDs
- Can represent any CNF φ in size $\mathcal{O}(n \cdot 2^{\mathsf{tw}(\varphi)})$

Unknown whether SDDs support polynomial-time restructuring (r)

$\rightarrow$ Use restricted restructuring (r^*) instead



Roadmap

How do we find separations between our proof systems?

Roadmap

How do we find separations between our proof systems?

There are many well-known unsatisfiable CNFs

$\rightarrow$ Many do not separate OBDD from SDD

Roadmap

How do we find separations between our proof systems?

There are many well-known unsatisfiable CNFs
$\rightarrow$ Many do not separate OBDD from SDD

However: There are many superpolynomial separations for satisfiable formulas

- Lift knowledge compilation lower bounds to proof Complexity
- Make satisfiable CNFs unsatisfiable
- Ideally works on many CNFs
- Ideally works on many different representation formats

Making CNFs unsatisfiable

Definition ($\mathcal{Z}(\varphi)$)

Let $\varphi := \bigwedge_{i\in[m]} C_i$ be a CNF. We define

$$\mathcal{Z}(\varphi) = \varphi \wedge \bigwedge_{i\in[m]} (C_i \to (z_i \to z_{i+1})) \wedge z_1 \wedge \neg z_{m+1}$$

Making CNFs unsatisfiable

Definition ($\mathcal{Z}(\varphi)$)

Let $\varphi := \bigwedge_{i \in [m]} C_i$ be a CNF. We define

$$\begin{aligned}\mathcal{Z}(\varphi) &= \varphi \wedge \bigwedge_{i \in [m]} (C_i \rightarrow (z_i \rightarrow z_{i+1})) \wedge z_1 \wedge \neg z_{m+1} \\ &= \varphi \wedge \bigwedge_{i \in [m]} \bigwedge_{\lambda \in C_i} (\neg \lambda \vee \neg z_i \vee z_{i+1}) \wedge z_1 \wedge \neg z_{m+1}\end{aligned}$$

For every satisfiable φ the transformed $\mathcal{Z}(\varphi)$ is unsatisfiable

Making CNFs unsatisfiable

Definition ($\mathcal{Z}(\varphi)$)

Let $\varphi := \bigwedge_{i\in[m]} C_i$ be a CNF. We define

$$\begin{aligned}\mathcal{Z}(\varphi) &= \varphi \wedge \bigwedge_{i\in[m]} (C_i \to (z_i \to z_{i+1})) \wedge z_1 \wedge \neg z_{m+1} \\ &= \varphi \wedge \bigwedge_{i\in[m]} \bigwedge_{\lambda\in C_i} (\neg\lambda \vee \neg z_i \vee z_{i+1}) \wedge z_1 \wedge \neg z_{m+1}\end{aligned}$$

For every satisfiable φ the transformed $\mathcal{Z}(\varphi)$ is unsatisfiable

φ is called *reduced* if it does not contain unnecessary clauses or literals

$\to$ For every reduced CNF φ, the transformed $\mathcal{Z}(\varphi)$ is *minimally* unsatisfiable

Lower bounds for $\mathcal{Z}(\varphi)$

φ is a *monotone* (k, ℓ)-CNF, if

- Every literal only appears positively
- Every clause contains k literals
- Every variable appears in at most ℓ clauses

Let $G = (V, E)$ be a graph of maximal degree Δ. Then

$$\mathsf{VC}(G) = \bigwedge_{\{u,v\} \in E} (u \vee v)$$

is a monotone $(2, \Delta)$-CNF

Lower bounds for $\mathcal{Z}(\varphi)$

φ is a *monotone* (k, ℓ)-CNF, if

- Every literal only appears positively
- Every clause contains k literals
- Every variable appears in at most ℓ clauses

Looks simple, but may be very hard to represent

Let $G = (V, E)$ be a graph of maximal degree Δ. Then

$$\mathsf{VC}(G) = \bigwedge_{\{u,v\} \in E} (u \vee v)$$

is a monotone $(2, \Delta)$-CNF

Lower bounds for $\mathcal{Z}(\varphi)$

φ is a *monotone* (k, ℓ)-CNF, if

- Every literal only appears positively
- Every clause contains k literals
- Every variable appears in at most ℓ clauses

Looks simple, but may be very hard to represent

Let $G = (V, E)$ be a graph of maximal degree Δ. Then

$$\mathsf{VC}(G) = \bigwedge_{\{u,v\} \in E} (u \vee v)$$

is a monotone $(2, \Delta)$-CNF

Theorem

Let φ be a reduced monotone $(k, \sqrt{\log n})$-CNF, $\mathfrak{C} \in \{\text{OBDD}, \text{SDD}\}$, and $R_1, \ldots, R_\ell$ be $\mathfrak{C}(\wedge, r)$-refutation of $\mathcal{Z}(\varphi)$ of size $t(n)$.
Then there is a $\mathfrak{C}$-representation of φ of size $\mathcal{O}\left(t(n)^2 \cdot n^{k^2}\right)$.

Proof (Sketch)

Let $D_1 \wedge D_2 = \bot$ be the last step of the refutation
Goal: find partial assignments α_1, α_2 and a poly-sized D^* such that

$$D_1|_{\alpha_1} \wedge D_2|_{\alpha_2} \wedge D^* \equiv \varphi$$

Proof (Sketch)

Let $D_1 \land D_2 = \bot$ be the last step of the refutation
Goal: find partial assignments a_1, a_2 and a poly-sized D^* such that

$$D_1|_{a_1} \land D_2|_{a_2} \land D^* \equiv \varphi$$

→ Each D_i is missing a clause C_i

→ Because $\mathcal{Z}(\varphi)$ is minimally unsatisfiable, D_1 and D_2 contain all clauses of $\mathcal{Z}(\varphi)$

Proof (Sketch)

Let $D_1 \wedge D_2 = \bot$ be the last step of the refutation
Goal: find partial assignments α_1, α_2 and a poly-sized D^* such that

$$D_1|_{\alpha_1} \wedge D_2|_{\alpha_2} \wedge D^* \equiv \varphi$$

→ Each D_i is missing a clause C_i

→ Because $\mathcal{Z}(\varphi)$ is minimally unsatisfiable, D_1 and D_2 contain all clauses of $\mathcal{Z}(\varphi)$

Goal: define α_i such that $D_1|_{\alpha_1} \wedge D_2|_{\alpha_2}$ is a sub-CNF of φ

→ Do not leave "broken" clauses

→ Do not remove too many clauses

Proof (cont.)

Assume $C_1 \in \varphi$, and let $\mathscr{b}_1 \coloneqq \bot_{\mathrm{var}(C_1)}$.
Because φ is reduced, it holds that

- $D_1|_{\mathscr{b}_1}$ is satisfiable, but
- $\mathscr{b}_1 \models \neg C_1$

Therefore, $D_1|_{\mathscr{b}_1}$ "breaks" the implication chain at the clauses of $C_1 \rightarrow (z_i \rightarrow z_{i+1})$

$\bot_X : X \rightarrow \{0,1\}$
such that
$\bot_X(x) = 0$ for all $x \in X$

Proof (cont.)

Assume $C_1 \in \varphi$, and let $\mathscr{b}_1 \coloneqq \bot_{\mathrm{var}(C_1)}$.

Because φ is reduced, it holds that

- $D_1|_{\mathscr{b}_1}$ is satisfiable, but
- $\mathscr{b}_1 \models \neg C_1$

Therefore, $D_1|_{\mathscr{b}_1}$ "breaks" the implication chain at the clauses of $C_1 \to (z_i \to z_{i+1})$

$\bot_X : X \to \{0,1\}$
such that
$\bot_X(x) = 0$ for all $x \in X$

$$z_1 \longrightarrow z_2 \longrightarrow \cdots \longrightarrow z_i \longrightarrow z_{i+1} \longrightarrow \cdots \longrightarrow z_{m-1} \longrightarrow z_m$$

Proof (cont.)

Assume $C_1 \in \varphi$, and let $\mathscr{b}_1 \coloneqq \bot_{\text{var}(C_1)}$.
Because φ is reduced, it holds that

- $D_1|_{\mathscr{b}_1}$ is satisfiable, but
- $\mathscr{b}_1 \models \neg C_1$

Therefore, $D_1|_{\mathscr{b}_1}$ "breaks" the implication chain at the clauses of $C_1 \to (z_i \to z_{i+1})$

$\bot_X : X \to \{0, 1\}$
such that
$\bot_X(x) = 0$ for all $x \in X$

$$z_1 \longrightarrow z_2 \longrightarrow \cdots \longrightarrow z_i \longrightarrow z_{i+1} \longrightarrow \cdots \longrightarrow z_{m-1} \longrightarrow z_m$$

Proof (cont.)

Assume $C_1 \in \varphi$, and let $\mathscr{b}_1 := \bot_{\mathrm{var}(C_1)}$.
Because φ is reduced, it holds that

- $D_1|_{\mathscr{b}_1}$ is satisfiable, but
- $\mathscr{b}_1 \models \neg C_1$

Therefore, $D_1|_{\mathscr{b}_1}$ "breaks" the implication chain at the clauses of $C_1 \to (z_i \to z_{i+1})$

$\bot_X : X \to \{0,1\}$
such that
$\bot_X(x) = 0$ for all $x \in X$

$$z_1 \longrightarrow z_2 \longrightarrow \cdots \longrightarrow z_i \longrightarrow z_{i+1} \longrightarrow \cdots \longrightarrow z_{m-1} \longrightarrow z_m$$

This allows us to assign all z-variables without any contradiction

$\to$ let d be such an assignment to the z-variables

It remains to show that we did not remove too many clauses

Proof (cont.)

Let $\mathcal{C} := \{C_j \setminus C_1 \mid C_j \cap C_1 \neq \emptyset\}$ the set of *broken clauses*
Let $\mathscr{c}_1$ be a minimal model of $\mathcal{C}$ and $\mathscr{a}_1 := \mathscr{b}_1 \cup \mathscr{c}_1 \cup \mathscr{d}$

Proof (cont.)

Let $\mathcal{C} := \{C_j \setminus C_1 \mid C_j \cap C_1 \neq \emptyset\}$ the set of *broken clauses*
Let $\mathscr{c}_1$ be a minimal model of $\mathcal{C}$ and $\mathscr{a}_1 := \mathscr{b}_1 \cup \mathscr{c}_1 \cup \mathscr{d}$

→ Every clause in φ is either satisfied by $\mathscr{a}_1$, or not assigned at all

→ at most $k \cdot \log n$ clauses are assigned

$\mathscr{a}_2$ defined analogously for D_2

Proof (cont.)

Let $\mathcal{C} := \{C_j \setminus C_1 \mid C_j \cap C_1 \neq \emptyset\}$ the set of *broken clauses*
Let $\mathbf{c}_1$ be a minimal model of $\mathcal{C}$ and $\mathbf{a}_1 := \mathbf{b}_1 \cup \mathbf{c}_1 \cup \mathbf{d}$

- $\rightarrow$ Every clause in φ is either satisfied by $\mathbf{a}_1$, or not assigned at all
- $\rightarrow$ at most $k \cdot \log n$ clauses are assigned

$\mathbf{a}_2$ defined analogously for D_2

Then $D_1|_{\mathbf{a}_1} \wedge D_2|_{\mathbf{a}_2}$ is equivalent to a proper sub-CNF of φ

Proof (cont.)

Let $\mathcal{C} := \{C_j \setminus C_1 \mid C_j \cap C_1 \neq \emptyset\}$ the set of *broken clauses*
Let $\mathscr{c}_1$ be a minimal model of $\mathcal{C}$ and $\mathscr{a}_1 := \mathscr{b}_1 \cup \mathscr{c}_1 \cup \mathscr{d}$

→ Every clause in φ is either satisfied by $\mathscr{a}_1$, or not assigned at all

→ at most $k \cdot \log n$ clauses are assigned

$\mathscr{a}_2$ defined analogously for D_2

Then $D_1|_{\mathscr{a}_1} \wedge D_2|_{\mathscr{a}_2}$ is equivalent to a proper sub-CNF of φ

Let D^* be the OBDD (SDD) collecting all clauses satisfied by either $\mathscr{a}_i$

→ $|D^*| \in \mathcal{O}\left(2^{k^2 \cdot \log n}\right) = \mathcal{O}\left(n^{k^2}\right)$

→ $D_1|_{\mathscr{a}_1} \wedge D_2|_{\mathscr{a}_2} \wedge D^* = \varphi$

Proof (cont.)

Let $\mathcal{C} := \{C_j \setminus C_1 \mid C_j \cap C_1 \neq \emptyset\}$ the set of *broken clauses*

Let $\mathscr{c}_1$ be a minimal model of $\mathcal{C}$ and $\mathscr{a}_1 := \mathscr{b}_1 \cup \mathscr{c}_1 \cup \mathscr{d}$

→ Every clause in φ is either satisfied by $\mathscr{a}_1$, or not assigned at all

→ at most $k \cdot \log n$ clauses are assigned

$\mathscr{a}_2$ defined analogously for D_2

Then $D_1|_{\mathscr{a}_1} \wedge D_2|_{\mathscr{a}_2}$ is equivalent to a proper sub-CNF of φ

Let D^* be the OBDD (SDD) collecting all clauses satisfied by either $\mathscr{a}_i$

→ $|D^*| \in \mathcal{O}\left(2^{k^2 \cdot \log n}\right) = \mathcal{O}\left(n^{k^2}\right)$

→ $D_1|_{\mathscr{a}_1} \wedge D_2|_{\mathscr{a}_2} \wedge D^* = \varphi$

Therefore, $|D_\varphi|$ bounded in $\mathcal{O}\left(|D_1| \cdot |D_2| \cdot |D^*| = t(n)^2 n^{k^2}\right)$

Separation Example

Known superpolynomial lower bounds on φ imply superpolynomial lower bounds on $\mathcal{Z}(\varphi)$

Lemma (Razgon (2014))

For every $k > 50$ there is a class $\mathcal{G} := (G_i)_{i \in \mathbb{N}}$ of constant degree graphs s.t. for every $G_n \in \mathcal{G}$

- *tw$(G_n) = k$ and $|V(G_n)| \in \mathcal{O}(n)$*
- *Every OBDD representing $VC(G_n)$ has size $n^{\Omega(k)}$*

Separation Example

Known superpolynomial lower bounds on φ imply superpolynomial lower bounds on $\mathcal{Z}(\varphi)$

Lemma (Razgon (2014))

For every $k > 50$ there is a class $\mathcal{G} := (G_i)_{i \in \mathbb{N}}$ of constant degree graphs s.t. for every $G_n \in \mathcal{G}$

- *tw$(G_n) = k$ and $|V(G_n)| \in \mathcal{O}(n)$*
- *Every OBDD representing $VC(G_n)$ has size $n^{\Omega(k)}$*

Consequence: Every OBDD$(\wedge, r)$-refutation of $\mathcal{Z}(VC(G_n))$ has size $n^{\Omega(k)}$

Separation Example

Known superpolynomial lower bounds on φ imply superpolynomial lower bounds on $\mathcal{Z}(\varphi)$

Lemma (Razgon (2014))

For every $k > 50$ there is a class $\mathcal{G} := (G_i)_{i\in\mathbb{N}}$ of constant degree graphs s.t. for every $G_n \in \mathcal{G}$

- *tw$(G_n) = k$ and $|V(G_n)| \in \mathcal{O}(n)$*
- *Every OBDD representing $VC(G_n)$ has size $n^{\Omega(k)}$*

Consequence: Every OBDD$(\wedge, r)$-refutation of $\mathcal{Z}(VC(G_n))$ has size $n^{\Omega(k)}$

if φ has treewidth k, then there there is a SDD$(\wedge)$-refutation of size $\mathcal{O}(n^2 \cdot 2^k)$ of $\mathcal{Z}(VC(G_n))$

Separation Example

Known superpolynomial lower bounds on φ imply superpolynomial lower bounds on $\mathcal{Z}(\varphi)$

Lemma (Razgon (2014))

For every $k > 50$ there is a class $\mathcal{G} := (G_i)_{i \in \mathbb{N}}$ of constant degree graphs s.t. for every $G_n \in \mathcal{G}$

- *tw$(G_n) = k$ and $|V(G_n)| \in \mathcal{O}(n)$*
- *Every OBDD representing $VC(G_n)$ has size $n^{\Omega(k)}$*

Consequence: Every OBDD$(\wedge, r)$-refutation of $\mathcal{Z}(VC(G_n))$ has size $n^{\Omega(k)}$

if φ has treewidth k, then there there is a SDD$(\wedge)$-refutation of size $\mathcal{O}(n^2 \cdot 2^k)$ of $\mathcal{Z}(VC(G_n))$

$\rightarrow$ Quasipolynomial separation between OBDD$(\wedge, r)$ and SDD$(\wedge)$

Outlook

It is possible to further generalise our results

- Our main method also works for CNF classes beyond monotone $(k, \sqrt{\log n})$-CNFs
  - $\rightarrow$ Potential exponential separation between OBDD$(\wedge, r)$ and SDD$(\wedge)$
- Lifting method also works on classes beyond OBDD and SDD (e.g. structured-d-DNNF)

Unfortunately, weakening can easily refute every $\mathcal{Z}(\varphi)$

$\rightarrow$ More tools necessary to separate OBDD$(\wedge, w)$ and SDD$(\wedge, w)$

Outlook

It is possible to further generalise our results

- Our main method also works for CNF classes beyond monotone $(k, \sqrt{\log n})$-CNFs
  $\rightarrow$ Potential exponential separation between OBDD$(\wedge, r)$ and SDD$(\wedge)$
- Lifting method also works on classes beyond OBDD and SDD (e.g. structured-d-DNNF)

Unfortunately, weakening can easily refute every $\mathcal{Z}(\varphi)$

$\rightarrow$ More tools necessary to separate OBDD$(\wedge, w)$ and SDD$(\wedge, w)$

Some questions regarding SDD equivalence still open

- Is SDD equivalence polynomially verifiable?

Restricted Restructuring

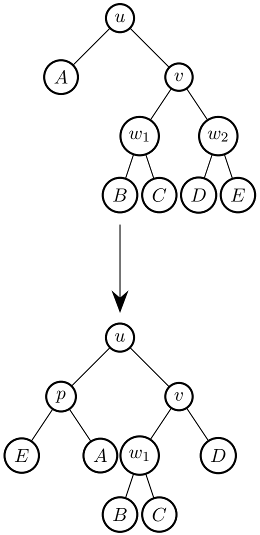
Definition (SDD equivalence)

Input: Two SDDs D_1, D_2 respecting vtrees T_1, T_2
Question: Are D_1 and D_2 equivalent?

- If $T_1 = T_2$, SDD poly-time decidable
- If T_2 and T_2 arbitrary, Unknown
- If T_1 and T_2 very similar, again poly-time
  - Allow only local changes to the vtree

Restricted restructuring (r^*):

- Each step at most quadratic growth
- polynomially verifiable
- can simulate OBDD reordering



Local CNFs

Lifting method can be expanded to larger classes of CNFs called *local CNFs*

Definition (Local CNF)

Let φ be a reduced CNF with n variables. partition (ψ_1, ψ_2) of φ is called *monotone*, if each $x \in \mathrm{var}(\psi_1) \cap \mathrm{var}(\psi_2)$ appears only positively or only negatively

Let $C \in \varphi$. C *has monotone protrusion*, if there is a $\psi \subset \varphi$ such that

- $\mathrm{var}(C) \cap \mathrm{var}(\varphi \setminus \psi) = \emptyset$
- $(\psi, \varphi \setminus \psi)$ is a monotone protrusion
- $|\mathrm{var}(\{C \mid \mathrm{var}(C) \cap \mathrm{var}(\psi) \neq \emptyset\})| \in \mathcal{O}(\log n)$

A CNF φ is called *local*, if every $C \in \varphi$ has a monotone protrusion.

Every reduced monotone $(k, \sqrt{\log n})$-CNF is local, but the class of local CNFs is larger. For example, if the Gaifman graph of φ consists of many small components, then φ is local

lifting method also works on all local CNFs with the same proof idea