Compilation de CNF vers les circuits booléens : une borne inférieure

Simone Bova Florent Capelli Friedrich Slivovski Stefan Mengel

15 Janvier 2015

Compilation de CNF

CNF $F = \bigwedge_{i=1}^{n} C_i$
$\rightarrow$ une *base de connaissances* sur var(F) (communauté IA).

Compilation de CNF

CNF $F = \bigwedge_{i=1}^{n} C_i$
$\rightarrow$ une *base de connaissances* sur var(F) (communauté IA).

But : effectuer rapidement des requêtes sur F comme :

- est-ce que F est satisfiable ?
- est-ce que $F \Rightarrow C$ pour une clause C ? (*entailment problem*)
- combien F a-t-elle de solutions ? combien $F[x_1 = 0]$ a-t-elle de solution ?

Problème : on ne sait répondre à aucune de ces requêtes en temps polynomial sur les CNFs.

Compilation de CNF

Penser F comme une CNF "constante". On veut éviter :

- (requête 1, F) $\rightarrow$ réponse 1 en temps exponentiel en $|F|$.
- ...
- (requête n, F) $\rightarrow$ réponse n en temps exponentiel en $|F|$.

Compilation de CNF

Penser F comme une CNF "constante". On veut éviter :

- (requête 1, F) $\rightarrow$ réponse 1 en temps exponentiel en $|F|$.
- ...
- (requête n, F) $\rightarrow$ réponse n en temps exponentiel en $|F|$.

Idéal à atteindre :

- $F \rightarrow D$, structure de données particulière, en temps exponentiel en $|F|$.
- $|D|$ polynomial en $|F|$ si possible
- (requête 1, D) $\rightarrow$ réponse 1 en temps polynomial en $|D|$.
- ...
- (requête n, D) $\rightarrow$ réponse n en temps polynomial en $|D|$

Que choisir comme structure de données ?

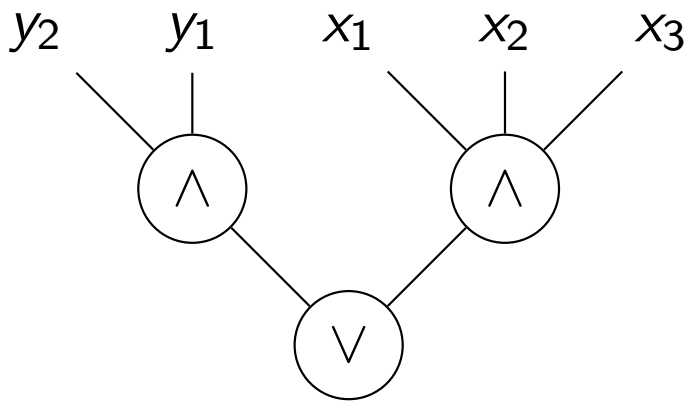
D : en général une famille de circuits avec de bonnes propriétés.
Ici $D = \mathrm{DNNF} =$ *Decomposable Negation Normal Form.*

Une DNNF c'est :

- un circuit booléen C avec des portes $\vee$ et $\wedge$ de fan-in 2
- *Negation Normal Form :*
  les entrées sont étiquetées par x ou $\neg x$ avec $x \in \mathrm{var}(F)$
- *Multilinéarité (Decomposable) :*
  Soit α une porte $\wedge$ ayant pour entrée α_1 et α_2. Alors
  $\mathrm{var}(\alpha_1) \cap \mathrm{var}(\alpha_2) = \emptyset$

Remarques

- les DNFs sont un cas particulier de DNNFs.



- SAT peut être résolu en temps linéaire sur les DNNFs
- Généralise beaucoup d'autres structures de données : OBDDs, FBDDs, Read-Once Branching Programs ...
- Restriction des DNNFs plus utile : *deterministic* DNNF = les portes $\vee$ ont des modèles disjoints, permet le comptage en temps linéaire

Question

Question

Est-ce que toute CNF F peut être compilée vers une DNNF *de taille polynomiale en $|F|$?*

Dans cette présentation, on prouve que non. Et plus précisément:

Théorème

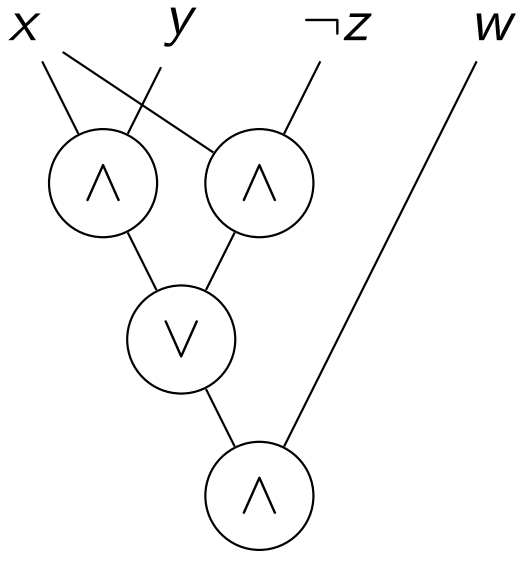
Il existe une classe infinie $\mathcal{C}$ de formules CNF telle que pour tout $F \in \mathcal{C}$, toute DNNF *calculant F est de taille $2^{\Omega(|F|)}$.*

Remarque : on sait transformer une CNF F en une DNF de taille $2^{\Omega(|F|)}$, cette borne inférieure est donc optimale.

Les DNNFs : proof trees

Proof tree T : sous-ensemble de portes de la DNNF D tel que

- T contient la sortie de D
- si une porte α de T est une porte $\wedge$, alors toutes ses entrées sont dans T
- si une porte α de T est une porte $\vee$, alors exactement une de ses entrées est dans T

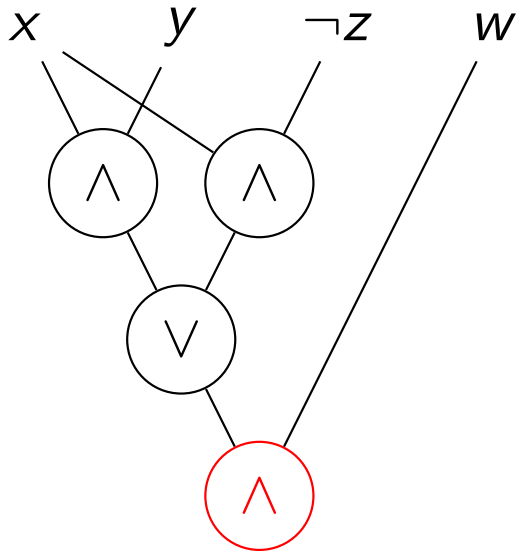


- T est un arbre
- on note lit(T) les étiquettes des feuilles de T
- on note $m(T) = \bigwedge_{x \in \text{lit}(T)} x$

Les DNNFs : proof trees

Proof tree T : sous-ensemble de portes de la DNNF D tel que

- T contient la sortie de D
- si une porte α de T est une porte $\wedge$, alors toutes ses entrées sont dans T
- si une porte α de T est une porte $\vee$, alors exactement une de ses entrées est dans T

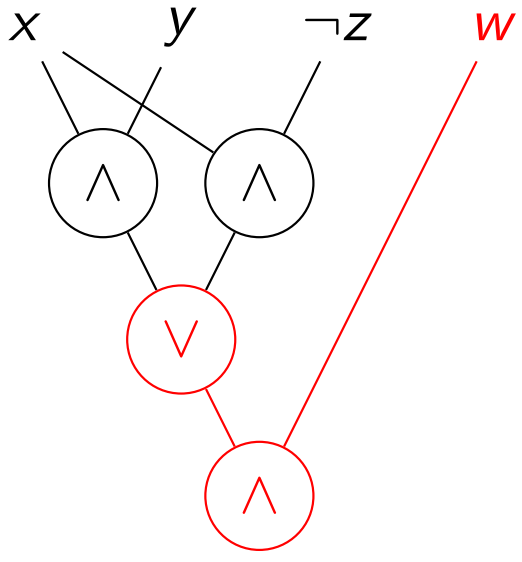


- T est un arbre
- on note lit(T) les étiquettes des feuilles de T
- on note $m(T) = \bigwedge_{x \in \text{lit}(T)} x$

Les DNNFs : proof trees

Proof tree T : sous-ensemble de portes de la DNNF D tel que

- T contient la sortie de D
- si une porte α de T est une porte $\wedge$, alors toutes ses entrées sont dans T
- si une porte α de T est une porte $\vee$, alors exactement une de ses entrées est dans T

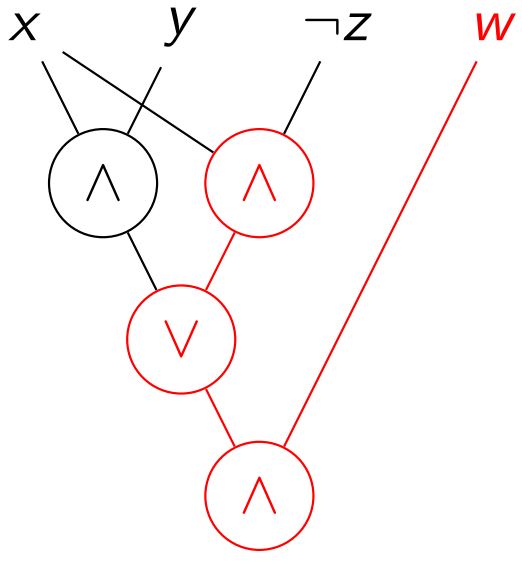


- T est un arbre
- on note $\mathrm{lit}(T)$ les étiquettes des feuilles de T
- on note $m(T) = \bigwedge_{x \in \mathrm{lit}(T)} x$

Les DNNFs : proof trees

Proof tree T : sous-ensemble de portes de la DNNF D tel que

- T contient la sortie de D
- si une porte α de T est une porte $\wedge$, alors toutes ses entrées sont dans T
- si une porte α de T est une porte $\vee$, alors exactement une de ses entrées est dans T

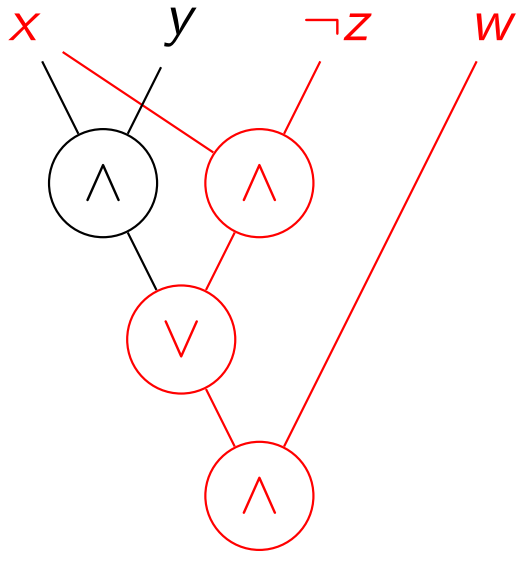


- T est un arbre
- on note $\mathrm{lit}(T)$ les étiquettes des feuilles de T
- on note $m(T) = \bigwedge_{x \in \mathrm{lit}(T)} x$

Les DNNFs : proof trees

Proof tree T : sous-ensemble de portes de la DNNF D tel que

- T contient la sortie de D
- si une porte α de T est une porte $\wedge$, alors toutes ses entrées sont dans T
- si une porte α de T est une porte $\vee$, alors exactement une de ses entrées est dans T



$m(T) = x \wedge \neg z \wedge w$

- T est un arbre
- on note $\mathrm{lit}(T)$ les étiquettes des feuilles de T
- on note $m(T) = \bigwedge_{x \in \mathrm{lit}(T)} x$

Les DNNFs : quelques résultats

Soit F calculée par une DNNF D :

- $F = \bigvee_{T \text{ proof tree de } D} m(T)$
- F monotone, alors il existe D' de même taille que D sans négation calculant F (remplacer $\neg x$ par 1)

Les DNNFs : quelques résultats

Soit F calculée par une DNNF D :

- $F = \bigvee_{T \text{ proof tree de } D} m(T)$
- F monotone, alors il existe D' de même taille que D sans négation calculant F (remplacer $\neg x$ par 1)

Dans la suite : que des formules monotones donc que des DNNFs sans négation.

CNF associée à un graphe G

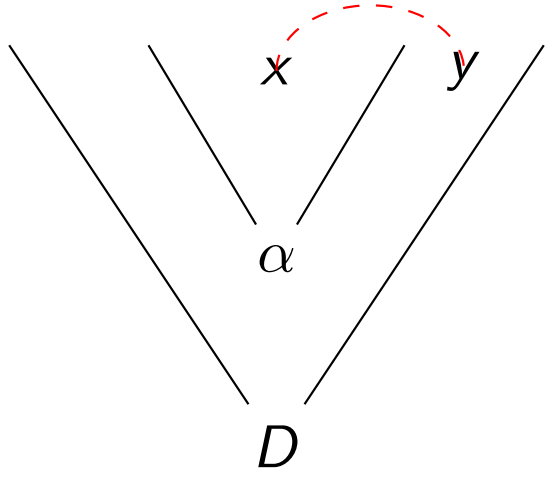
À un graphe $G = (V, E)$, on associe une CNF F_G :

- $\text{var}(F_G) = V$
- $F_G = \bigwedge_{(u,v)\in E}(u \vee v)$
- F_G est une 2-CNF, monotone

Lien entre proof trees et graphes

G un graphe, D une DNNF sans négation calculant F_G, α une porte de D

$x \in \mathrm{var}(D_\alpha)$, $y \notin \mathrm{var}(D_\alpha)$, $(x, y) \in E$.

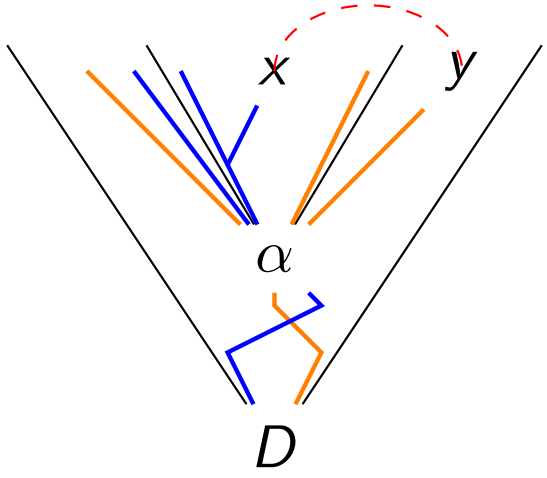


- soit x est dans tous les proof tree passant par α
- soit y est dans tous les proof tree passant par α

Lien entre proof trees et graphes

G un graphe, D une DNNF sans négation calculant F_G, α une porte de D

$x \in \mathrm{var}(D_\alpha)$, $y \notin \mathrm{var}(D_\alpha)$, $(x, y) \in E$.

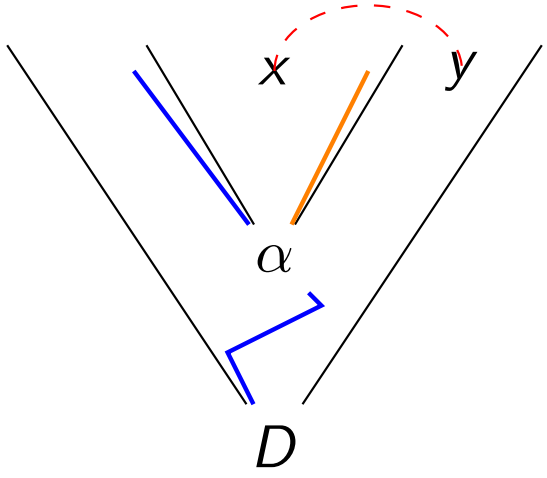


- soit x est dans tous les proof tree passant par α
- soit y est dans tous les proof tree passant par α

Lien entre proof trees et graphes

G un graphe, D une DNNF sans négation calculant F_G, α une porte de D

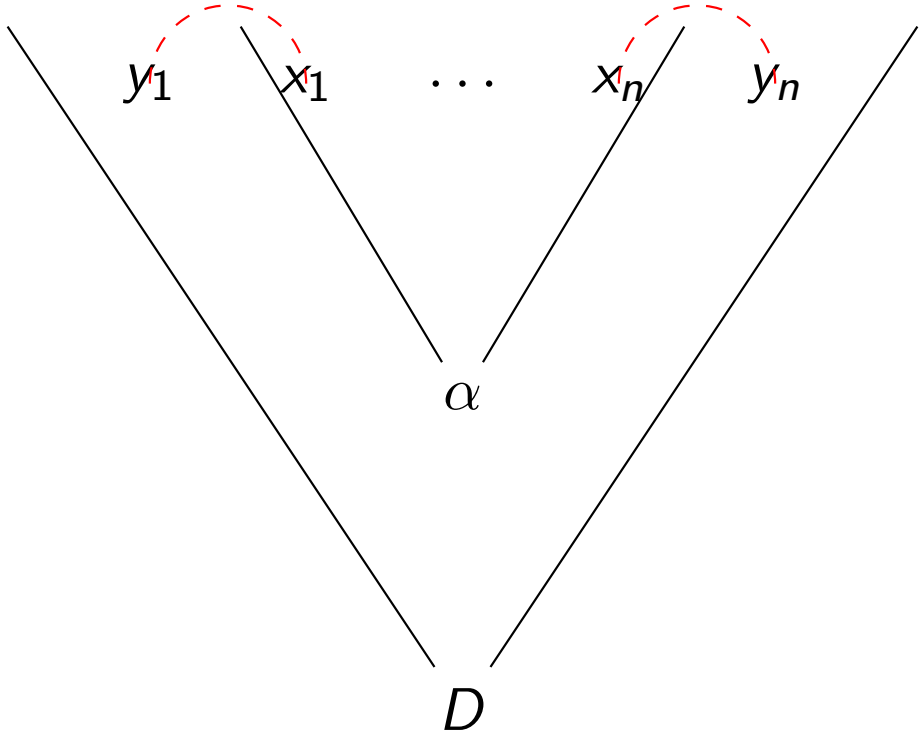
$x \in \mathrm{var}(D_\alpha)$, $y \notin \mathrm{var}(D_\alpha)$, $(x, y) \in E$.



- soit x est dans tous les proof tree passant par α
- soit y est dans tous les proof tree passant par α

Lien entre proof trees et graphes

$(x_1, y_1), \ldots, (x_n, y_n)$ un matching de G :



On trouve $S \subseteq X \cup Y$, $|S| = n$ tel que tout proof tree passant par α contient S.

Question

Quelle est la proportion des solutions concernées ?

Vertex covers

- Isomorphisme naturel entre les solutions de F_G et les vertex covers de G :
- $W \subseteq V \mapsto \tau_W$ avec $\tau_W(x) = 1$ ssi $x \in W$
- τ_W solution de F_G ssi W vertex cover de G
- Notation : $\mathrm{VC}(G, S) =$ vertex covers de G contenant S

Vertex covers

- Isomorphisme naturel entre les solutions de F_G et les vertex covers de G :
- $W \subseteq V \mapsto \tau_W$ avec $\tau_W(x) = 1$ ssi $x \in W$
- τ_W solution de F_G ssi W vertex cover de G
- Notation : $\mathrm{VC}(G, S) =$ vertex covers de G contenant S

Théorème

Soit G un graphe de degré d et $\mu_d = (1 + 2^{-d}) > 1$:

$$\#\mathrm{VC}(G, S) \leq \mu_d^{-|S|} \#\mathrm{VC}(G)$$

Moralité : si d est borné et $|S|$ grand, on rate beaucoup de solution !

Démonstration du théorème

Pour $S = \{s\}$, $N_s = \text{voisin}(s)$:

- On note $\text{VC}(G, \backslash s) = \{\mathcal{C} \in \text{VC}(G) \mid s \notin \mathcal{C}\}$
- On a $\#\text{VC}(G) = \#\text{VC}(G, s) + \#\text{VC}(G, \backslash s)$

Démonstration du théorème

Pour $S = \{s\}$, $N_s = \text{voisin}(s)$:

- On note $\text{VC}(G, \setminus s) = \{\mathcal{C} \in \text{VC}(G) \mid s \notin \mathcal{C}\}$
- On a $\#\text{VC}(G) = \#\text{VC}(G, s) + \#\text{VC}(G, \setminus s)$
- De plus $2^d \#\text{VC}(G, \setminus s) \geq \#\text{VC}(G, s)$
- car $\mathcal{C} \in \text{VC}(G, s) \mapsto (\mathcal{C} \cap N_s, (\mathcal{C} \setminus \{s\}) \cup N_s)$ est une injection

Démonstration du théorème

Pour $S = \{s\}$, $N_s = \text{voisin}(s)$:

- On note $\text{VC}(G, \backslash s) = \{\mathcal{C} \in \text{VC}(G) \mid s \notin \mathcal{C}\}$
- On a $\#\text{VC}(G) = \#\text{VC}(G, s) + \#\text{VC}(G, \backslash s)$
- De plus $2^d \#\text{VC}(G, \backslash s) \geq \#\text{VC}(G, s)$
- car $\mathcal{C} \in \text{VC}(G, s) \mapsto (\mathcal{C} \cap N_s, (\mathcal{C} \setminus \{s\}) \cup N_s)$ est une injection

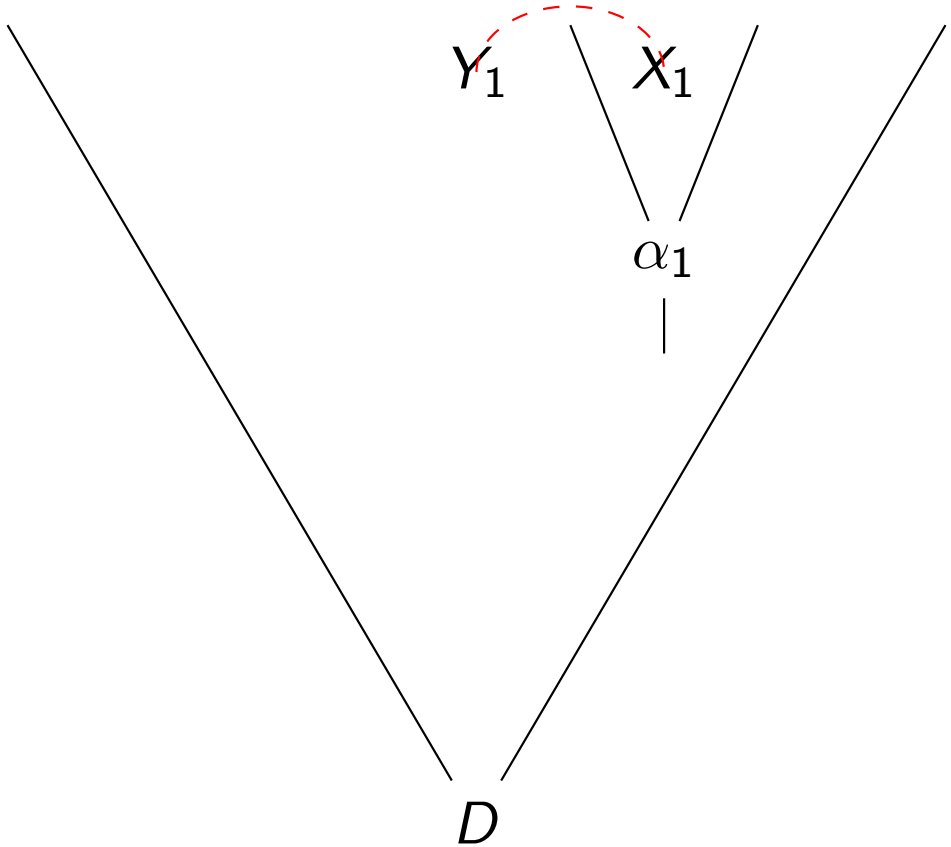
D'où

$$(1 + 2^{-d})\#\text{VC}(G, s) = \mu_d \#\text{VC}(G, s) \leq \#\text{VC}(G)$$

Pour $|S| > 1$, on peut faire une récurrence.

Stratégie de la preuve

On choisit "bien" G:

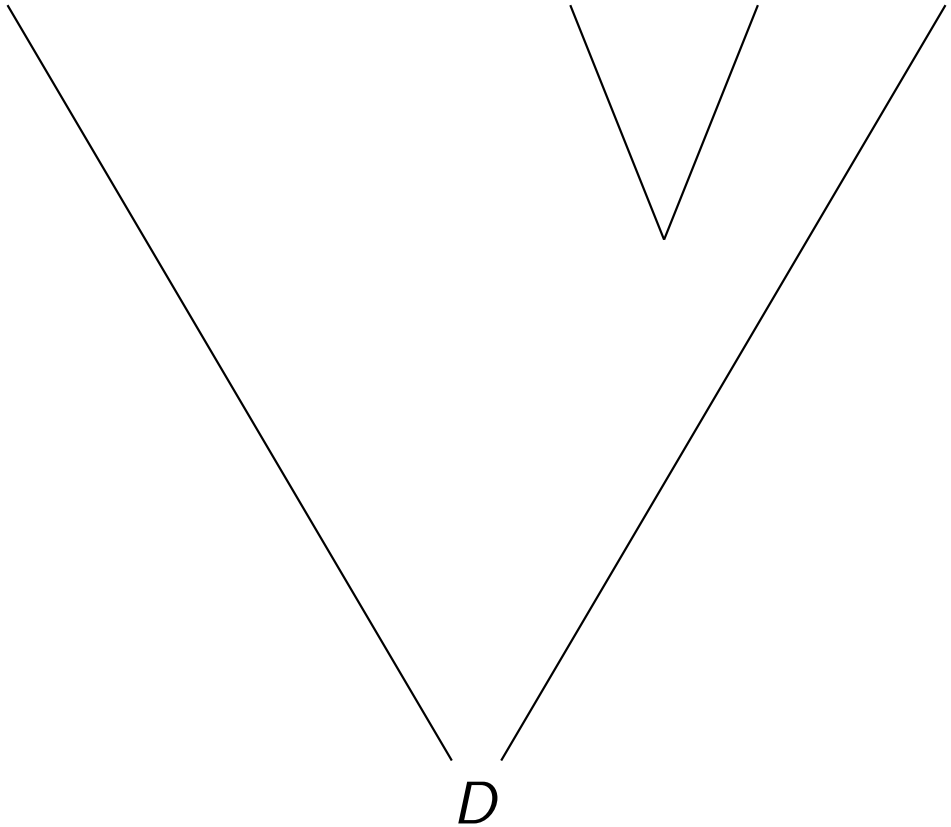


$|X_i|, |Y_i| \geq \Omega(n)$

Quand D est vide, on a construit $S_1, \ldots, S_m$ de taille $\Omega(n)$ et on couvre toutes les solutions donc $m = \mu_d^{\Omega(n)} \leq |D|$

Stratégie de la preuve

On choisit “bien” G:

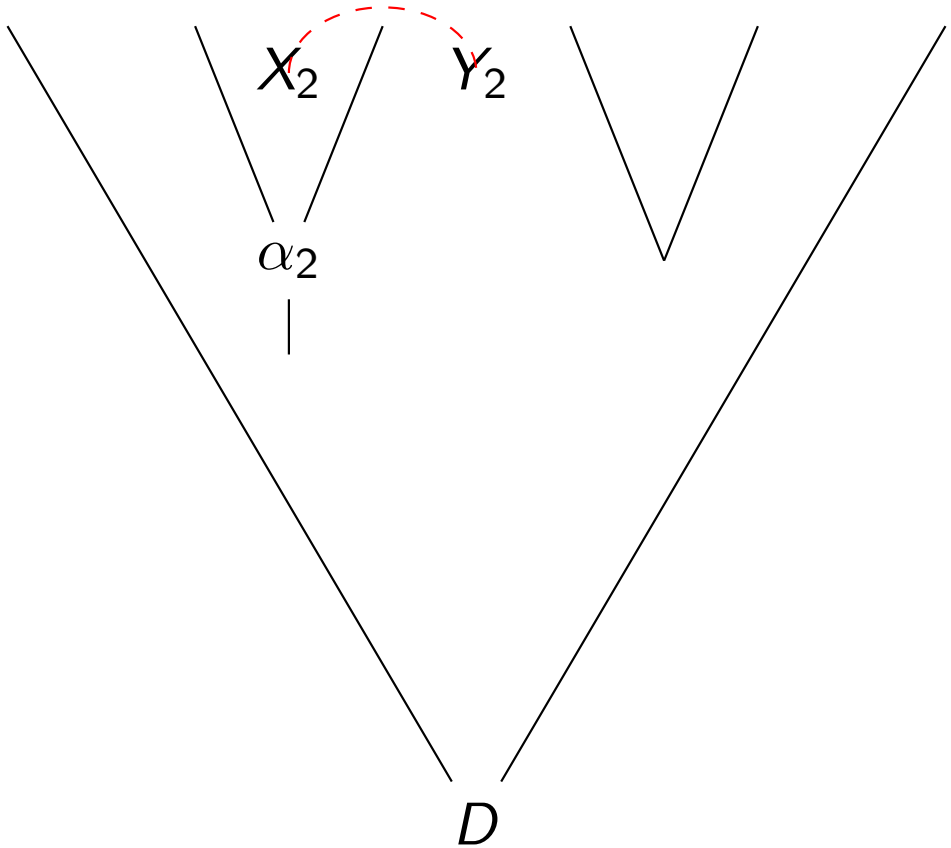


$|X_i|, |Y_i| \geq \Omega(n)$

Quand D est vide, on a construit $S_1, \ldots, S_m$ de taille $\Omega(n)$ et on couvre toutes les solutions donc $m = \mu_d^{\Omega(n)} \leq |D|$

Stratégie de la preuve

On choisit "bien" G:

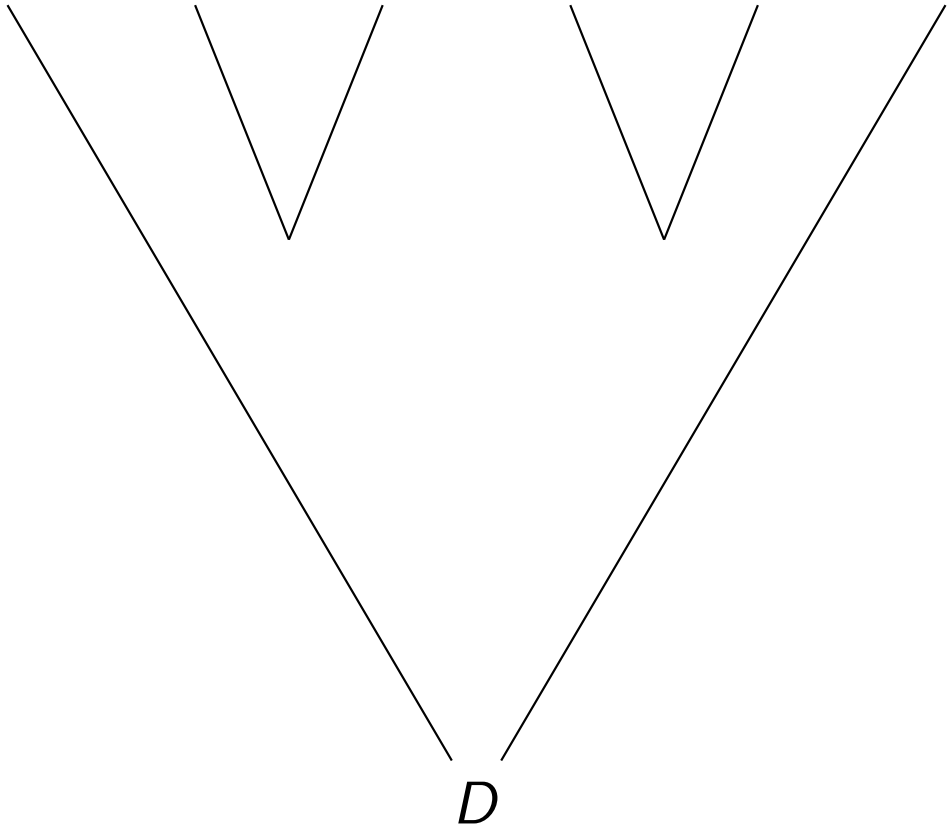


$|X_i|, |Y_i| \geq \Omega(n)$

Quand D est vide, on a construit $S_1, \ldots, S_m$ de taille $\Omega(n)$ et on couvre toutes les solutions donc $m = \mu_d^{\Omega(n)} \leq |D|$

Stratégie de la preuve

On choisit "bien" G:



$|X_i|, |Y_i| \geq \Omega(n)$

Quand D est vide, on a construit $S_1, \ldots, S_m$ de taille $\Omega(n)$ et on couvre toutes les solutions donc $m = \mu_d^{\Omega(n)} \leq |D|$

Choix de G

On choisit $G =$ un (c, d)-expander, pouquoi ?

- Degré borné par d : borne inf sur les vertex covers
- Propriété : $\frac{|V|}{d} \leq |S| \leq \frac{|V|}{2}$ alors $|\text{voisin}(S)| \geq c|S|$
- Donc trouver $\alpha \in D$ avec $\frac{|V|}{d} \leq |\text{var}(\alpha)| \leq \frac{|V|}{2}$ suffit pour trouver un matching de taille $\Omega(|V|)$
- On sait en trouver de n'importe quelle taille, donc cela nous donne une famille de CNF pour la borne inférieure

Corollaire

Question

Étant donné une DNNF D*, existe-t-il une* DNNF D' *de taille polynomial en* $|D|$ *qui calcule* $\neg D$ *?*

Non :

- on a construit une famille F_n de CNF ayant des DNNF de taille $\geq 2^{\Omega(|F_n|)}$
- mais $\neg F_n$ est sous forme DNF : admet une petite DNNF, de taille $\Omega(|F_n|)$

Conclusion

- Ferme plusieurs questions ouvertes dans le domaine de la compilation de connaissance (Marquis, Darwich, 2002)
- Améliore les bornes inférieures qu'on avait sur les FBDD (qui étaient en $2^{\Omega(\sqrt{n})}$) : utilisation des expanders pour avoir une treewidth linéaire en n
- Peut-on utiliser ces techniques pour séparer d'autres classes de circuits booléens ?