Problems and solutions
courtesy of James Worrell

Logic & Proof Sheet 1

Jingjie Yang
HT'26

Problems and solutions
courtesy of James Worrell

Logic & Proof Sheet 1

Jingjie Yang
HT'26

Propositional logic

Propositional logic

A formula in n variables amounts to a function $\{0,1\}^n \to \{0,1\}$.

Propositional logic

(CNF)

A formula in n variables amounts to a function $\{0,1\}^n \to \{0,1\}$.

Propositional logic

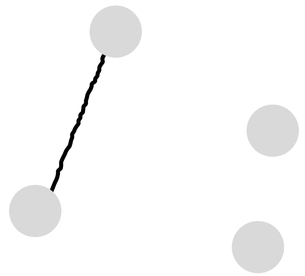
(CNF)

A formula in n variables amounts to a function $\{0,1\}^n \to \{0,1\}$.

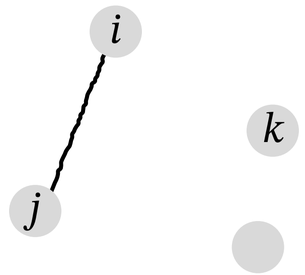
/ straight-line program

Example: graph homomorphism

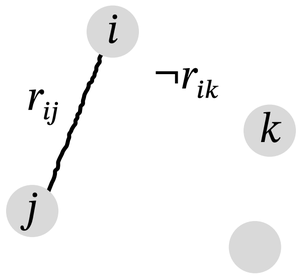
Example: graph homomorphism



Example: graph homomorphism

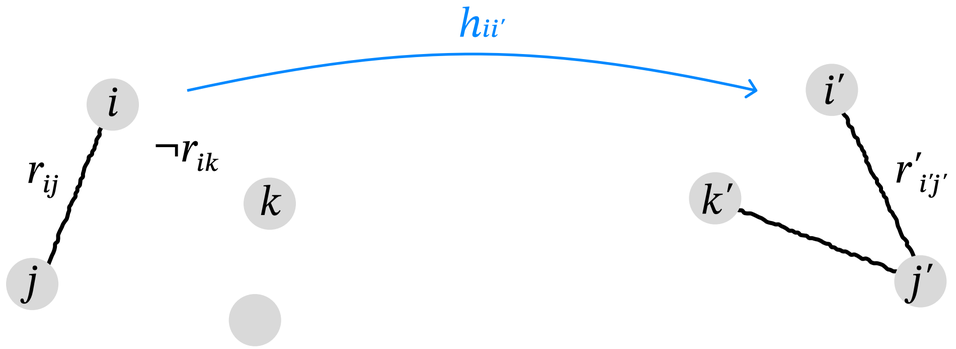


Example: graph homomorphism

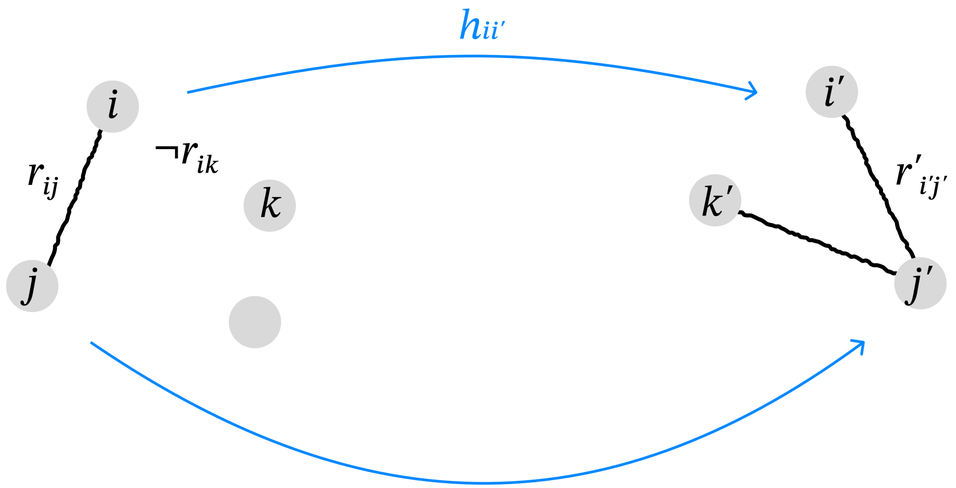


Example: graph homomorphism

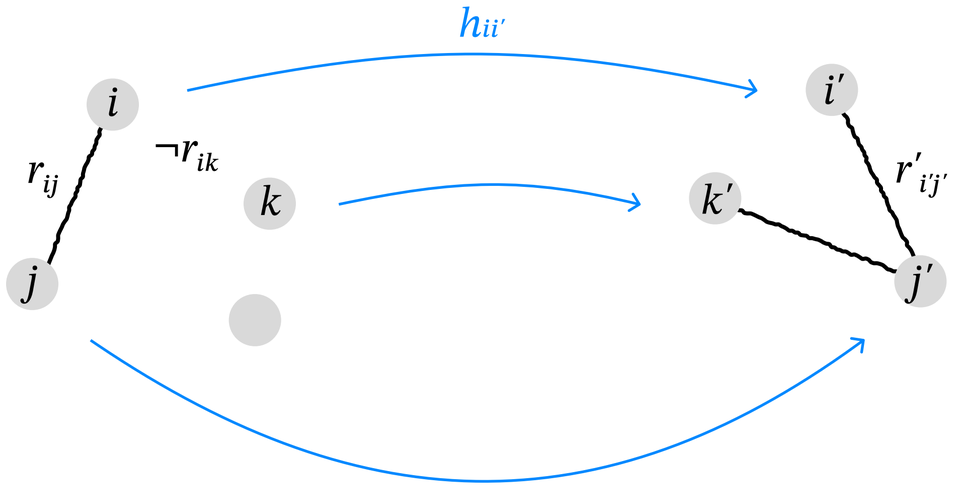
Example: graph homomorphism



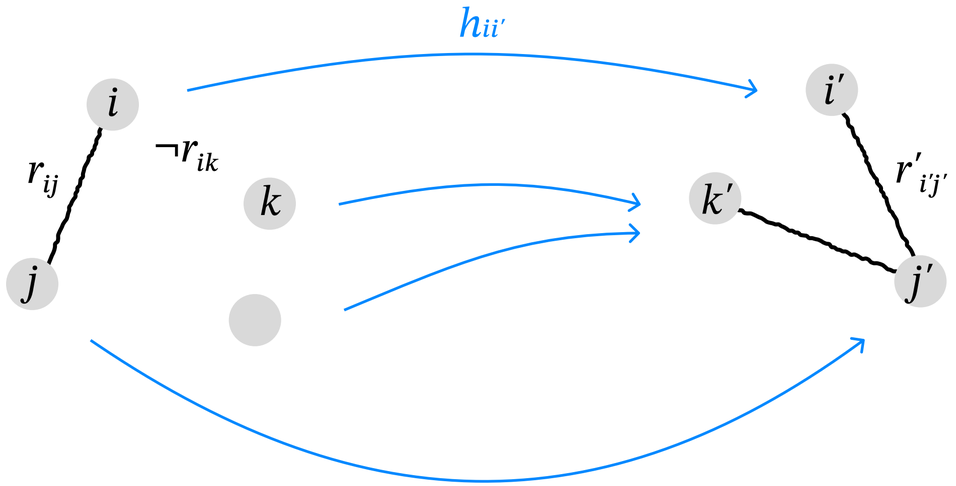
Example: graph homomorphism



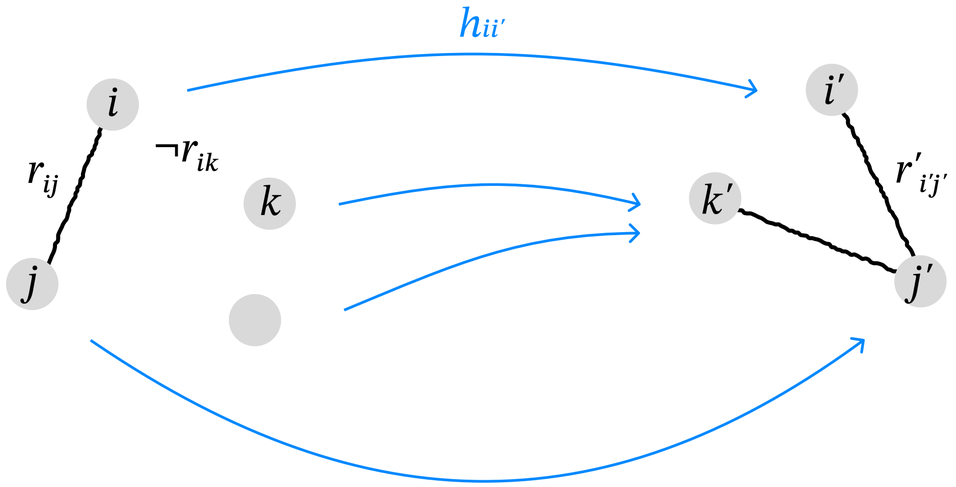
Example: graph homomorphism



Example: graph homomorphism

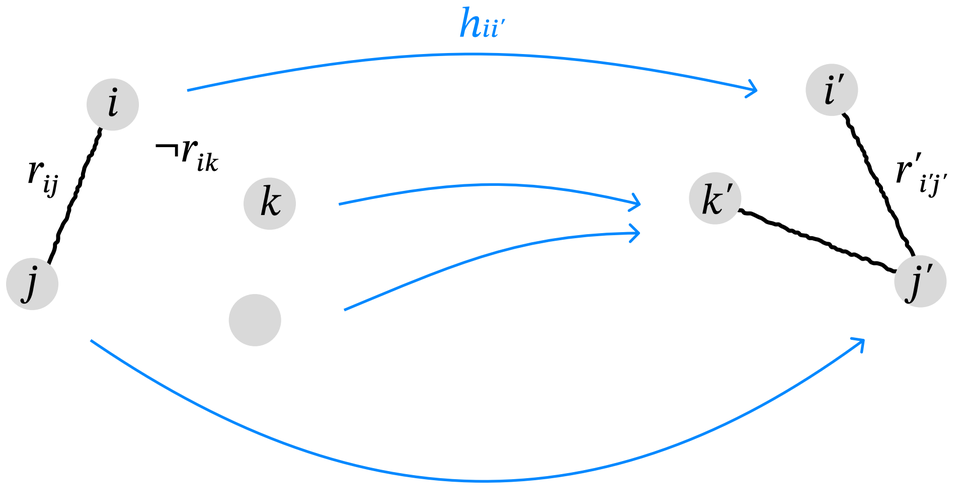


Example: graph homomorphism



$$\bigwedge_{i}\bigvee_{i'} h_{ii'} \quad \wedge \quad \bigwedge_{i}\bigwedge_{i' \neq j'} \neg h_{ii'} \vee \neg h_{ij'} \quad \wedge \quad \bigwedge_{i,j,i',j'} \neg h_{ii'} \vee \neg h_{jj'} \vee \neg r_{ij} \vee r'_{i'j'}$$

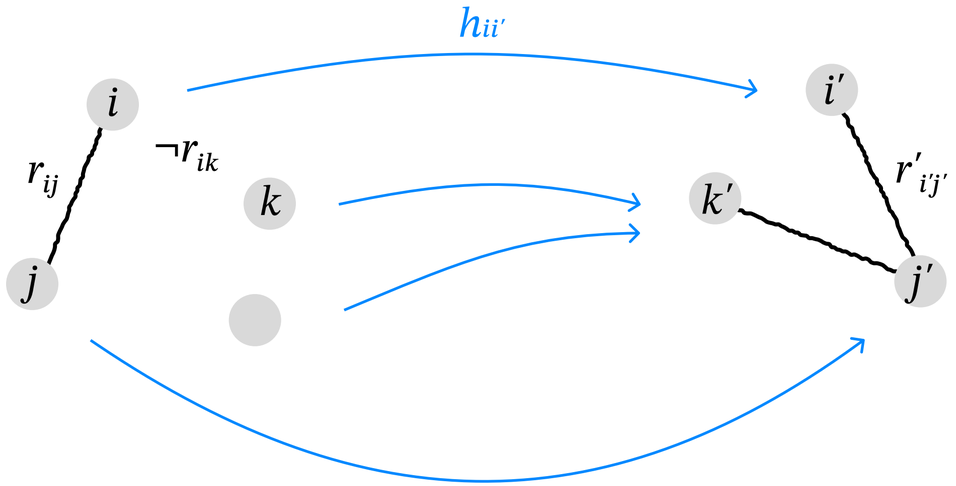
Example: graph homomorphism



every vertex maps to ≥ 1 vertex′

$$\bigwedge_{i}\bigvee_{i'} h_{ii'} \quad \wedge \quad \bigwedge_{i}\bigwedge_{i' \neq j'} \neg h_{ii'} \vee \neg h_{ij'} \quad \wedge \quad \bigwedge_{i,j,i',j'} \neg h_{ii'} \vee \neg h_{jj'} \vee \neg r_{ij} \vee r'_{i'j'}$$

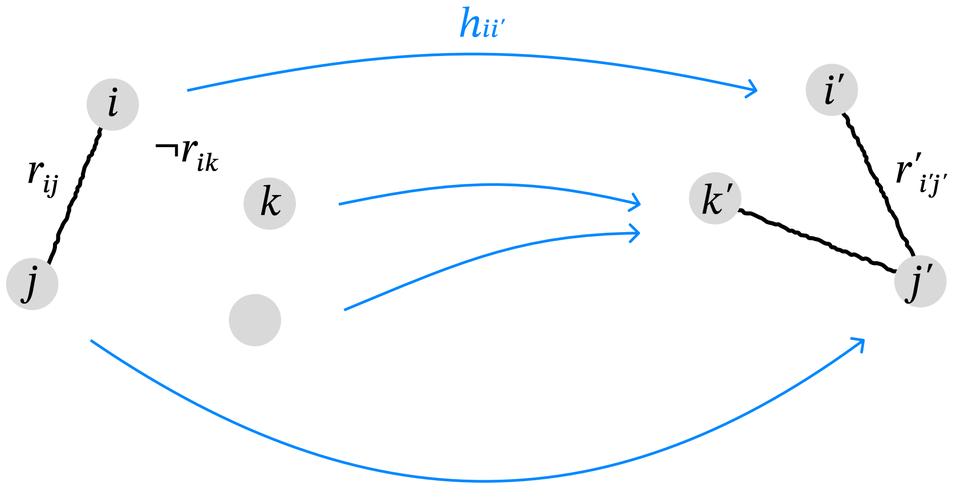
Example: graph homomorphism



every vertex maps to ≥ 1 vertex′ every vertex maps to ≤ 1 vertex′

$$\bigwedge_{i} \bigvee_{i'} h_{ii'} \quad \wedge \quad \bigwedge_{i} \bigwedge_{i' \neq j'} \neg h_{ii'} \vee \neg h_{ij'} \quad \wedge \quad \bigwedge_{i,j,i',j'} \neg h_{ii'} \vee \neg h_{jj'} \vee \neg r_{ij} \vee r'_{i'j'}$$

Example: graph homomorphism



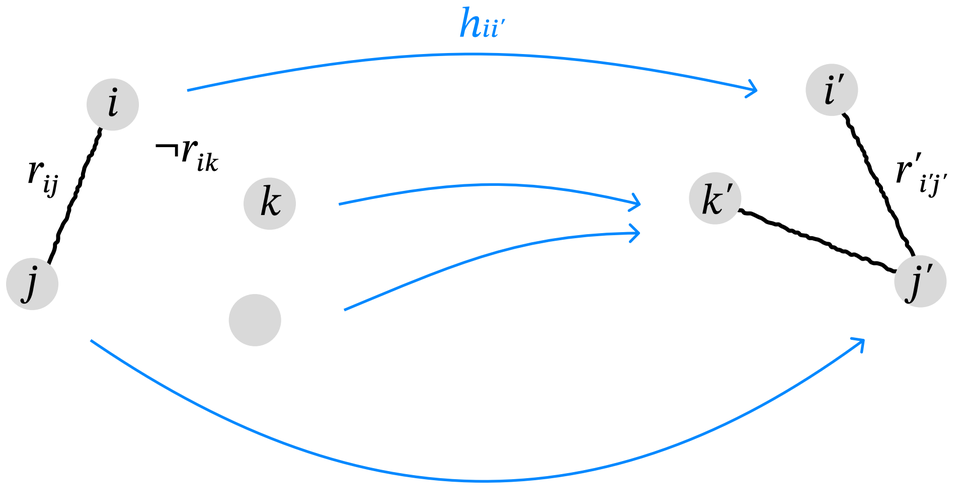
every vertex maps to ≥ 1 vertex'

every vertex maps to ≤ 1 vertex'

edges map to edges'

$$\bigwedge_{i} \bigvee_{i'} h_{ii'} \quad \wedge \quad \bigwedge_{i} \bigwedge_{i' \neq j'} \neg h_{ii'} \vee \neg h_{ij'} \quad \wedge \quad \bigwedge_{i,j,i',j'} \neg h_{ii'} \vee \neg h_{jj'} \vee \neg r_{ij} \vee r'_{i'j'}$$

Example: graph homomorphism



every vertex maps to ≥ 1 vertex'

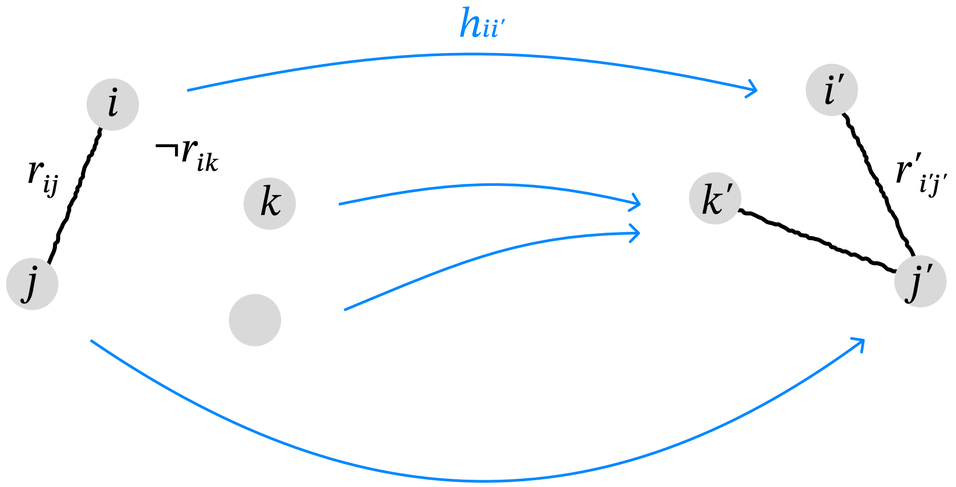
every vertex maps to ≤ 1 vertex'

edges map to edges'

$$\bigwedge_{i}\bigvee_{i'} h_{ii'} \quad \wedge \quad \bigwedge_{i}\bigwedge_{i' \neq j'} \neg h_{ii'} \vee \neg h_{ij'} \quad \wedge \quad \bigwedge_{i,j,i',j'} \neg h_{ii'} \vee \neg h_{jj'} \vee \neg r_{ij} \vee r'_{i'j'}$$

This formula in **h**, **r**, **r'** is satisfiable iff there is a graph homomorphism **h**.

Example: graph homomorphism



every vertex maps to ≥ 1 vertex′ every vertex maps to ≤ 1 vertex′ edges map to edges′

$$\bigwedge_{i}\bigvee_{i'} h_{ii'} \quad \wedge \quad \bigwedge_{i}\bigwedge_{i'\neq j'} \neg h_{ii'} \vee \neg h_{ij'} \quad \wedge \quad \bigwedge_{i,j,i',j'} \neg h_{ii'} \vee \neg h_{jj'} \vee \neg r_{ij} \vee r'_{i'j'}$$

neg pos

This formula in **h**, **r**, **r'** is satisfiable iff there is a graph homomorphism **h**.

Exercise 1

Write K_m for the complete graph on m vertices.

Exercise 1

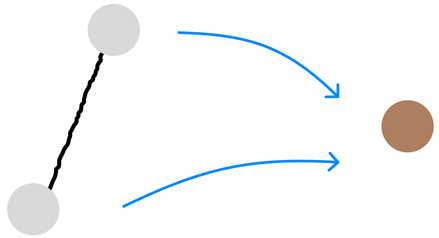
Write K_m for the complete graph on m vertices.

(a) homomorphism $K_m \to G$ $=$ m-clique in G

Exercise 1

Write K_m for the complete graph on m vertices.

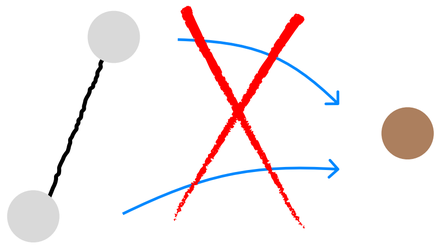
(a) homomorphism $K_m \to G$ $=$ m-clique in G



Exercise 1

Write K_m for the complete graph on m vertices.

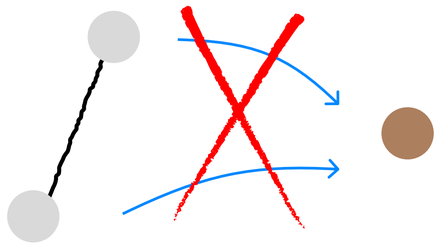
(a) homomorphism $K_m \to G$ $=$ m-clique in G



Exercise 1

Write K_m for the complete graph on m vertices.

(a) homomorphism $K_m \to G$ = m-clique in G



(b) homomorphism $G \to K_m$ = m-colouring of G

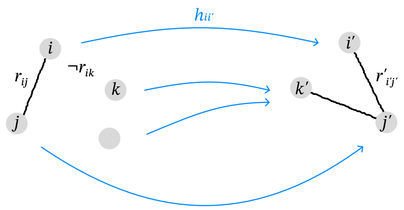
Propositional logic

(CNF)

A formula in n variables amounts to a function $\{0,1\}^n \to \{0,1\}$.

/ straight-line program

Example: graph homomorphism



every vertex maps to ≥ 1 vertex' every vertex maps to ≤ 1 vertex' edges map to edges'

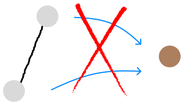
$$\bigwedge_i \bigvee_{i'} h_{ii'} \quad \wedge \quad \bigwedge_i \bigwedge_{i' \neq j'} \neg h_{ii'} \vee \neg h_{ij'} \quad \wedge \quad \bigwedge_{i,j,i',j'} \neg h_{ii'} \vee \neg h_{jj'} \vee \underbrace{\neg r_{ij}}_{\text{neg}} \vee \underbrace{r'_{i'j'}}_{\text{pos}}$$

This formula in **h**, **r**, **r'** is satisfiable iff there is a graph homomorphism **h**.

Exercise 1

Write K_m for the complete graph on m vertices.

(a) homomorphism $K_m \to G$ = m-clique in G



(b) homomorphism $G \to K_m$ = m-colouring of G

Problems and solutions
courtesy of James Worrell

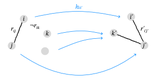
Logic & Proof Sheet 1

Jingjie Yang
HT'26

Propositional logic

(CNF)
A formula in n variables amounts to a function $\{0,1\}^n \to \{0,1\}$.
/ straight-line program

Example: graph homomorphism



every vertex maps to ≥ 1 vertex' every vertex maps to ≤ 1 vertex' edges map to edges'

$$\bigwedge_i \bigvee_{i'} h_{ii'} \quad \wedge \quad \bigwedge_i \bigwedge_{i' \neq j'} \neg h_{ii'} \vee \neg h_{ij'} \quad \wedge \quad \bigwedge_{i,j,i',j'} \neg h_{ii'} \vee \neg h_{jj'} \vee \underbrace{\neg r_{ij}}_{\text{neg}} \vee \underbrace{r'_{i'j'}}_{\text{pos}}$$

This formula in $\mathbf{h}$, $\mathbf{r}$, $\mathbf{r'}$ is satisfiable iff there is a graph homomorphism $\mathbf{h}$.

Exercise 1

Write K_m for the complete graph on m vertices.

(a) homomorphism $K_m \to G$ = m-clique in G

(b) homomorphism $G \to K_m$ = m-colouring of G

Resolution proof

Resolution proof

Given F in CNF: $C_1,\ C_2,\ C_3,\ \ldots,\ C_m$

Resolution proof

Given F in CNF: $\overset{\vee \text{ clause in } F}{C_1,\ C_2,\ C_3,\ \ldots,\ C_m}$

Resolution proof

Given F in CNF: $\overset{\vee \text{ clause in } F}{\overset{\text{clause} \in F}{C_1,\ C_2,\ C_3,\ \ldots,\ C_m}}$

Resolution proof

Given F in CNF: $\overset{\vee \text{ clause in } F}{\overset{\text{clause} \in F}{C_1,\ C_2,\ \underset{C_1 = C_1' \sqcup \{p\}}{C_3},\ \ldots,\ C_m}}$

Resolution proof

Given F in CNF: $\overset{\vee \text{ clause in } F}{C_1}, \ \overset{\text{clause} \in F}{C_2}, \ C_3, \ \ldots, \ C_m$

$C_1 = C_1' \sqcup \{p\}$

$C_2 = C_2' \sqcup \{\neg p\}$

Resolution proof

Given F in CNF: $\overset{\vee \text{ clause in } F}{C_1}, \overset{\text{clause} \in F}{C_2}, C_3, \ldots, C_m$

$C_1 = C_1' \sqcup \{p\}$

$C_2 = C_2' \sqcup \{\neg p\}$

$C_3 = C_1' \cup C_2'$

Claim: $F(p_1, \ldots, p_n)$ unsatisfiable $\Rightarrow$ $\exists$resolution proof of $\square$.

Claim: $F(p_1, \ldots, p_n)$ unsatisfiable $\Rightarrow$ $\exists$resolution proof of $\square$.

Proof: by induction on n.

Claim: $F(p_1, \ldots, p_n)$ unsatisfiable $\Rightarrow$ $\exists$resolution proof of $\square$.

Proof: by induction on n.

Base case: $n = 0$; only clause is $\square$;

Claim: $F(p_1, \ldots, p_n)$ unsatisfiable $\Rightarrow$ $\exists$resolution proof of $\square$.

Proof: by induction on n.

Base case: $n = 0$; only clause is $\square$; only CNF's are {} or {$\square$}.

Claim: $F(p_1, \ldots, p_n)$ unsatisfiable $\Rightarrow$ $\exists$resolution proof of $\square$.

Proof: by induction on n.

Base case: $n = 0$; only clause is $\square$; only CNF's are $\{\}$ or $\{\square\}$.

Claim: $F(p_1, \ldots, p_n)$ unsatisfiable $\Rightarrow$ $\exists$resolution proof of $\square$.

Proof: by induction on n.

Base case: $n = 0$; only clause is $\square$; only CNF's are $\{\}$ or $\{\square\}$.

Inductive step:

$$F[\mathbf{false}/p_n]$$

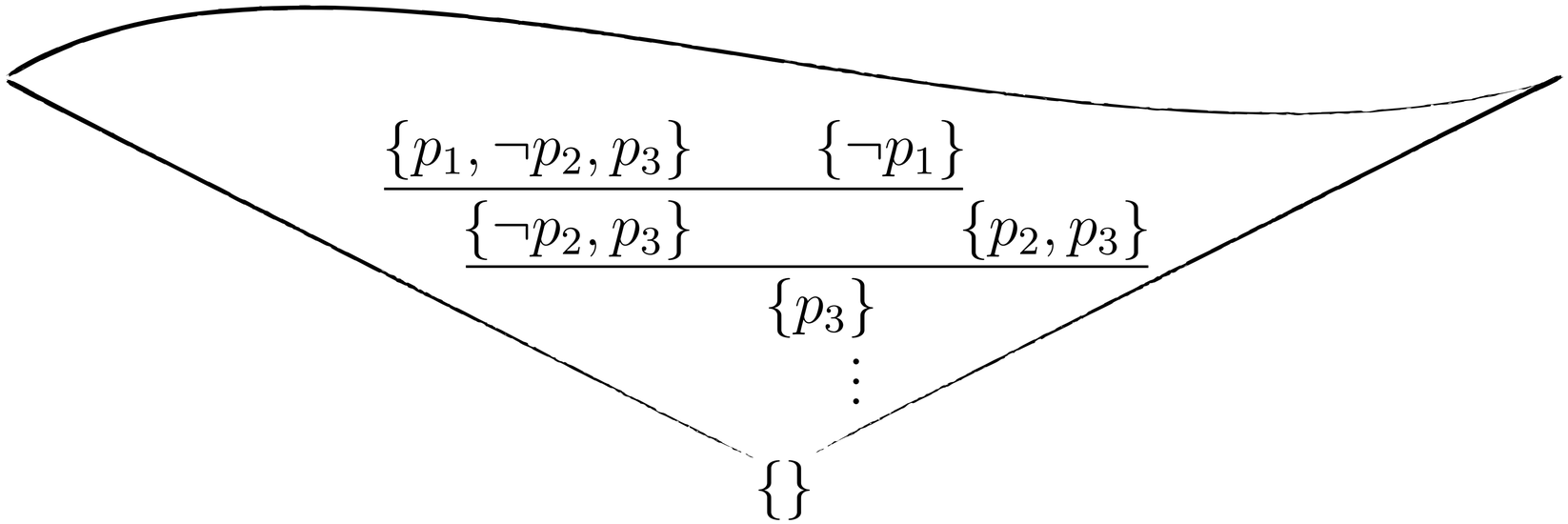
$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

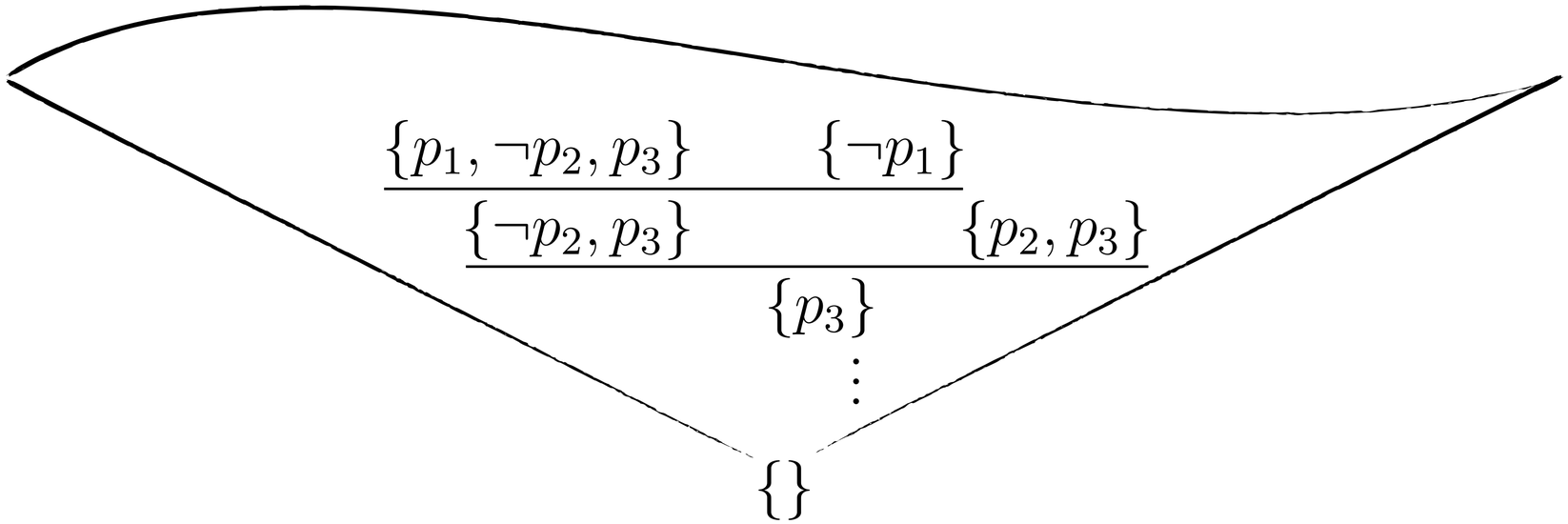
This is a CNF, still unsatisfiable, in fewer variables. So:



$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

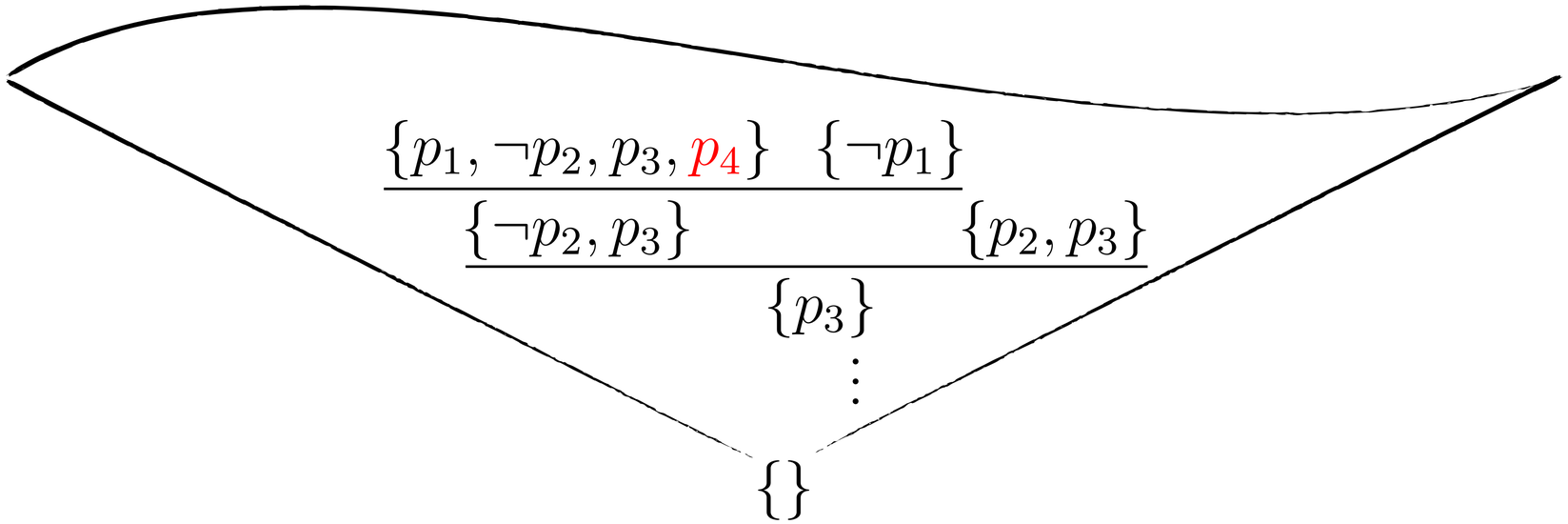
Either $\forall C : C \setminus \{p_n\} = C$ — done



$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

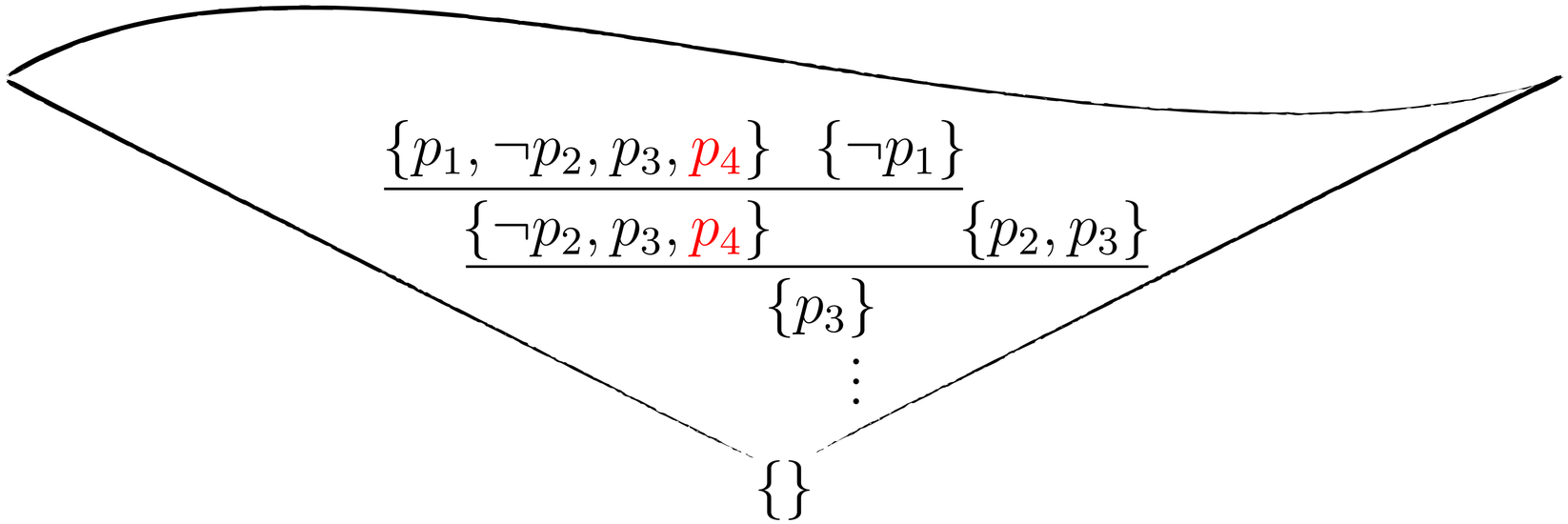
Either $\forall C : C \setminus \{p_n\} = C$ — done, or



$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

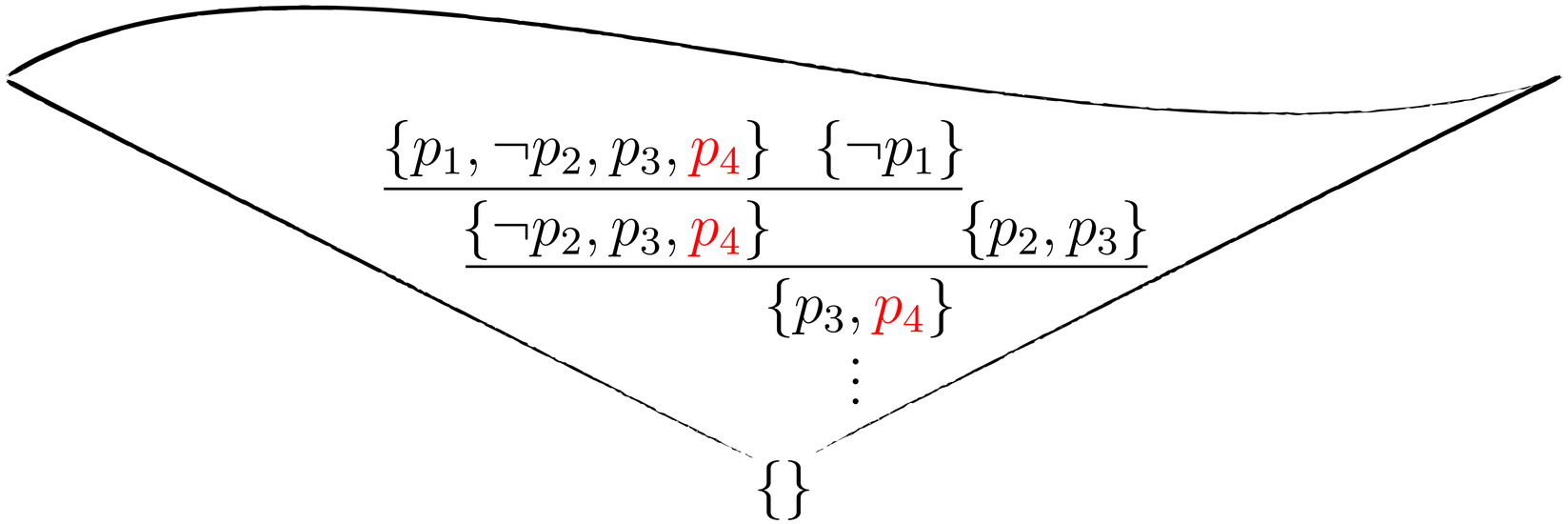
Either $\forall C : C \setminus \{p_n\} = C$ — done, or



$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

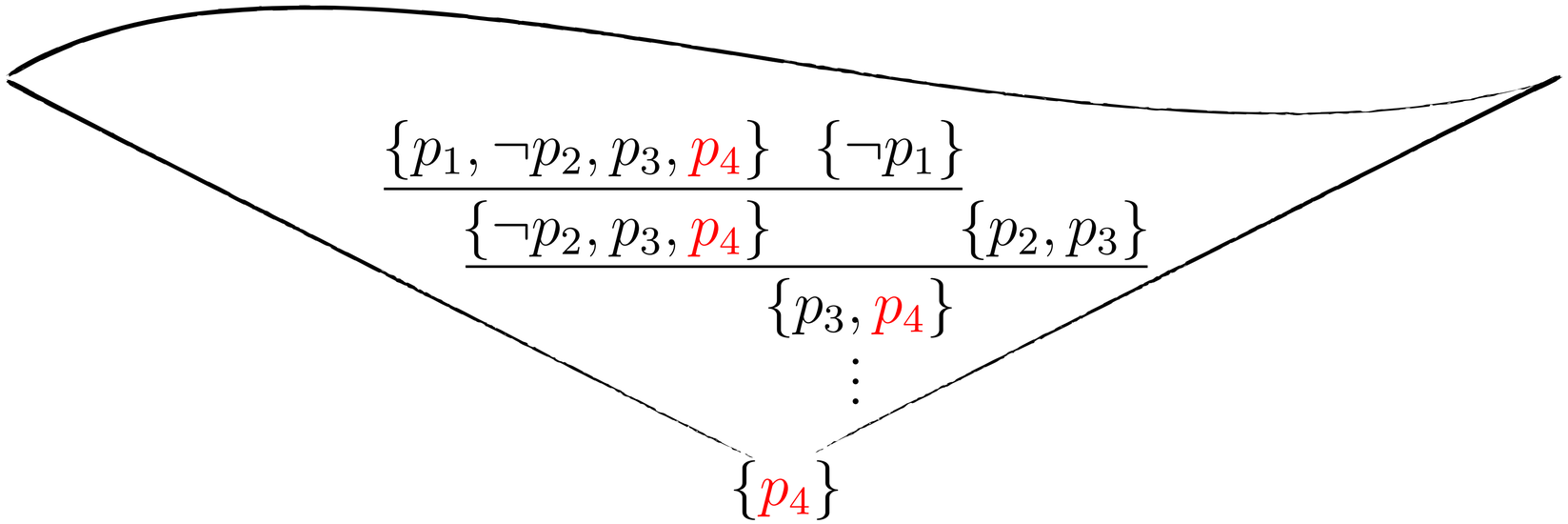
Either $\forall C : C \setminus \{p_n\} = C$ — done, or



$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

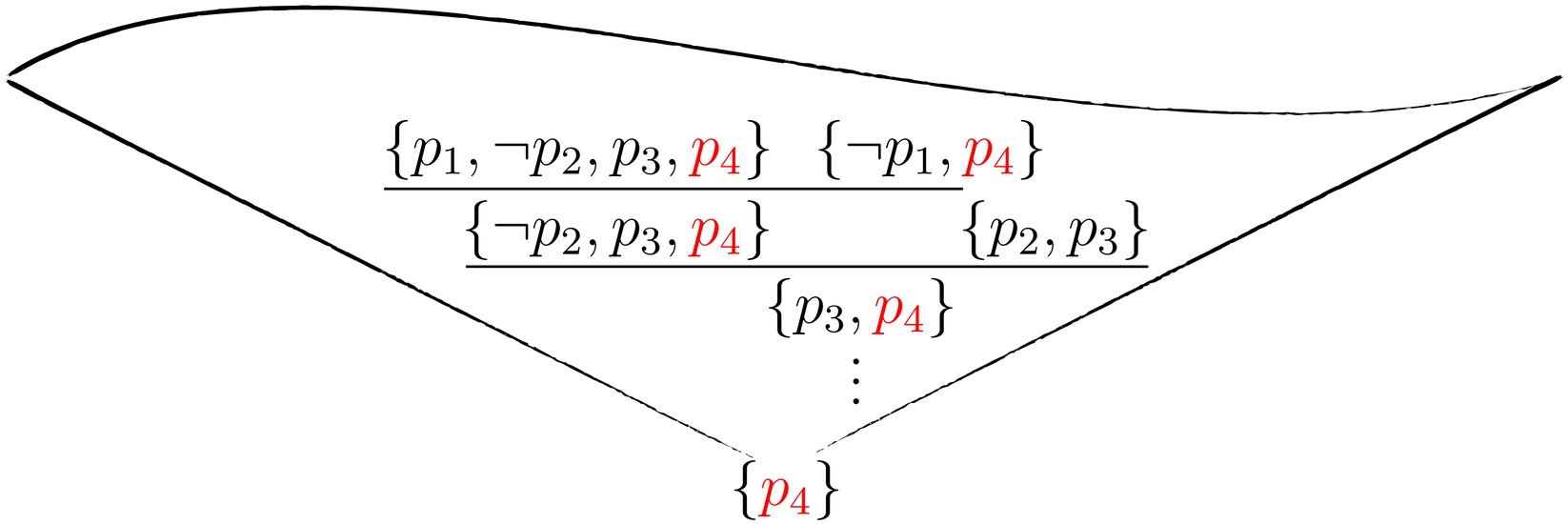
Either $\forall C : C \setminus \{p_n\} = C$ — done, or



$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

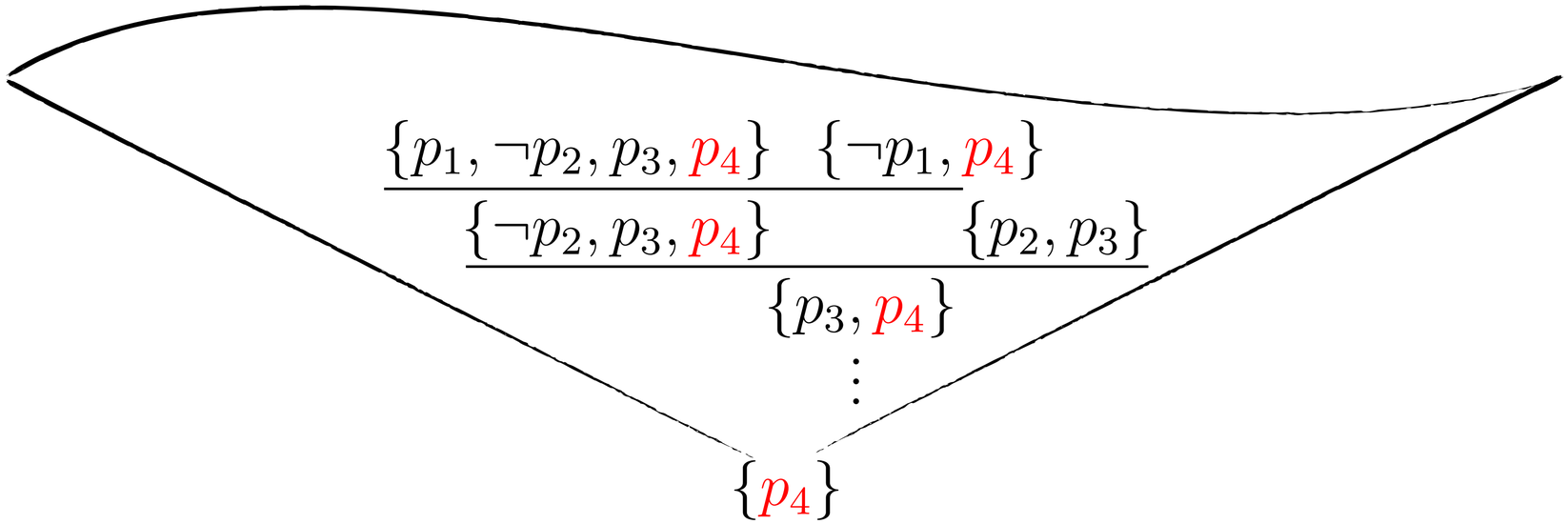
Either $\forall C : C \setminus \{p_n\} = C$ — done, or



$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{p_n\} = C$ — done, or we have a proof of p_n.



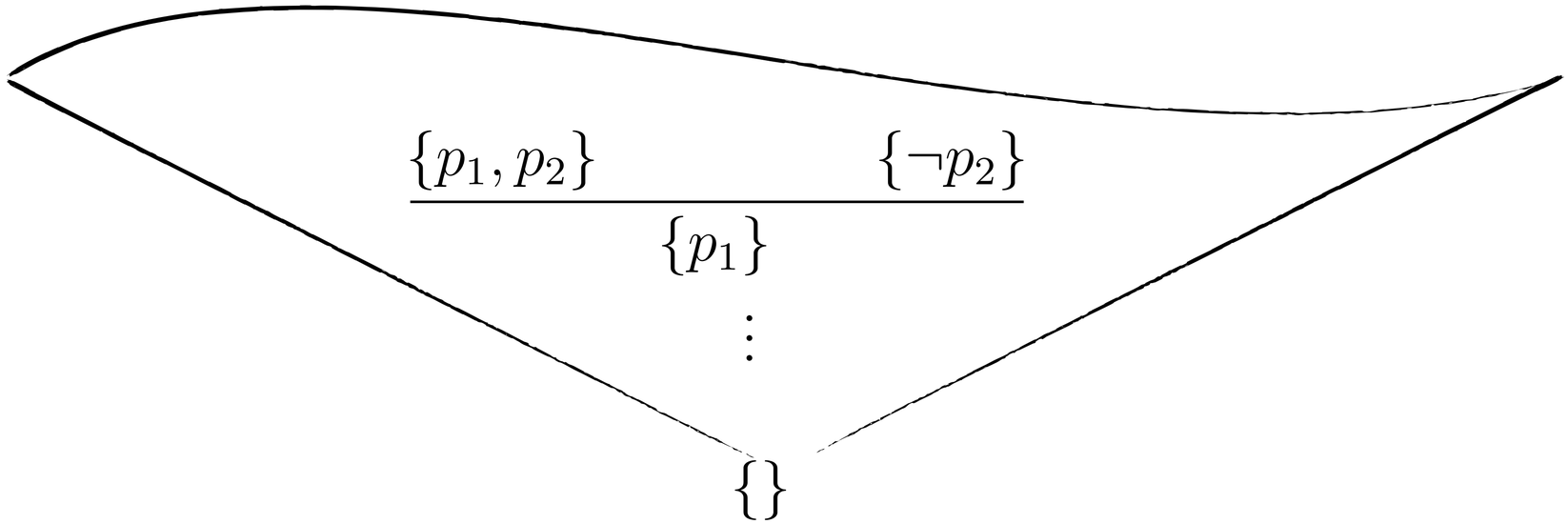
$$F[\mathbf{true}/p_n] = \bigcup_{C \in F : p_n \notin C} C \setminus \{\neg p_n\}$$

$$F[\mathbf{true}/p_n] = \bigcup_{C \in F : p_n \notin C} C \setminus \{\neg p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

$$F[\mathbf{true}/p_n] = \bigcup_{C \in F : p_n \notin C} C \setminus \{\neg p_n\}$$

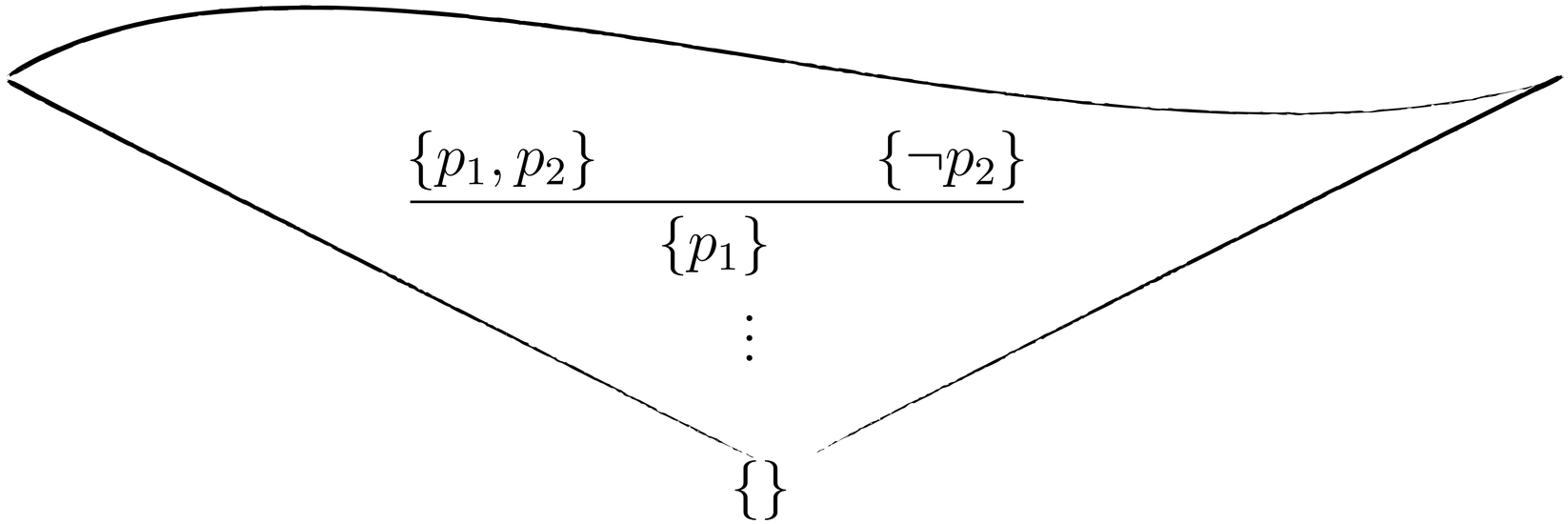
This is a CNF, still unsatisfiable, in fewer variables. So:



$$F[\mathbf{true}/p_n] = \bigcup_{C \in F : p_n \notin C} C \setminus \{\neg p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

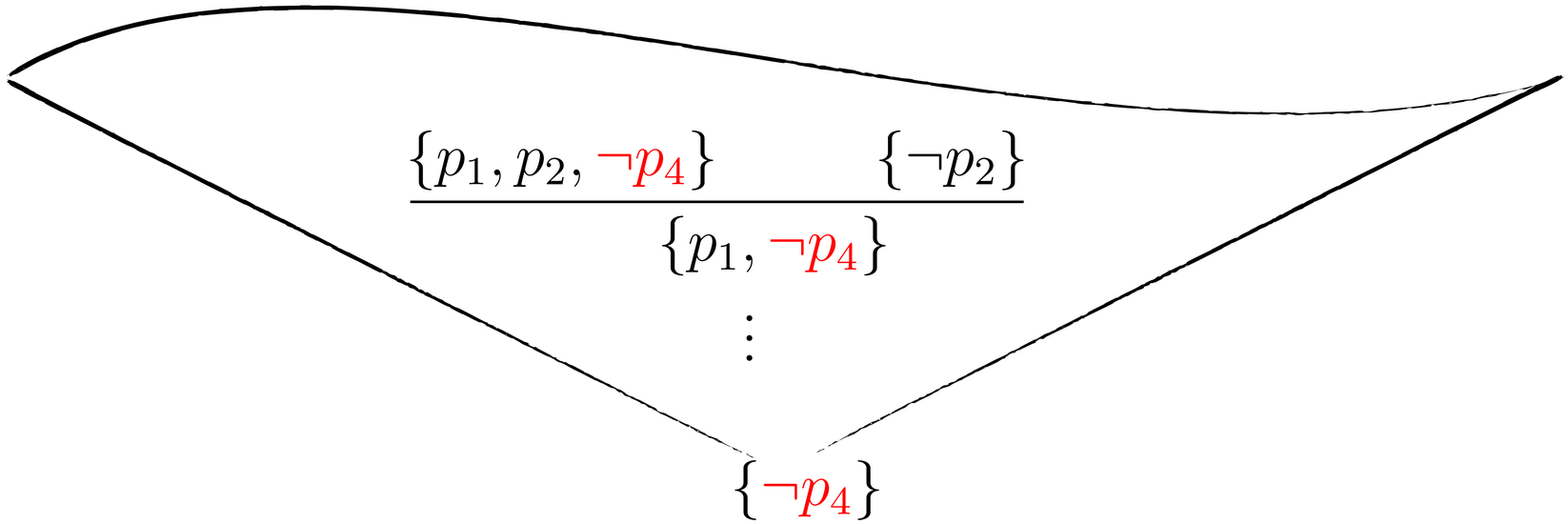
Either $\forall C : C \setminus \{\neg p_n\} = C$ — done



$$F[\mathbf{true}/p_n] = \bigcup_{C \in F : p_n \notin C} C \setminus \{\neg p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{\neg p_n\} = C$ — done, or have a proof of $\neg p_n$.



Claim: $F(p_1, \ldots, p_n)$ unsatisfiable $\Rightarrow$ $\exists$resolution proof of $\square$.

Proof: by induction on n.

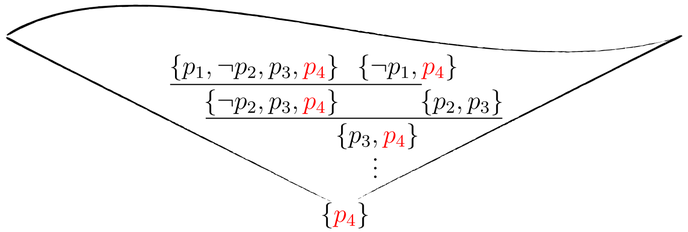
Base case: $n = 0$; only clause is $\square$; only CNF's are ~~$\{\}$~~ or $\{\square\}$.

Inductive step:

$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

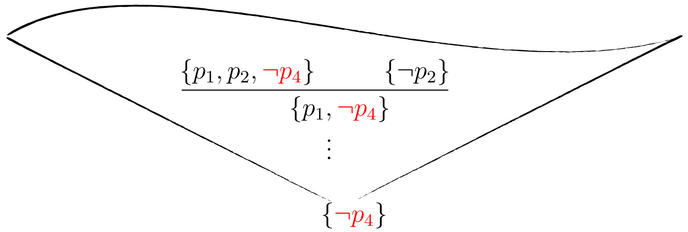
Either $\forall C : C \setminus \{p_n\} = C$ — done, or we have a proof of p_n.



$$F[\mathbf{true}/p_n] = \bigcup_{C \in F : p_n \notin C} C \setminus \{\neg p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{\neg p_n\} = C$ — done, or have a proof of $\neg p_n$.



Claim: $F(p_1, \ldots, p_n)$ unsatisfiable $\Rightarrow$ $\exists$resolution proof of $\square$.

Proof: by induction on n.

Base case: $n = 0$; only clause is $\square$; only CNF's are $\{\}$ or $\{\square\}$.

Inductive step:

$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{p_n\} = C$ — done, or we have a proof of p_n.

$$F[\mathbf{true}/p_n] = \bigcup_{C \in F : p_n \notin C} C \setminus \{\neg p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{\neg p_n\} = C$ — done, or have a proof of $\neg p_n$.

of length $\leq 2^{n+1} - 1$

Claim: $F(p_1, \ldots, p_n)$ unsatisfiable $\Rightarrow$ $\exists$resolution proof of $\square$.

Proof: by induction on n.

Base case: $n = 0$; only clause is $\square$; only CNF's are $\{\}$ or $\{\square\}$.

Inductive step:

$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

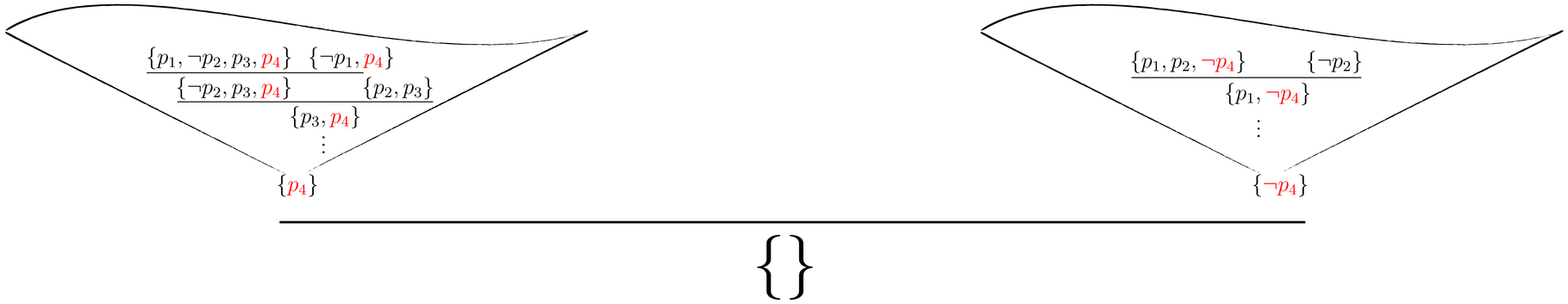
This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{p_n\} = C$ — done, or we have a proof of p_n.

$$F[\mathbf{true}/p_n] = \bigcup_{C \in F : p_n \notin C} C \setminus \{\neg p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{\neg p_n\} = C$ — done, or have a proof of $\neg p_n$.



of length $\leq 2^{n+1} - 1$

Claim: $F(p_1, \ldots, p_n)$ unsatisfiable $\Rightarrow$ $\exists$resolution proof of $\square$.

Proof: by induction on n.

length 1

Base case: $n = 0$; only clause is $\square$; only CNF's are ~~$\{\}$~~ or $\{\square\}$.

Inductive step:

$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{p_n\} = C$ — done, or we have a proof of p_n.

$$F[\mathbf{true}/p_n] = \bigcup_{C \in F : p_n \notin C} C \setminus \{\neg p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{\neg p_n\} = C$ — done, or have a proof of $\neg p_n$.

$$\{\}$$

of length $\leq 2^{n+1} - 1$

Claim: $F(p_1, \ldots, p_n)$ unsatisfiable $\Rightarrow$ $\exists$resolution proof of $\square$.

Proof: by induction on n.

length $1 \leq 2^1 - 1$

Base case: $n = 0$; only clause is $\square$; only CNF's are $\{\}$ or $\{\square\}$.

Inductive step:

$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{p_n\} = C$ — done, or we have a proof of p_n.

$$F[\mathbf{true}/p_n] = \bigcup_{C \in F : p_n \notin C} C \setminus \{\neg p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{\neg p_n\} = C$ — done, or have a proof of $\neg p_n$.

$$\{\}$$

of length $\leq 2^{n+1} - 1$

Claim: $F(p_1, \ldots, p_n)$ unsatisfiable $\Rightarrow$ $\exists$resolution proof of $\square$.

Proof: by induction on n.

length $1 \leq 2^1 - 1$

Base case: $n = 0$; only clause is $\square$; only CNF's are $\{\}$ or $\{\square\}$.

Inductive step:

$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

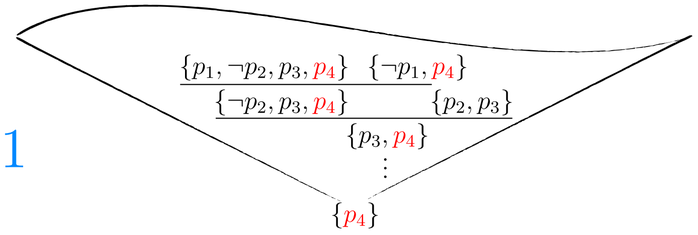
Either $\forall C : C \setminus \{p_n\} = C$ — done, or we have a proof of p_n.

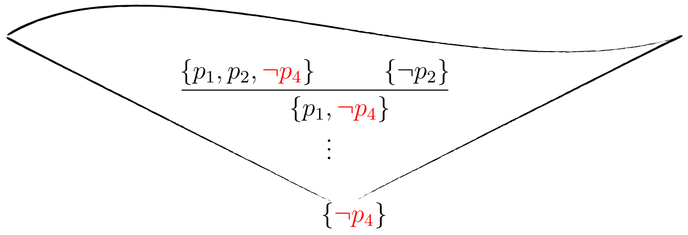
$$F[\mathbf{true}/p_n] = \bigcup_{C \in F : p_n \notin C} C \setminus \{\neg p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{\neg p_n\} = C$ — done, or have a proof of $\neg p_n$.

length $\leq 2^n - 1$





{}

of length $\leq 2^{n+1} - 1$

Claim: $F(p_1, \ldots, p_n)$ unsatisfiable $\Rightarrow$ $\exists$resolution proof of $\square$.

Proof: by induction on n.

length $1 \leq 2^1 - 1$

Base case: $n = 0$; only clause is $\square$; only CNF's are $\{\}$ or $\{\square\}$.

Inductive step:

$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{p_n\} = C$ — done, or we have a proof of p_n.

$$F[\mathbf{true}/p_n] = \bigcup_{C \in F : p_n \notin C} C \setminus \{\neg p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{\neg p_n\} = C$ — done, or have a proof of $\neg p_n$.

length $\leq 2^n - 1$
$(\leq 2^{n+1} - 1)$

of length $\leq 2^{n+1} - 1$

Claim: $F(p_1, \ldots, p_n)$ unsatisfiable $\Rightarrow$ $\exists$resolution proof of $\square$.

Proof: by induction on n.

length $1 \leq 2^1 - 1$

Base case: $n = 0$; only clause is $\square$; only CNF's are $\{\}$ or $\{\square\}$.

Inductive step:

$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

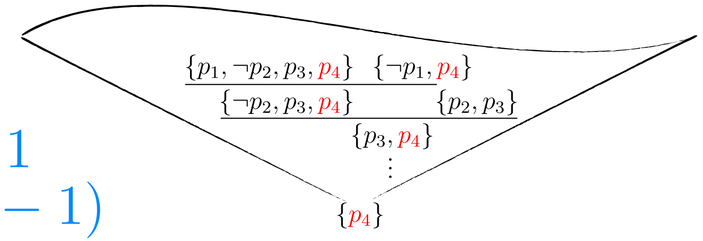
Either $\forall C : C \setminus \{p_n\} = C$ — done, or we have a proof of p_n.

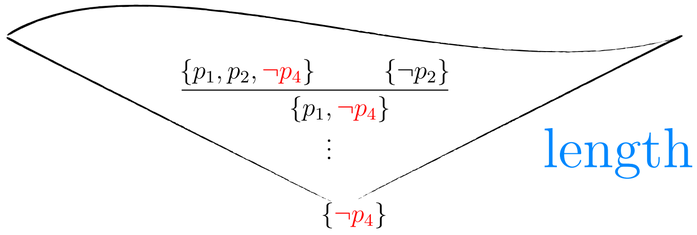
$$F[\mathbf{true}/p_n] = \bigcup_{C \in F : p_n \notin C} C \setminus \{\neg p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{\neg p_n\} = C$ — done, or have a proof of $\neg p_n$.

length $\leq 2^n - 1$
$(\leq 2^{n+1} - 1)$





length $\leq 2^n - 1$

$\{\}$

of length $\leq 2^{n+1} - 1$

Claim: $F(p_1, \ldots, p_n)$ unsatisfiable $\Rightarrow$ $\exists$resolution proof of $\square$.

Proof: by induction on n.

length $1 \leq 2^1 - 1$

Base case: $n = 0$; only clause is $\square$; only CNF's are ~~$\{\}$~~ or $\{\square\}$.

Inductive step:

$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

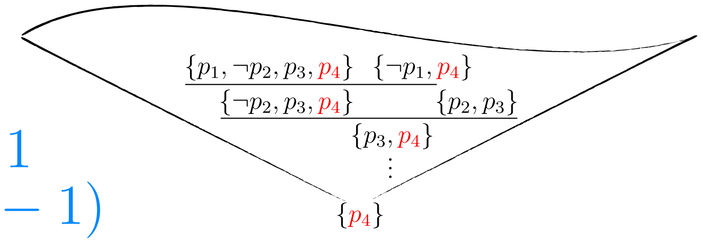
Either $\forall C : C \setminus \{p_n\} = C$ — done, or we have a proof of p_n.

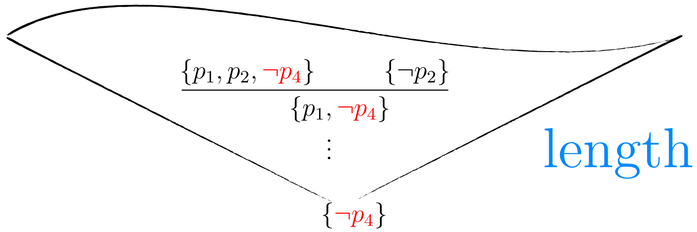
$$F[\mathbf{true}/p_n] = \bigcup_{C \in F : p_n \notin C} C \setminus \{\neg p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{\neg p_n\} = C$ — done, or have a proof of $\neg p_n$.

length $\leq 2^n - 1$
$(\leq 2^{n+1} - 1)$





length $\leq 2^n - 1$

$\{\}$

length $\leq 2 \cdot (2^n - 1) + 1$

of length $\leq 2^{n+1} - 1$

Claim: $F(p_1, \ldots, p_n)$ unsatisfiable $\Rightarrow$ $\exists$resolution proof of $\square$.

Proof: by induction on n.

length $1 \leq 2^1 - 1$

Base case: $n = 0$; only clause is $\square$; only CNF's are $\{\}$ or $\{\square\}$.

Inductive step:

$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

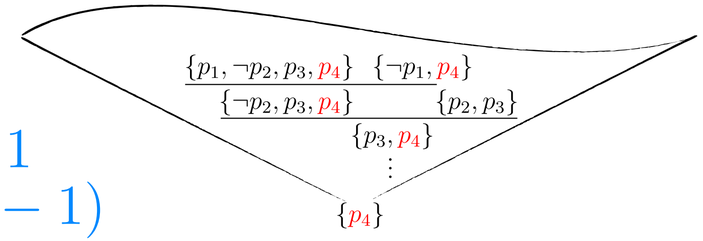
Either $\forall C : C \setminus \{p_n\} = C$ — done, or we have a proof of p_n.

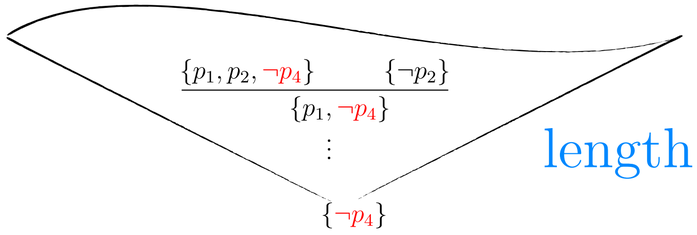
$$F[\mathbf{true}/p_n] = \bigcup_{C \in F : p_n \notin C} C \setminus \{\neg p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{\neg p_n\} = C$ — done, or have a proof of $\neg p_n$.

length $\leq 2^n - 1$
$(\leq 2^{n+1} - 1)$





length $\leq 2^n - 1$

$\{\}$

length $\leq 2 \cdot (2^n - 1) + 1 = 2^{n+1} - 1$

Claim: $F(p_1, \ldots, p_n)$ unsatisfiable $\Rightarrow$ $\exists$resolution proof of $\square$.

Claim: $F(p_1, \ldots, p_n)$ unsatisfiable $\Rightarrow$ $\exists$ positive resolution proof of $\square$.

Claim: $F(p_1, \ldots, p_n)$ unsatisfiable $\Rightarrow$ $\exists$resolution proof of $\square$.

positive — $C_1 = C_1' \sqcup \{p\}$ where $\forall q : \neg q \notin C_1$

$C_2 = C_2' \sqcup \{\neg p\}$

$C_3 = C_1' \cup C_2'$

Claim: $F(p_1, \ldots, p_n)$ unsatisfiable $\Rightarrow$ $\exists$resolution proof of $\square$.

positive — $C_1 = C_1' \sqcup \{p\}$ where $\forall q : \neg q \notin C_1$

$C_2 = C_2' \sqcup \{\neg p\}$

$C_3 = C_1' \cup C_2'$

(Constructive) Proof: by induction on n.

Base case: $n = 0$; only clause is $\square$; only CNF's are ~~$\{\}$~~ or $\{\square\}$.

Claim: $F(p_1, \ldots, p_n)$ unsatisfiable $\Rightarrow$ $\exists$resolution proof of $\square$.

positive — $C_1 = C_1' \sqcup \{p\}$ where $\forall q : \neg q \notin C_1$

$C_2 = C_2' \sqcup \{\neg p\}$

$C_3 = C_1' \cup C_2'$

(Constructive) Proof: by induction on n.

Base case: $n = 0$; only clause is $\square$; only CNF's are ~~$\{\}$~~ or $\{\square\}$.

1-line, certainly positive

Claim: $F(p_1, \ldots, p_n)$ unsatisfiable $\Rightarrow$ $\exists$resolution proof of $\square$.

positive — $C_1 = C_1' \sqcup \{p\}$ where $\forall q : \neg q \notin C_1$

$C_2 = C_2' \sqcup \{\neg p\}$

$C_3 = C_1' \cup C_2'$

(Constructive) Proof: by induction on n.

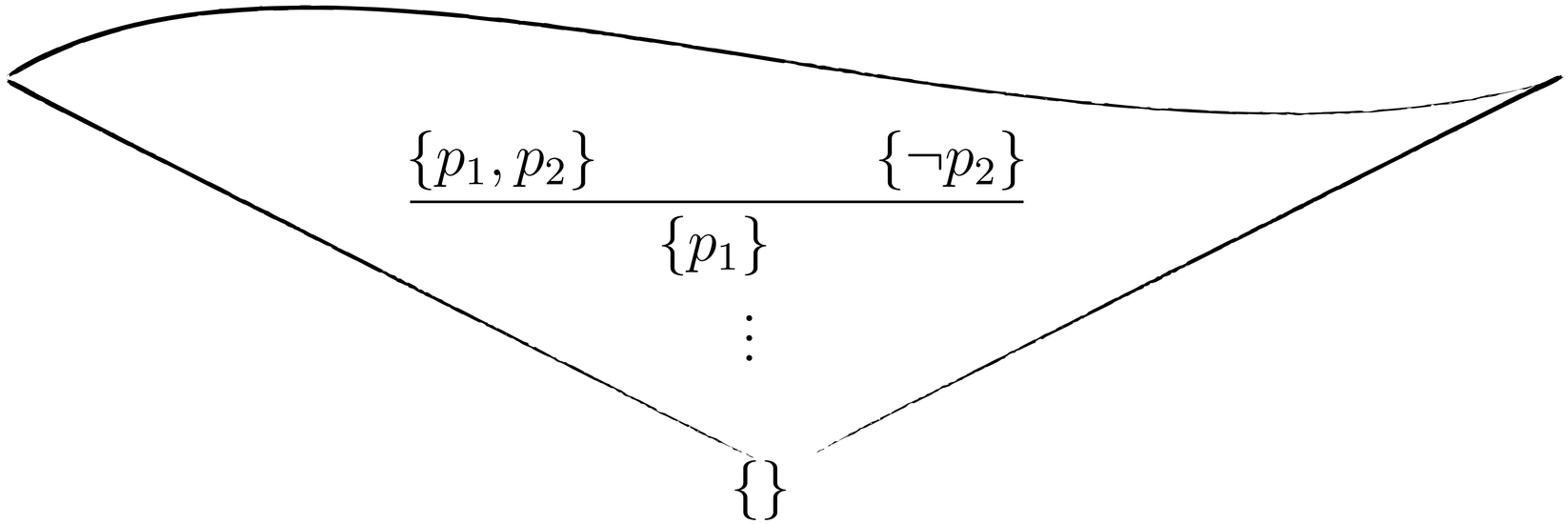
Base case: $n = 0$; only clause is $\square$; only CNF's are ~~$\{\}$~~ or $\{\square\}$.

1-line, certainly positive

Inductive step:

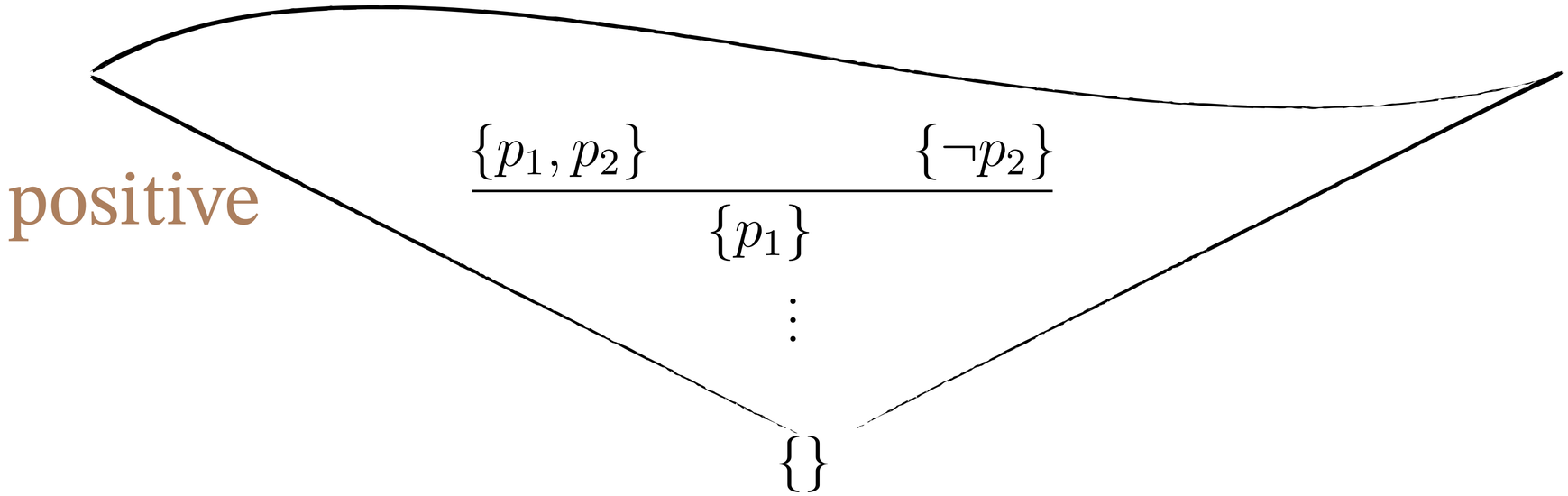
$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:



$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

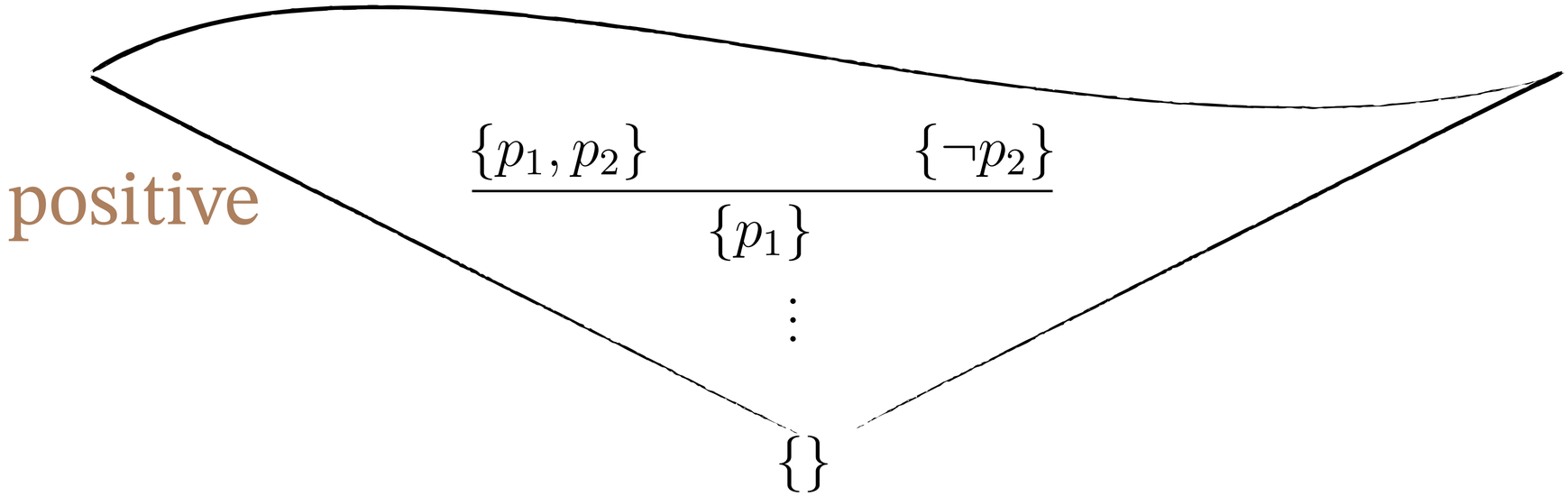
This is a CNF, still unsatisfiable, in fewer variables. So:



$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

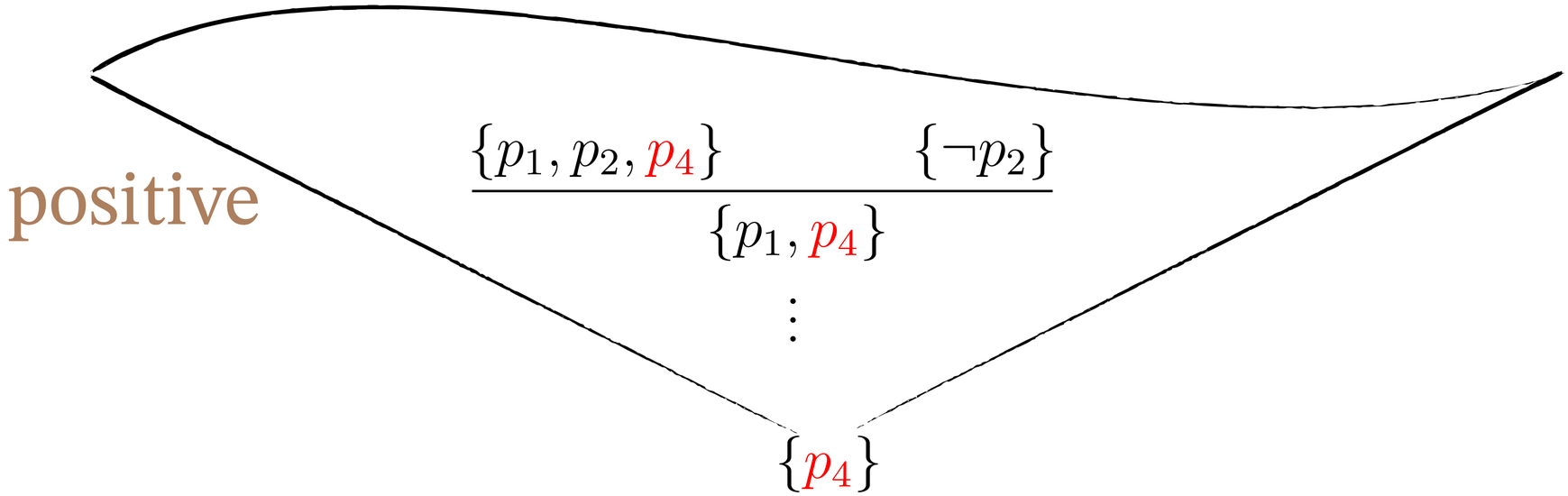
Either $\forall C : C \setminus \{p_n\} = C$ — done



$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

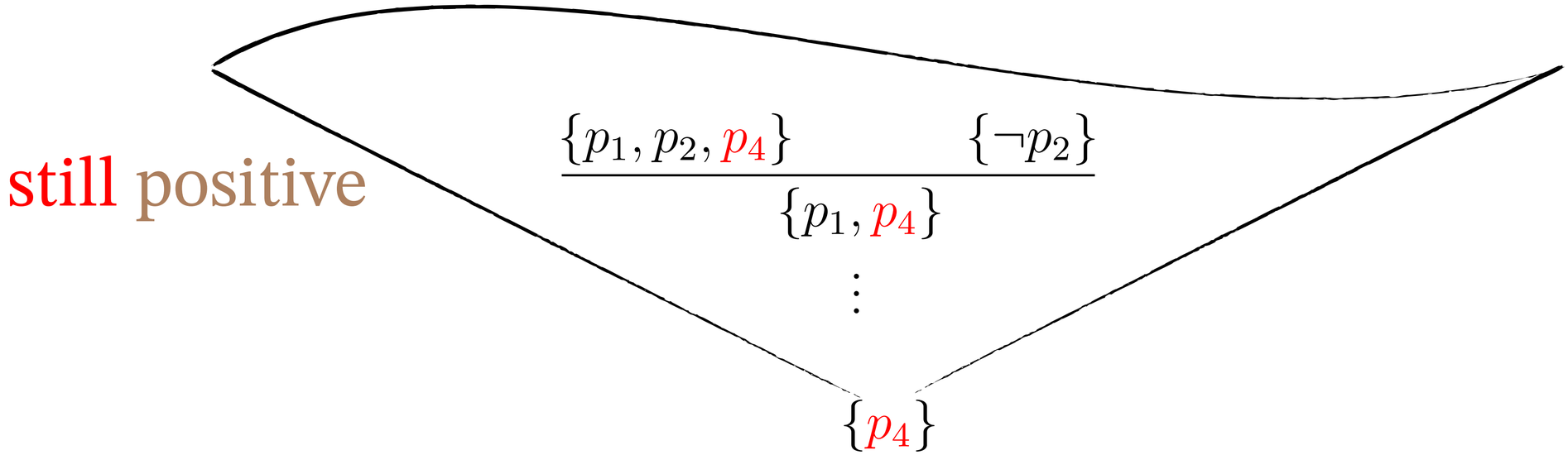
Either $\forall C : C \setminus \{p_n\} = C$ — done, or



$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{p_n\} = C$ — done, or

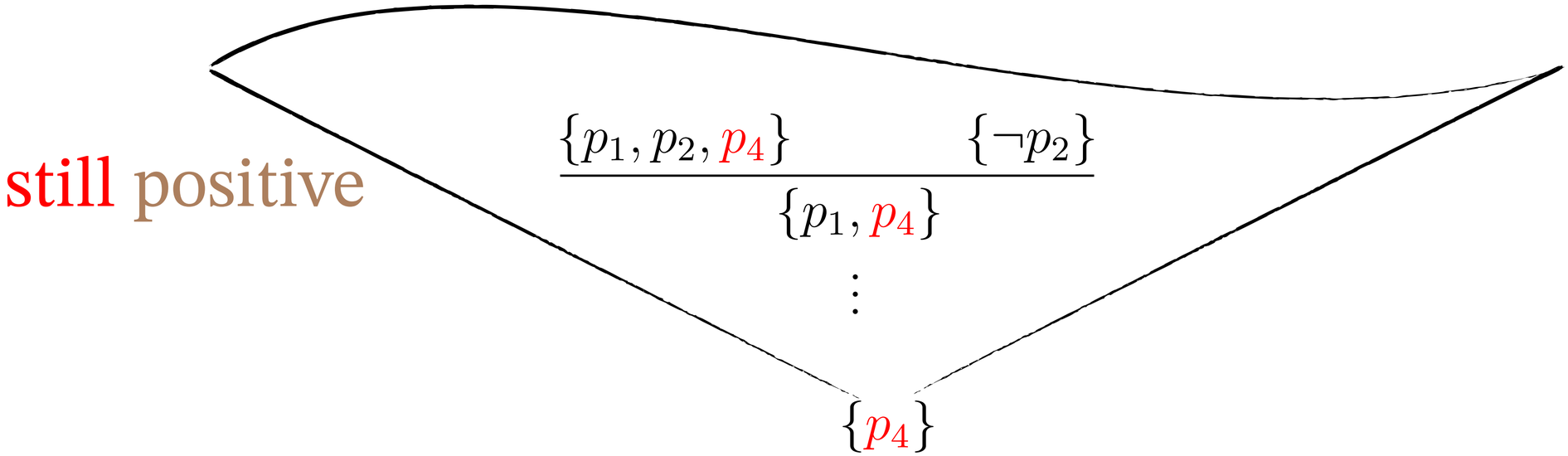


$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

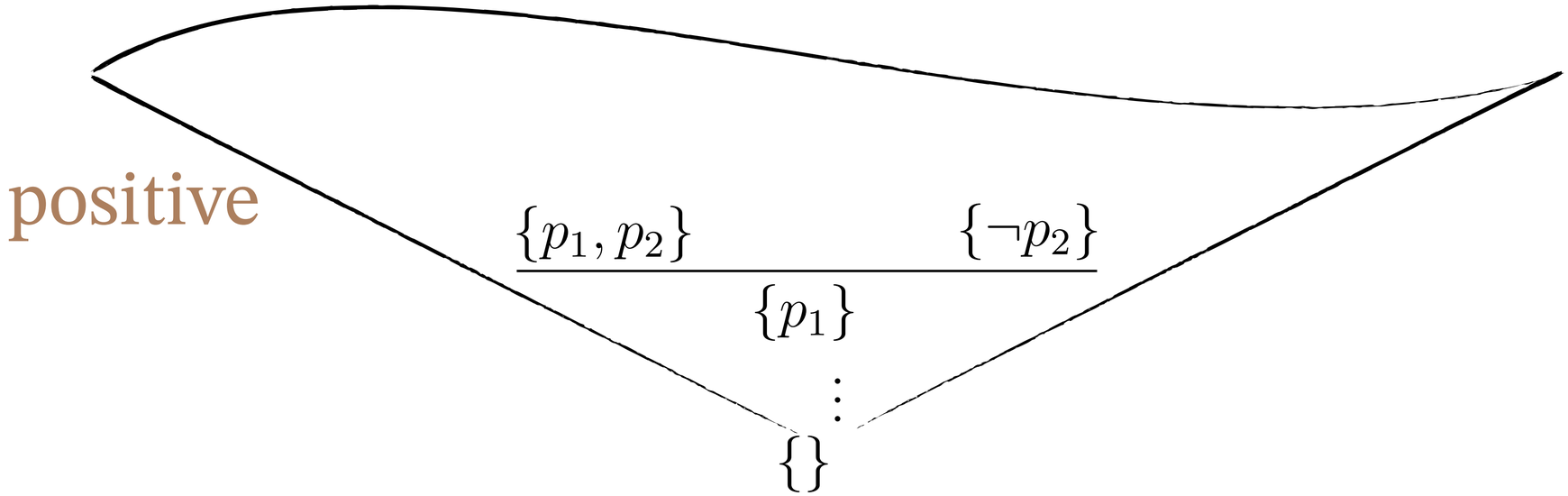
Either $\forall C : C \setminus \{p_n\} = C$ — done, or we have a proof of p_n.

positive



$$F[\mathbf{true}/p_n] = \bigcup_{C \in F : p_n \notin C} C \setminus \{\neg p_n\}$$

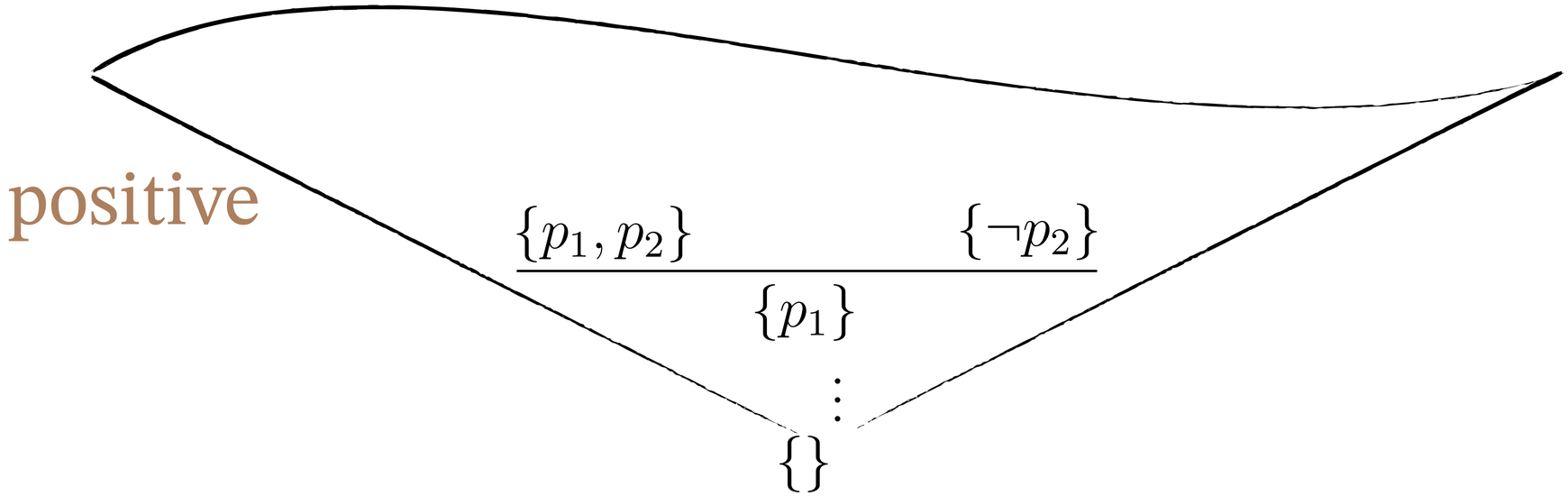
This is a CNF, still unsatisfiable, in fewer variables. So:



$$F[\mathbf{true}/p_n] = \bigcup_{C \in F : p_n \notin C} C \setminus \{\neg p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

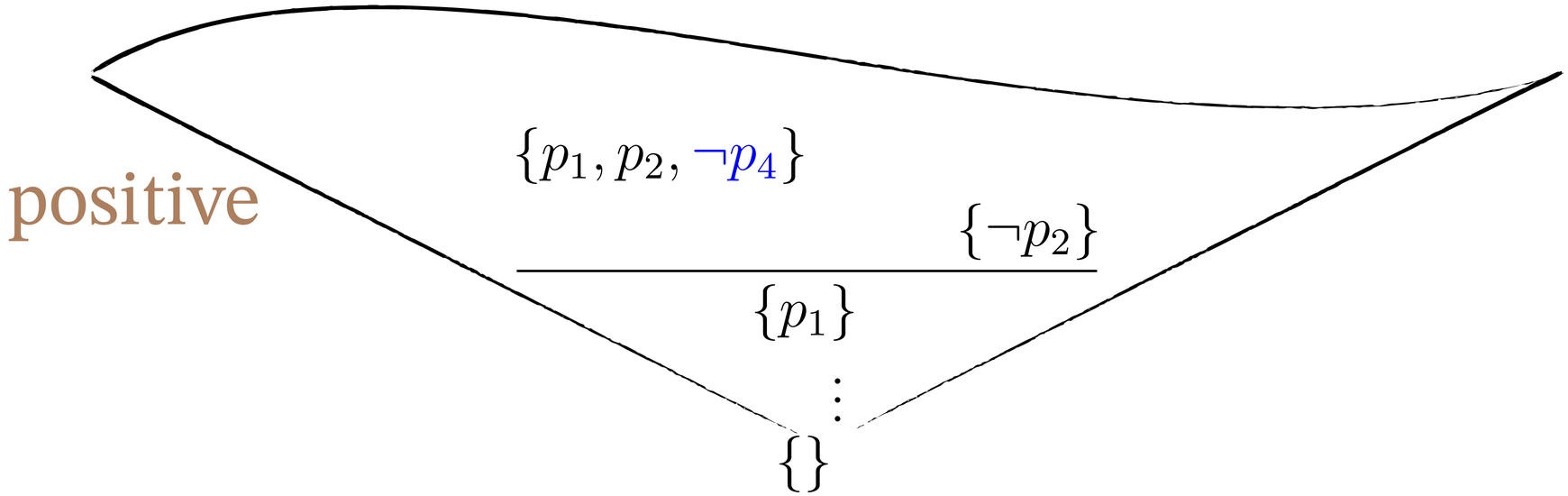
Either $\forall C : C \setminus \{\neg p_n\} = C$ — done



$$F[\mathbf{true}/p_n] = \bigcup_{C \in F : p_n \notin C} C \setminus \{\neg p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

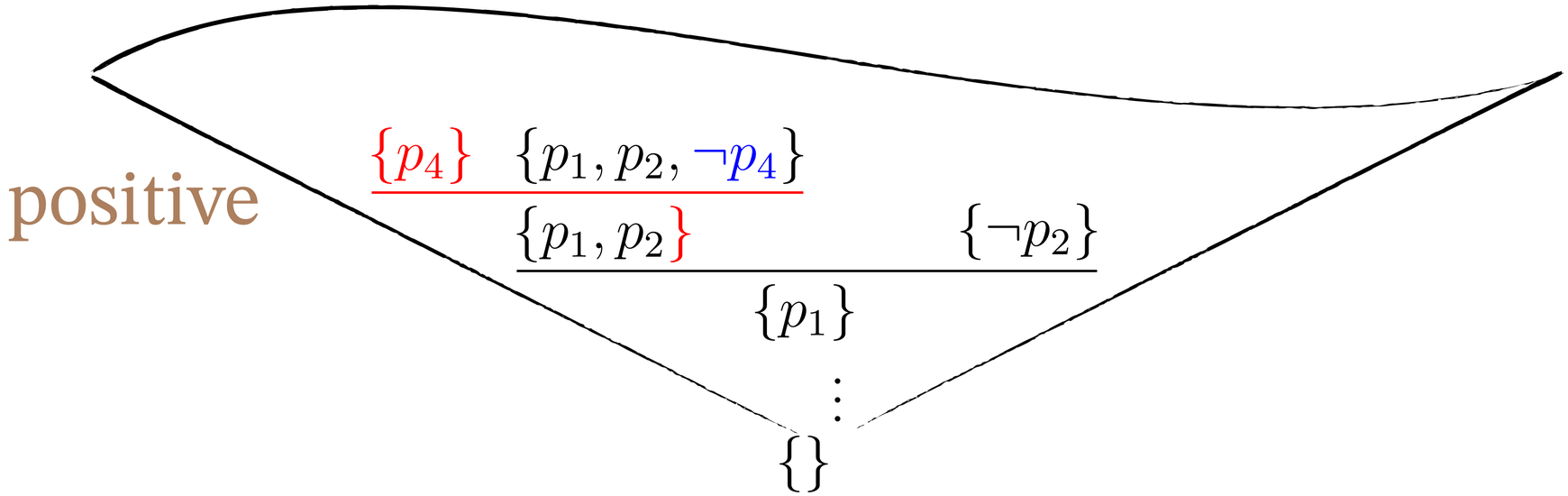
Either $\forall C : C \setminus \{\neg p_n\} = C$ — done, or



$$F[\mathbf{true}/p_n] = \bigcup_{C \in F : p_n \notin C} C \setminus \{\neg p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

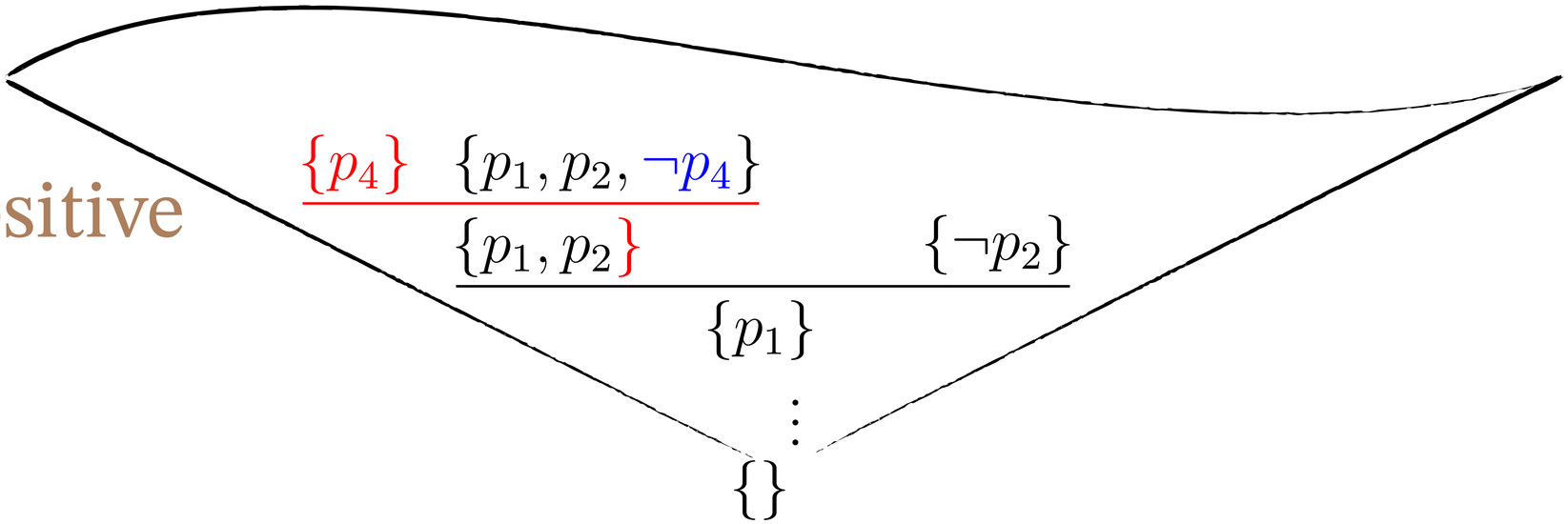
Either $\forall C : C \setminus \{\neg p_n\} = C$ — done, or



$$F[\mathbf{true}/p_n] = \bigcup_{C \in F : p_n \notin C} C \setminus \{\neg p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

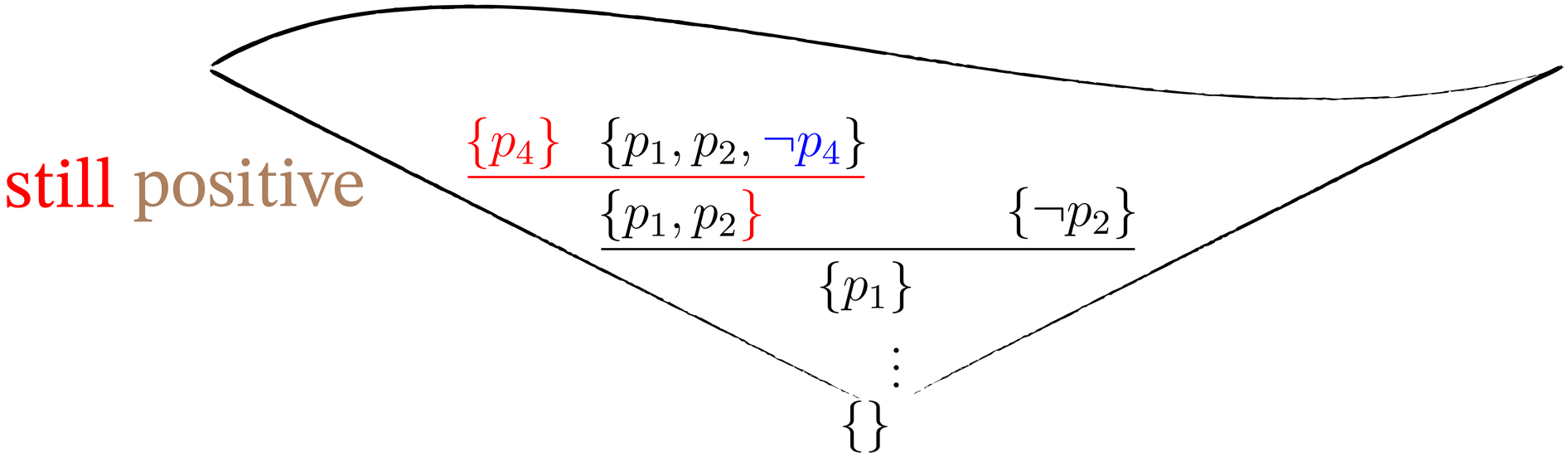
Either $\forall C : C \setminus \{\neg p_n\} = C$ — done, or



$$F[\mathbf{true}/p_n] = \bigcup_{C \in F : p_n \notin C} C \setminus \{\neg p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{\neg p_n\} = C$ — done, or have a longer,
pos. proof of $\square$.



Claim: $F(p_1, \ldots, p_n)$ unsatisfiable $\Rightarrow$ $\exists$resolution proof of $\square$.

positive — $C_1 = C_1' \sqcup \{p\}$ where $\forall q : \neg q \notin C_1$

$C_2 = C_2' \sqcup \{\neg p\}$

$C_3 = C_1' \cup C_2'$

(Constructive) Proof: by induction on n.

Base case: $n = 0$; only clause is $\square$; only CNF's are ~~$\{\}$~~ or $\{\square\}$.

1-line, certainly positive

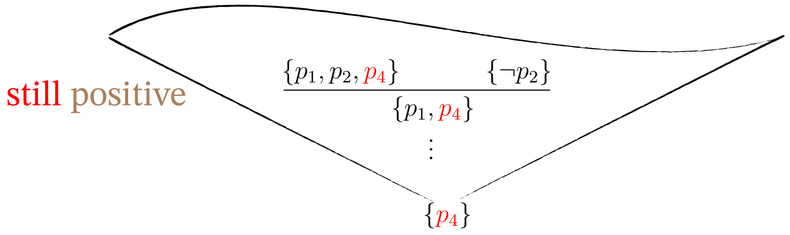
Inductive step:

$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{p_n\} = C$ — done, or we have a proof of p_n.

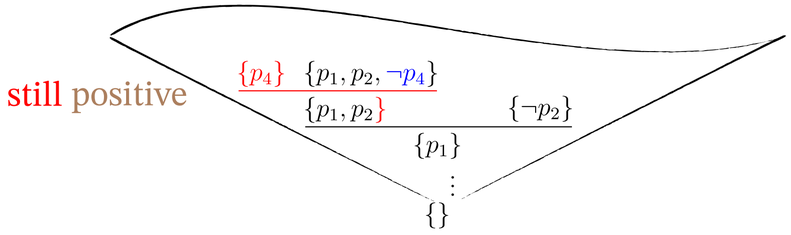
positive



$$F[\mathbf{true}/p_n] = \bigcup_{C \in F : p_n \notin C} C \setminus \{\neg p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{\neg p_n\} = C$ — done, or have a longer, pos. proof of $\square$.



Resolution proof

Given F in CNF: $\overset{\vee \text{ clause in } F}{C_1}, \overset{\text{clause} \in F}{C_2}, C_3, \ldots, C_m$

$C_1 = C_1' \sqcup \{p\}$

$C_2 = C_2' \sqcup \{\neg p\}$

$C_3 = C_1' \cup C_2'$

Claim: $F(p_1, \ldots, p_n)$ unsatisfiable $\Rightarrow$ $\exists$resolution proof of $\square$. (of length $\leq 2^{n+1} - 1$)

Proof: by induction on n.

Base case: $n = 0$; only clause is $\square$; only CNF's are ~~$\{\}$~~ or $\{\square\}$. (length $1 \leq 2^1 - 1$)

Inductive step:

$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

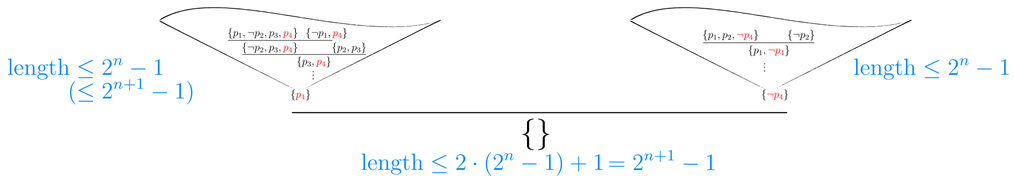
This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{p_n\} = C$ — done, or we have a proof of p_n.

$$F[\mathbf{true}/p_n] = \bigcup_{C \in F : p_n \notin C} C \setminus \{\neg p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{\neg p_n\} = C$ — done, or have a proof of $\neg p_n$.



Claim: $F(p_1, \ldots, p_n)$ unsatisfiable $\Rightarrow$ $\exists$resolution proof of $\square$.

positive — $C_1 = C_1' \sqcup \{p\}$ where $\forall q : \neg q \notin C_1$

$C_2 = C_2' \sqcup \{\neg p\}$

$C_3 = C_1' \cup C_2'$

(Constructive) Proof: by induction on n.

Base case: $n = 0$; only clause is $\square$; only CNF's are ~~$\{\}$~~ or $\{\square\}$. (1-line, certainly positive)

Inductive step:

$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{p_n\} = C$ — done, or we have a proof of p_n. (positive)

$$F[\mathbf{true}/p_n] = \bigcup_{C \in F : p_n \notin C} C \setminus \{\neg p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{\neg p_n\} = C$ — done, or have a longer, pos. proof of $\square$.

Resolution proof

Given F in CNF: $\overset{\vee \text{ clause in } F}{C_1}, \overset{\text{clause} \in F}{C_2}, C_3, \ldots, C_m$

$C_1 = C_1' \sqcup \{p\}$

$C_2 = C_2' \sqcup \{\neg p\}$

$C_3 = C_1' \cup C_2'$

Exercise 2

Claim: $F(p_1, \ldots, p_n)$ unsatisfiable $\Rightarrow$ $\exists$resolution proof of $\square$. (of length $\leq 2^{n+1} - 1$)

Proof: by induction on n.

Base case: $n = 0$; only clause is $\square$; only CNF's are ~~$\{\}$~~ or $\{\square\}$. (length $1 \leq 2^1 - 1$)

Inductive step:

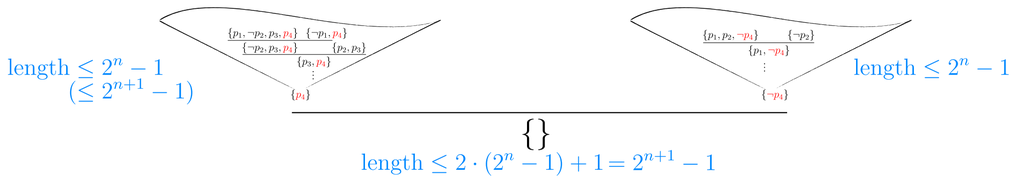
$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\} \qquad F[\mathbf{true}/p_n] = \bigcup_{C \in F : p_n \notin C} C \setminus \{\neg p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{p_n\} = C$ — done, or we have a proof of p_n.

This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{\neg p_n\} = C$ — done, or have a proof of $\neg p_n$.



Exercise 3

Claim: $F(p_1, \ldots, p_n)$ unsatisfiable $\Rightarrow$ $\exists$resolution proof of $\square$.

positive — $C_1 = C_1' \sqcup \{p\}$ where $\forall q : \neg q \notin C_1$

$C_2 = C_2' \sqcup \{\neg p\}$

$C_3 = C_1' \cup C_2'$

(Constructive) Proof: by induction on n.

Base case: $n = 0$; only clause is $\square$; only CNF's are ~~$\{\}$~~ or $\{\square\}$. (1-line, certainly positive)

Inductive step:

$$F[\mathbf{false}/p_n] = \bigcup_{C \in F : \neg p_n \notin C} C \setminus \{p_n\} \qquad F[\mathbf{true}/p_n] = \bigcup_{C \in F : p_n \notin C} C \setminus \{\neg p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{p_n\} = C$ — done, or we have a proof of p_n. (positive)

This is a CNF, still unsatisfiable, in fewer variables. So:

Either $\forall C : C \setminus \{\neg p_n\} = C$ — done, or have a longer, pos. proof of $\square$.

Interpolant

Interpolant

"Resolution has feasible monotone interpolation."

Interpolant

"Resolution has feasible monotone interpolation."

Input:

- $F(\mathbf{x}, \mathbf{z}) \wedge G(\mathbf{y}, \mathbf{z})$ is unsatisfiable
- $C_1, ..., C_m$ is a resolution refutation
- $\mathbf{z}$ only appears positively in F

Interpolant

"Resolution has feasible monotone interpolation."

Input:

- $F(\mathbf{x}, \mathbf{z}) \wedge G(\mathbf{y}, \mathbf{z})$ is unsatisfiable
- $C_1, ..., C_m$ is a resolution refutation
- $\mathbf{z}$ only appears positively in F

Output:

straight-line programs $e_1(\mathbf{z}), ..., e_m(\mathbf{z})$,

Interpolant

"Resolution has feasible monotone interpolation."

Input:

- $F(\mathbf{x}, \mathbf{z}) \wedge G(\mathbf{y}, \mathbf{z})$ is unsatisfiable
- $C_1, \ldots, C_m$ is a resolution refutation
- $\mathbf{z}$ only appears positively in F

Output:

straight-line programs $e_1(\mathbf{z}), \ldots, e_m(\mathbf{z})$,

1. $C_i \in F$
   $e_i = \mathbf{false}$

2. $C_i \in G$
   $e_i = \mathbf{true}$

Interpolant

"Resolution has feasible monotone interpolation."

Input:

- $F(\mathbf{x}, \mathbf{z}) \wedge G(\mathbf{y}, \mathbf{z})$ is unsatisfiable
- $C_1, ..., C_m$ is a resolution refutation
- $\mathbf{z}$ only appears positively in F

Output:

straight-line programs $e_1(\mathbf{z}), ..., e_m(\mathbf{z})$,

1. $C_i \in F$
   $e_i =$ **false**

2. $C_i \in G$
   $e_i =$ **true**

3. $C_i = C_j \backslash \{x\} \cup C_k \backslash \{\neg x\}$
   $e_i = e_j \vee e_k$

4. $C_i = C_j \backslash \{y\} \cup C_k \backslash \{\neg y\}$
   $e_i = e_j \wedge e_k$

5. $C_i = C_j \backslash \{z\} \cup C_k \backslash \{\neg z\}$
   $e_i = (e_j \vee z) \wedge e_k$

Interpolant

"Resolution has feasible monotone interpolation."

Input:

- $F(\mathbf{x}, \mathbf{z}) \wedge G(\mathbf{y}, \mathbf{z})$ is unsatisfiable
- $C_1, \ldots, C_m$ is a resolution refutation
- $\mathbf{z}$ only appears positively in F

Output:

straight-line programs $e_1(\mathbf{z}), \ldots, e_m(\mathbf{z})$,

1. $C_i \in F$
   $e_i = \mathbf{false}$

2. $C_i \in G$
   $e_i = \mathbf{true}$

3. $C_i = C_j \backslash \{x\} \cup C_k \backslash \{\neg x\}$
   $e_i = e_j \vee e_k$

4. $C_i = C_j \backslash \{y\} \cup C_k \backslash \{\neg y\}$
   $e_i = e_j \wedge e_k$

5. $C_i = C_j \backslash \{z\} \cup C_k \backslash \{\neg z\}$
   $e_i = (e_j \vee z) \wedge e_k$
   z only appears positively in e_i

Interpolant

"Resolution has feasible monotone interpolation."

Input:

- $F(\mathbf{x}, \mathbf{z}) \wedge G(\mathbf{y}, \mathbf{z})$ is unsatisfiable
- $C_1, \ldots, C_m$ is a resolution refutation
- $\mathbf{z}$ only appears positively in F

Output:

straight-line programs $e_1(\mathbf{z}), \ldots, e_m(\mathbf{z})$, monotone in $\mathbf{z}$,

1. $C_i \in F$
   $e_i = \mathbf{false}$

2. $C_i \in G$
   $e_i = \mathbf{true}$

3. $C_i = C_j \backslash \{x\} \cup C_k \backslash \{\neg x\}$
   $e_i = e_j \vee e_k$

4. $C_i = C_j \backslash \{y\} \cup C_k \backslash \{\neg y\}$
   $e_i = e_j \wedge e_k$

5. $C_i = C_j \backslash \{z\} \cup C_k \backslash \{\neg z\}$
   $e_i = (e_j \vee z) \wedge e_k$
   z only appears positively in e_i

Interpolant

"Resolution has feasible monotone interpolation."

Input:

- $F(\mathbf{x}, \mathbf{z}) \wedge G(\mathbf{y}, \mathbf{z})$ is unsatisfiable
- $C_1, \ldots, C_m$ is a resolution refutation
- $\mathbf{z}$ only appears positively in F

Output:

straight-line programs $e_1(\mathbf{z}), \ldots, e_m(\mathbf{z})$, monotone in $\mathbf{z}$, such that:

$$\mathrm{v}[\![e_i]\!] = 0 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{x},+\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![F]\!] = 0$$

1. $C_i \in F$
   $e_i =$ **false**

2. $C_i \in G$
   $e_i =$ **true**

3. $C_i = C_j \backslash \{x\} \cup C_k \backslash \{\neg x\}$
   $e_i = e_j \vee e_k$

4. $C_i = C_j \backslash \{y\} \cup C_k \backslash \{\neg y\}$
   $e_i = e_j \wedge e_k$

5. $C_i = C_j \backslash \{z\} \cup C_k \backslash \{\neg z\}$
   $e_i = (e_j \vee z) \wedge e_k$
   z only appears positively in e_i

Interpolant

"Resolution has feasible monotone interpolation."

Input:

- $F(\mathbf{x}, \mathbf{z}) \wedge G(\mathbf{y}, \mathbf{z})$ is unsatisfiable
- $C_1, \ldots, C_m$ is a resolution refutation
- $\mathbf{z}$ only appears positively in F

Output:

straight-line programs $e_1(\mathbf{z}), \ldots, e_m(\mathbf{z})$, monotone in $\mathbf{z}$, such that:

$$\mathrm{v}[\![e_i]\!] = 0 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{x},+\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![F]\!] = 0$$

discard any y and any $\neg$ z

1. $C_i \in F$
   $e_i =$ **false**

2. $C_i \in G$
   $e_i =$ **true**

3. $C_i = C_j \backslash \{x\} \cup C_k \backslash \{\neg x\}$
   $e_i = e_j \vee e_k$

4. $C_i = C_j \backslash \{y\} \cup C_k \backslash \{\neg y\}$
   $e_i = e_j \wedge e_k$

5. $C_i = C_j \backslash \{z\} \cup C_k \backslash \{\neg z\}$
   $e_i = (e_j \vee z) \wedge e_k$
   z only appears positively in e_i

Interpolant

"Resolution has feasible monotone interpolation."

Input:

- $F(\mathbf{x}, \mathbf{z}) \wedge G(\mathbf{y}, \mathbf{z})$ is unsatisfiable
- $C_1, \ldots, C_m$ is a resolution refutation
- $\mathbf{z}$ only appears positively in F

Output:

straight-line programs $e_1(\mathbf{z}), \ldots, e_m(\mathbf{z})$, monotone in $\mathbf{z}$, such that:

discard any y and any $\neg z$

$$\mathrm{v}[\![e_i]\!] = 0 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{x},+\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![F]\!] = 0$$

$$\mathrm{v}[\![e_i]\!] = 1 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{y},\mathbf{z}}]\!] = 0 \implies$$

1. $C_i \in F$
   $e_i = \mathbf{false}$

2. $C_i \in G$
   $e_i = \mathbf{true}$

3. $C_i = C_j \backslash \{x\} \cup C_k \backslash \{\neg x\}$
   $e_i = e_j \vee e_k$

4. $C_i = C_j \backslash \{y\} \cup C_k \backslash \{\neg y\}$
   $e_i = e_j \wedge e_k$

5. $C_i = C_j \backslash \{z\} \cup C_k \backslash \{\neg z\}$
   $e_i = (e_j \vee z) \wedge e_k$
   z only appears positively in e_i

Interpolant

"Resolution has feasible monotone interpolation."

Input:

- $F(\mathbf{x}, \mathbf{z}) \wedge G(\mathbf{y}, \mathbf{z})$ is unsatisfiable
- $C_1, \ldots, C_m$ is a resolution refutation
- $\mathbf{z}$ only appears positively in F

Output:

straight-line programs $e_1(\mathbf{z}), \ldots, e_m(\mathbf{z})$, monotone in $\mathbf{z}$, such that:

discard any y and any $\neg z$

$$\mathrm{v}[\![e_i]\!] = 0 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{x},+\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![F]\!] = 0$$

$$\mathrm{v}[\![e_i]\!] = 1 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{y},\mathbf{z}}]\!] = 0 \implies$$

discard any x

1. $C_i \in F$
   $e_i = \mathbf{false}$

2. $C_i \in G$
   $e_i = \mathbf{true}$

3. $C_i = C_j \backslash \{x\} \cup C_k \backslash \{\neg x\}$
   $e_i = e_j \vee e_k$

4. $C_i = C_j \backslash \{y\} \cup C_k \backslash \{\neg y\}$
   $e_i = e_j \wedge e_k$

5. $C_i = C_j \backslash \{z\} \cup C_k \backslash \{\neg z\}$
   $e_i = (e_j \vee z) \wedge e_k$
   z only appears positively in e_i

Interpolant

"Resolution has feasible monotone interpolation."

Input:

- $F(\mathbf{x}, \mathbf{z}) \wedge G(\mathbf{y}, \mathbf{z})$ is unsatisfiable
- $C_1, \ldots, C_m$ is a resolution refutation
- $\mathbf{z}$ only appears positively in F

Output:

straight-line programs $e_1(\mathbf{z}), \ldots, e_m(\mathbf{z})$, monotone in $\mathbf{z}$, such that:

discard any y and any $\neg z$

$$\mathrm{v}[\![e_i]\!] = 0 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{x},+\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![F]\!] = 0$$

$$\mathrm{v}[\![e_i]\!] = 1 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{y},\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![G]\!] = 0$$

discard any x

1. $C_i \in F$
   $e_i = \mathbf{false}$

2. $C_i \in G$
   $e_i = \mathbf{true}$

3. $C_i = C_j \backslash \{x\} \cup C_k \backslash \{\neg x\}$
   $e_i = e_j \vee e_k$

4. $C_i = C_j \backslash \{y\} \cup C_k \backslash \{\neg y\}$
   $e_i = e_j \wedge e_k$

5. $C_i = C_j \backslash \{z\} \cup C_k \backslash \{\neg z\}$
   $e_i = (e_j \vee z) \wedge e_k$
   z only appears positively in e_i

Interpolant

"Resolution has feasible monotone interpolation."

Input:

- $F(\mathbf{x}, \mathbf{z}) \wedge G(\mathbf{y}, \mathbf{z})$ is unsatisfiable
- $C_1, \ldots, C_m$ is a resolution refutation
- $\mathbf{z}$ only appears positively in F

Output:

straight-line programs $e_1(\mathbf{z}), \ldots, e_m(\mathbf{z})$, monotone in $\mathbf{z}$, such that:

discard any y and any $\neg z$

$$\mathrm{v}[\![e_i]\!] = 0 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{x},+\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![F]\!] = 0$$

$$\mathrm{v}[\![e_i]\!] = 1 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{y},\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![G]\!] = 0$$

discard any x

Thanks to Mahesh Viswanathan for the correct and convenient inductive hypothesis!
https://courses.grainger.illinois.edu/cs498mv/fa2018/CraigInterpolation.pdf

1. $C_i \in F$
   $e_i = \mathbf{false}$

2. $C_i \in G$
   $e_i = \mathbf{true}$

3. $C_i = C_j \backslash \{x\} \cup C_k \backslash \{\neg x\}$
   $e_i = e_j \vee e_k$

4. $C_i = C_j \backslash \{y\} \cup C_k \backslash \{\neg y\}$
   $e_i = e_j \wedge e_k$

5. $C_i = C_j \backslash \{z\} \cup C_k \backslash \{\neg z\}$
   $e_i = (e_j \vee z) \wedge e_k$
   z only appears positively in e_i

Interpolant

"Resolution has feasible monotone interpolation."

Input:

- $F(\mathbf{x}, \mathbf{z}) \wedge G(\mathbf{y}, \mathbf{z})$ is unsatisfiable
- $C_1, \ldots, C_m$ is a resolution refutation
- $\mathbf{z}$ only appears positively in F

Output:

straight-line programs $e_1(\mathbf{z}), \ldots, e_m(\mathbf{z})$, monotone in $\mathbf{z}$, such that:

discard any y and any $\neg z$

$$\mathrm{v}[\![e_i]\!] = 0 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{x},+\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![F]\!] = 0$$

$$\mathrm{v}[\![e_i]\!] = 1 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{y},\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![G]\!] = 0$$

discard any x

so $\vDash \neg e_m(\mathbf{z}) \rightarrow F(\mathbf{x}, \mathbf{z})$

and $\vDash e_m(\mathbf{z}) \rightarrow G(\mathbf{y}, \mathbf{z})$

Thanks to Mahesh Viswanathan for the correct and convenient inductive hypothesis!
https://courses.grainger.illinois.edu/cs498mv/fa2018/CraigInterpolation.pdf

1. $C_i \in F$
   $e_i = \mathbf{false}$

2. $C_i \in G$
   $e_i = \mathbf{true}$

3. $C_i = C_j \backslash \{x\} \cup C_k \backslash \{\neg x\}$
   $e_i = e_j \vee e_k$

4. $C_i = C_j \backslash \{y\} \cup C_k \backslash \{\neg y\}$
   $e_i = e_j \wedge e_k$

5. $C_i = C_j \backslash \{z\} \cup C_k \backslash \{\neg z\}$
   $e_i = (e_j \vee z) \wedge e_k$
   z only appears positively in e_i

$$\mathrm{v}[\![e_i]\!] = 0 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{x},+\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![F]\!] = 0$$

1. $C_i \in F$
   $e_i = \mathbf{false}$

$$\mathrm{v}[\![e_i]\!] = 1 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{y},\,\pm\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![G]\!] = 0$$

$$\mathrm{v}[\![e_i]\!] = 0 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{x},+\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![F]\!] = 0$$

1. $C_i \in F$
   $e_i = \textbf{false}$

never

$$\mathrm{v}[\![e_i]\!] = 1 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{y},\,\pm\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![G]\!] = 0$$

$$\mathrm{v}[\![e_i]\!] = 0 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{x},+\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![F]\!] = 0$$

1. $C_i \in F$
   $e_i = \textbf{false}$

never — yes, vacuously

$$\mathrm{v}[\![e_i]\!] = 1 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{y},\,\pm\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![G]\!] = 0$$

$$\mathrm{v}[\![e_i]\!] = 0 \text{ and } \mathrm{v}[\![\underbrace{C_i|_{\mathbf{x},+\mathbf{z}}}_{= C_i}]\!] = 0 \implies \mathrm{v}[\![F]\!] = 0$$

1. $C_i \in F$

   $e_i = \mathbf{false}$

never — yes, vacuously

$$\mathrm{v}[\![e_i]\!] = 1 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{y},\,\pm\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![G]\!] = 0$$

$$\mathrm{v}[\![e_i]\!] = 0 \text{ and } \mathrm{v}[\![\underbrace{C_i|_{\mathbf{x},+\mathbf{z}}}_{= C_i}]\!] = 0 \implies \mathrm{v}[\![F]\!] = 0$$

yes, as C_i is a clause in F

1. $C_i \in F$

   $e_i = \textbf{false}$

never

$$\mathrm{v}[\![e_i]\!] = 1 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{y},\,\pm\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![G]\!] = 0$$

yes, vacuously

$$\mathrm{v}[\![e_i]\!] = 0 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{x},+\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![F]\!] = 0$$

2. $C_i \in G$

   $e_i = \mathbf{true}$

$$\mathrm{v}[\![e_i]\!] = 1 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{y},\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![G]\!] = 0$$

$$\mathrm{v}[\![e_i]\!] = 0 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{x},+\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![F]\!] = 0$$

never — yes, vacuously

2. $C_i \in G$

   $e_i = \mathbf{true}$

$$\mathrm{v}[\![e_i]\!] = 1 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{y},\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![G]\!] = 0$$

$$\mathrm{v}[\![e_i]\!] = 0 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{x},+\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![F]\!] = 0$$

never — yes, vacuously

2. $C_i \in G$

   $e_i = \mathbf{true}$

$= C_i$ — yes, as C_i is a clause in G

$$\mathrm{v}[\![e_i]\!] = 1 \text{ and } \mathrm{v}[\![\overline{C_i|_{\mathbf{y},\mathbf{z}}}]\!] = 0 \implies \mathrm{v}[\![G]\!] = 0$$

$$\mathrm{v}[\![e_i]\!] = 0 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{x},+\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![F]\!] = 0$$

Note $C_i|_{\mathrm{x},+\mathrm{z}} = C_j\backslash\{x\}|_{\mathrm{x},+\mathrm{z}} \cup C_k\backslash\{\neg x\}|_{\mathrm{x},+\mathrm{z}}$
$= C_j|_{\mathrm{x},+\mathrm{z}}\backslash\{x\} \cup C_k|_{\mathrm{x},+\mathrm{z}}\backslash\{\neg x\}$

Assume:

- $\mathrm{v}[\![e_i]\!] = 0$ — so $\mathrm{v}[\![e_j]\!] = 0$ and $\mathrm{v}[\![e_k]\!] = 0$;
- $\mathrm{v}[\![C_i|_{\mathrm{x},+\mathrm{z}}]\!] = 0$ — so $\mathrm{v}[\![C_j|_{\mathrm{x},+\mathrm{z}} \setminus \{x\}]\!] = 0$ and $\mathrm{v}[\![C_k|_{\mathrm{x},+\mathrm{z}} \setminus \{\neg x\}]\!] = 0$.

Then:

- if $\mathrm{v}[\![x]\!] = 0$ then $\mathrm{v}[\![C_j|_{\mathrm{x},+\mathrm{z}}]\!] = 0$, so $\mathrm{v}[\![F]\!] = 0$ by induction;
- if $\mathrm{v}[\![x]\!] = 1$ then $\mathrm{v}[\![C_k|_{\mathrm{x},+\mathrm{z}}]\!] = 0$, so $\mathrm{v}[\![F]\!] = 0$ by induction.

3. $C_i = C_j\backslash\{x\} \cup C_k\backslash\{\neg x\}$
   $e_i = e_j \vee e_k$

$$\mathrm{v}[\![e_i]\!] = 1 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{y},\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![G]\!] = 0$$

Note $C_i|_{\mathrm{y},\mathrm{z}} = C_j|_{\mathrm{y},\mathrm{z}} \cup C_k|_{\mathrm{y},\mathrm{z}}$

Assume:

- $\mathrm{v}[\![C_i|_{\mathrm{y},\mathrm{z}}]\!] = 0$ — so $\mathrm{v}[\![C_j|_{\mathrm{y},\mathrm{z}}]\!] = 0$ and $\mathrm{v}[\![C_k|_{\mathrm{y},\mathrm{z}}]\!] = 0$;
- $\mathrm{v}[\![e_i]\!] = 1$ — so
  - either $\mathrm{v}[\![e_j]\!] = 1$, which gives $\mathrm{v}[\![G]\!] = 0$ by induction;
  - or $\mathrm{v}[\![e_k]\!] = 1$, which gives $\mathrm{v}[\![G]\!] = 0$ by induction.

$$\mathrm{v}[\![e_i]\!] = 0 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{x},+\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![F]\!] = 0$$

- $\mathrm{v}[\![C_j|_{\mathrm{x},+\mathrm{z}}]\!] = 0$ and $\mathrm{v}[\![C_k|_{\mathrm{y},+\mathrm{z}}]\!] = 0$;
- $\mathrm{v}[\![e_j]\!] = 0$ or $\mathrm{v}[\![e_k]\!] = 0$;
- so $\mathrm{v}[\![F]\!] = 0$ by induction.

4. $C_i = C_j \backslash \{y\} \cup C_k \backslash \{\neg y\}$
   $e_i = e_j \wedge e_k$

$$\mathrm{v}[\![e_i]\!] = 1 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{y},\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![G]\!] = 0$$

- $\mathrm{v}[\![C_j|_{\mathrm{y},\mathrm{z}} \backslash \{y\}]\!] = 0$ and $\mathrm{v}[\![C_k|_{\mathrm{y},\mathrm{z}} \backslash \{\neg y\}]\!] = 0$;
- $\mathrm{v}[\![e_j]\!] = 1$ and $\mathrm{v}[\![e_k]\!] = 1$;
- so $\mathrm{v}[\![G]\!] = 0$ by induction.

$$\mathrm{v}[\![e_i]\!] = 0 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{x},+\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![F]\!] = 0$$

Assume:

- $\mathrm{v}[\![C_i|_{\mathrm{x},+\mathrm{z}}]\!] = 0$ — so $\mathrm{v}[\![C_j|_{\mathrm{x},+\mathrm{z}} \setminus \{z\}]\!] = 0$ and $\mathrm{v}[\![C_k|_{\mathrm{x},+\mathrm{z}}]\!] = 0$.
- $\mathrm{v}[\![e_i]\!] = 0$ — so ($\mathrm{v}[\![e_j]\!] = 0$ and $\mathrm{v}[\![z]\!] = 0$) or $\mathrm{v}[\![e_k]\!] = 0$;

Then:

- $\mathrm{v}[\![C_j|_{\mathrm{x},+\mathrm{z}}]\!] = 0$, so $\mathrm{v}[\![F]\!] = 0$ by induction.

Then:

- $\mathrm{v}[\![F]\!] = 0$ by induction.

5. $C_i = C_j \setminus \{z\} \cup C_k \setminus \{\neg z\}$
   $e_i = (e_j \vee z) \wedge e_k$

$$\mathrm{v}[\![e_i]\!] = 1 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{y},\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![G]\!] = 0$$

Assume:

- $\mathrm{v}[\![C_i|_{\mathrm{y},\mathrm{z}}]\!] = 0$ — so $\mathrm{v}[\![C_j|_{\mathrm{y},\mathrm{z}} \setminus \{z\}]\!] = 0$ and $\mathrm{v}[\![C_k|_{\mathrm{y},\mathrm{z}} \setminus \{\neg z\}]\!] = 0$.
- $\mathrm{v}[\![e_i]\!] = 1$ — so ($\mathrm{v}[\![e_j]\!] = 1$ or $\mathrm{v}[\![z]\!] = 1$) and $\mathrm{v}[\![e_k]\!] = 1$.

Then:

- either $\mathrm{v}[\![z]\!] = 0$,
  so $\mathrm{v}[\![C_j|_{\mathrm{y},\mathrm{z}}]\!] = 0$, $\mathrm{v}[\![G]\!] = 1$;
- or $\mathrm{v}[\![z]\!] = 1$ ...

Then:

- $\mathrm{v}[\![C_k|_{\mathrm{y},\mathrm{z}}]\!] = 0$, so $\mathrm{v}[\![G]\!] = 1$.

Interpolant

"Resolution has feasible monotone interpolation."

Input:

- $F(\mathbf{x}, \mathbf{z}) \wedge G(\mathbf{y}, \mathbf{z})$ is unsatisfiable
- $C_1, \ldots, C_m$ is a resolution refutation
- $\mathbf{z}$ only appears positively in F

Output:

straight-line programs $e_1(\mathbf{z}), \ldots, e_m(\mathbf{z})$, monotone in $\mathbf{z}$, such that:

discard any y and any $\neg$ z

$$\mathrm{v}[\![e_i]\!] = 0 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{x},+\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![F]\!] = 0$$

$$\mathrm{v}[\![e_i]\!] = 1 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{y},\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![G]\!] = 0$$

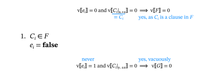
discard any x

so $\vDash \neg e_m(\mathbf{z}) \rightarrow F(\mathbf{x}, \mathbf{z})$

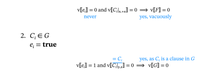
and $\vDash e_m(\mathbf{z}) \rightarrow G(\mathbf{y}, \mathbf{z})$

Thanks to Mahesh Viswanathan for the correct and convenient inductive hypothesis!
https://courses.grainger.illinois.edu/cs498mv/fa2018/CraigInterpolation.pdf

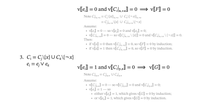
1. $C_i \in F$
   $e_i = \mathbf{false}$



2. $C_i \in G$
   $e_i = \mathbf{true}$

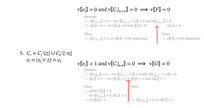


3. $C_i = C_j \backslash \{x\} \cup C_k \backslash \{\neg x\}$
   $e_i = e_j \vee e_k$



4. $C_i = C_j \backslash \{y\} \cup C_k \backslash \{\neg y\}$
   $e_i = e_j \wedge e_k$

5. $C_i = C_j \backslash \{z\} \cup C_k \backslash \{\neg z\}$
   $e_i = (e_j \vee z) \wedge e_k$
   z only appears positively in e_i



Interpolant

"Resolution has feasible monotone interpolation."

Input:

- $F(\mathbf{x}, \mathbf{z}) \wedge G(\mathbf{y}, \mathbf{z})$ is unsatisfiable
- $C_1, \ldots, C_m$ is a resolution refutation
- $\mathbf{z}$ only appears positively in F

Output:

straight-line programs $e_1(\mathbf{z}), \ldots, e_m(\mathbf{z})$, monotone in $\mathbf{z}$, such that:

discard any y and any $\neg$ z

$$\mathrm{v}[\![e_i]\!] = 0 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{x},+\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![F]\!] = 0$$

$$\mathrm{v}[\![e_i]\!] = 1 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{y},\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![G]\!] = 0$$

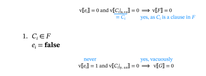
discard any x

so $\vDash \neg e_m(\mathbf{z}) \rightarrow F(\mathbf{x}, \mathbf{z})$

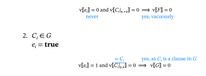
and $\vDash e_m(\mathbf{z}) \rightarrow G(\mathbf{y}, \mathbf{z})$

Thanks to Mahesh Viswanathan for the correct and convenient inductive hypothesis!
https://courses.grainger.illinois.edu/cs498mv/fa2018/CraigInterpolation.pdf

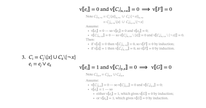
1. $C_i \in F$
   $e_i = \mathbf{false}$



2. $C_i \in G$
   $e_i = \mathbf{true}$

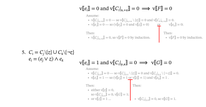


3. $C_i = C_j \backslash \{x\} \cup C_k \backslash \{\neg x\}$
   $e_i = e_j \vee e_k$



4. $C_i = C_j \backslash \{y\} \cup C_k \backslash \{\neg y\}$
   $e_i = e_j \wedge e_k$

5. $C_i = C_j \backslash \{z\} \cup C_k \backslash \{\neg z\}$
   $e_i = (e_j \vee z) \wedge e_k$
   z only appears positively in e_i



Exercise 4

Problems and solutions
courtesy of James Worrell

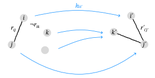
Logic & Proof Sheet 1

Jingjie Yang
HT'26

Propositional logic

(CNF)
A formula in n variables amounts to a function $\{0,1\}^n \to \{0,1\}$.
/ straight-line program

Example: graph homomorphism



every vertex maps to ≥ 1 vertex' every vertex maps to ≤ 1 vertex' edges map to edges'

$$\bigwedge_i \bigvee_{i'} h_{ii'} \quad \wedge \quad \bigwedge_i \bigwedge_{i' \neq j'} \neg h_{ii'} \vee \neg h_{ij'} \quad \wedge \quad \bigwedge_{i,j,i',j'} \neg h_{ii'} \vee \neg h_{jj'} \vee \neg r_{ij} \vee r'_{i'j'}$$

neg pos

This formula in $\mathbf{h}$, $\mathbf{r}$, $\mathbf{r'}$ is satisfiable iff there is a graph homomorphism $\mathbf{h}$.

Exercise 1

Write K_m for the complete graph on m vertices.

(a) homomorphism $K_m \to G$ = m-clique in G

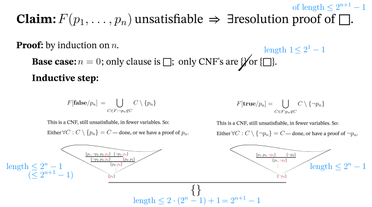
(b) homomorphism $G \to K_m$ = m-colouring of G

Resolution proof

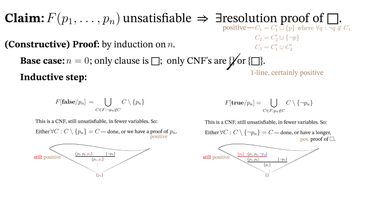
Given F in CNF: $\overset{\vee \text{ clause in } F}{C_1}, \overset{\text{clause} \in F}{C_2}, C_3, \ldots, C_m$

$C_1 = C_1' \sqcup \{p\}$
$C_2 = C_2' \sqcup \{\neg p\}$
$C_3 = C_1' \cup C_2'$

Exercise 2



Exercise 3



Interpolant

"Resolution has feasible monotone interpolation."

Input:
- $F(\mathbf{x}, \mathbf{z}) \wedge G(\mathbf{y}, \mathbf{z})$ is unsatisfiable
- C_1, ..., C_m is a resolution refutation
- $\mathbf{z}$ only appears positively in F

Output:
straight-line programs $e_1(\mathbf{z})$, ..., $e_m(\mathbf{z})$, monotone in $\mathbf{z}$, such that:

discard any y and any $\neg z$

$$\mathrm{v}[\![e_i]\!] = 0 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{x},+\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![F]\!] = 0$$

$$\mathrm{v}[\![e_i]\!] = 1 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{y},\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![G]\!] = 0$$

discard any x

so $\models \neg e_m(\mathbf{z}) \to F(\mathbf{x}, \mathbf{z})$
and $\models e_m(\mathbf{z}) \to G(\mathbf{y}, \mathbf{z})$

Thanks to Mahesh Viswanathan for the correct and convenient inductive hypothesis!
https://courses.grainger.illinois.edu/cs498mv/fa2018/CraigInterpolation.pdf

1. $C_i \in F$
   e_i = **false**
2. $C_i \in G$
   e_i = **true**
3. $C_i = C_j \backslash \{x\} \cup C_k \backslash \{\neg x\}$
   $e_i = e_j \vee e_k$
4. $C_i = C_j \backslash \{y\} \cup C_k \backslash \{\neg y\}$
   $e_i = e_j \wedge e_k$
5. $C_i = C_j \backslash \{z\} \cup C_k \backslash \{\neg z\}$
   $e_i = (e_j \vee z) \wedge e_k$
   z only appears positively in e_i

Exercise 4

Easy instances of SAT

Horn-CNF

2-CNF

Horn formula: ≤ 1 positive literal in each clause

Horn formula: ≤ 1 positive literal in each clause

1 positive literal

0 positive literals

$n \geq 1$ negative literals

0 negative literals

Horn formula: ≤ 1 positive literal in each clause

1 positive literal

0 positive literals

$n \geq 1$ negative literals

$$p_1 \wedge \cdots \wedge p_n \rightarrow q$$

0 negative literals

Horn formula: ≤ 1 positive literal in each clause

	1 positive literal	0 positive literals
$n \geq 1$ negative literals	$p_1 \wedge \cdots \wedge p_n \rightarrow q$	$p_1 \wedge \cdots \wedge p_n \rightarrow \mathbf{false}$
0 negative literals		

Horn formula: ≤ 1 positive literal in each clause

	1 positive literal	0 positive literals
$n \geq 1$ negative literals	$p_1 \wedge \cdots \wedge p_n \rightarrow q$	$p_1 \wedge \cdots \wedge p_n \rightarrow \mathbf{false}$
0 negative literals	$\mathbf{true} \rightarrow q$	

Horn formula: ≤ 1 positive literal in each clause

	1 positive literal	0 positive literals
$n \geq 1$ negative literals	$p_1 \wedge \cdots \wedge p_n \rightarrow q$	$p_1 \wedge \cdots \wedge p_n \rightarrow \mathbf{false}$
0 negative literals	$\mathbf{true} \rightarrow q$	$\mathbf{true} \rightarrow \mathbf{false}$

Horn formula: ≤ 1 positive literal in each clause

	1 positive literal	0 positive literals
$n \geq 1$ negative literals	$p_1 \wedge \cdots \wedge p_n \rightarrow q$	$p_1 \wedge \cdots \wedge p_n \rightarrow \mathbf{false}$
0 negative literals	$\mathbf{true} \rightarrow q$	$\mathbf{true} \rightarrow \mathbf{false}$

1. if ∃ □ :
   return UNSAT

Horn formula: ≤ 1 positive literal in each clause

	1 positive literal	0 positive literals
$n \geq 1$ negative literals	$p_1 \wedge \cdots \wedge p_n \rightarrow q$	$p_1 \wedge \cdots \wedge p_n \rightarrow \mathbf{false}$
0 negative literals	$\mathbf{true} \rightarrow q$	$\mathbf{true} \rightarrow \mathbf{false}$

2. if ∃ unit clause:
   consider $F' = F[\mathbf{true}/q]$, which
   - has 1 fewer variable;
   - is still a Horn formula; and
   - is equisatisfiable with F.

1. if ∃ □ :
   return UNSAT

Horn formula: ≤ 1 positive literal in each clause

	1 positive literal	0 positive literals
$n \geq 1$ negative literals	$p_1 \wedge \cdots \wedge p_n \to q$ 3. otherwise ∀ of this form: return SAT (with $p_i, q \mapsto 0$)	$p_1 \wedge \cdots \wedge p_n \to \mathbf{false}$
0 negative literals	$\mathbf{true} \to q$ 2. if ∃ unit clause: consider $F' = F[\mathbf{true}/q]$, which • has 1 fewer variable; • is still a Horn formula; and • is equisatisfiable with F.	$\mathbf{true} \to \mathbf{false}$ 1. if ∃ □ : return UNSAT

Easy instances of SAT

Horn-CNF

Horn formula: ≤ 1 positive literal in each clause

	1 positive literal	0 positive literals
$n \geq 1$ negative literals	$p_1 \wedge \cdots \wedge p_n \to q$	$p_1 \wedge \cdots \wedge p_n \to \mathbf{false}$
0 negative literals	$\mathbf{true} \to q$	$\mathbf{true} \to \mathbf{false}$

3. otherwise ∀ of this form:
   return SAT (with $p_i, q \mapsto 0$)

2. if ∃ unit clause:
   consider $F' = F[\mathbf{true}/q]$, which
   - has 1 fewer variable;
   - is still a Horn formula; and
   - is equisatisfiable with F.

1. if ∃ □ :
   return UNSAT

2-CNF

Easy instances of SAT

Horn-CNF

Horn formula: ≤ 1 positive literal in each clause

	1 positive literal	0 positive literals
$n \geq 1$ negative literals	$p_1 \wedge \cdots \wedge p_n \to q$	$p_1 \wedge \cdots \wedge p_n \to \mathbf{false}$
0 negative literals	$\mathbf{true} \to q$	$\mathbf{true} \to \mathbf{false}$

3. otherwise ∀ of this form:
   return SAT (with p_i, $q \mapsto 0$)

2. if ∃ unit clause:
   consider $F' = F[\mathbf{true}/q]$, which
   - has 1 fewer variable;
   - is still a Horn formula; and
   - is equisatisfiable with F.

1. if ∃ □ :
   return UNSAT

2-CNF

renamable Horn

Horn up to flips: after $p \leftrightarrow \neg p$ for some p's,
≤ 1 positive literal in each clause

Horn up to flips: after $p \leftrightarrow \neg p$ for some p's,
≤ 1 positive literal in each clause

$$\cdots \wedge \ (\neg p \vee \neg q \vee r \vee s) \ \wedge \cdots$$

Use -p to mean: do this flip

Horn up to flips: after $p \leftrightarrow \neg p$ for some p's,
≤ 1 positive literal in each clause

$$\cdots \wedge \ (\neg p \vee \neg q \vee r \vee s) \ \wedge \cdots$$

Horn up to flips: after $p \leftrightarrow \neg p$ for some p's,
≤ 1 positive literal in each clause

Use -p to mean: do this flip

$$\cdots \wedge (\neg p \vee \neg q \vee r \vee s) \wedge \cdots$$

need to flip r, s, or both

Use -p to mean: do this flip

Horn up to flips: after $p \leftrightarrow \neg p$ for some p's,
≤ 1 positive literal in each clause

$$\cdots \wedge (\neg p \vee \neg q \vee r \vee s) \wedge \cdots$$

need to flip r, s, or both

$-r \vee -s$

Use -p to mean: do this flip

Horn up to flips: after $p \leftrightarrow \neg p$ for some p's,
≤ 1 positive literal in each clause

don't flip q, or do but flip r too

$$\cdots \wedge (\neg p \vee \neg q \vee r \vee s) \wedge \cdots$$

need to flip r, s, or both

$-r \vee -s$

Use -p to mean: do this flip

Horn up to flips: after $p \leftrightarrow \neg p$ for some p's,
≤ 1 positive literal in each clause

$\neg$-$q \vee$ -r

don't flip q, or do but flip r too

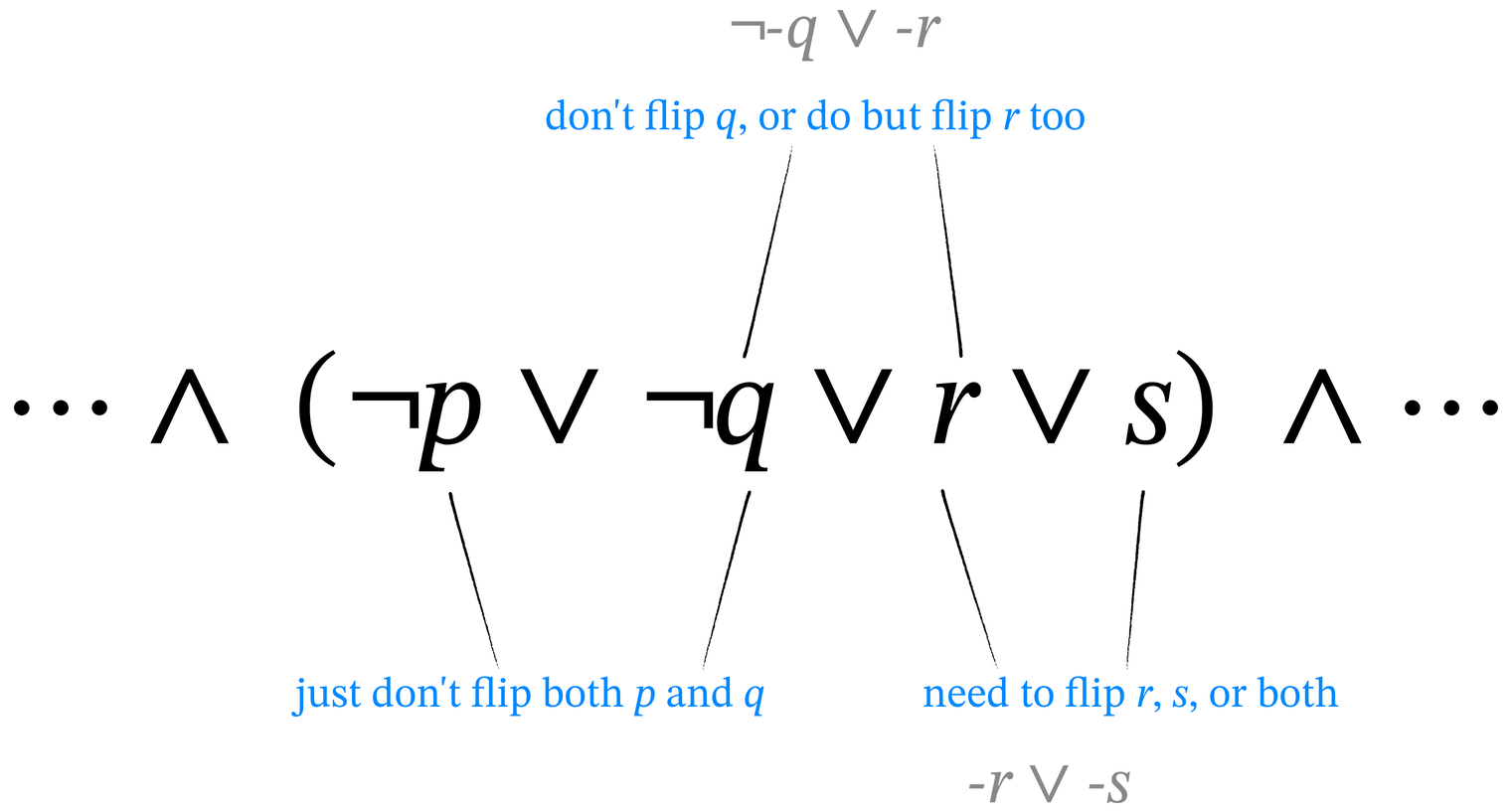
$$\cdots \wedge (\neg p \vee \neg q \vee r \vee s) \wedge \cdots$$

need to flip r, s, or both

-$r \vee$ -s

Use -p to mean: do this flip

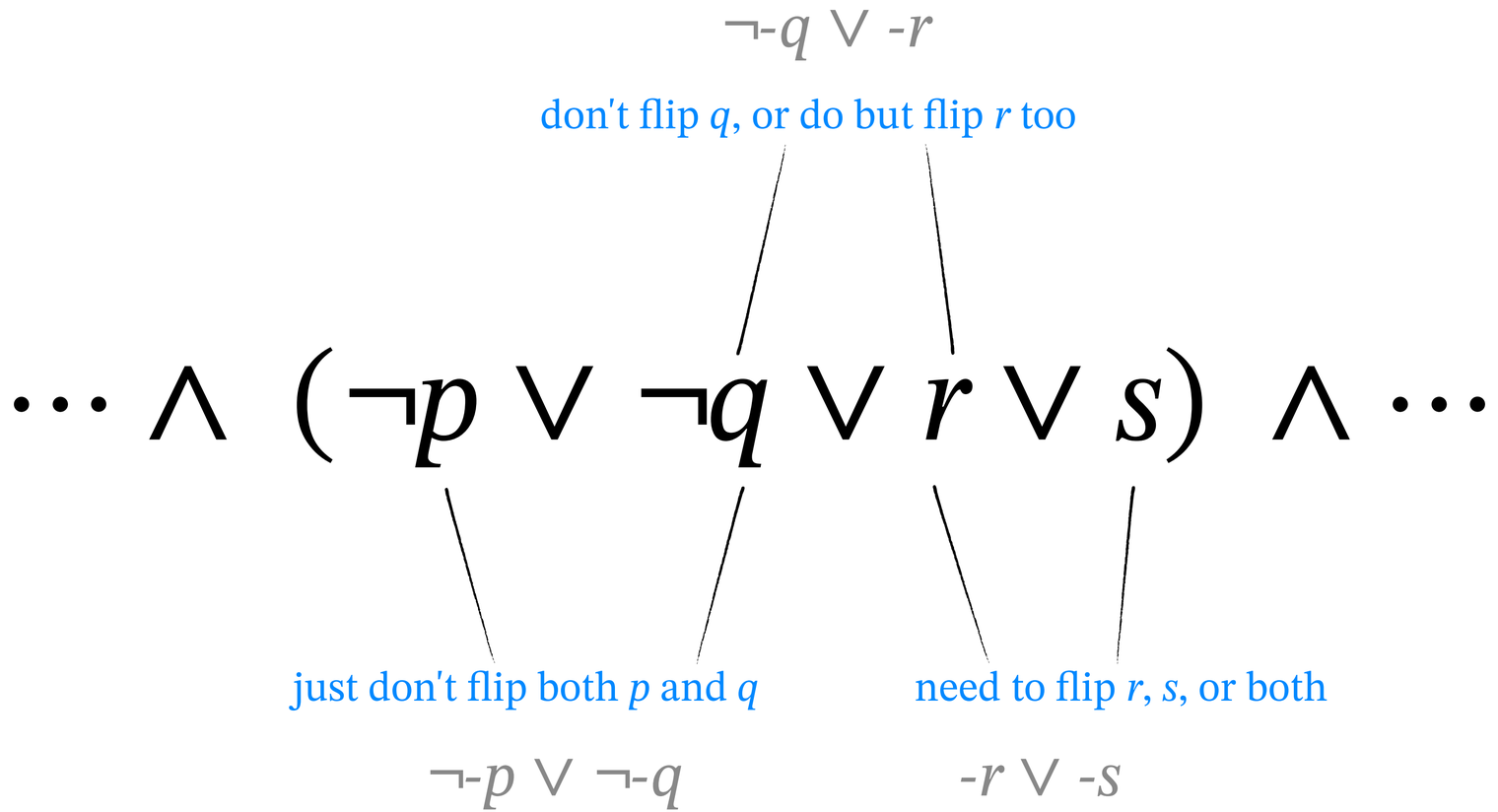
Horn up to flips: after $p \leftrightarrow \neg p$ for some p's,
≤ 1 positive literal in each clause



Horn up to flips:

Use -p to mean: do this flip

after $p \leftrightarrow \neg p$ for some p's,
≤ 1 positive literal in each clause



Use -p to mean: do this flip

Horn up to flips: after $p \leftrightarrow \neg p$ for some p's,
≤ 1 positive literal in each clause

$\neg\text{-}q \vee \text{-}r$

don't flip q, or do but flip r too

$$\cdots \wedge (\neg p \vee \neg q \vee r \vee s) \wedge \cdots$$

just don't flip both p and q

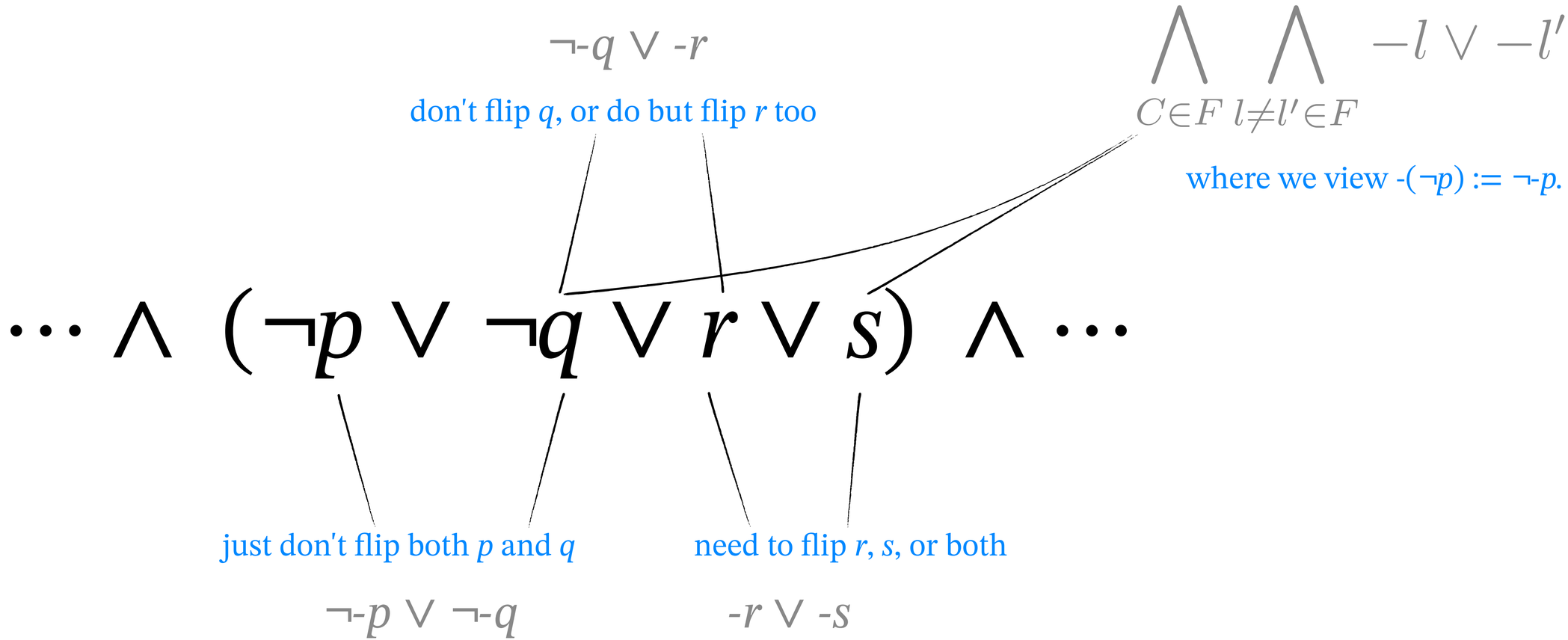
need to flip r, s, or both

$\neg\text{-}p \vee \neg\text{-}q$

$\text{-}r \vee \text{-}s$

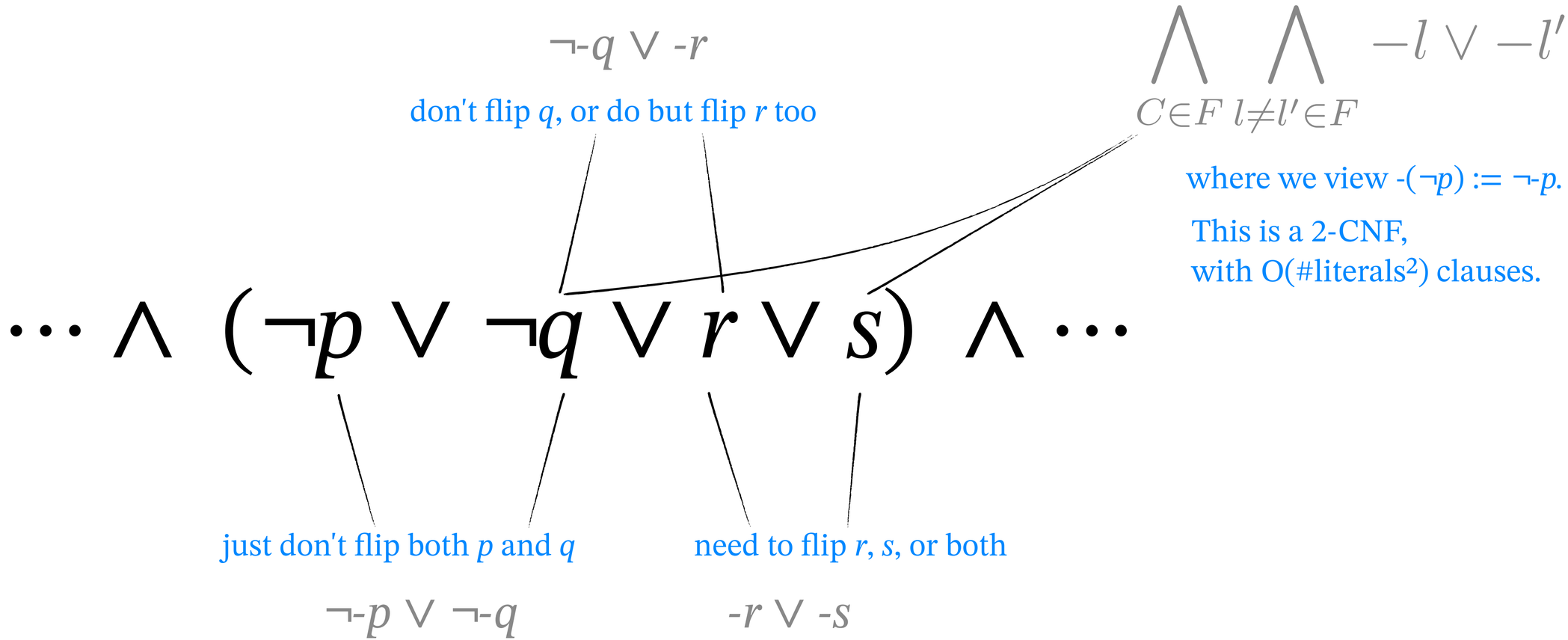
Use -p to mean: do this flip

Horn up to flips: after $p \leftrightarrow \neg p$ for some p's,
≤ 1 positive literal in each clause



Use -p to mean: do this flip

Horn up to flips: after $p \leftrightarrow \neg p$ for some p's,
≤ 1 positive literal in each clause



Exercise 5

Horn up to flips:

Use -p to mean: do this flip

after $p \leftrightarrow \neg p$ for some p's,
≤ 1 positive literal in each clause

$\neg$-$q \vee$ -r

don't flip q, or do but flip r too

$\bigwedge_{C \in F} \bigwedge_{l \neq l' \in F} -l \vee -l'$

where we view -($\neg p$) := $\neg$-p.

This is a 2-CNF,
with O(#literals^2) clauses.

$$\cdots \wedge (\neg p \vee \neg q \vee r \vee s) \wedge \cdots$$

just don't flip both p and q

need to flip r, s, or both

$\neg$-$p \vee \neg$-q

-$r \vee$ -s

Easy instances of SAT

Horn-CNF

Horn formula: ≤ 1 positive literal in each clause

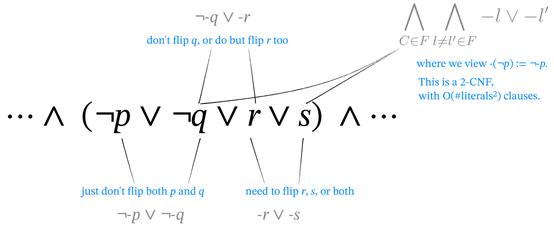
	1 positive literal	0 positive literals
$n \geq 1$ negative literals	3. otherwise ∀ of this form: return SAT (with p_i, $q \mapsto 0$) $p_1 \wedge \cdots \wedge p_n \to q$	$p_1 \wedge \cdots \wedge p_n \to \mathbf{false}$
0 negative literals	$\mathbf{true} \to q$ 2. if ∃ unit clause: consider $F' = F[\mathbf{true}/q]$, which • has 1 fewer variable; • is still a Horn formula; and • is equisatisfiable with F.	$\mathbf{true} \to \mathbf{false}$ 1. if ∃ □ : return UNSAT

2-CNF

renamable Horn

Exercise 5

Horn up to flips: Use $\neg p$ to mean: do this flip
after $p \leftrightarrow \neg p$ for some p's,
≤ 1 positive literal in each clause



2-CNF: $\cdots \wedge (p \vee q) \wedge \cdots$

2-CNF: $\cdots \wedge (p \vee q) \wedge \cdots$

- (Krom '67) use resolution

2-CNF: $\cdots \wedge (p \vee q) \wedge \cdots$

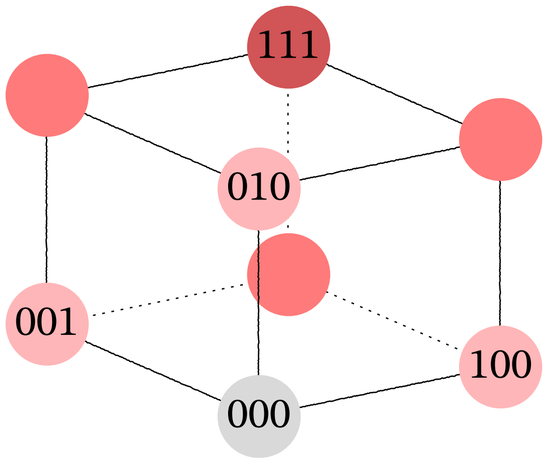
- (Tarjan et al. '79) SCC's of implication graph
- (Krom '67) use resolution

2-CNF: $\cdots \wedge (p \vee q) \wedge \cdots$

- (Tarjan et al. '79) SCC's of implication graph
- (Krom '67) use resolution
- **Walk-SAT**

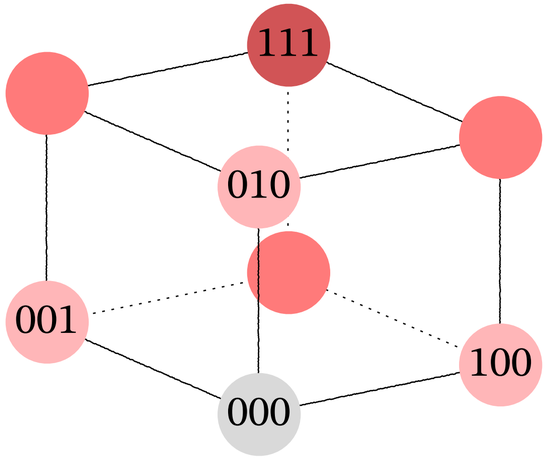
2-CNF: $\cdots \wedge (p \vee q) \wedge \cdots$

- (Tarjan et al. '79) SCC's of implication graph
- (Krom '67) use resolution
- **Walk-SAT**



2-CNF: $\cdots \wedge (p \vee q) \wedge \cdots$

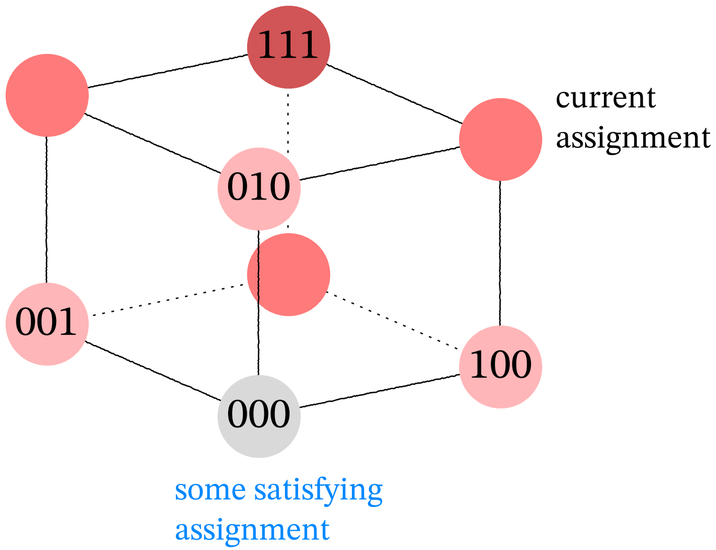
- (Tarjan et al. '79) SCC's of implication graph
- (Krom '67) use resolution
- **Walk-SAT**



some satisfying assignment

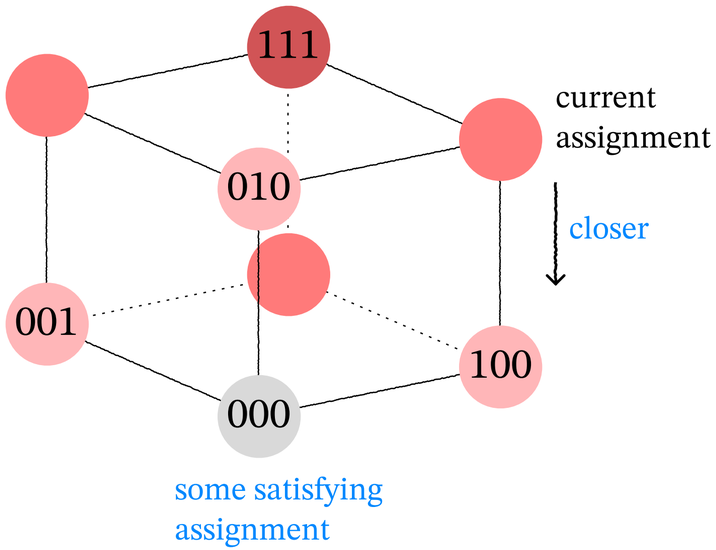
2-CNF: $\cdots \wedge (p \vee q) \wedge \cdots$

- (Tarjan et al. '79) SCC's of implication graph
- (Krom '67) use resolution
- **Walk-SAT**



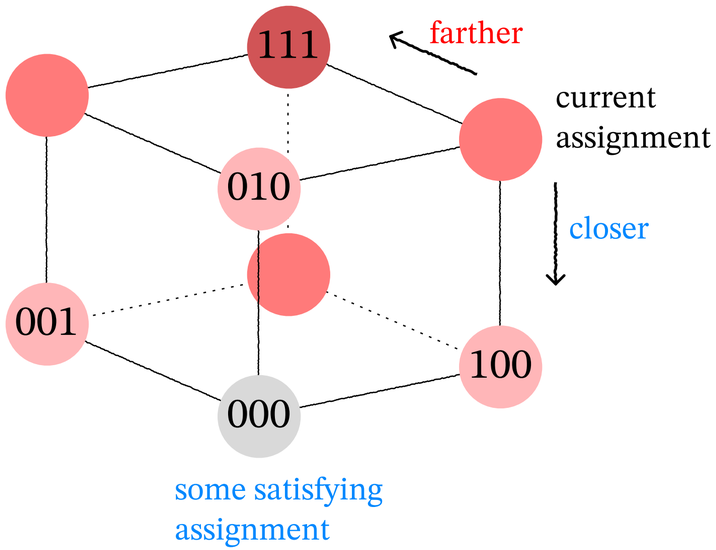
2-CNF: $\cdots \wedge (p \vee q) \wedge \cdots$

- (Tarjan et al. '79) SCC's of implication graph
- (Krom '67) use resolution
- **Walk-SAT**



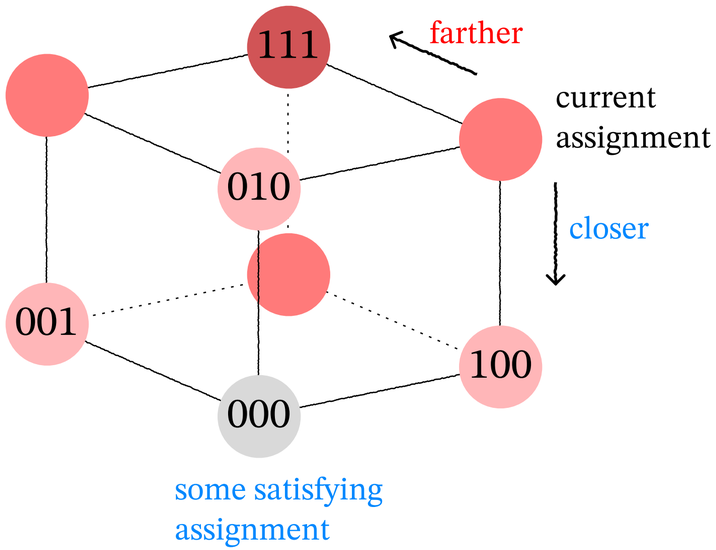
2-CNF: $\cdots \wedge (p \vee q) \wedge \cdots$

- (Tarjan et al. '79) SCC's of implication graph
- (Krom '67) use resolution
- **Walk-SAT**



2-CNF: $\cdots \wedge (p \vee q) \wedge \cdots$

- (Tarjan et al. '79) SCC's of implication graph
- (Krom '67) use resolution
- **Walk-SAT**



Repeat $2n^2$ times:

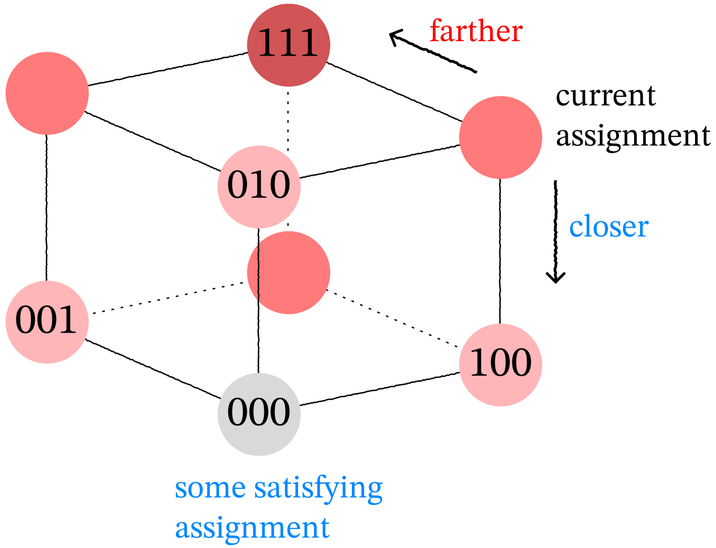
Pick an unsatisfied clause;

pick a literal within;

flip its value.

2-CNF: $\cdots \wedge (p \vee q) \wedge \cdots$

- (Tarjan et al. '79) SCC's of implication graph
- (Krom '67) use resolution
- **Walk-SAT**

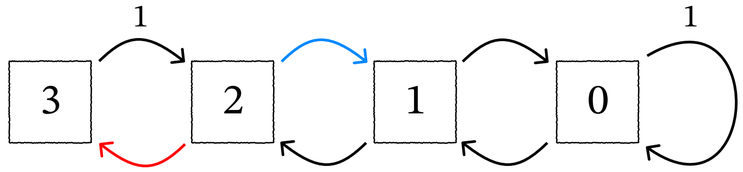


Repeat $2n^2$ times:

Pick an unsatisfied clause;

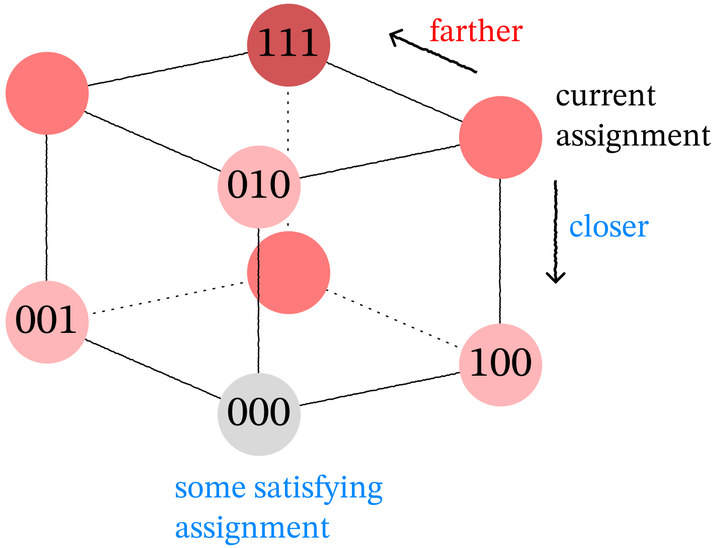
pick a literal within;

flip its value.



2-CNF: $\cdots \wedge (p \vee q) \wedge \cdots$

- (Tarjan et al. '79) SCC's of implication graph
- (Krom '67) use resolution
- **Walk-SAT**

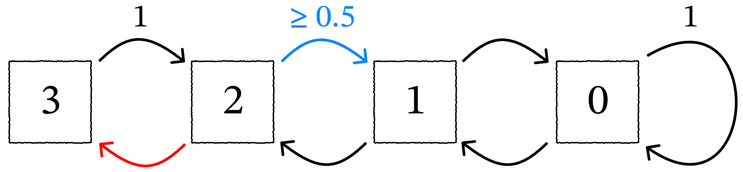


Repeat $2n^2$ times:

Pick an unsatisfied clause;

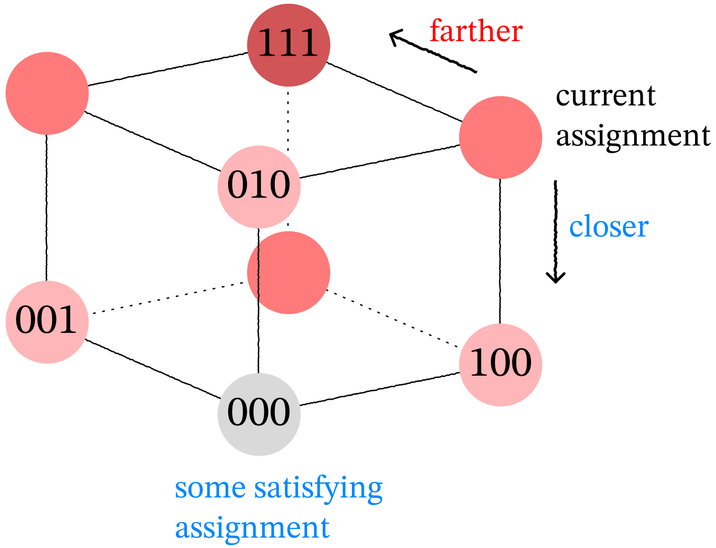
pick a literal within;

flip its value.



2-CNF: $\cdots \wedge (p \vee q) \wedge \cdots$

- (Tarjan et al. '79) SCC's of implication graph
- (Krom '67) use resolution
- **Walk-SAT**

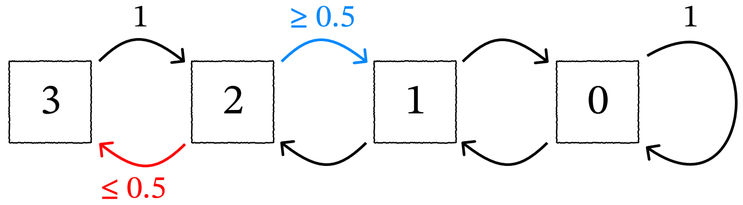


Repeat $2n^2$ times:

Pick an unsatisfied clause;

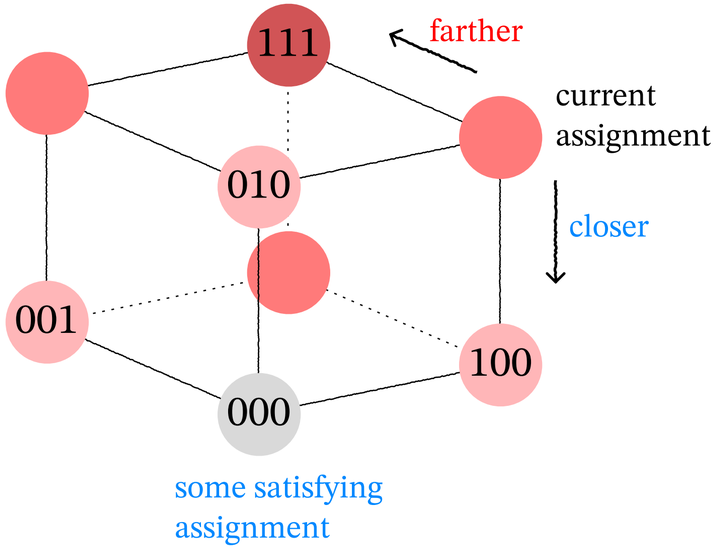
pick a literal within;

flip its value.



2-CNF: $\cdots \wedge (p \vee q) \wedge \cdots$

- (Tarjan et al. '79) SCC's of implication graph
- (Krom '67) use resolution
- **Walk-SAT**

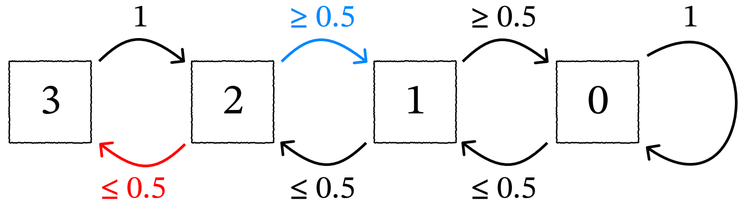


Repeat $2n^2$ times:

Pick an unsatisfied clause;

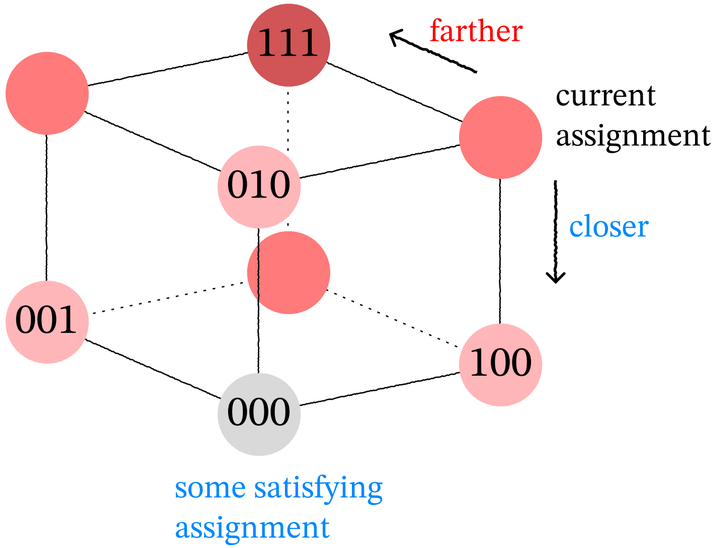
pick a literal within;

flip its value.



2-CNF: $\cdots \wedge (p \vee q) \wedge \cdots$

- (Tarjan et al. '79) SCC's of implication graph
- (Krom '67) use resolution
- **Walk-SAT**



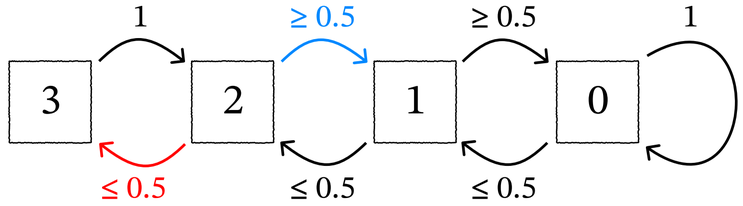
Repeat $2n^2$ times:

Pick an unsatisfied clause;

pick a literal within;

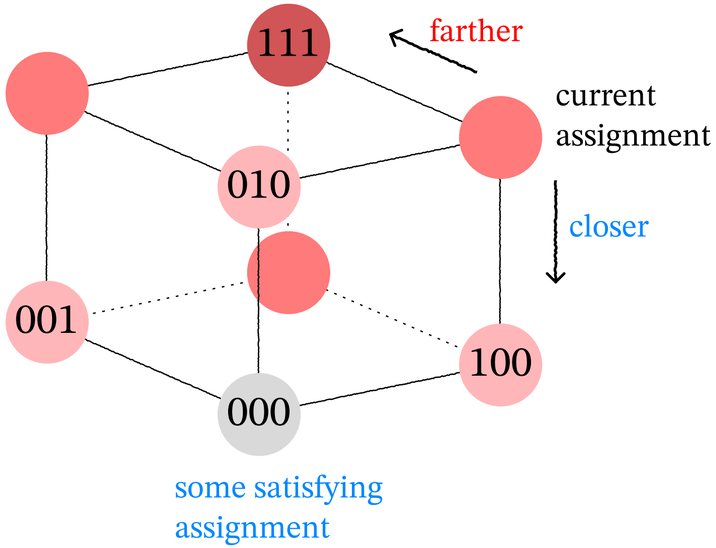
flip its value.

here with probability $\geq 1/2$



2-CNF: $\cdots \wedge (p \vee q) \wedge \cdots$

- (Tarjan et al. '79) SCC's of implication graph
- (Krom '67) use resolution
- **Walk-SAT** — also works for Horn-CNF without unit clauses.



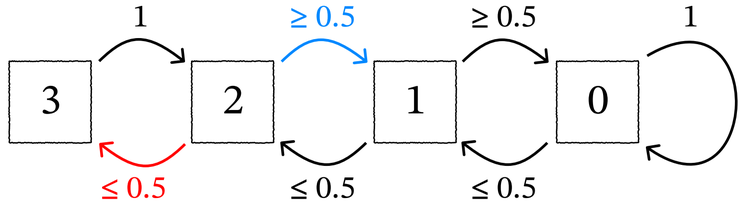
Repeat $2n^2$ times:

Pick an unsatisfied clause;

pick a literal within;

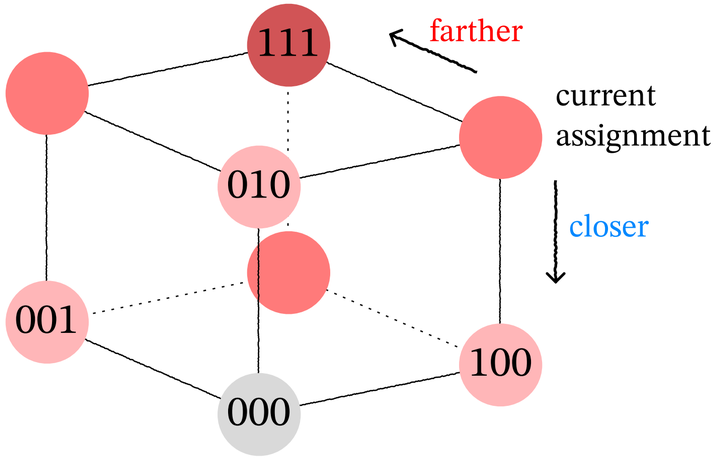
flip its value.

here with probability $\geq 1/2$



2-CNF: $\cdots \wedge (p \vee q) \wedge \cdots$

- (Tarjan et al. '79) SCC's of implication graph
- (Krom '67) use resolution
- **Walk-SAT** — also works for Horn-CNF without unit clauses.



some satisfying
assignment

* ↦ 0 is satisfying,
as there are no unit clauses

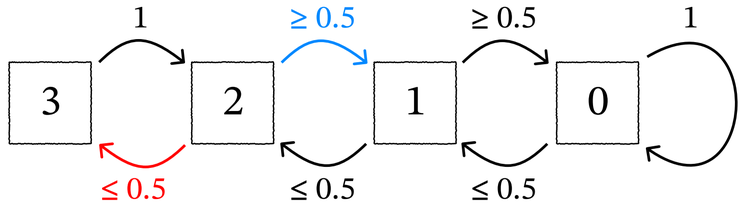
Repeat $2n^2$ times:

Pick an unsatisfied clause;

pick a literal within;

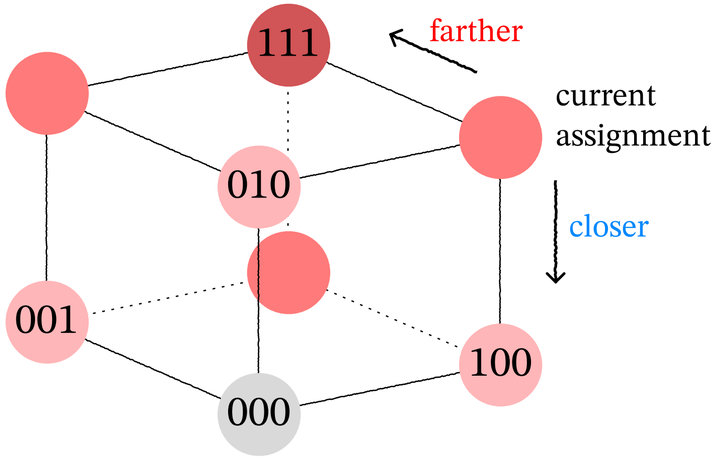
flip its value.

here with
probability
≥ 1/2



2-CNF: $\cdots \wedge (p \vee q) \wedge \cdots$

- (Tarjan et al. '79) SCC's of implication graph
- (Krom '67) use resolution
- **Walk-SAT** — also works for Horn-CNF without unit clauses.



some satisfying
assignment

* ↦ 0 is satisfying,
as there are no unit clauses

Repeat $2n^2$ times:

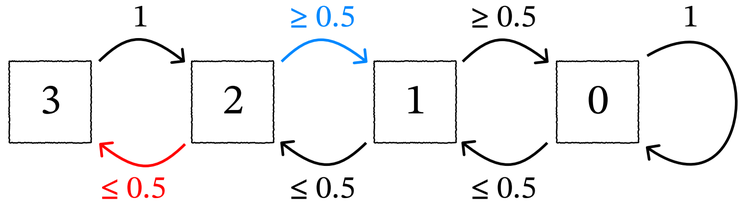
Pick an unsatisfied clause;

$p_1 \wedge \cdots \wedge p_k \to x$

pick a literal within;

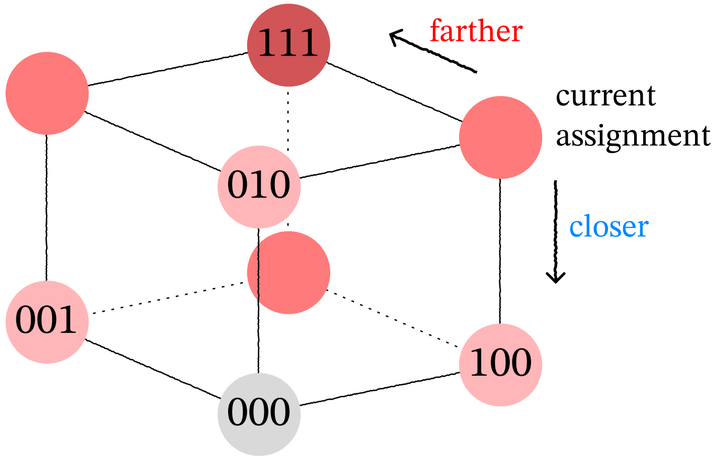
flip its value.

here with
probability
≥ 1/2



2-CNF: $\cdots \wedge (p \vee q) \wedge \cdots$

- (Tarjan et al. '79) SCC's of implication graph
- (Krom '67) use resolution
- **Walk-SAT** — also works for Horn-CNF without unit clauses.



some satisfying
assignment

* ↦ 0 is satisfying,
as there are no unit clauses

Repeat $2n^2$ times:

Pick an unsatisfied clause;

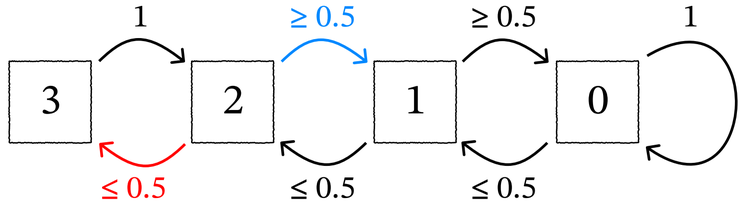
$p_1 \wedge \cdots \wedge p_k \to x$

pick a literal within;

$k \geq 1$ variables set to 1, and x to 0

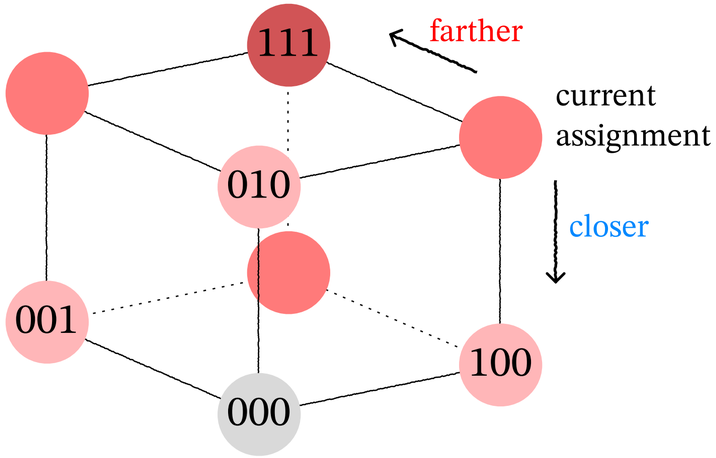
flip its value.

here with
probability
$\geq 1/2$



2-CNF: $\cdots \wedge (p \vee q) \wedge \cdots$

- (Tarjan et al. '79) SCC's of implication graph
- (Krom '67) use resolution
- **Walk-SAT** — also works for Horn-CNF without unit clauses.



some satisfying
assignment

$* \mapsto 0$ is satisfying,
as there are no unit clauses

Repeat $2n^2$ times:

Pick an unsatisfied clause;

$p_1 \wedge \cdots \wedge p_k \to x$

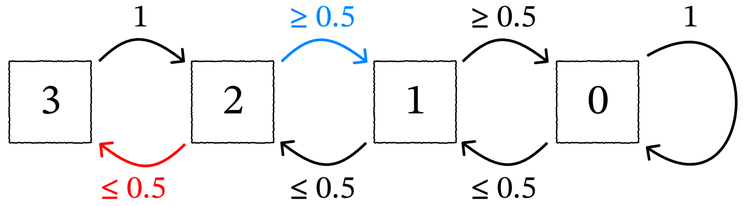
pick a literal within;

$k \geq 1$ variables set to 1, and x to 0

flip its value.

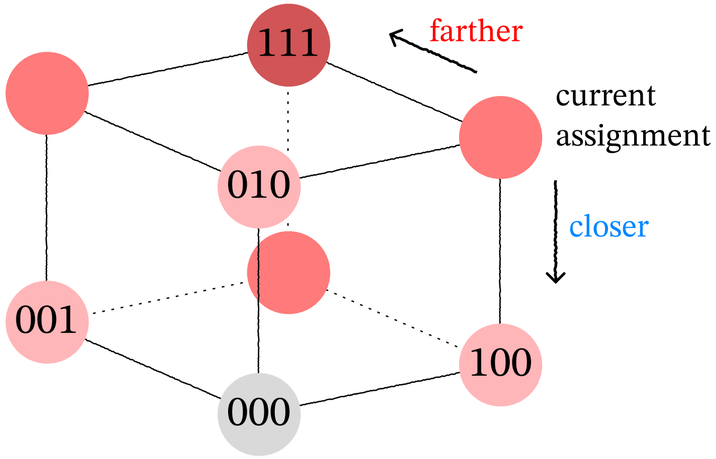
this is closer to $* \mapsto 0$ as long as
x (= **false** possibly) was not picked

here with
probability
$\geq 1/2$



2-CNF: $\cdots \wedge (p \vee q) \wedge \cdots$

- (Tarjan et al. '79) SCC's of implication graph
- (Krom '67) use resolution
- **Walk-SAT** — also works for Horn-CNF without unit clauses.



some satisfying assignment

$* \mapsto 0$ is satisfying,
as there are no unit clauses

Repeat $2n^2$ times:

Pick an unsatisfied clause;

$p_1 \wedge \cdots \wedge p_k \to x$

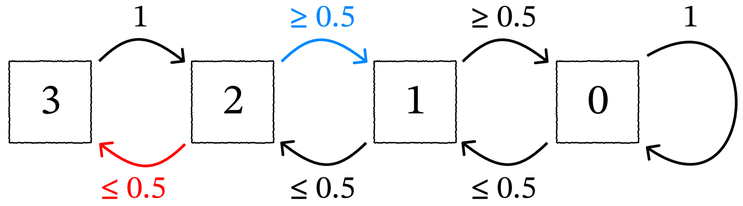
pick a literal within;

$k \geq 1$ variables set to 1, and x to 0

flip its value.

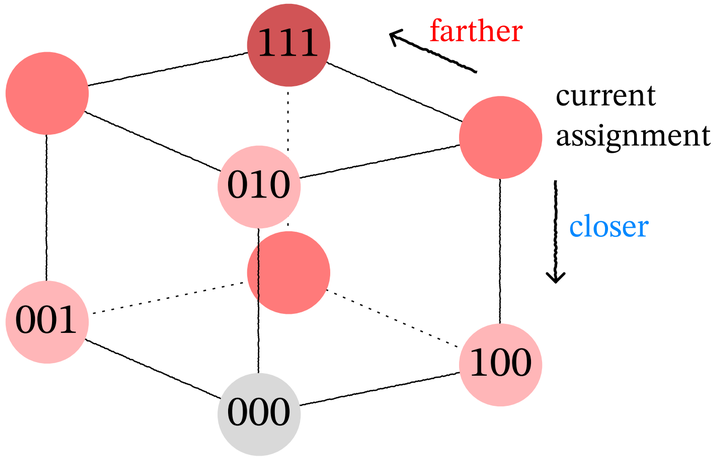
this is closer to $* \mapsto 0$ as long as
x (= **false** possibly) was not picked

also here with probability $\geq 1/2$



2-CNF: $\cdots \wedge (p \vee q) \wedge \cdots$

- (Tarjan et al. '79) SCC's of implication graph
- (Krom '67) use resolution
- **Walk-SAT** — also works for Horn-CNF without unit clauses.



some satisfying assignment

$* \mapsto 0$ is satisfying,
as there are no unit clauses

Repeat $2n^2$ times:

Pick an unsatisfied clause;

$p_1 \wedge \cdots \wedge p_k \to x$

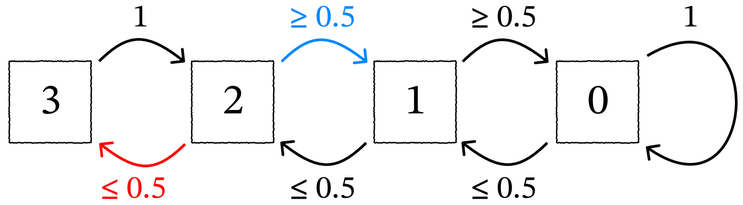
pick a literal within;

$k \geq 1$ variables set to 1, and x to 0

flip its value.

this is closer to $* \mapsto 0$ as long as
x (= **false** possibly) was not picked

also here with probability $\geq 1/2$



Exercise 6

Easy instances of SAT

Horn-CNF

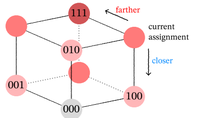
Horn formula: ≤ 1 positive literal in each clause

	1 positive literal	0 positive literals
	3. otherwise ∀ of this form: return SAT (with $p_i, q \mapsto 0$)	
$n \geq 1$ negative literals	$p_1 \wedge \cdots \wedge p_n \to q$	$p_1 \wedge \cdots \wedge p_n \to \mathbf{false}$
0 negative literals	$\mathbf{true} \to q$ 2. if ∃ unit clause: consider $F' = F[\mathbf{true}/q]$, which • has 1 fewer variable; • is still a Horn formula; and • is equisatisfiable with F.	$\mathbf{true} \to \mathbf{false}$ 1. if ∃ □ : return UNSAT

2-CNF

2-CNF: $\cdots \wedge (p \vee q) \wedge \cdots$

- (Tarjan et al. '79) SCC's of implication graph
- (Krom '67) use resolution
- **Walk-SAT** — also works for Horn-CNF without unit clauses.



some satisfying assignment

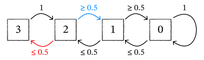
$* \mapsto 0$ is satisfying, as there are no unit clauses

Repeat $2n^2$ times:

Pick an unsatisfied clause;
$p_i \wedge \cdots \wedge p_k \to x$

pick a literal within;
$k \geq 1$ variables set to 1, and x to 0

flip its value.
this is closer to $* \mapsto 0$ as long as x (= **false** possibly) was not picked



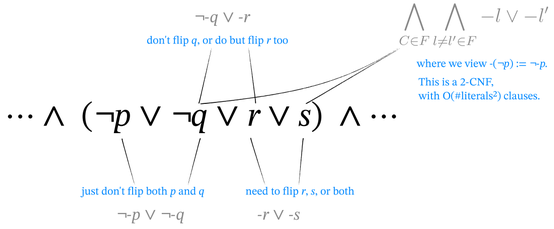
also here with probability ≥ 1/2

Exercise 6

renamable Horn

Exercise 5

Horn up to flips: Use $\neg p$ to mean: do this flip
after $p \leftrightarrow \neg p$ for some p's,
≤ 1 positive literal in each clause



Easy instances of SAT

Horn-CNF

Horn formula: ≤ 1 positive literal in each clause

	1 positive literal	0 positive literals
$n \geq 1$ negative literals	$p_1 \wedge \cdots \wedge p_n \to q$	$p_1 \wedge \cdots \wedge p_n \to \mathbf{false}$
0 negative literals	$\mathbf{true} \to q$	$\mathbf{true} \to \mathbf{false}$

3. otherwise ∀ of this form: return SAT (with $p_i, q \mapsto 0$)

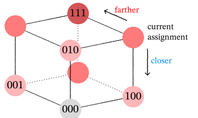
2. if ∃ unit clause: consider $F' = F[\mathbf{true}/q]$, which
   - has 1 fewer variable;
   - is still a Horn formula; and
   - is equisatisfiable with F.

1. if ∃ □ : return UNSAT

2-CNF

2-CNF: $\cdots \wedge (p \vee q) \wedge \cdots$

- (Tarjan et al. '79) SCC's of implication graph
- (Krom '67) use resolution
- **Walk-SAT** — also works for Horn-CNF without unit clauses.



some satisfying assignment

$* \mapsto 0$ is satisfying, as there are no unit clauses

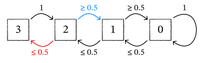
Repeat $2n^2$ times:

Pick an unsatisfied clause; $p_i \wedge \cdots \wedge p_k \to x$

pick a literal within; $k \geq 1$ variables set to 1, and x to 0

flip its value.

this is closer to $* \mapsto 0$ as long as x (= **false** possibly) was not picked



also here with probability ≥ 1/2

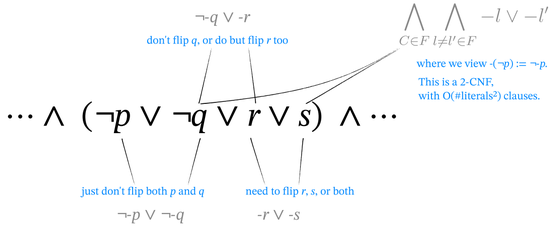
Exercise 6

renamable Horn

Exercise 5

Horn up to flips: Use $\neg p$ to mean: do this flip

after $p \leftrightarrow \neg p$ for some p's,
≤ 1 positive literal in each clause



2-use-CNF

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0) \quad \wedge \quad (\neg p \vee \cdots_1)$$

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0) \quad \wedge \quad (\neg p \vee \cdots_1)$$

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0) \quad \wedge \quad (p \vee \cdots_1)$$

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0) \quad \wedge \quad (\neg p \vee \cdots_1)$$

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0) \quad \wedge \quad (p \vee \cdots_1)$$

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0)$$

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0) \quad \wedge \quad (\neg p \vee \cdots_1)$$

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0) \quad \wedge \quad (p \vee \cdots_1)$$

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0)$$

If p only appears positively, then

$$F' = \overset{\text{no } p \text{ here}}{\cdots} = F[\mathbf{true}/p]$$

has fewer variables; still 2-use; equisatisfiable:

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0) \quad \wedge \quad (\neg p \vee \cdots_1)$$

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0) \quad \wedge \quad (p \vee \cdots_1)$$

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0)$$

If p only appears positively, then

$$F' = \overset{\text{no } p \text{ here}}{\cdots} = F[\mathbf{true}/p]$$

has fewer variables; still 2-use; equisatisfiable:

v	
⋮	⋮
p	0

v′	
⋮	⋮

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0) \quad \wedge \quad (\neg p \vee \cdots_1)$$

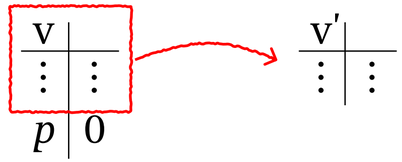
$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0) \quad \wedge \quad (p \vee \cdots_1)$$

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0)$$

If p only appears positively, then

$$F' = \overset{\text{no } p \text{ here}}{\cdots} = F[\mathbf{true}/p]$$

has fewer variables; still 2-use; equisatisfiable:



$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0) \quad \wedge \quad (\neg p \vee \cdots_1)$$

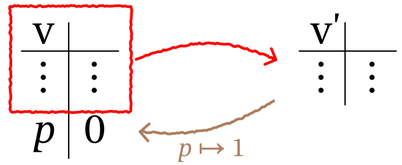
$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0) \quad \wedge \quad (p \vee \cdots_1)$$

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0)$$

If p only appears positively, then

$$F' = \overset{\text{no } p \text{ here}}{\cdots} = F[\mathbf{true}/p]$$

has fewer variables; still 2-use; equisatisfiable:



$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0) \quad \wedge \quad (\neg p \vee \cdots_1)$$

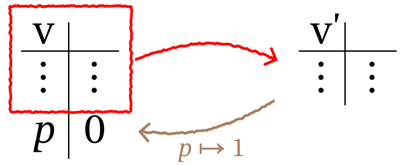
$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0) \quad \wedge \quad (p \vee \cdots_1)$$

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0)$$

If p only appears positively, then

$$F' = \overset{\text{no } p \text{ here}}{\cdots} = F[\mathbf{true}/p]$$

has fewer variables; still 2-use; equisatisfiable:



$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (\neg p \vee p \vee \cdots_0)$$

Symmetric if p only appears negatively or trivially.

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \ (p \vee \cdots_0) \ \wedge \ (\neg p \vee \cdots_1) \qquad F' = \overset{\text{no } p \text{ here}}{\cdots} \wedge \ (\cdots_0 \vee \cdots_1)$$

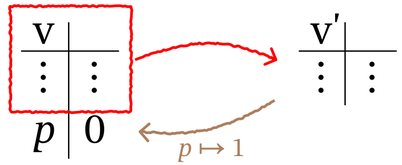
$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \ (p \vee \cdots_0) \ \wedge \ (p \vee \cdots_1)$$

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \ (p \vee \cdots_0)$$

If p only appears positively, then

$$F' = \overset{\text{no } p \text{ here}}{\cdots} = F[\mathbf{true}/p]$$

has fewer variables; still 2-use; equisatisfiable:



$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \ (\neg p \vee p \vee \cdots_0)$$

Symmetric if p only appears negatively or trivially.

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \ (p \vee \cdots_0) \ \wedge \ (\neg p \vee \cdots_1)$$

$$F' = \overset{\text{no } p \text{ here}}{\cdots} \wedge \ (\cdots_0 \vee \cdots_1)$$

v satisfies $F \implies \mathrm{v}' = \mathrm{v}|_{\setminus\{p\}}$ satisfies F':

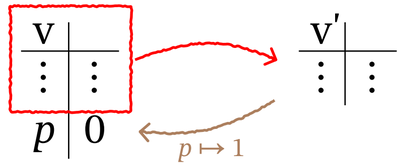
$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \ (p \vee \cdots_0) \ \wedge \ (p \vee \cdots_1)$$

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \ (p \vee \cdots_0)$$

If p only appears positively, then

$$F' = \overset{\text{no } p \text{ here}}{\cdots} = F[\mathbf{true}/p]$$

has fewer variables; still 2-use; equisatisfiable:



$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \ (\neg p \vee p \vee \cdots_0)$$

Symmetric if p only appears negatively or trivially.

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0) \quad \wedge \quad (\neg p \vee \cdots_1)$$

$$F' = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (\cdots_0 \vee \cdots_1)$$

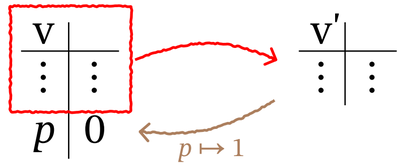
v satisfies $F \implies \mathrm{v}' = \mathrm{v}|_{\setminus\{p\}}$ satisfies F':
because resolution is sound.

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0) \quad \wedge \quad (p \vee \cdots_1)$$

If p only appears positively, then

$$F' = \overset{\text{no } p \text{ here}}{\cdots} = F[\mathbf{true}/p]$$

has fewer variables; still 2-use; equisatisfiable:



$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0)$$

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (\neg p \vee p \vee \cdots_0)$$

Symmetric if p only appears negatively or trivially.

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0) \quad \wedge \quad (\neg p \vee \cdots_1)$$

v′ satisfies F' $\Longrightarrow$

- v′ satisfies $\cdots_0$

$$F' = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (\cdots_0 \vee \cdots_1)$$

v satisfies F $\Longrightarrow$ $\mathrm{v}' = \mathrm{v}|_{\setminus\{p\}}$ satisfies F':
because resolution is sound.

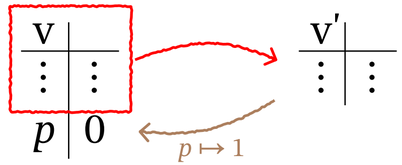
$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0) \quad \wedge \quad (p \vee \cdots_1)$$

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0)$$

If p only appears positively, then

$$F' = \overset{\text{no } p \text{ here}}{\cdots} = F[\mathbf{true}/p]$$

has fewer variables; still 2-use; equisatisfiable:



$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (\neg p \vee p \vee \cdots_0)$$

Symmetric if p only appears negatively or trivially.

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0) \quad \wedge \quad (\neg p \vee \cdots_1)$$

v′ satisfies $F' \implies$

- v′ satisfies $\cdots_0$ — then $\mathrm{v}'_{[p \mapsto 0]}$ satisfies F; or

$$F' = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (\cdots_0 \vee \cdots_1)$$

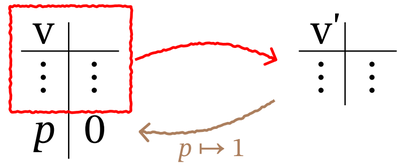
v satisfies $F \implies \mathrm{v}' = \mathrm{v}|_{\setminus\{p\}}$ satisfies F':
because resolution is sound.

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0) \quad \wedge \quad (p \vee \cdots_1)$$

If p only appears positively, then

$$F' = \overset{\text{no } p \text{ here}}{\cdots} = F[\mathbf{true}/p]$$

has fewer variables; still 2-use; equisatisfiable:



$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0)$$

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (\neg p \vee p \vee \cdots_0)$$

Symmetric if p only appears negatively or trivially.

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0) \quad \wedge \quad (\neg p \vee \cdots_1)$$

v′ satisfies $F' \implies$

- v′ satisfies $\cdots_0$ — then $\mathrm{v}'_{[p \mapsto 0]}$ satisfies F; or
- v′ satisfies $\cdots_1$ — then $\mathrm{v}'_{[p \mapsto 1]}$ satisfies F.

$$F' = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (\cdots_0 \vee \cdots_1)$$

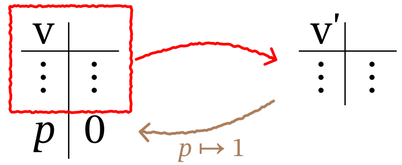
v satisfies $F \implies \mathrm{v}' = \mathrm{v}|_{\setminus\{p\}}$ satisfies F':
because resolution is sound.

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0) \quad \wedge \quad (p \vee \cdots_1)$$

If p only appears positively, then

$$F' = \overset{\text{no } p \text{ here}}{\cdots} = F[\mathbf{true}/p]$$

has fewer variables; still 2-use; equisatisfiable:



$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (p \vee \cdots_0)$$

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \quad (\neg p \vee p \vee \cdots_0)$$

Symmetric if p only appears negatively or trivially.

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \ (p \vee \cdots_0) \ \wedge \ (\neg p \vee \cdots_1)$$

Can eliminate variables one by one,
without the size blowing up

v′ satisfies $F' \implies$

- v′ satisfies $\cdots_0$ — then $\mathrm{v}'_{[p \mapsto 0]}$ satisfies F; or
- v′ satisfies $\cdots_1$ — then $\mathrm{v}'_{[p \mapsto 1]}$ satisfies F.

$$F' = \overset{\text{no } p \text{ here}}{\cdots} \wedge \ (\cdots_0 \vee \cdots_1)$$

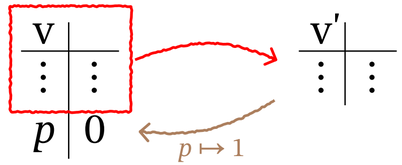
v satisfies $F \implies \mathrm{v}' = \mathrm{v}|_{\setminus\{p\}}$ satisfies F':
because resolution is sound.

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \ (p \vee \cdots_0) \ \wedge \ (p \vee \cdots_1)$$

If p only appears positively, then

$$F' = \overset{\text{no } p \text{ here}}{\cdots} = F[\mathbf{true}/p]$$

has fewer variables; still 2-use; equisatisfiable:



$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \ (p \vee \cdots_0)$$

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \ (\neg p \vee p \vee \cdots_0)$$

Symmetric if p only appears negatively or trivially.

Exercise 7

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \ (p \vee \cdots_0) \ \wedge \ (\neg p \vee \cdots_1)$$

Can eliminate variables one by one,
without the size blowing up

v′ satisfies F' $\Longrightarrow$

- v′ satisfies $\cdots_0$ — then $\mathrm{v}'_{[p \mapsto 0]}$ satisfies F; or
- v′ satisfies $\cdots_1$ — then $\mathrm{v}'_{[p \mapsto 1]}$ satisfies F.

$$F' = \overset{\text{no } p \text{ here}}{\cdots} \wedge \ (\cdots_0 \vee \cdots_1)$$

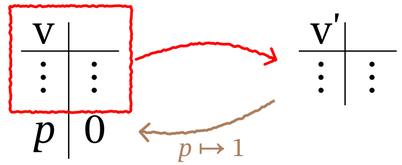
v satisfies F $\Longrightarrow$ $\mathrm{v}' = \mathrm{v}|_{\setminus\{p\}}$ satisfies F':
because resolution is sound.

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \ (p \vee \cdots_0) \ \wedge \ (p \vee \cdots_1)$$

If p only appears positively, then

$$F' = \overset{\text{no } p \text{ here}}{\cdots} = F[\mathbf{true}/p]$$

has fewer variables; still 2-use; equisatisfiable:



$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \ (p \vee \cdots_0)$$

$$F = \overset{\text{no } p \text{ here}}{\cdots} \wedge \ (\neg p \vee p \vee \cdots_0)$$

Symmetric if p only appears negatively or trivially.

Easy instances of SAT

Horn-CNF

Horn formula: ≤ 1 positive literal in each clause

	1 positive literal	0 positive literals
$n \geq 1$ negative literals	$p_1 \wedge \cdots \wedge p_n \to q$	$p_1 \wedge \cdots \wedge p_n \to \mathbf{false}$
0 negative literals	$\mathbf{true} \to q$	$\mathbf{true} \to \mathbf{false}$

3. otherwise ∀ of this form: return SAT (with $p_i, q \mapsto 0$)

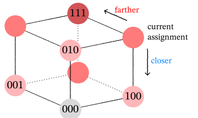
2. if ∃ unit clause: consider $F' = F[\mathbf{true}/q]$, which
   - has 1 fewer variable;
   - is still a Horn formula; and
   - is equisatisfiable with F.

1. if ∃ $\square$: return UNSAT

2-CNF

2-CNF: $\cdots \wedge (p \vee q) \wedge \cdots$

- (Tarjan et al. '79) SCC's of implication graph
- (Krom '67) use resolution
- **Walk-SAT** — also works for Horn-CNF without unit clauses.



some satisfying assignment

$* \mapsto 0$ is satisfying, as there are no unit clauses

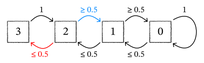
Repeat $2n^2$ times:

Pick an unsatisfied clause;
$p_i \wedge \cdots \wedge p_k \to x$

pick a literal within;
$k \geq 1$ variables set to 1, and x to 0

flip its value.

this is closer to $* \mapsto 0$ as long as x (= **false** possibly) was not picked



also here with probability ≥ 1/2

Exercise 6

renamable Horn

Exercise 5

Horn up to flips: after $p \leftrightarrow \neg p$ for some p's, ≤ 1 positive literal in each clause

Use $\neg p$ to mean: do this flip

$\neg q \vee \neg r$ — don't flip q, or do but flip r too

$\bigwedge_{C \in F} \bigwedge_{l \neq l' \in F} \neg l \vee \neg l'$

where we view $\neg(\neg p) := \neg p$.

This is a 2-CNF, with $O(\#\text{literals}^2)$ clauses.

$$\cdots \wedge (\neg p \vee \neg q \vee r \vee s) \wedge \cdots$$

just don't flip both p and q — $\neg p \vee \neg q$

need to flip r, s, or both — $\neg r \vee \neg s$

2-use-CNF

Exercise 7

$F =^{\text{no } p \text{ here}} \cdots \wedge (p \vee \cdots_0) \wedge (\neg p \vee \cdots_1)$

Can eliminate variables one by one, without the size blowing up

v′ satisfies $F' \implies$
- v′ satisfies $\cdots_0$ — then $v'|_{p \mapsto 0}$ satisfies F; or
- v′ satisfies $\cdots_1$ — then $v'|_{p \mapsto 1}$ satisfies F.

$F' =^{\text{no } p \text{ here}} \cdots \wedge (\cdots_0 \vee \cdots_1)$

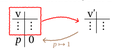
v satisfies $F \implies v = v|_{\setminus\{p\}}$ satisfies F': because resolution is sound.

$F =^{\text{no } p \text{ here}} \cdots \wedge (p \vee \cdots_0) \wedge (p \vee \cdots_1)$

If p only appears positively, then

$F' =^{\text{no } p \text{ here}} \cdots = F[\mathbf{true}/p]$

has fewer variables; still 2-use; equisatisfiable:



$F =^{\text{no } p \text{ here}} \cdots \wedge (p \vee \cdots_0)$

$F =^{\text{no } p \text{ here}} \cdots \wedge (\neg p \vee p \vee \cdots_0)$

Symmetric if p only appears negatively or trivially.

Problems and solutions
courtesy of James Worrell

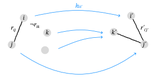
Logic & Proof Sheet 1

Jingjie Yang
HT'26

Propositional logic

(CNF)
A formula in n variables amounts to a function $\{0,1\}^n \to \{0,1\}$.
/ straight-line program

Example: graph homomorphism



every vertex maps to ≥ 1 vertex' every vertex maps to ≤ 1 vertex' edges map to edges'

$$\bigwedge_i \bigvee_{i'} h_{ii'} \quad \wedge \quad \bigwedge_i \bigwedge_{i' \neq j'} \neg h_{ii'} \vee \neg h_{ij'} \quad \wedge \quad \bigwedge_{i,j,i',j'} \neg h_{ii'} \vee \neg h_{jj'} \vee \neg r_{ij} \vee r'_{i'j'}$$

neg pos

This formula in $\mathbf{h}$, $\mathbf{r}$, $\mathbf{r}'$ is satisfiable iff there is a graph homomorphism $\mathbf{h}$.

Exercise 1

Write K_m for the complete graph on m vertices.

(a) homomorphism $K_m \to G$ = m-clique in G

(b) homomorphism $G \to K_m$ = m-colouring of G

Resolution proof

Given F in CNF: $C_1,\ C_2,\ C_3,\ \ldots,\ C_m$ (∨ clause in F, clause ∈ F)

$C_1 = C_1' \sqcup \{p\}$
$C_2 = C_2' \sqcup \{\neg p\}$
$C_3 = C_1' \cup C_2'$

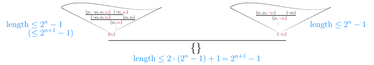
Exercise 2

Claim: $F(p_1, \ldots, p_n)$ unsatisfiable ⇒ ∃resolution proof of $\square$. (of length $\leq 2^{n+1} - 1$)

Proof: by induction on n.
Base case: $n = 0$; only clause is $\square$; only CNF's are $\square$ (length $1 \leq 2^1 - 1$) or $\{\square\}$.
Inductive step:

$$F[\mathbf{false}/p_n] = \bigcup_{C \in F,\, \neg p_n \notin C} C \setminus \{p_n\} \qquad F[\mathbf{true}/p_n] = \bigcup_{C \in F,\, p_n \notin C} C \setminus \{\neg p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:
Either $\forall C' : C \setminus \{p_n\} = C'$— done, or we have a proof of p_n.
Either $\forall C' : C \setminus \{\neg p_n\} = C'$— done, or have a proof of $\neg p_n$.



length $\leq 2^n - 1$ ($\leq 2^{n+1} - 1$) + length $\leq 2^n - 1$
length $\leq 2 \cdot (2^n - 1) + 1 = 2^{n+1} - 1$

Exercise 3

Claim: $F(p_1, \ldots, p_n)$ unsatisfiable ⇒ ∃resolution proof of $\square$.

(Constructive) Proof: by induction on n.
Base case: $n = 0$; only clause is $\square$; only CNF's are $\square$ or $\{\square\}$.
Inductive step:

positive — $C_1 = C_1' \sqcup \{p\}$ where $\forall C_1' : \neg p \notin C_1'$
$C_2 = C_2' \sqcup \{\neg p\}$
$C_3 = C_1' \cup C_2'$
1-line, certainly positive

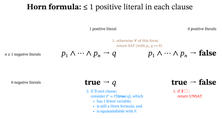
$$F[\mathbf{false}/p_n] = \bigcup_{C \in F,\, \neg p_n \notin C} C \setminus \{p_n\} \qquad F[\mathbf{true}/p_n] = \bigcup_{C \in F,\, p_n \notin C} C \setminus \{\neg p_n\}$$

This is a CNF, still unsatisfiable, in fewer variables. So:
Either $\forall C' : C \setminus \{p_n\} = C'$— done, or we have a proof of p_n.
pos
Either $\forall C' : C \setminus \{\neg p_n\} = C'$— done, or have a longer, pos. proof of $\square$.

Easy instances of SAT

Horn-CNF

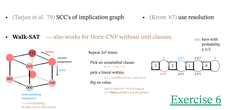
Horn formula: ≤ 1 positive literal in each clause



2-CNF

2-CNF: $\cdots \wedge (p \vee q) \wedge \cdots$

- (Tarjan et al. 79) SCC's of implication graph
- (Krom '67) use resolution
- **Walk-SAT** — also works for Horn-CNF without unit clauses.



Exercise 6

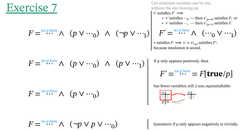
renamable Horn

Exercise 5

Horn up to flips: after $p \mapsto \neg p$ for some p's, ≤ 1 positive literal in each clause

2-use-CNF

Exercise 7



Interpolant

"Resolution has feasible monotone interpolation."

Input:
- $F(\mathbf{x}, \mathbf{z}) \wedge G(\mathbf{y}, \mathbf{z})$ is unsatisfiable
- C_1, ..., C_m is a resolution refutation
- $\mathbf{z}$ only appears positively in F

Output:
straight-line programs $e_1(\mathbf{z})$, ..., $e_m(\mathbf{z})$, monotone in $\mathbf{z}$, such that:

$$\mathrm{v}[\![e_i]\!] = 0 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{x},+\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![F]\!] = 0 \quad \text{(discard any } y \text{ and any } \neg z\text{)}$$

$$\mathrm{v}[\![e_i]\!] = 1 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{y},\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![G]\!] = 0 \quad \text{(discard any } x\text{)}$$

so $\vDash \neg e_m(\mathbf{z}) \to F(\mathbf{x}, \mathbf{z})$
and $\vDash e_m(\mathbf{z}) \to G(\mathbf{y}, \mathbf{z})$

Thanks to Mahesh Viswanathan for the correct and convenient inductive hypothesis! https://courses.grainger.illinois.edu/cs498mv/fa2018/CraigInterpolation.pdf

1. $C_i \in F$
   $e_i = \mathbf{false}$
2. $C_i \in G$
   $e_i = \mathbf{true}$
3. $C_i = C_j \setminus \{x\} \cup C_k \setminus \{\neg x\}$
   $e_i = e_j \vee e_k$
4. $C_i = C_j \setminus \{y\} \cup C_k \setminus \{\neg y\}$
   $e_i = e_j \wedge e_k$
5. $C_i = C_j \setminus \{z\} \cup C_k \setminus \{\neg z\}$
   $e_i = (e_j \vee z) \wedge e_k$
   z only appears positively in e_i

Exercise 4

Problems and solutions
courtesy of James Worrell

Logic & Proof Sheet 1

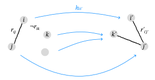
Jingjie Yang
HT'26

Propositional logic

(CNF)
A formula in n variables amounts to a function $\{0,1\}^n \to \{0,1\}$.

/ straight-line program

Example: graph homomorphism



every vertex maps to ≥ 1 vertex' every vertex maps to ≤ 1 vertex' edges map to edges'

$$\bigwedge_i \bigvee_{i'} h_{ii'} \quad \wedge \quad \bigwedge_i \bigwedge_{i' \neq j'} \neg h_{ii'} \vee \neg h_{ij'} \quad \wedge \quad \bigwedge_{i,j,i',j'} \neg h_{ii'} \vee \neg h_{jj'} \vee \neg r_{ij} \vee r'_{i'j'}$$

neg pos

This formula in $\mathbf{h}$, $\mathbf{r}$, $\mathbf{r}'$ is satisfiable iff there is a graph homomorphism $\mathbf{h}$.

Exercise 1

Write K_m for the complete graph on m vertices.

(a) homomorphism $K_m \to G$ = m-clique in G

(b) homomorphism $G \to K_m$ = m-colouring of G

Feedback!

Resolution proof

Given F in CNF: $C_1, C_2, C_3, \ldots, C_m$ (∨ clause in F, clause ∈ F)

$C_1 = C_1' \sqcup \{p\}$
$C_2 = C_2' \sqcup \{\neg p\}$
$C_3 = C_1' \cup C_2'$

Exercise 2

Claim: $F(p_1, \ldots, p_n)$ unsatisfiable ⇒ ∃resolution proof of $\square$. (of length $\leq 2^{n+1} - 1$)

Proof: by induction on n.
Base case: $n = 0$; only clause is $\square$; only CNF's are $\square$ or $\{\square\}$. (length $1 \leq 2^1 - 1$)
Inductive step:

$F[\mathbf{false}/p_n] = \bigcup_{C \in F,\, \neg p_n \notin C} C \setminus \{p_n\}$

This is a CNF, still unsatisfiable, in fewer variables. So:
Either $\forall C \in F,\ C \setminus \{p_n\} = C$ — done, or we have a proof of p_n.

$F[\mathbf{true}/p_n] = \bigcup_{C \in F,\, p_n \notin C} C \setminus \{\neg p_n\}$

This is a CNF, still unsatisfiable, in fewer variables. So:
Either $\forall C \in F,\ C \setminus \{\neg p_n\} = C$ — done, or we have a proof of $\neg p_n$.

length $\leq 2^n - 1$ ($\leq 2^{n+1} - 1$) — length $\leq 2^n - 1$

length $\leq 2 \cdot (2^n - 1) + 1 = 2^{n+1} - 1$

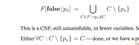
Exercise 3

Claim: $F(p_1, \ldots, p_n)$ unsatisfiable ⇒ ∃resolution proof of $\square$.

(Constructive) Proof: by induction on n.
Base case: $n = 0$; only clause is $\square$; only CNF's are $\square$ or $\{\square\}$.
Inductive step:

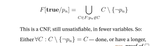
$F[\mathbf{false}/p_n] = \bigcup_{C \in F,\, \neg p_n \notin C} C \setminus \{p_n\}$

This is a CNF, still unsatisfiable, in fewer variables. So:
Either $\forall C \in F,\ C \setminus \{p_n\} = C$ — done, or we have a proof of p_n.



$F[\mathbf{true}/p_n] = \bigcup_{C \in F,\, p_n \notin C} C \setminus \{\neg p_n\}$

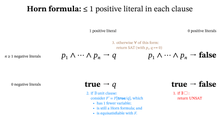
This is a CNF, still unsatisfiable, in fewer variables. So:
Either $\forall C \in F,\ C \setminus \{\neg p_n\} = C$ — done, or have a longer, pos. proof of $\square$.



Easy instances of SAT

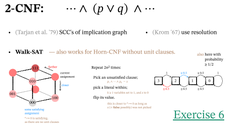
Horn-CNF

Horn formula: ≤ 1 positive literal in each clause



2-CNF

2-CNF: $\cdots \wedge (p \vee q) \wedge \cdots$



Exercise 6

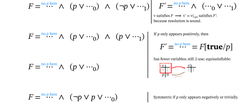
renamable Horn

Exercise 5

Horn up to flips: after $p \mapsto \neg p$ for some p's, ≤ 1 positive literal in each clause

2-use-CNF

Exercise 7



Interpolant

"Resolution has feasible monotone interpolation."

Input:
- $F(\mathbf{x}, \mathbf{z}) \wedge G(\mathbf{y}, \mathbf{z})$ is unsatisfiable
- C_1, ..., C_m is a resolution refutation
- $\mathbf{z}$ only appears positively in F

Output:
straight-line programs $e_1(\mathbf{z})$, ..., $e_m(\mathbf{z})$, monotone in $\mathbf{z}$, such that:

$$\mathrm{v}[\![e_i]\!] = 0 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{x},+\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![F]\!] = 0 \quad \text{(discard any } y \text{ and any } \neg z\text{)}$$

$$\mathrm{v}[\![e_i]\!] = 1 \text{ and } \mathrm{v}[\![C_i|_{\mathbf{y},\mathbf{z}}]\!] = 0 \implies \mathrm{v}[\![G]\!] = 0 \quad \text{(discard any } x\text{)}$$

so $\vDash \neg e_m(\mathbf{z}) \to F(\mathbf{x}, \mathbf{z})$
and $\vDash e_m(\mathbf{z}) \to G(\mathbf{y}, \mathbf{z})$

Thanks to Mahesh Viswanathan for the correct and convenient inductive hypothesis! https://courses.grainger.illinois.edu/cs498mv/fa2018/CraigInterpolation.pdf

1. $C_i \in F$
   $e_i = \mathbf{false}$
2. $C_i \in G$
   $e_i = \mathbf{true}$
3. $C_i = C_j \setminus \{x\} \cup C_k \setminus \{\neg x\}$
   $e_i = e_j \vee e_k$
4. $C_i = C_j \setminus \{y\} \cup C_k \setminus \{\neg y\}$
   $e_i = e_j \wedge e_k$
5. $C_i = C_j \setminus \{z\} \cup C_k \setminus \{\neg z\}$
   $e_i = (e_j \vee z) \wedge e_k$
   z only appears positively in e_i

Exercise 4