binary decision diagrams

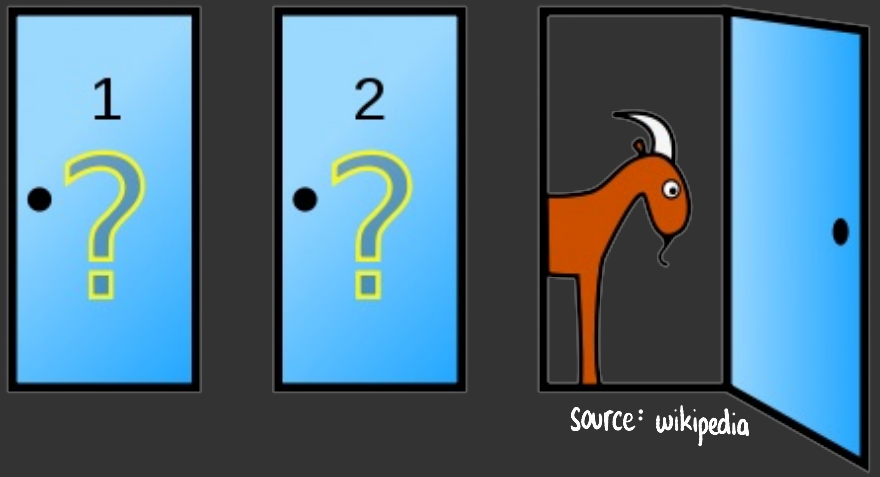
CSCI 373

Ask Marilyn™

BY MARILYN VOS SAVANT

Suppose you're on a game show, and you're given the choice of three doors: Behind one door is a car; behind the others, goats. You pick a door, say No. 1, and the host, who knows what's behind the doors, opens another door, say No. 3, which has a goat. He then says to you, "Do you want to pick door No. 2?" Is it to your advantage to switch your choice?

—Craig F. Whitaker, Columbia, Md.



should you switch?

let C_d be whether the car is behind door d

let G_d be whether you guess door d

let W be whether you win

express the monty hall problem as a cnf sentence

let C_d be whether the car is behind door d

let G_d be whether you guess door d

let W be whether you win

the car is behind exactly one door

$$(C_1 \vee C_2 \vee C_3) \wedge (\neg C_1 \vee \neg C_2) \wedge (\neg C_1 \vee \neg C_3) \wedge (\neg C_2 \vee \neg C_3)$$

you make exactly one guess

$$(G_1 \vee G_2 \vee G_3) \wedge (\neg G_1 \vee \neg G_2) \wedge (\neg G_1 \vee \neg G_3) \wedge (\neg G_2 \vee \neg G_3)$$

you win if you guess the door with the car

$$W \Leftrightarrow (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)$$

let C_d be whether the car is behind door d

let G_d be whether you guess door d

let W be whether you win

the car is behind exactly one door

$$(C_1 \vee C_2 \vee C_3) \wedge (\neg C_1 \vee \neg C_2) \wedge (\neg C_1 \vee \neg C_3) \wedge (\neg C_2 \vee \neg C_3)$$ ← cnf

you make exactly one guess

$$(G_1 \vee G_2 \vee G_3) \wedge (\neg G_1 \vee \neg G_2) \wedge (\neg G_1 \vee \neg G_3) \wedge (\neg G_2 \vee \neg G_3)$$ ← cnf

you win if you guess the door with the car

$$W \Leftrightarrow (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)$$ ← not cnf

recall how to convert to cnf

1. eliminate implications

pattern	rewrite as
$\alpha \Rightarrow \beta$	$\neg\alpha \vee \beta$
until no more rewrites possible	

2. move $\neg$ inward

pattern	rewrite as
$\neg\neg\alpha$	α
$\neg(\alpha \vee \beta)$	$\neg\alpha \wedge \neg\beta$
$\neg(\alpha \wedge \beta)$	$\neg\alpha \vee \neg\beta$
until no more rewrites possible	

3. distribute

pattern	rewrite as
$\alpha \wedge (\beta \vee \gamma)$	$(\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$
$(\alpha \vee \beta) \wedge \gamma$	$(\alpha \wedge \gamma) \vee (\beta \wedge \gamma)$
$\alpha \vee (\beta \wedge \gamma)$	$(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$
$(\alpha \wedge \beta) \vee \gamma$	$(\alpha \vee \gamma) \wedge (\beta \vee \gamma)$
until no more rewrites possible	

you win if you guess the door with the car

$$W \Leftrightarrow (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)$$

← not cnf

1. eliminate implications

pattern	rewrite as
$\alpha \Rightarrow \beta$	$\neg\alpha \vee \beta$
until no more rewrites possible	

$$W \Leftrightarrow (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)$$

$$\left(W \Rightarrow (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)\right) \wedge \left((C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3) \Rightarrow W\right)$$

$$\left(\neg W \vee (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)\right) \wedge \neg\left((C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)\right) \vee W$$

1. eliminate implications

pattern	rewrite as
$\alpha \Rightarrow \beta$	$\neg\alpha \vee \beta$
until no more rewrites possible	

$$W \Leftrightarrow (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)$$

$$\left(W \Rightarrow (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)\right) \wedge \left((C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3) \Rightarrow W\right)$$

$$\left(\neg W \vee (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)\right) \wedge \neg\left((C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)\right) \vee W$$

1. eliminate implications

pattern	rewrite as
$\alpha \Rightarrow \beta$	$\neg\alpha \vee \beta$
until no more rewrites possible	

$$W \Leftrightarrow (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)$$

$$\Big(W \Rightarrow (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)\Big) \wedge \Big((C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3) \Rightarrow W\Big)$$

$$\Big(\neg W \vee (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)\Big) \wedge \neg\Big((C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)\Big) \vee W$$

2. move ¬ inward

pattern	rewrite as
$\neg\neg\alpha$	α
$\neg(\alpha \vee \beta)$	$\neg\alpha \wedge \neg\beta$
$\neg(\alpha \wedge \beta)$	$\neg\alpha \vee \neg\beta$
until no more rewrites possible	

$$\left(\neg W \vee (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)\right) \wedge \left(\neg\left((C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)\right) \vee W\right)$$

$$\left(\neg W \vee (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)\right) \wedge \left(\neg(C_1 \wedge G_1) \wedge \neg(C_2 \wedge G_2) \wedge \neg(C_3 \wedge G_3)) \vee W\right)$$

$$\left(\neg W \vee (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)\right) \wedge \left((\neg C_1 \vee \neg G_1) \wedge (\neg C_2 \vee \neg G_2) \wedge (\neg C_3 \vee \neg G_3)) \vee W\right)$$

2. move ¬ inward

pattern	rewrite as
$\neg\neg\alpha$	α
$\neg(\alpha \vee \beta)$	$\neg\alpha \wedge \neg\beta$
$\neg(\alpha \wedge \beta)$	$\neg\alpha \vee \neg\beta$
until no more rewrites possible	

$$\left(\neg W \vee (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)\right) \wedge \left(\neg\left((C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)\right) \vee W\right)$$

$$\left(\neg W \vee (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)\right) \wedge \left(\neg(C_1 \wedge G_1) \wedge \neg(C_2 \wedge G_2) \wedge \neg(C_3 \wedge G_3)\right) \vee W\Big)$$

$$\left(\neg W \vee (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)\right) \wedge \left((\neg C_1 \vee \neg G_1) \wedge (\neg C_2 \vee \neg G_2) \wedge (\neg C_3 \vee \neg G_3)\right) \vee W\Big)$$

2. move ¬ inward

pattern	rewrite as
$\neg\neg\alpha$	α
$\neg(\alpha \vee \beta)$	$\neg\alpha \wedge \neg\beta$
$\neg(\alpha \wedge \beta)$	$\neg\alpha \vee \neg\beta$
until no more rewrites possible	

$$\left(\neg W \vee (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)\right) \wedge \left(\neg\left((C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)\right) \vee W\right)$$

$$\left(\neg W \vee (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)\right) \wedge \left(\neg(C_1 \wedge G_1) \wedge \neg(C_2 \wedge G_2) \wedge \neg(C_3 \wedge G_3)) \vee W\right)$$

$$\left(\neg W \vee (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)\right) \wedge \left((\neg C_1 \vee \neg G_1) \wedge (\neg C_2 \vee \neg G_2) \wedge (\neg C_3 \vee \neg G_3)) \vee W\right)$$

3. distribute

pattern	rewrite as
$\alpha \wedge (\beta \vee \gamma)$	$(\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$
$(\alpha \vee \beta) \wedge \gamma$	$(\alpha \wedge \gamma) \vee (\beta \wedge \gamma)$
$\alpha \vee (\beta \wedge \gamma)$	$(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$
$(\alpha \wedge \beta) \vee \gamma$	$(\alpha \vee \gamma) \wedge (\beta \vee \gamma)$
until no more rewrites possible	

$$\left(\neg W \vee (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)\right) \wedge \left((\neg C_1 \vee \neg G_1) \wedge (\neg C_2 \vee \neg G_2) \wedge (\neg C_3 \vee \neg G_3)\right) \vee W\Big)$$

$$\left(\neg W \vee (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)\right) \wedge \left((\neg C_1 \vee \neg G_1 \vee W) \wedge (\neg C_2 \vee \neg G_2 \vee W) \wedge (\neg C_3 \vee \neg G_3 \vee W)\right)$$

3. distribute

pattern	rewrite as
$\alpha \wedge (\beta \vee \gamma)$	$(\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$
$(\alpha \vee \beta) \wedge \gamma$	$(\alpha \wedge \gamma) \vee (\beta \wedge \gamma)$
$\alpha \vee (\beta \wedge \gamma)$	$(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$
$(\alpha \wedge \beta) \vee \gamma$	$(\alpha \vee \gamma) \wedge (\beta \vee \gamma)$
until no more rewrites possible	

$$\left(\neg W \vee (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)\right) \wedge \left((\neg C_1 \vee \neg G_1) \wedge (\neg C_2 \vee \neg G_2) \wedge (\neg C_3 \vee \neg G_3)) \vee W\right)$$

$$\left(\neg W \vee (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)\right) \wedge \left((\neg C_1 \vee \neg G_1 \vee W) \wedge (\neg C_2 \vee \neg G_2 \vee W) \wedge (\neg C_3 \vee \neg G_3 \vee W)\right)$$

3. distribute

pattern	rewrite as
$\alpha \wedge (\beta \vee \gamma)$	$(\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$
$(\alpha \vee \beta) \wedge \gamma$	$(\alpha \wedge \gamma) \vee (\beta \wedge \gamma)$
$\alpha \vee (\beta \wedge \gamma)$	$(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$
$(\alpha \wedge \beta) \vee \gamma$	$(\alpha \vee \gamma) \wedge (\beta \vee \gamma)$
until no more rewrites possible	

$$\left(\neg W \vee (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)\right) \wedge \left((\neg C_1 \vee \neg G_1) \wedge (\neg C_2 \vee \neg G_2) \wedge (\neg C_3 \vee \neg G_3)) \vee W\right)$$

$$\underbrace{\left(\neg W \vee (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)\right)}_{\text{not yet cnf}} \wedge \underbrace{\left((\neg C_1 \vee \neg G_1 \vee W) \wedge (\neg C_2 \vee \neg G_2 \vee W) \wedge (\neg C_3 \vee \neg G_3 \vee W)\right)}_{\text{cnf}}$$

3. distribute

pattern	rewrite as
$\alpha \wedge (\beta \vee \gamma)$	$(\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$
$(\alpha \vee \beta) \wedge \gamma$	$(\alpha \wedge \gamma) \vee (\beta \wedge \gamma)$
$\alpha \vee (\beta \wedge \gamma)$	$(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$
$(\alpha \wedge \beta) \vee \gamma$	$(\alpha \vee \gamma) \wedge (\beta \vee \gamma)$
until no more rewrites possible	

if we had n doors, we would have ended up with 2^n clauses

3 doors

$$\neg w \vee (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)$$

$$\neg w \vee (C_1 \wedge G_1) \vee ((C_2 \vee C_3) \wedge (C_2 \vee G_3) \wedge (G_2 \vee C_3) \wedge (G_2 \vee G_3))$$

$$\neg w \vee ((C_1 \vee C_2 \vee C_3) \wedge (C_1 \vee C_2 \vee G_3) \wedge (C_1 \vee G_2 \vee C_3) \wedge (C_1 \vee G_2 \vee G_3) \wedge (G_1 \vee C_2 \vee C_3) \wedge (G_1 \vee C_2 \vee G_3) \wedge (G_1 \vee G_2 \vee C_3) \wedge (G_1 \vee G_2 \vee G_3))$$

$$(\neg w \vee C_1 \vee C_2 \vee C_3) \wedge (\neg w \vee C_1 \vee C_2 \vee G_3) \wedge (\neg w \vee C_1 \vee G_2 \vee C_3) \wedge (\neg w \vee C_1 \vee G_2 \vee G_3) \wedge (\neg w \vee G_1 \vee C_2 \vee C_3) \wedge (\neg w \vee G_1 \vee C_2 \vee G_3) \wedge (\neg w \vee G_1 \vee G_2 \vee C_3) \wedge (\neg w \vee G_1 \vee G_2 \vee G_3)$$

2^3 clauses

$$W \Leftrightarrow (C_1 \wedge G_1) \vee (C_2 \wedge G_2) \vee (C_3 \wedge G_3)$$

becomes the following

cnf sentence:

$$(\neg W \vee C_1 \vee C_2 \vee C_3) \wedge (\neg W \vee C_1 \vee C_2 \vee G_3) \wedge (\neg W \vee C_1 \vee G_2 \vee C_3) \wedge (\neg W \vee C_1 \vee G_2 \vee G_3)$$

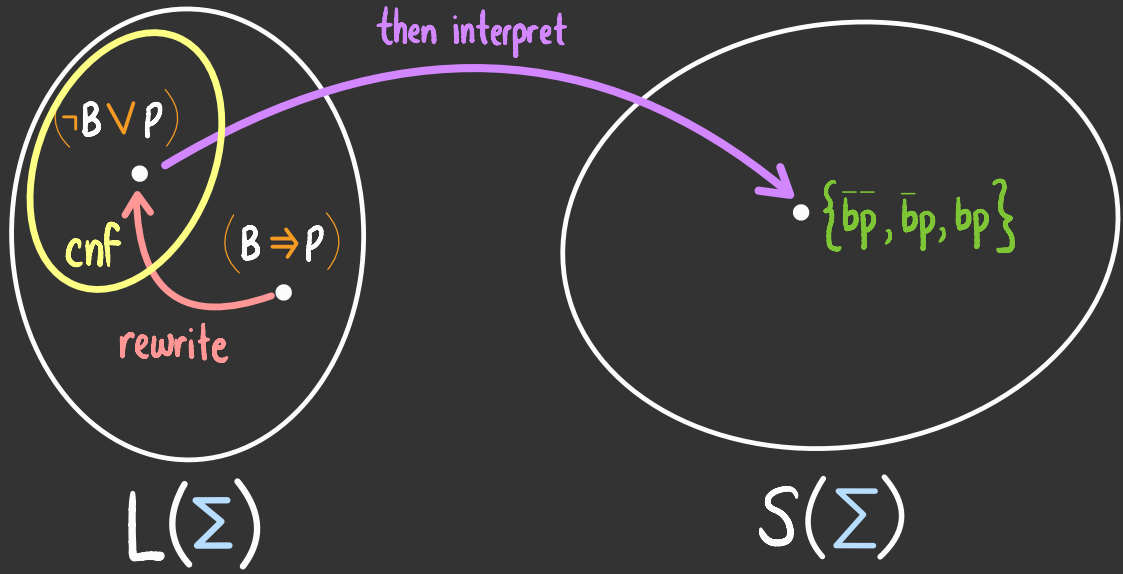
$$\wedge (\neg W \vee G_1 \vee C_2 \vee C_3) \wedge (\neg W \vee G_1 \vee C_2 \vee G_3) \wedge (\neg W \vee G_1 \vee G_2 \vee C_3) \wedge (\neg W \vee G_1 \vee G_2 \vee G)$$

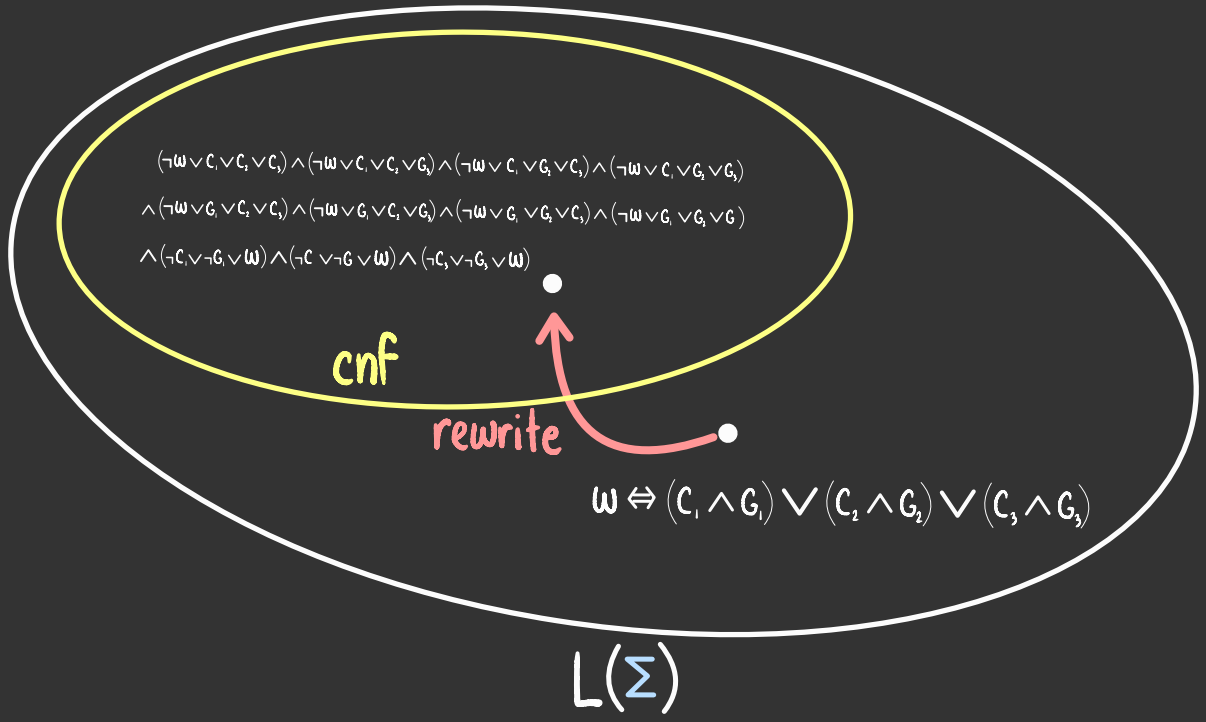
$$\wedge (\neg C_1 \vee \neg G_1 \vee W) \wedge (\neg C \vee \neg G \vee W) \wedge (\neg C_3 \vee \neg G_3 \vee W)$$

Can we avoid this exponential blowup?

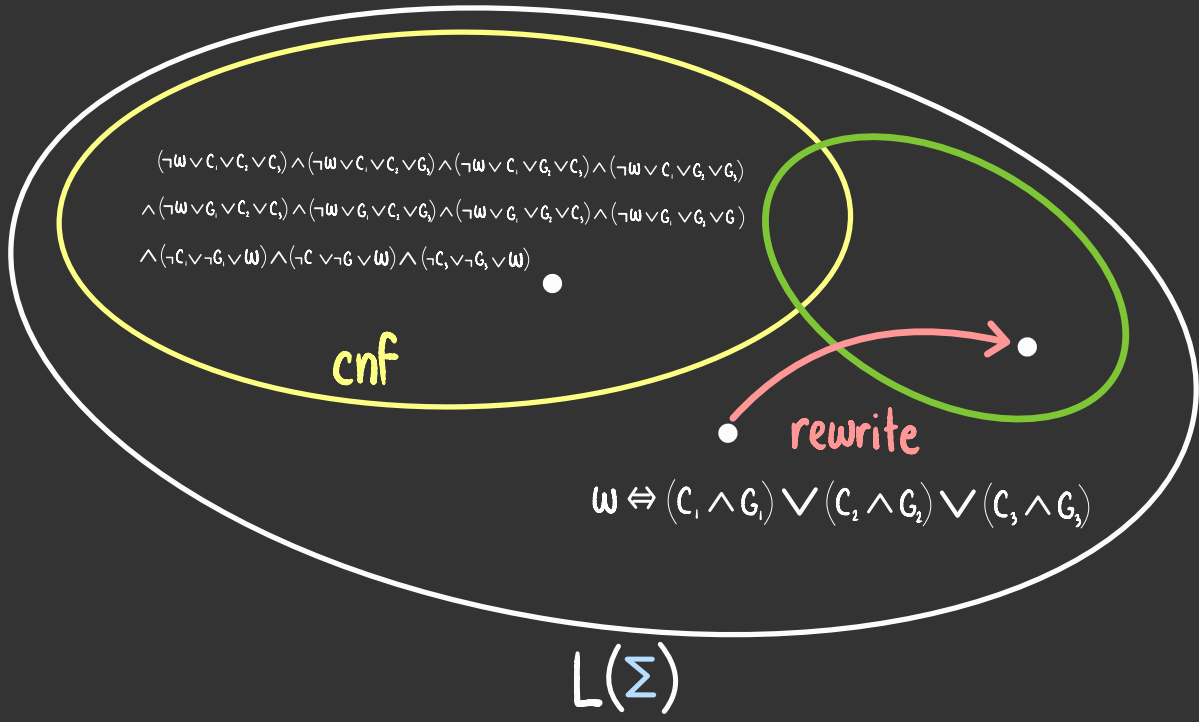
recall this earlier claim:

since any propositional sentence can be rewritten as a logically equivalent cnf sentence, we can focus on reasoning using cnf sentences





but some sentences only have cnf equivalents that are exponential in the size of the original sentence



are there alternative subspaces of the propositional language?

"the car is behind exactly one door"

in cnf: $(C_1 \vee C_2 \vee C_3) \wedge (\neg C_1 \vee \neg C_2) \wedge (\neg C_1 \vee \neg C_3) \wedge (\neg C_2 \vee \neg C_3)$

how would we express this as a disjunction of conjunctions?

"the car is behind exactly one door"

in cnf: $(C_1 \vee C_2 \vee C_3) \wedge (\neg C_1 \vee \neg C_2) \wedge (\neg C_1 \vee \neg C_3) \wedge (\neg C_2 \vee \neg C_3)$

in dnf: $(C_1 \wedge \neg C_2 \wedge \neg C_3) \vee (\neg C_1 \wedge C_2 \wedge \neg C_3) \vee (\neg C_1 \wedge \neg C_2 \wedge C_3)$

a disjunction of conjunctions is called

disjunctive normal form (dnf)

"the car is behind exactly one door"

in cnf: $(C_1 \vee C_2 \vee C_3) \wedge (\neg C_1 \vee \neg C_2) \wedge (\neg C_1 \vee \neg C_3) \wedge (\neg C_2 \vee \neg C_3)$

in dnf: $(C_1 \wedge \neg C_2 \wedge \neg C_3) \vee (\neg C_1 \wedge C_2 \wedge \neg C_3) \vee (\neg C_1 \wedge \neg C_2 \wedge C_3)$

but is this any better?

"the car is behind exactly one of n doors"

in cnf: $(C_1 \vee \cdots \vee C_n) \wedge \bigvee_{\substack{i,j \in \{1,\ldots,n\} \\ i<j}} (\neg C_i \vee \neg C_j)$

how many literals?

in dnf: $(C_1 \wedge \neg C_2 \wedge \cdots \wedge \neg C_n) \vee \cdots \vee (\neg C_1 \wedge \cdots \wedge \neg C_{n-1} \wedge C_n)$

how many literals?

"the car is behind exactly one of n doors"

in cnf: $(C_1 \vee \cdots \vee C_n) \wedge \bigvee_{\substack{i,j \in \{1,\ldots,n\} \\ i<j}} (\neg C_i \vee \neg C_j)$

$O(n^2)$ literals

in dnf: $(C_1 \wedge \neg C_2 \wedge \cdots \wedge \neg C_n) \vee \cdots \vee (\neg C_1 \wedge \cdots \wedge \neg C_{n-1} \wedge C_n)$

$O(n^2)$ literals

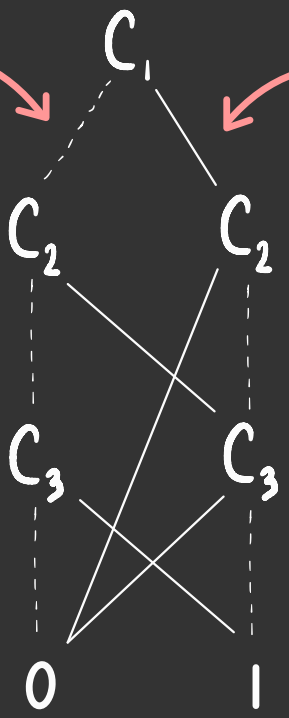
"the car is behind exactly one of n doors"

Claim: we can express this as a propositional sentence with $O(n)$ literals

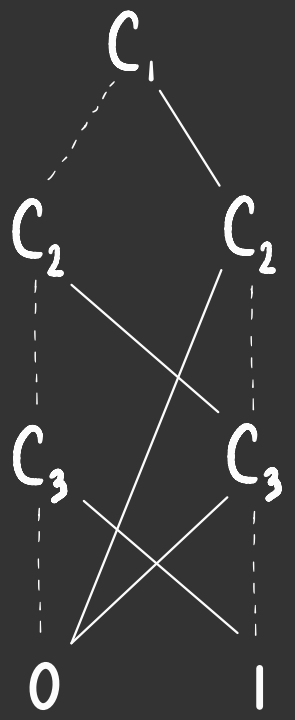
"the car is behind exactly one of 3 doors"

dotted line:
assign zero
to the parent

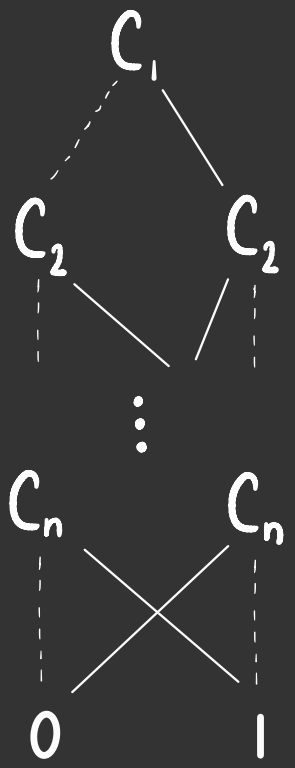
regular line:
assign one
to the parent



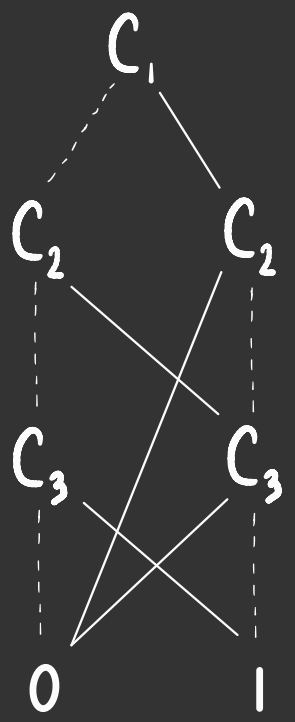
"the car is behind exactly one of 3 doors"



"the car is behind exactly one of n doors"



"the car is behind exactly one of 3 doors"

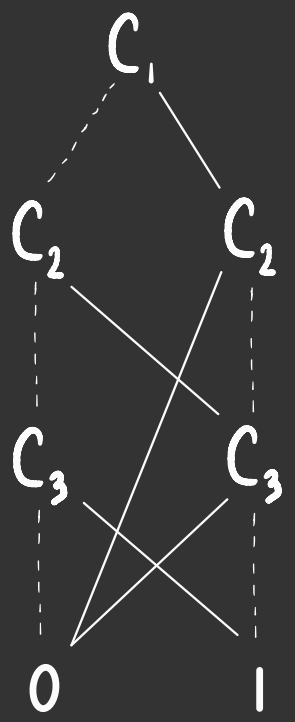


this is called a

binary decision diagram

(bdd)

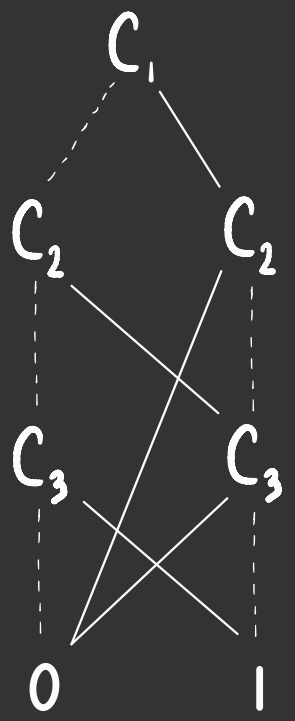
"the car is behind exactly one of 3 doors"



the rules of bdds:

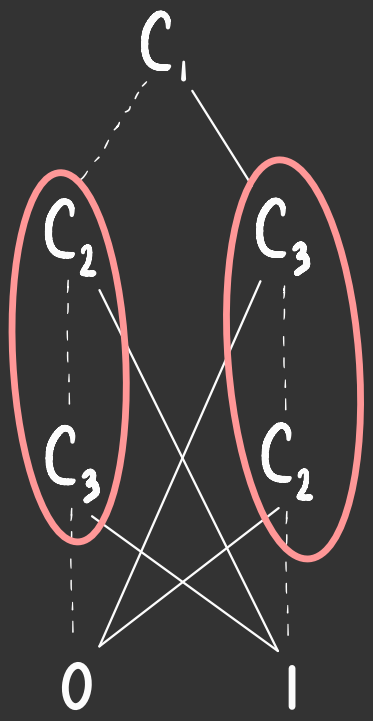
- bdds are directed acyclic graphs (directionality is top-down)
- internal nodes are labeled with signature variables
- sinks (leaves) are labeled 0 or 1
- bdds are ordered: if one path visits C_i before C_j, then no path can visit C_j before C_i
- bdds are reduced: no two nodes have identical subgraphs or two identical children

"the car is behind exactly one of 3 doors"



the rules of bdds:

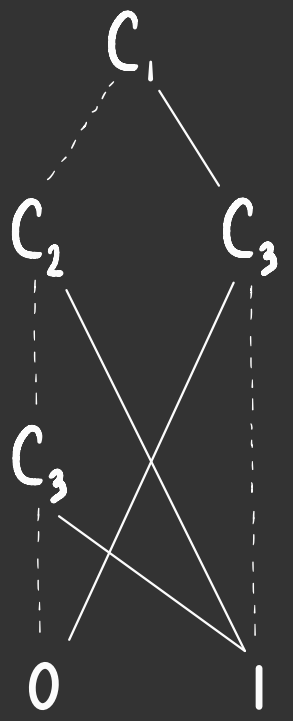
- bdds are directed acyclic graphs (directionality is top-down)
- internal nodes are labeled with signature variables
- sinks (leaves) are labeled 0 or 1
- bdds are ordered: if one path visits C_i before C_j, then no path can visit C_j before C_i
- bdds are reduced: no two nodes have identical subgraphs or two identical children



not ordered

the rules of bdds:

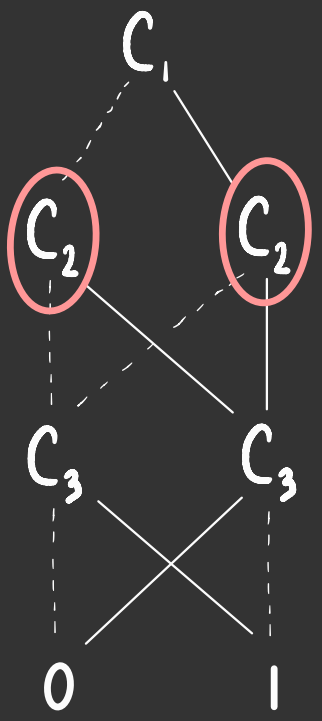
- bdds are directed acyclic graphs (directionality is top-down)
- internal nodes are labeled with signature variables
- sinks (leaves) are labeled 0 or 1
- bdds are ordered: if one path visits C_i before C_j, then no path can visit C_j before C_i
- bdds are reduced: no two nodes have identical subgraphs or two identical children



ordered

the rules of bdds:

- bdds are directed acyclic graphs (directionality is top-down)
- internal nodes are labeled with signature variables
- sinks (leaves) are labeled 0 or 1
- bdds are ordered: if one path visits C_i before C_j, then no path can visit C_j before C_i
- bdds are reduced: no two nodes have identical subgraphs or two identical children

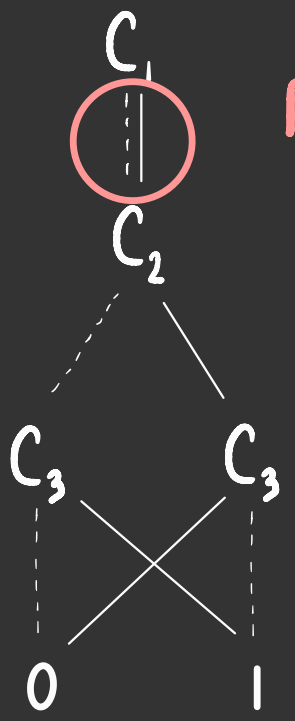


identical subgraphs

not reduced

the rules of bdds:

- bdds are directed acyclic graphs (directionality is top-down)
- internal nodes are labeled with signature variables
- sinks (leaves) are labeled 0 or 1
- bdds are ordered: if one path visits C_i before C_j, then no path can visit C_j before C_i
- bdds are reduced: no two nodes have identical subgraphs or two identical children

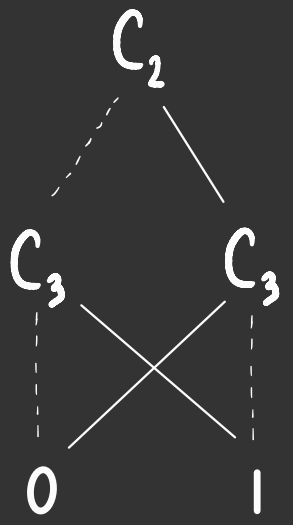


both edges point to the same node

not reduced

the rules of bdds:

- bdds are directed acyclic graphs (directionality is top-down)
- internal nodes are labeled with signature variables
- sinks (leaves) are labeled 0 or 1
- bdds are ordered: if one path visits C_i before C_j, then no path can visit C_j before C_i
- bdds are reduced: no two nodes have identical subgraphs or two identical children



reduced

the rules of bdds:

- bdds are directed acyclic graphs (directionality is top-down)
- internal nodes are labeled with signature variables
- sinks (leaves) are labeled 0 or 1
- bdds are ordered: if one path visits C_i before C_j, then no path can visit C_j before C_i
- bdds are reduced: no two nodes have identical subgraphs or two identical children

"one of the only really fundamental data structures that came out in the last twenty-five years"

– donald knuth, 2008

monty hall

let C_d be whether the car is behind door d

let G_d be whether you guess door d

let W be whether you win

- the car is behind exactly one door
- you make exactly one guess
- you win if you guess the door with the car

how can we express these rules as a bdd?

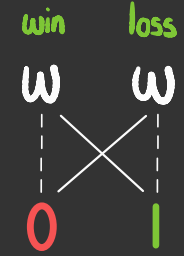
monty hall

let C_d be whether the car is behind door d

let G_d be whether you guess door d

let W be whether you win

- the car is behind exactly one door
- you make exactly one guess
- you win if you guess the door with the car



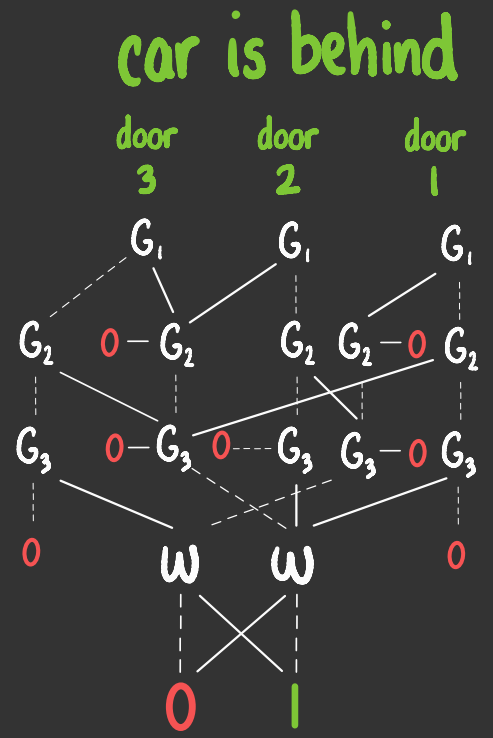
monty hall

let C_d be whether the car is behind door d

let G_d be whether you guess door d

let W be whether you win

- the car is behind exactly one door
- you make exactly one guess
- you win if you guess the door with the car



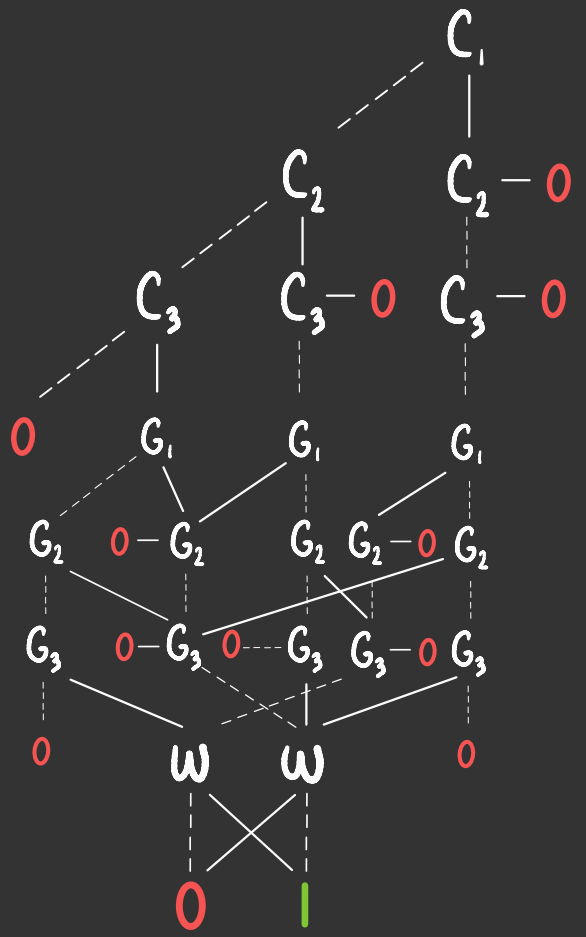
monty hall

let C_d be whether the car is behind door d

let G_d be whether you guess door d

let W be whether you win

- the car is behind exactly one door
- you make exactly one guess
- you win if you guess the door with the car



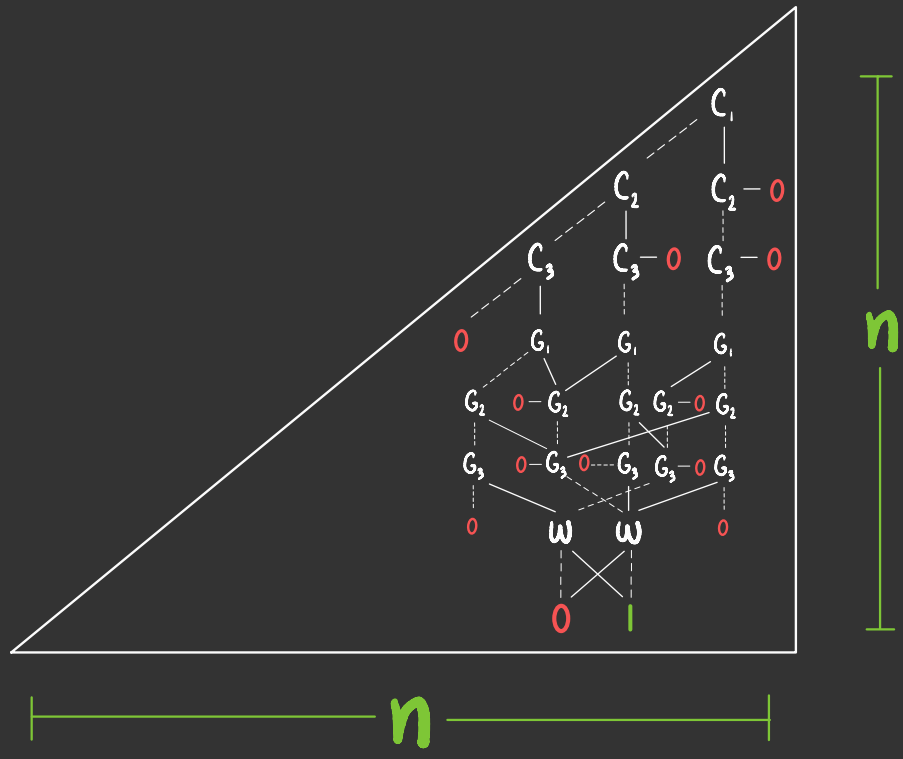
monty hall

let C_d be whether the car is behind door d

let G_d be whether you guess door d

let W be whether you win

- the car is behind exactly one door
- you make exactly one guess
- you win if you guess the door with the car



this is $O(n^2)$ nodes

monty hall

let C_d be whether the car is behind door d

let G_d be whether you guess door d

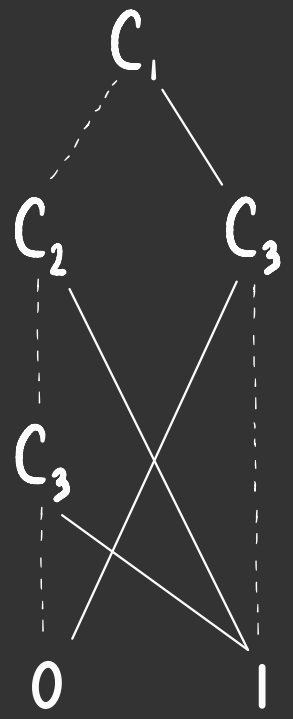
let W be whether you win

- the car is behind exactly one door
- you make exactly one guess
- you win if you guess the door with the car

$$(\neg W \vee C_1 \vee C_2 \vee C_3) \wedge (\neg W \vee C_1 \vee C_2 \vee G_3) \wedge (\neg W \vee C_1 \vee G_2 \vee C_3) \wedge (\neg W \vee C_1 \vee G_2 \vee G_3)$$
$$\wedge (\neg W \vee G_1 \vee C_2 \vee C_3) \wedge (\neg W \vee G_1 \vee C_2 \vee G_3) \wedge (\neg W \vee G_1 \vee G_2 \vee C_3) \wedge (\neg W \vee G_1 \vee G_2 \vee G)$$
$$\wedge (\neg C_1 \vee \neg G_1 \vee W) \wedge (\neg C_2 \vee \neg G \vee W) \wedge (\neg C_3 \vee \neg G_3 \vee W)$$

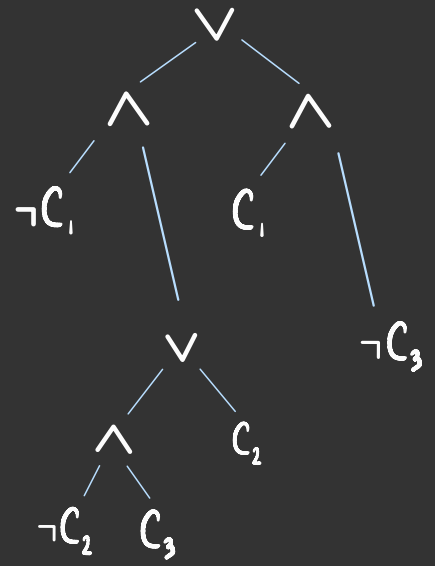
in contrast, the cnf representation has $O(2^n)$ clauses

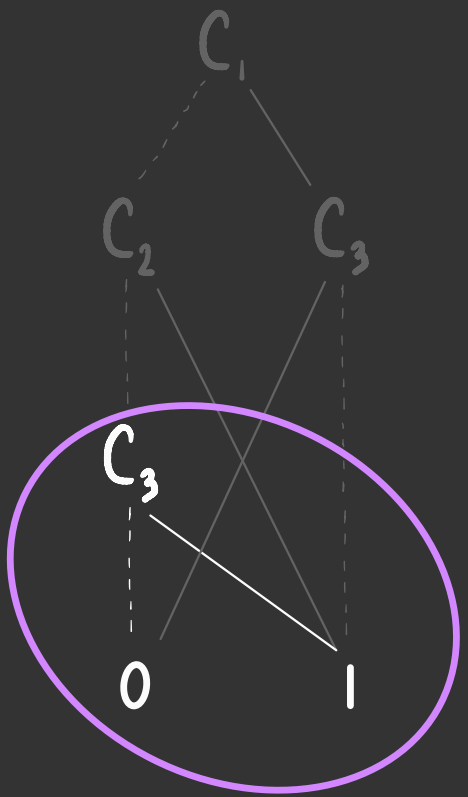
but this isn't a propositional logic sentence...



but this isn't a propositional logic sentence...

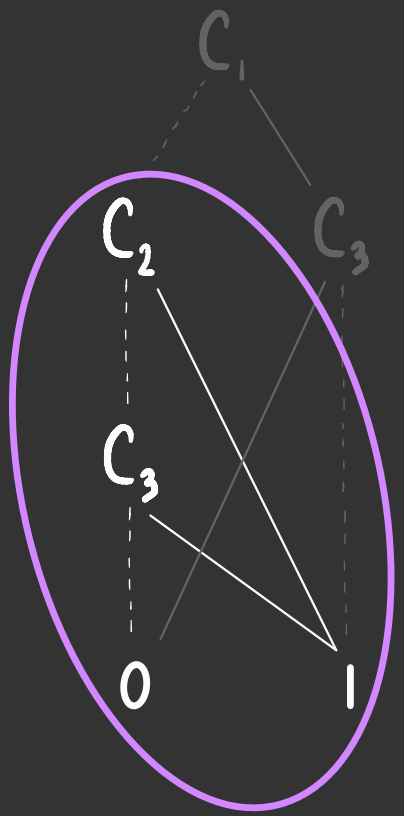
or is it?





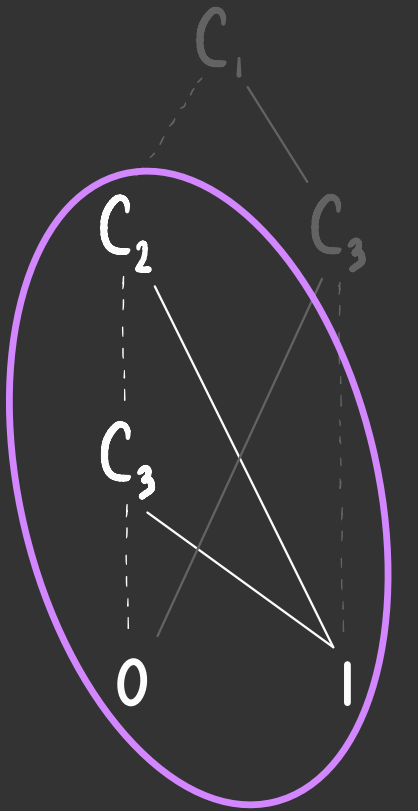
these models are

$I(C_3)$



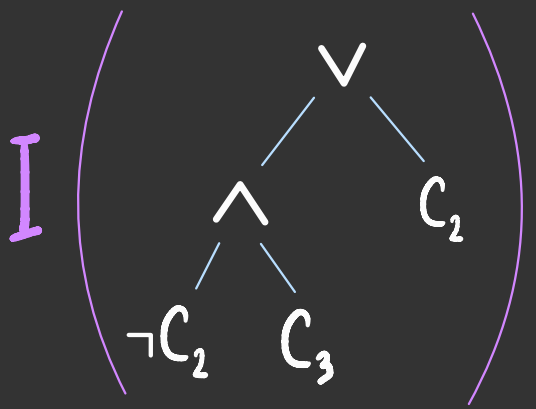
these models are

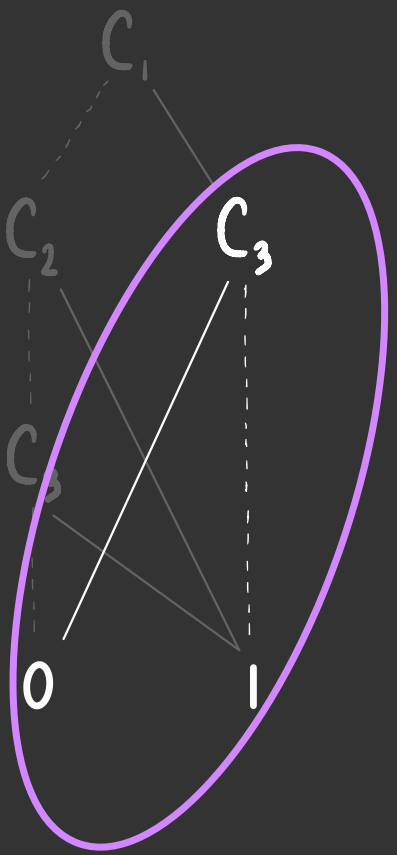
$$I((\neg C_2 \wedge C_3) \vee C_2)$$



$I((\neg C_2 \wedge C_3) \vee C_2)$

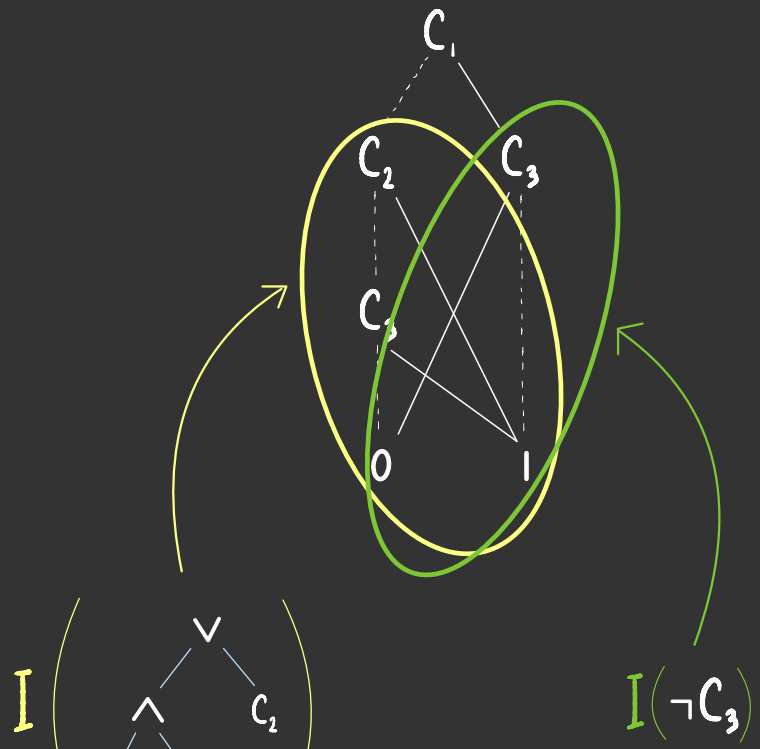
these models are

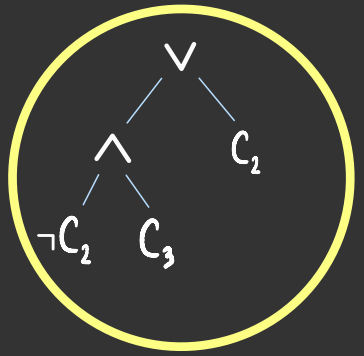




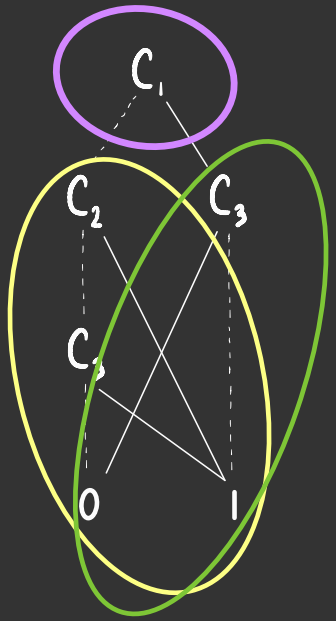
these models are

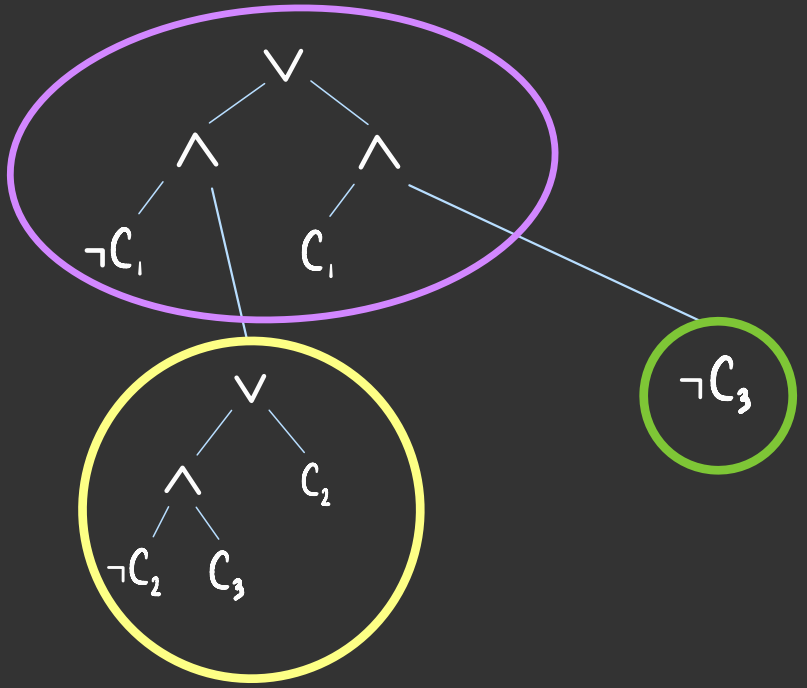
$$I(\neg C_3)$$

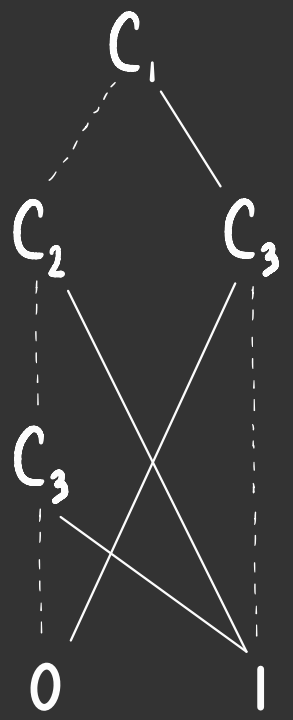

C1
C2
C3
C4
0
1
I
∨
∧
C2
¬C2
C3
I(¬C3)


∨
∧
C2
¬C2
C3

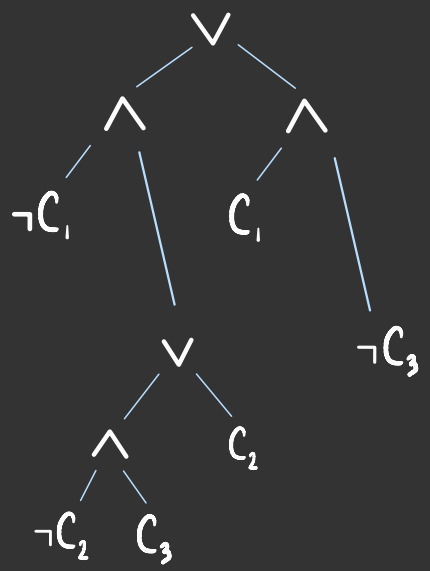

¬C3


C_1
C_2
C_3
C_3
0
1

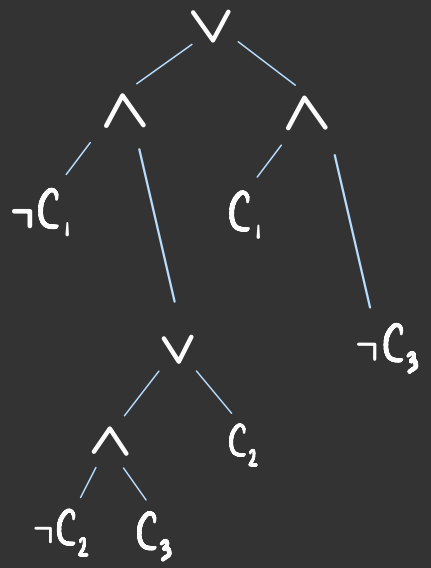

∨
∧
∧
$\neg C_1$
C_1
$\neg C_3$
∨
∧
C_2
$\neg C_2$
C_3



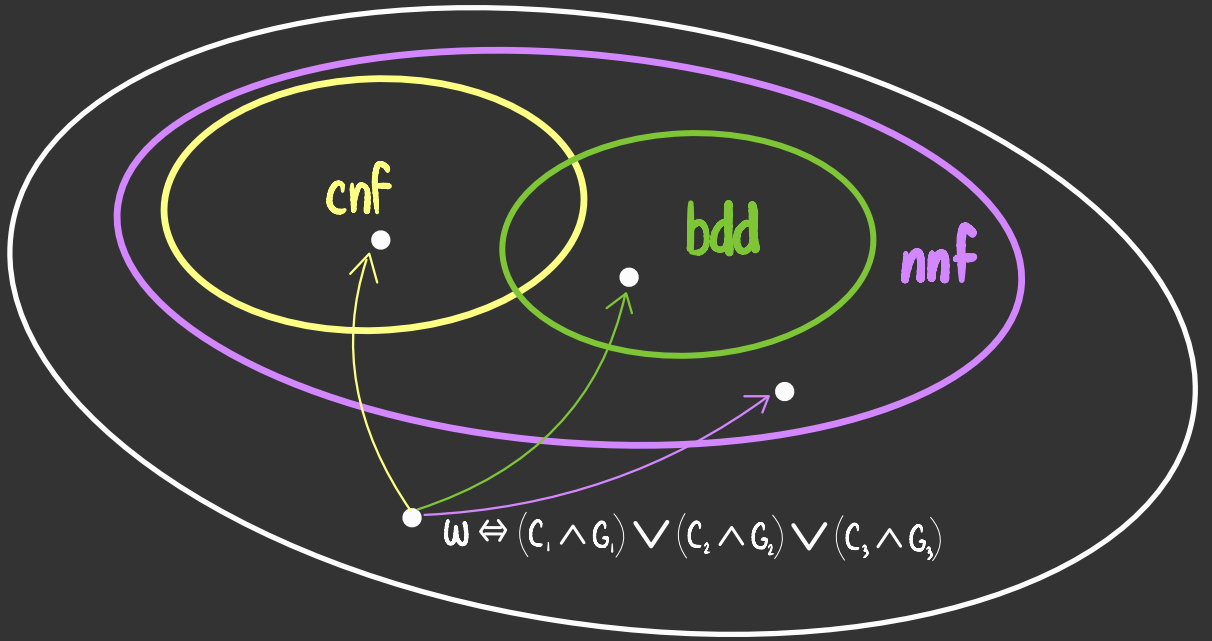
is shorthand for



this is negation normal form (nnf)



an and-or graph with literals at the leaves

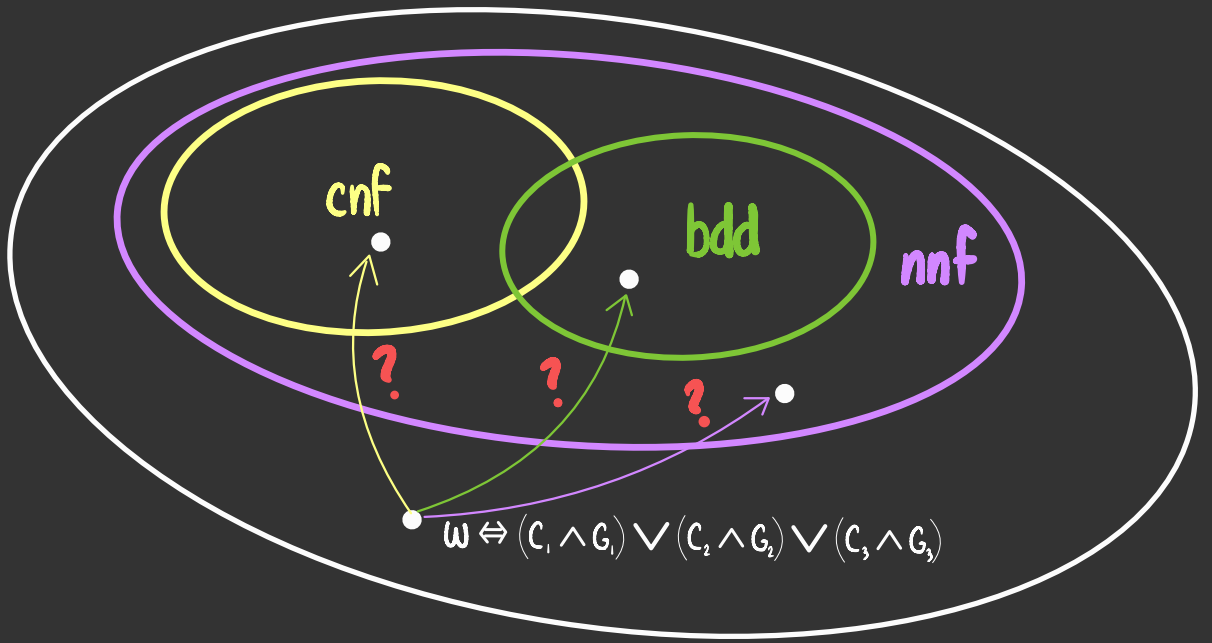


language space $L(\Sigma)$

every sentence in propositional logic

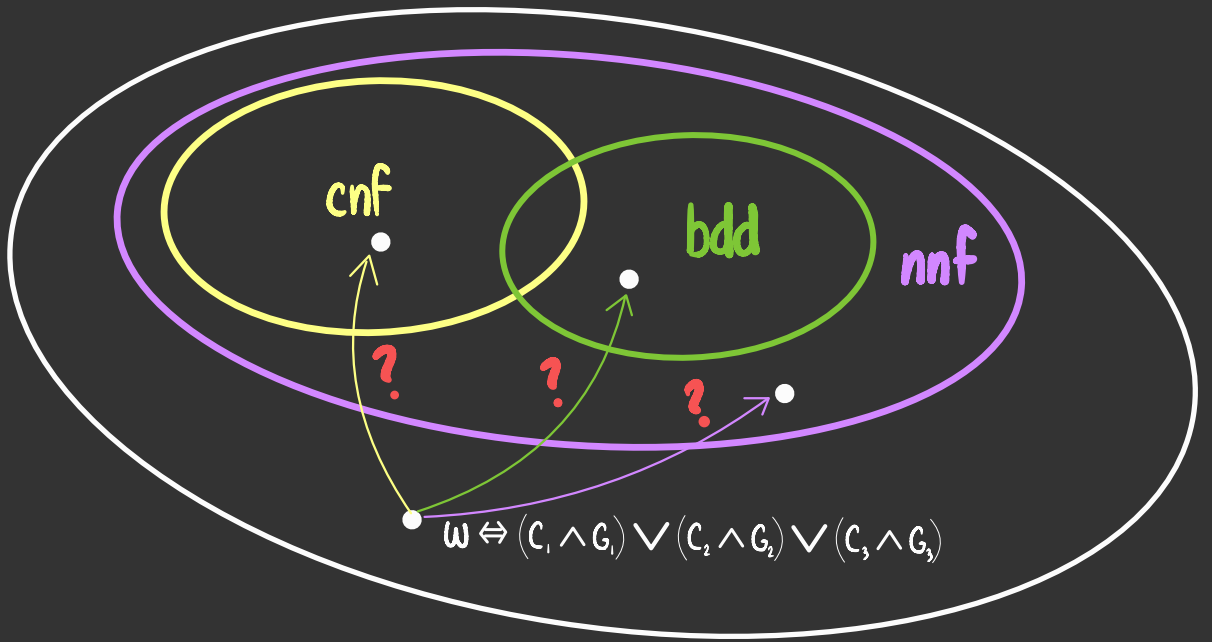
has a logically equivalent

cnf / bdd / nnf representation



language space $L(\Sigma)$

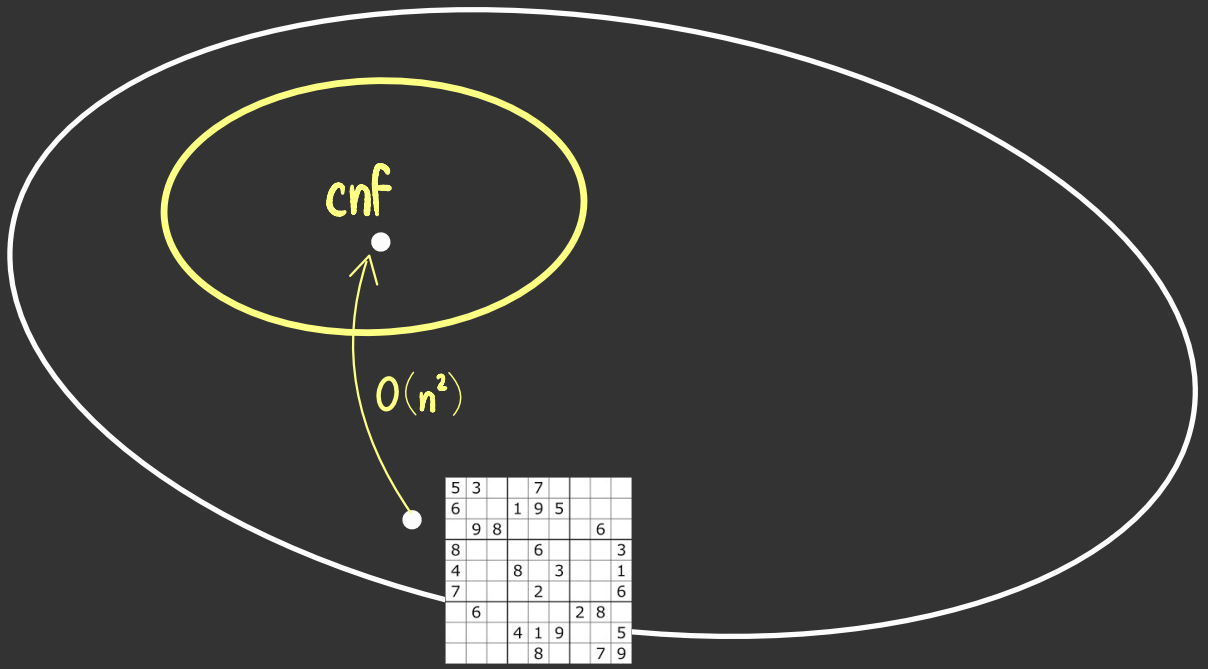
so how do we choose a target representation?



language space $L(\Sigma)$

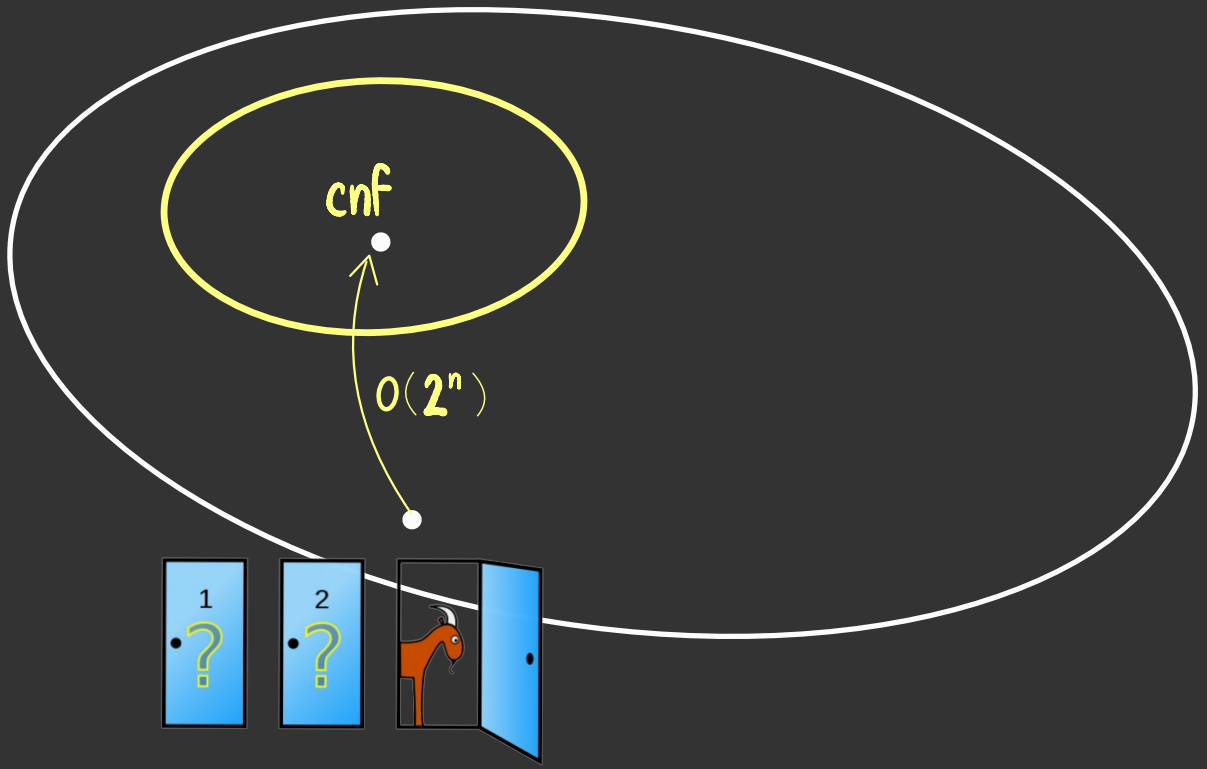
two criteria

1. how succinct is the representation?
2. what can we do with the representation?



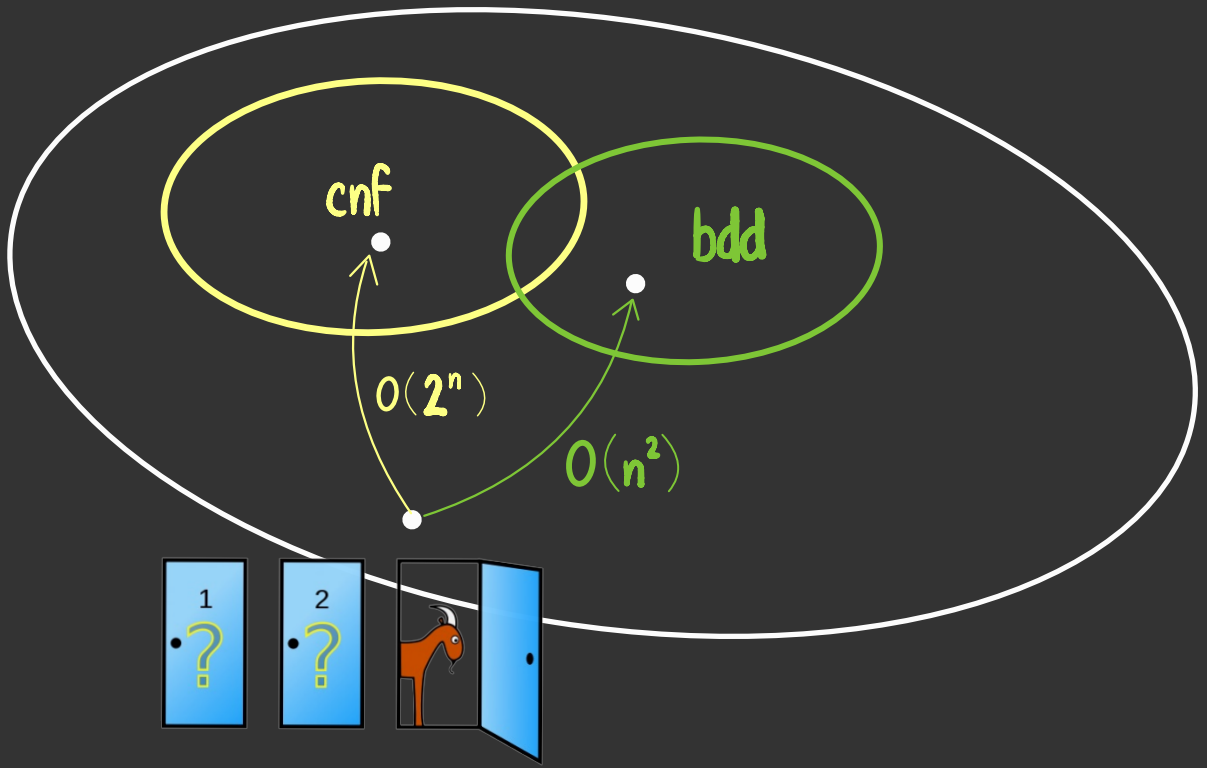
language space $L(\Sigma)$

1. how succinct is the representation? $O(n^2)$
2. what can we do with the representation? satisfiability, entailment (via dpll)
   (caveat: worst-case exponential time)



language space $L(\Sigma)$

1. how succinct is the representation? $O(2^n)$
2. what can we do with the representation? satisfiability, entailment (via dpll)
   (caveat: worst-case exponential time)

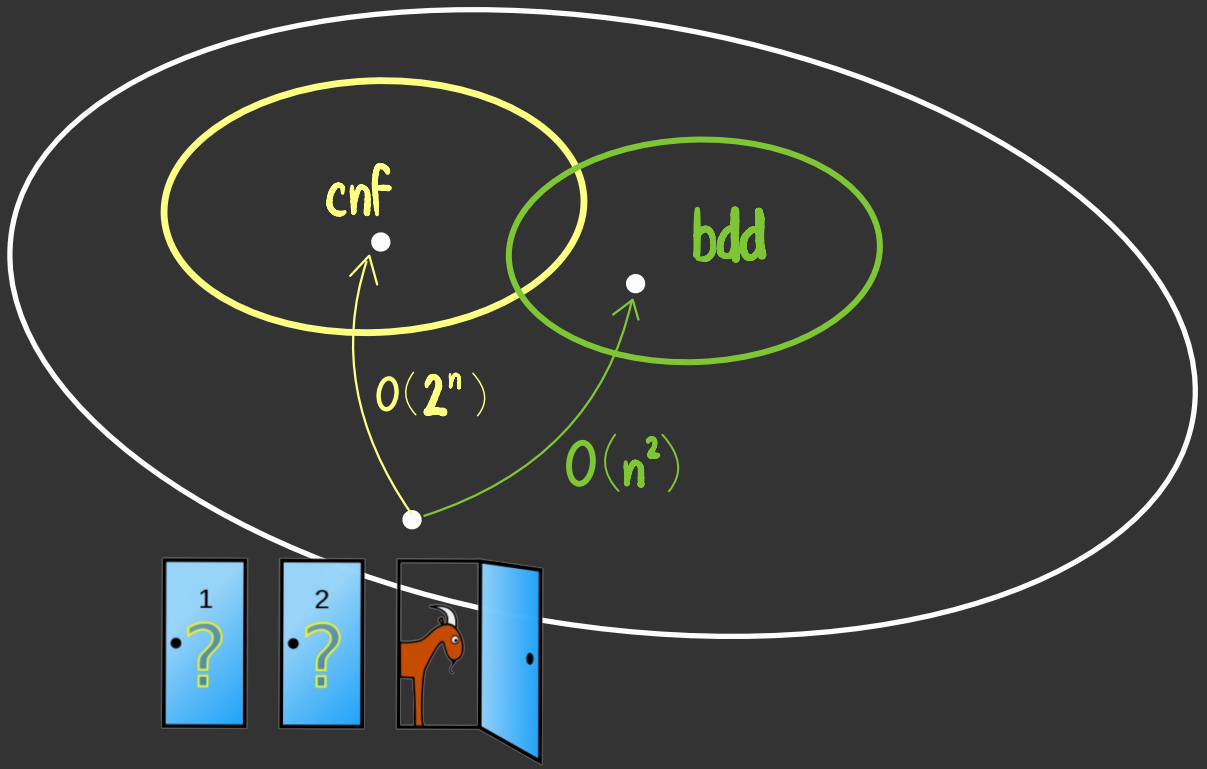


language

space

$L(\Sigma)$

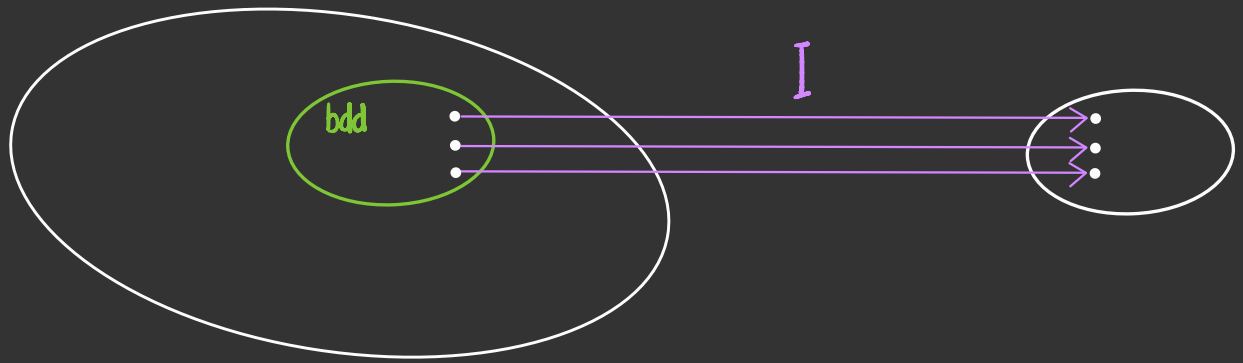
1. how succinct is the representation? $O(n^2)$
2. what can we do with the representation? satisfiability, entailment, equivalence (all polynomial time!!!)



language space $L(\Sigma)$

1. how succinct is the representation? $O(n^2)$
2. what can we do with the representation? satisfiability, entailment, equivalence (all polynomial time!!!)

how?



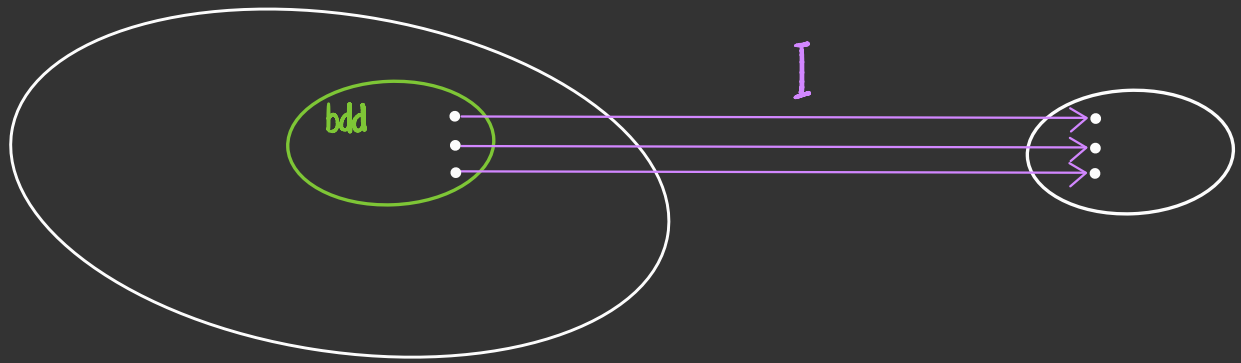
language space

$L(\Sigma)$

semantic space

$L(\Sigma)$

the interpretation function from bdds to truth tables is a bijection



language space

$L(\Sigma)$

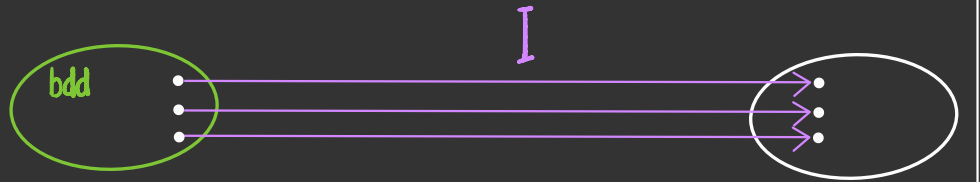
semantic space

$L(\Sigma)$

this means that there a unique bdd corresponding to each truth table

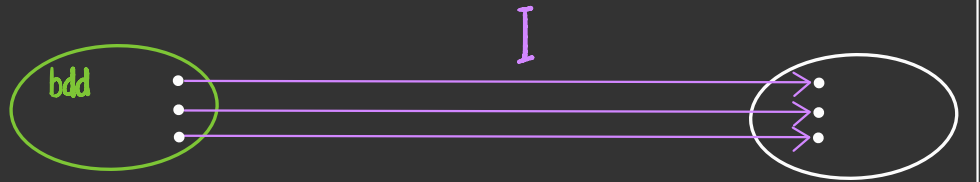
"the representation is canonical"

there a unique bdd corresponding to each truth table

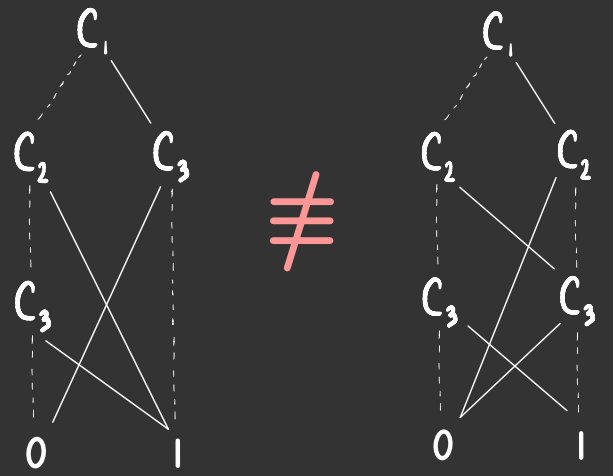


why does this give us a polynomial-time test for logical equivalence?

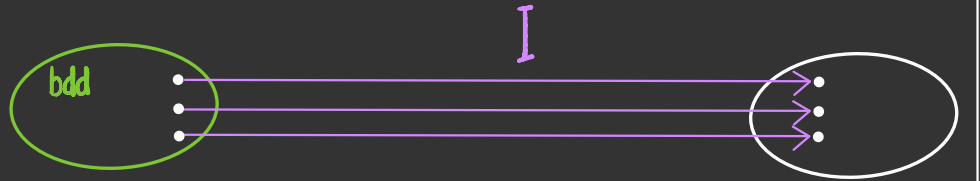
there a unique bdd corresponding to each truth table



just check that the bdds are identical

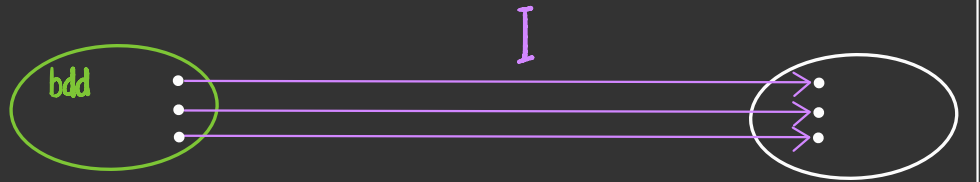


there a unique bdd corresponding to each truth table



why does this give us a polynomial-time test for satisfiability?

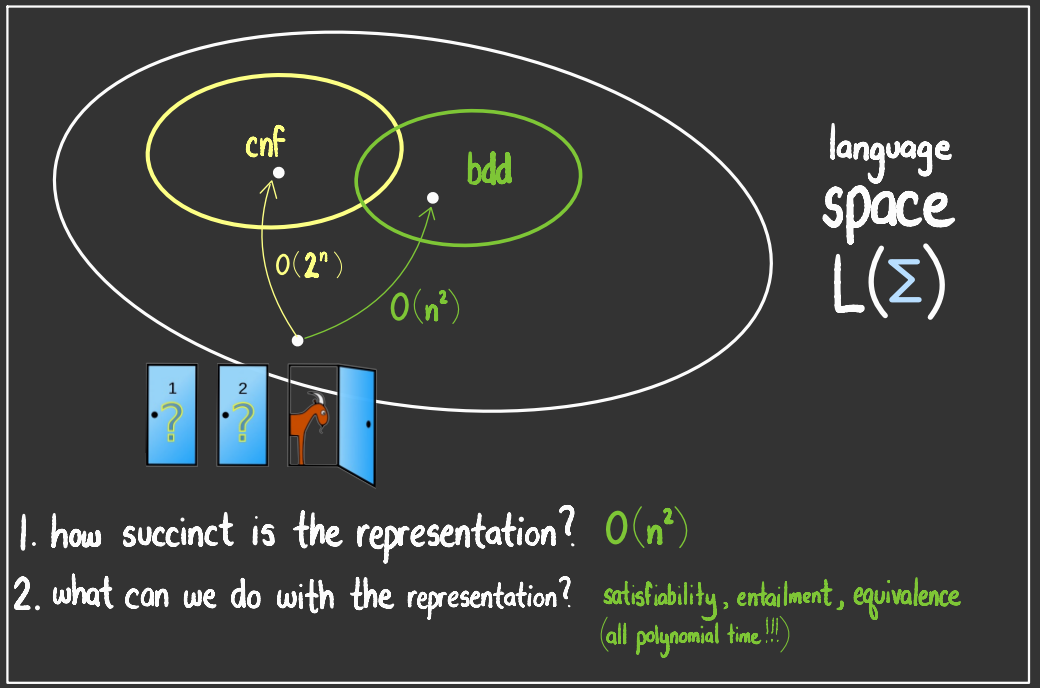
there a unique bdd corresponding to each truth table



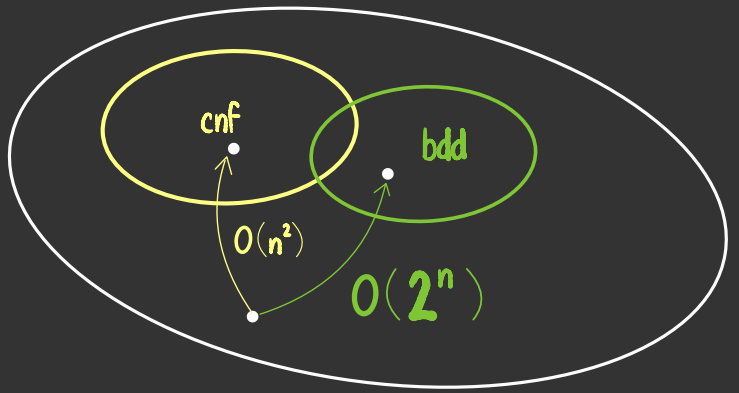
this is the only unsatisfiable bdd

0

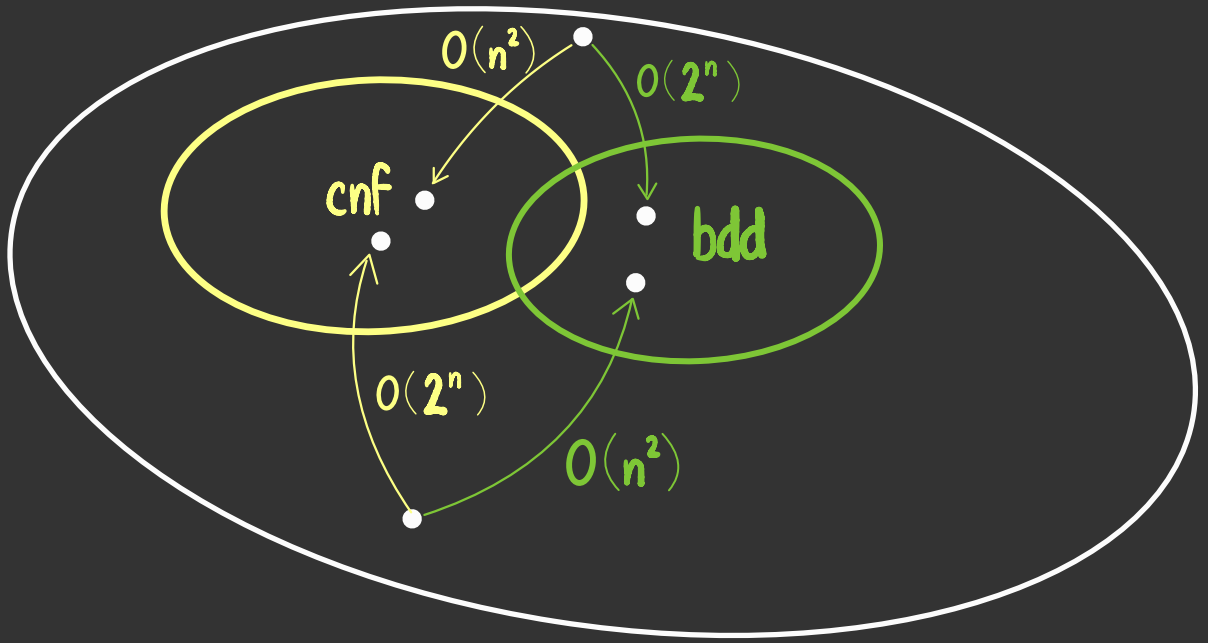
if bdds are so great, why don't we always use them?



this can also occur



1. how succinct is the representation? $O(2^n)$
2. what can we do with the representation? satisfiability, entailment, equivalence (all polynomial time!!!)



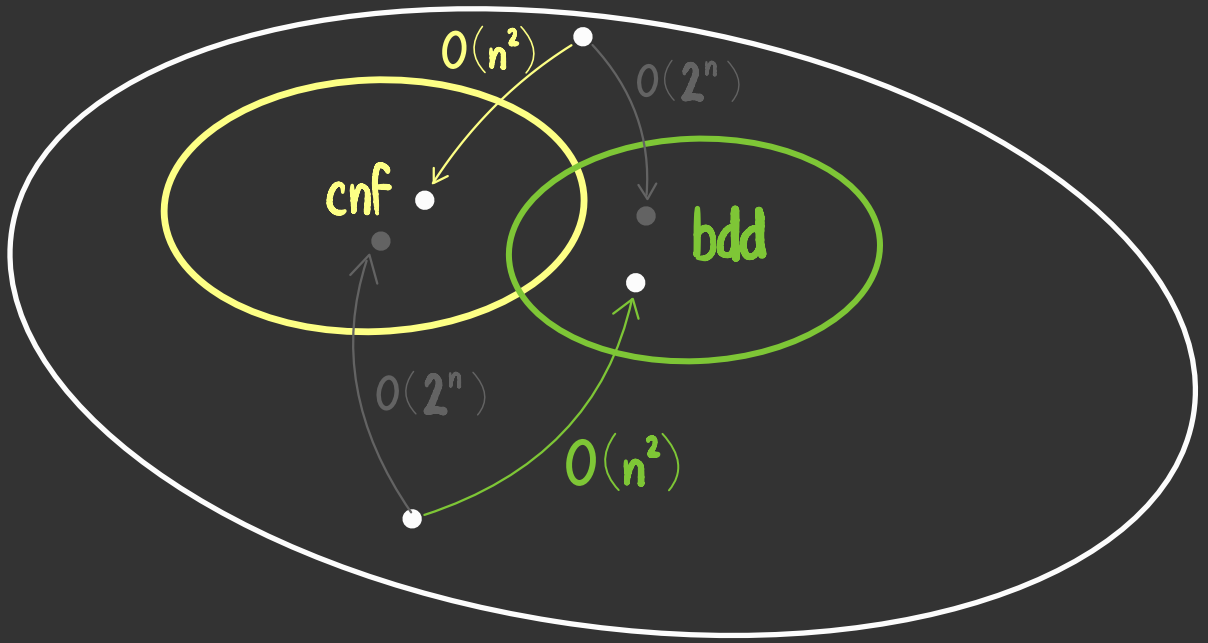
language space $L(\Sigma)$

for certain truth tables:

- the bdd is very large
- there is a reasonably sized cnf

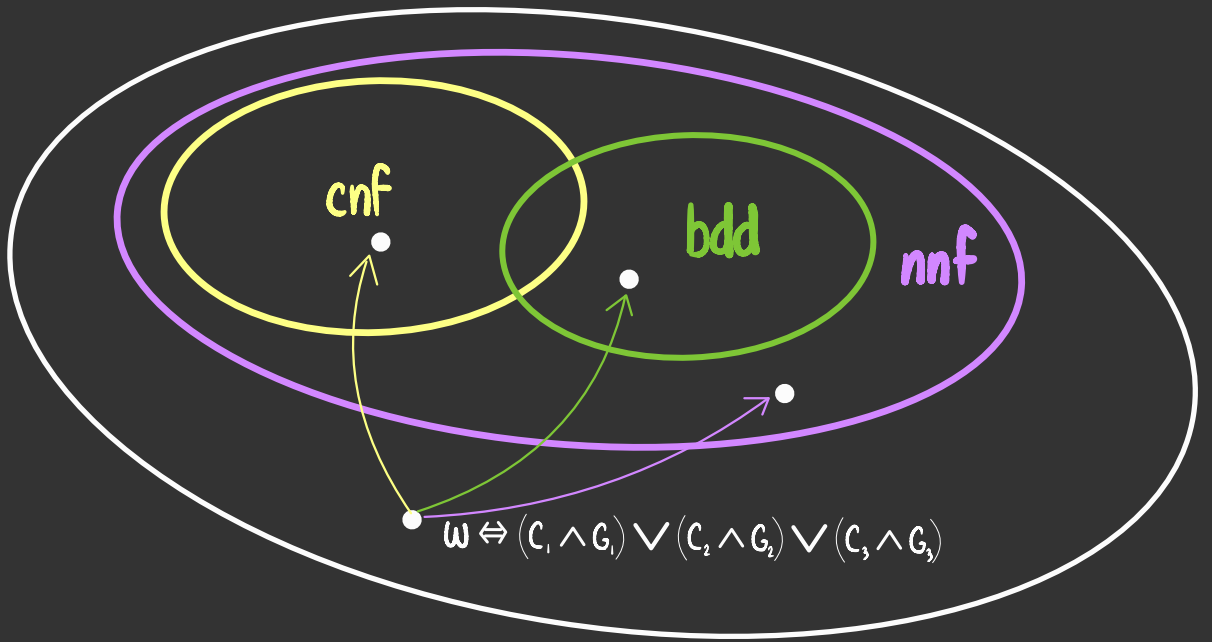
for certain truth tables:

- the bdd is reasonably sized
- all cnfs are very large



language space $L(\Sigma)$

knowledge compilation is the task of determining a good representation for a logical sentence



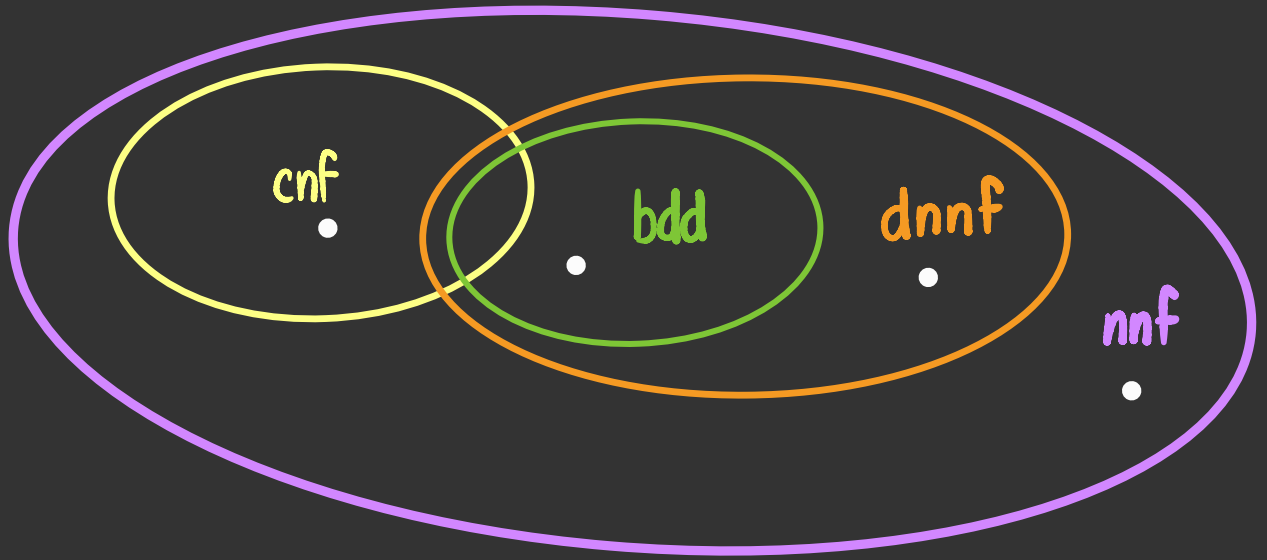
language space $L(\Sigma)$

two criteria

1. how succinct is the representation?
2. what can we do with the representation?

always at least as good as bdd or cnf

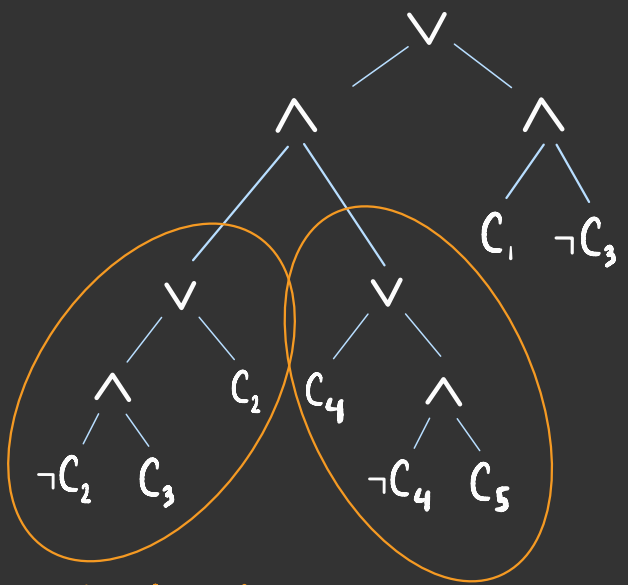
not much



two criteria

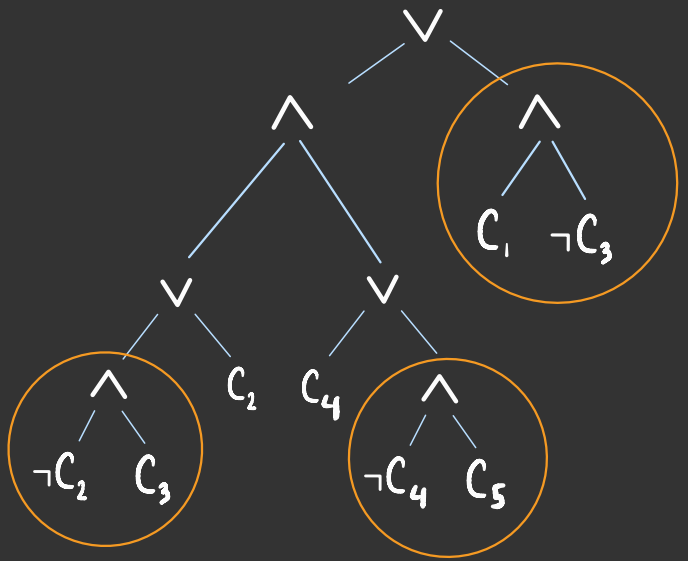
1. how succinct is the representation? always at least as good as bdd
2. what can we do with the representation? some stuff

an nnf sentence is **decomposable**
if the child subgraphs of $\wedge$ nodes
don't share signature variables



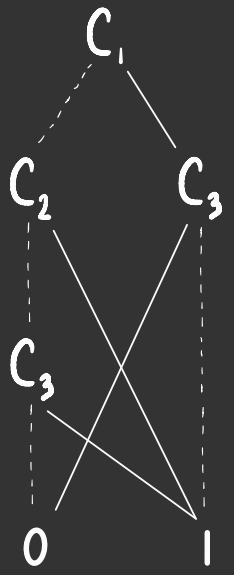
don't share symbols

an nnf sentence is **decomposable**
if the child subgraphs of $\wedge$ nodes
don't share signature variables

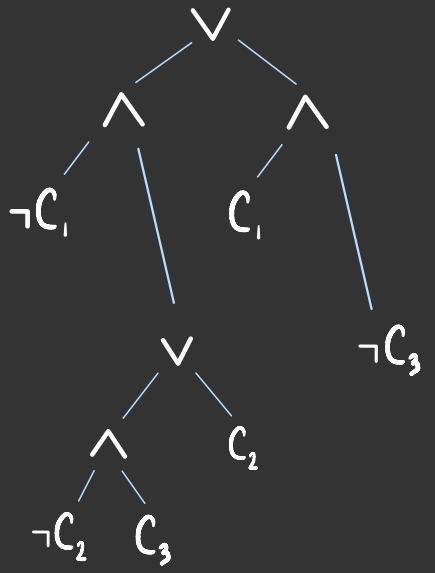


don't share symbols

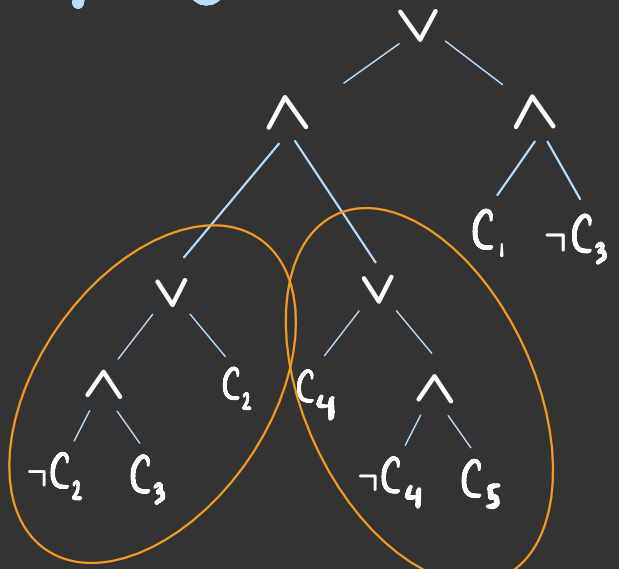
bdds are decomposable



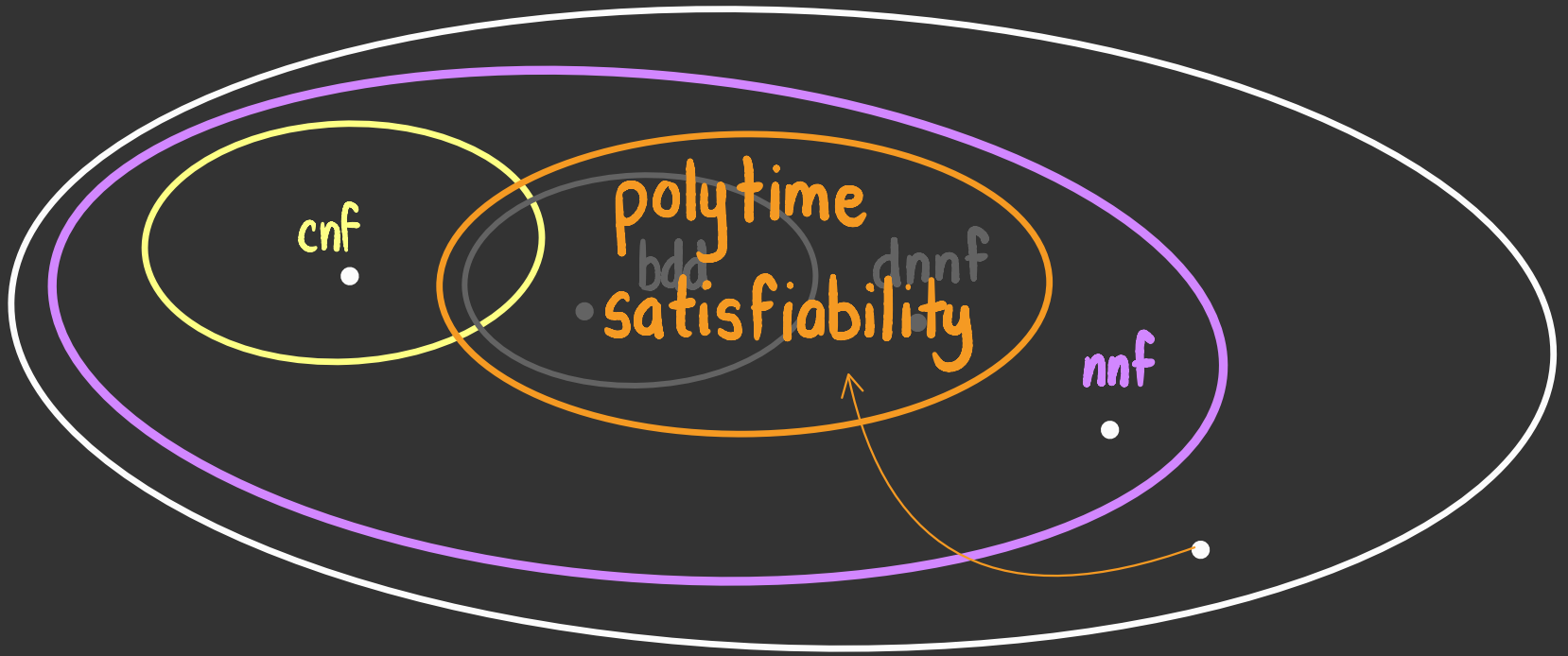
is shorthand for



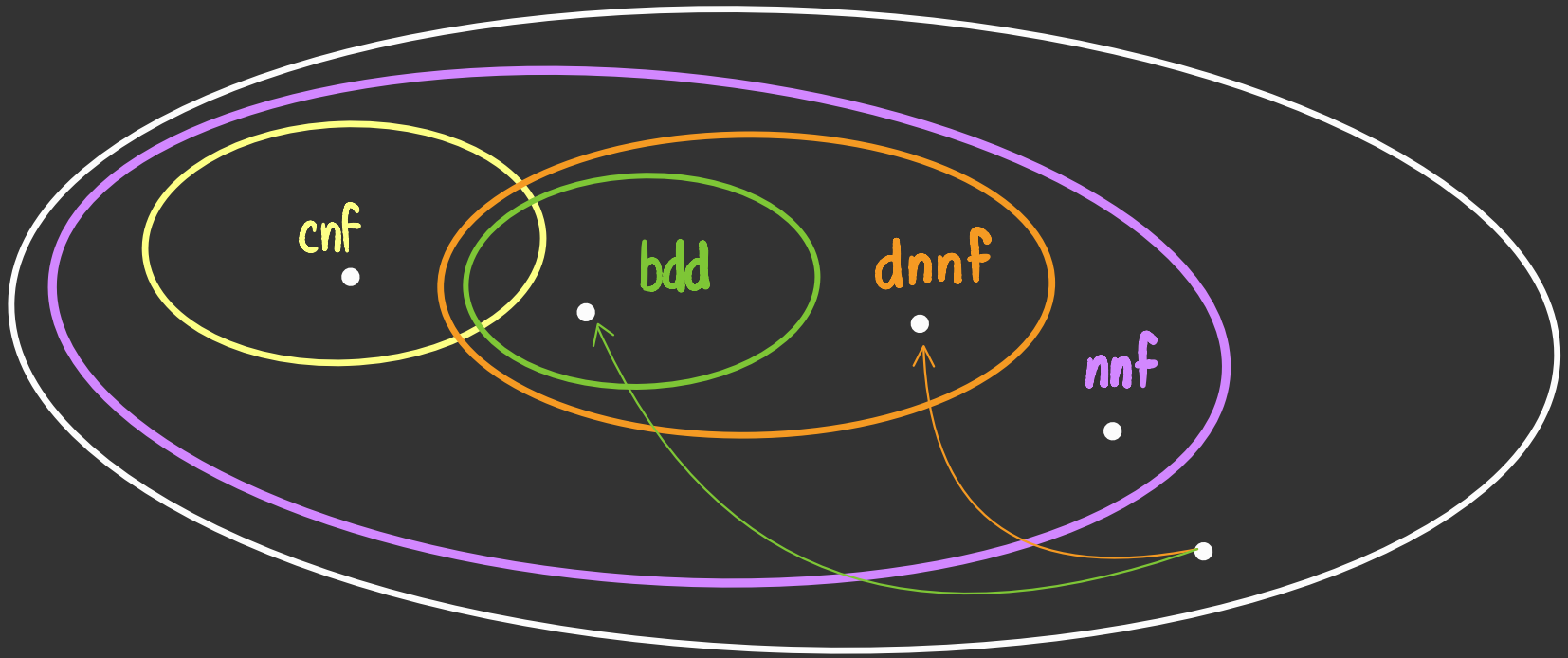
for a decomposable nnf sentence,
we can check satisfiability in
polynomial time



don't share symbols

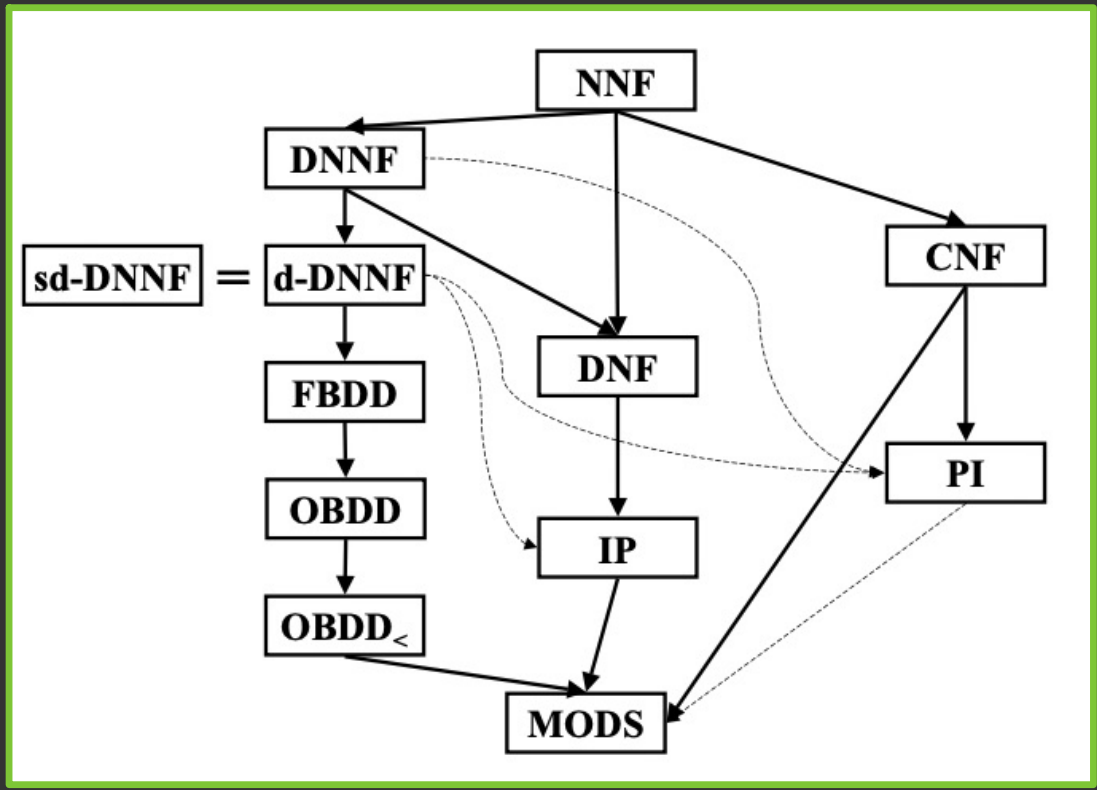


so if we're only interested in checking the satisfiability of a sentence, it makes more sense to compile into dnnf than bdd



because there might be a small dnnf
representation, even if there is no small bdd

1. how succinct is the representation?



darwiche and marquis 2001

2. what can we do with the representation?

L	CO	VA	CE	IM	EQ	SE	CT	ME
NNF	○	○	○	○	○	○	○	○
DNNF	√	○	√	○	○	○	○	√
d-NNF	○	○	○	○	○	○	○	○
s-NNF	○	○	○	○	○	○	○	○
f-NNF	○	○	○	○	○	○	○	○
d-DNNF	√	√	√	√	?	○	√	√
sd-DNNF	√	√	√	√	?	○	√	√
BDD	○	○	○	○	○	○	○	○
FBDD	√	√	√	√	?	○	√	√
OBDD	√	√	√	√	√	○	√	√
$\text{OBDD}_<$	√	√	√	√	√	√	√	√
DNF	√	○	√	○	○	○	○	√
CNF	○	√	○	√	○	○	○	○
PI	√	√	√	√	√	√	○	√
IP	√	√	√	√	√	√	○	√
MODS	√	√	√	√	√	√	√	√

darwiche and marquis 2001

2. what can we do with the representation?

L	CO	VA	CE	IM	EQ	SE	CT	ME
NNF	○	○	○	○	○	○	○	○
DNNF	√	○	√	○	○	○	○	√
d-NNF	○	○	○	○	○	○	○	○
s-NNF	○	○	○	○	○	○	○	○
f-NNF	○	○	○	○	○	○	○	○
d-DNNF	√	√	√	√	?	○	√	√
sd-DNNF	√	√	√	√	?	○	√	√
BDD	○	○	○	○	○	○	○	○
FBDD	√	√	√	√	?	○	√	√
OBDD	√	√	√	√	√	○	√	√
$\text{OBDD}_<$	√	√	√	√	√	√	√	√
DNF	√	○	√	○	○	○	○	√
CNF	○	√	○	√	○	○	○	○
PI	√	√	√	√	√	√	○	√
IP	√	√	√	√	√	√	○	√
MODS	√	√	√	√	√	√	√	√

darwiche and marquis 2001

this says we can do satisfiability but not entailment in polytime with dnnf

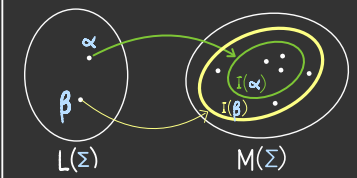
but didn't we say if we could check satisfiability, we could check entailment?

$\alpha \models \beta$

if and only if

$\alpha \wedge \neg\beta$ is unsatisfiable

entailment

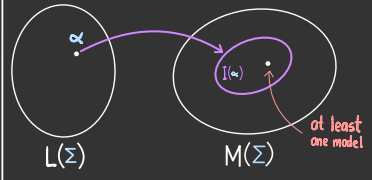


$\alpha \models \beta$

if and only if

$I(\alpha) \subseteq I(\beta)$

satisfiability



α is satisfiable

if and only if

$I(\alpha) \neq \{\}$

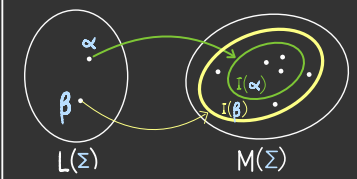
why does a polytime satisfiability check for dnnf not give us a polytime entailment algorithm?

$\alpha \models \beta$

if and only if

$\alpha \wedge \neg\beta$ is unsatisfiable

entailment

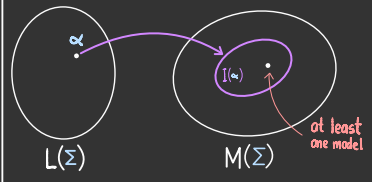


$\alpha \models \beta$

if and only if

$I(\alpha) \subseteq I(\beta)$

satisfiability

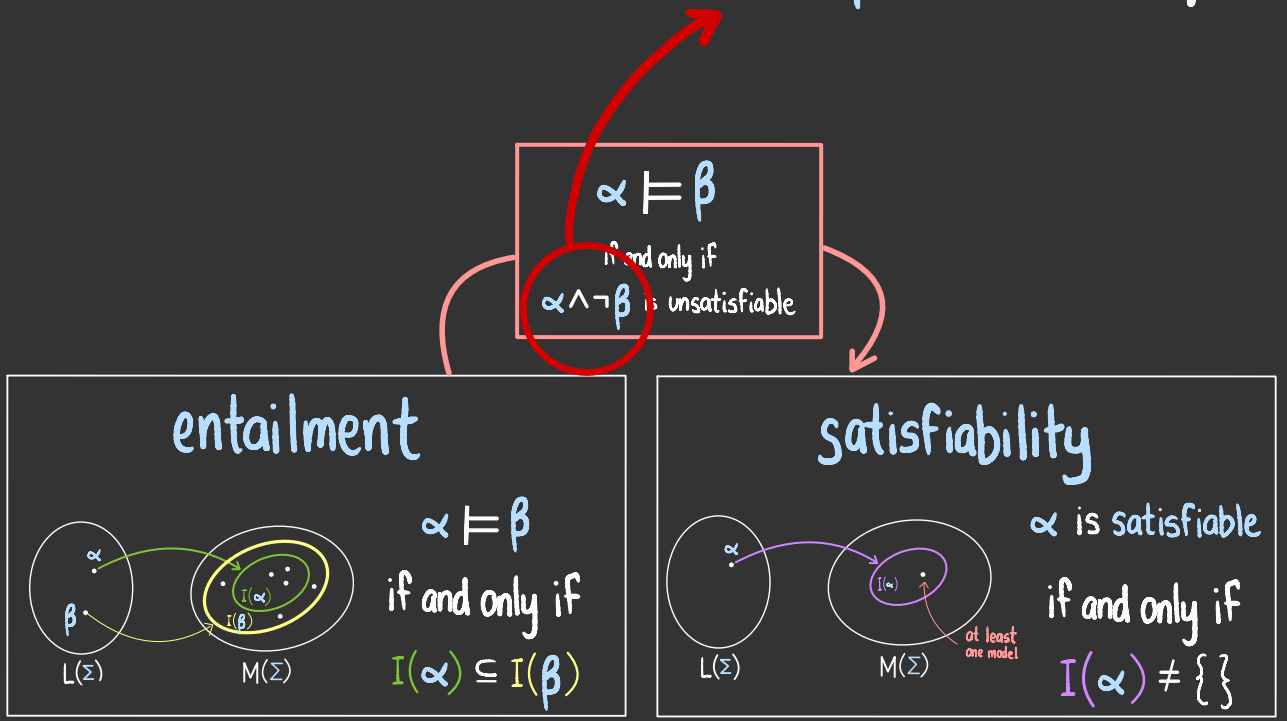


α is satisfiable

if and only if

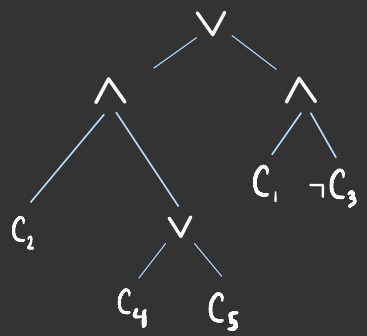
$I(\alpha) \neq \{\}$

even if α is decomposable and β is decomposable,
it doesn't mean that $\alpha \wedge \neg\beta$ is decomposable



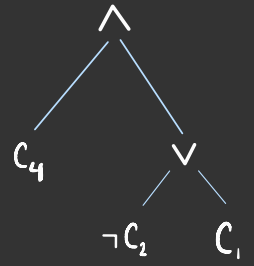
even if α is decomposable and β is decomposable,
it doesn't mean that $\alpha \wedge \neg\beta$ is decomposable

α



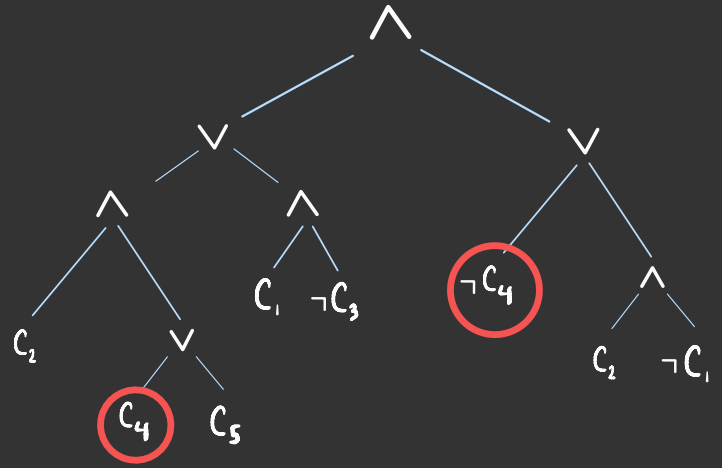
decomposable

β

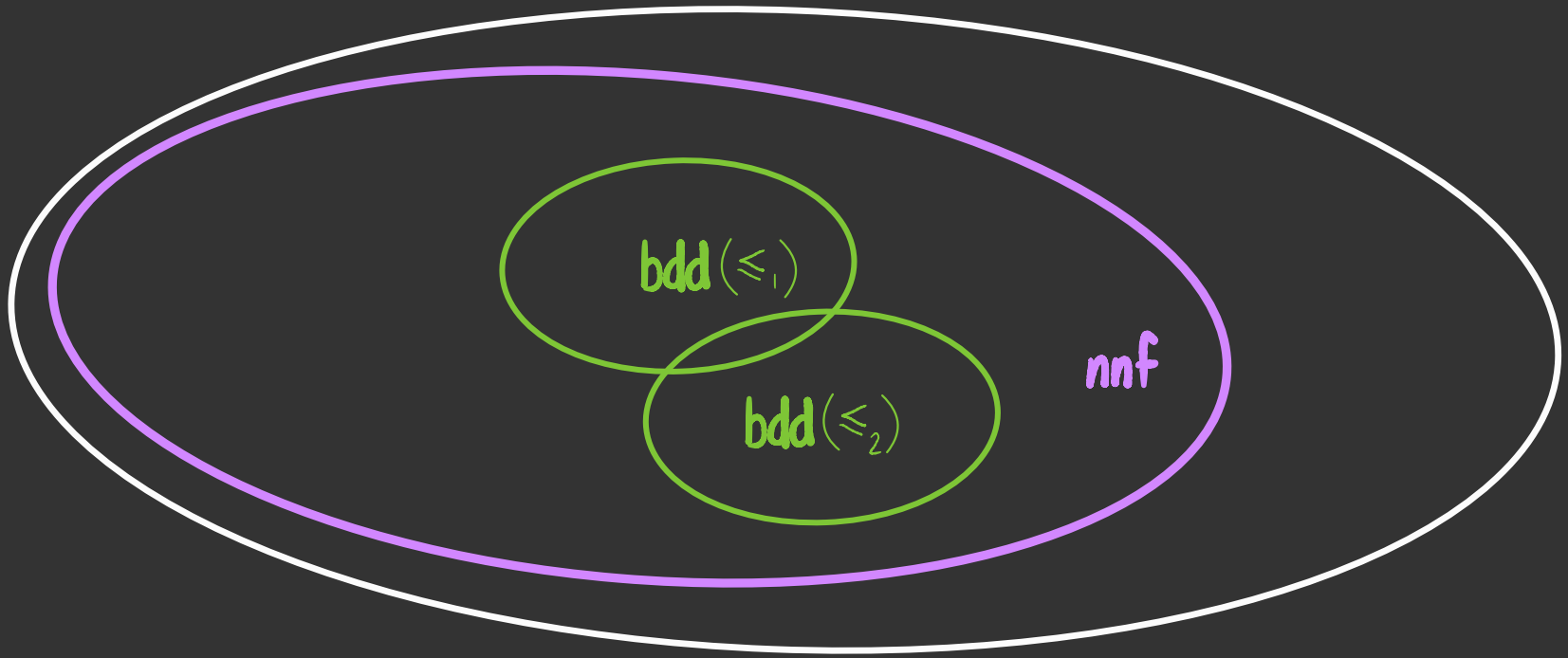


decomposable

$\alpha \wedge \neg\beta$



not decomposable



note: bdds are only canonical
when the signature ordering is fixed

represent all independent sets with a bdd