Satisfiability

16-Oct-2022 at 7:54 PM

A propositional sentence is in CNF

$$\alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_n$$

α_i is a clause

$$(A \vee B \vee \neg C) \wedge (\neg A \vee D) \wedge (A \vee \neg B \vee D)$$

$P \wedge \neg P \rightarrow$ Not valid

$\perp \rightarrow \{\} \rightarrow$ not valid

No clauses in CNF $\rightarrow$ valid

Set representation:

$$l_1 \vee l_2 \vee \cdots \vee l_m \rightarrow \{l_1, l_2, \cdots l_m\}$$

$$\alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_n \rightarrow \{\alpha_1, \alpha_2, \cdots, \alpha_n\}$$

$$\{\{A, B, \neg C\}, \{\neg A, D\}, \{A, \neg B, D\}\}$$

1) A CNF Δ is valid if $\Delta = \phi$

2) A CNF Δ is inconsistent if $\phi \in \Delta$

$\Delta \models \Gamma$ if every assignment that satisfies Δ also satisfies Γ.

C_i subsumes C_j

if $C_i \subseteq C_j$ then $C_i \models C_j$

$\Rightarrow$ there is no need to have clause C_j if C_i is present in a formula

if $\Delta \models \{\ldots, \{\Phi\}, \ldots\}$ — unsatisfiable

$\Delta \subseteq \{\ldots, \{\Phi\}, \ldots\}$

Resolution

→ Simplest complete algorithm for testing satisfiability.

Δ is in CNF and has two clauses C_i and C_j ⇒ Resolved clauses $P \in C_i$ and $\neg P \in C_j$

⇒ a derived clause

$$(C_i - \{P\}) \cup (C_j - \{\neg P\})$$

↓

Resolvent

$\{A, \neg B, C\}$ $\{B, D\}$

$\{A, C, D\}$ → B-resolvent

↓

If we can derive an empty clause ($\{\}$)

⇒ unsatisfiable.

Resolution is Sound but <u>not Complete</u>

↓

does not guarantee to derive every clause that is implied by Δ

⇒ Resolution is refutation complete

↳ guaranteed to derive empty clause if given σ is unsatisfiable.

↳ basis for using resolution as complete algo for testing satisfiability.

↳ Keep on applying resolution rule until

* Empty clause is derived (UNSAT)

* No more rule application is possible (SAT)

Δ = | 1. $\{\neg P, R\}$
| 2. $\{\neg Q, R\}$
| 3. $\{\neg R\}$
| 4. $\{P, Q\}$

---

5. $\{\neg P\}$ (1,3)
6. $\{\neg Q\}$ (2,3)
7. $\{P\}$ (4,6)
8. $\{\}$ (4,7)

↳ unsatisfiable

Unit Resolution:

↳ at least one of the resolved clauses has only one literal.

↳ unit clause

unit resolution is not refutation complete

↳ may not derive an empty clause from an unsatisfiable Δ.

↳ Application of unit resolution is time linear in the size of Δ

Key technique used by number of algorithms.

Conditioning:

Conditioning Δ on literal L

↳ Replacing every occurrence of L by TRUE

↳ replacing every occurrence of $\neg L$ by FALSE.

Δ has three types of clauses:

1) Set of clauses α containing L. Setting L to TRUE, α will become true
$\Rightarrow$ Truth of Δ then does not depend on α.
α should be removed from $\Delta|L$

2) Set of α containing $\neg L$. Setting $\neg L$ = FALSE, the truth value of α will depend on other literals
So, we can remove $\neg L$ from α.

3) Set of α that neither have L nor $\neg L$. α will appear in $\Delta|L$

$$\Delta|L = \begin{cases} \{\alpha \mid \alpha \in \Delta, L \notin \alpha \text{ or } \neg L \notin \alpha\} \\ \{\alpha - \{\neg L\} \mid \alpha \in \Delta, \neg L \in \alpha\} \\ \{\{\Delta - \alpha\} \mid \alpha \in \Delta, L \in \alpha\} \end{cases}$$

$$\Delta = \{\{A, B, \neg C\}, \{\neg A, D\}, \{B, C, D\}\}$$

$$\Delta|_C = \{\{A, B\}, \{\neg A, D\}\}$$

$$\Delta|_{\neg C} = \{\{\neg A, D\}, \{B, D\}\}$$

$$\Delta|_{C,A} = \{\{D\}\}$$

$$\Delta|_{C,A,\neg D} = \{\Phi\} \rightarrow \text{inconsistent}$$

$$\Delta|_{C,A,D} = \Phi \rightarrow \text{satisfiable}$$

$$\Delta|_{\neg C,D} = \Phi \rightarrow \text{satisfiable}$$

SAT by existential quantification

$$\exists P \Delta \overset{def}{=} (\Delta|_P) \vee (\Delta|_{\neg P})$$

$$\Delta = \{\{\neg A, B\}, \{\neg B, C\}\}$$

$A \Rightarrow B \qquad B \Rightarrow C$

$A \Rightarrow C$

$$\Delta|_B = \{\{C\}\}, \quad \Delta|_{\neg B} = \{\{\neg A\}\}$$

$$\exists B \Delta = \neg A \vee C \qquad A \Rightarrow C$$

Satisfiability of Φ does not depend on P

Δ is satisfiable if $\exists P \Delta$ is satisfiable

↳ Δ involve larger set of variables than FPS

↳ Hence, we are replacing a satisfiability problem of larger size to one with a smaller size.

Algo:

Keep on conditioning Δ with variables one by one until we are left with a trivial Δ with no variables:

Trivial Δ:

1) $\Delta = \phi$ [SAT]

or

2) $\phi \in \Delta$ [UNSAT]

Davis-Putnam (DP) Algo

CNF Δ,

CNF Γ obtained by adding all P-resolvents to Δ and then ignoring out all clauses that mention P

$$\Delta = \{\{\neg A, B\}, \{\neg B, C\}\}$$

B-resolvent $\rightarrow \{\neg A, C\}$

$$\Gamma = \{\{\neg A, C\}\} \equiv \exists B \Delta$$

DP algo → Directional Resolution
→ Bucket elimination.

1) Bucket construction - Entry for each var.
2) Decide an order π and place the variables accordingly

3) Each clause α∈Δ is placed in the first bucket P from the top s.t. P appears in α.
4) Apply P-resolution in each bucket (top to bottom)

$\Delta = \{\{\neg A, B\}, \{\neg A, C\}, \{\neg B, D\}, \{\neg C, \neg D\}, \{A, \neg C, E\}\}$

C	$\{\neg A, C\}, \{\neg C, \neg D\}, \{A, \neg C, E\}$	∃C
B	$\{\neg B, D\}, \{\neg A, B\}$	∃B
A	$\{\neg A, \neg D\}, \{\neg A, D\}$	∃A
D		
E		

$\exists A \Delta \equiv \Delta|_A \vee \Delta|_{\neg A}$

$= [\{\neg D\}, \{D\}] \vee \phi$

$= \{\} \vee \phi = \phi \Rightarrow SAT$

ordering 2 : E, A, B, C, D

E	$\{A, \neg C, E\}$
A	$\{\neg A, B\}, \{\neg A, C\}$
B	$\{\neg B, D\}$
C	$\{\neg C, \neg D\}$
D	

No resolvant

SAT

Extracting Satisfying Assignments:

- Process the variables in reverse order

- Let the order be

$$\pi = v_1, v_2, \cdots, v_n$$

- if v_i is empty $\Rightarrow$ assign any value

  else assign value to v_i such that together with v_j $i<j<n$ satisfies all the clauses in its bucket.

C: $\{\neg A, C\}, \{\neg C, \neg D\}, \{A, \neg C, E\}$

B: $\{A, B\}, \{\neg B, D\}$

A:

D:

E:

$\{\neg A, \neg D\}, \{\neg A, D\}$

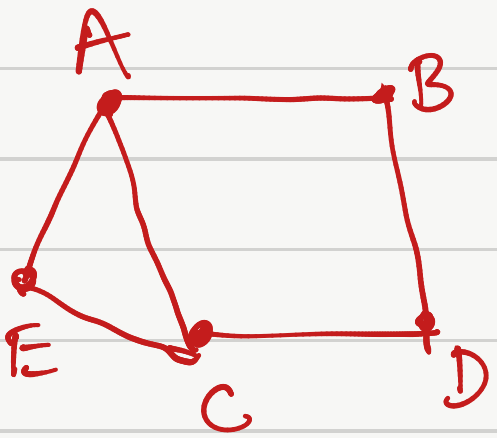
$\pi = C, B, A, D, E$

E (empty) → True D (empty) → False

A → False, B → False, C → True/False

Time and Space Complexity:

Connectivity Graph: undirected graph G over variables in Δ,
→ an edge exists between two variables if they are in the same clause.



$w = 2$

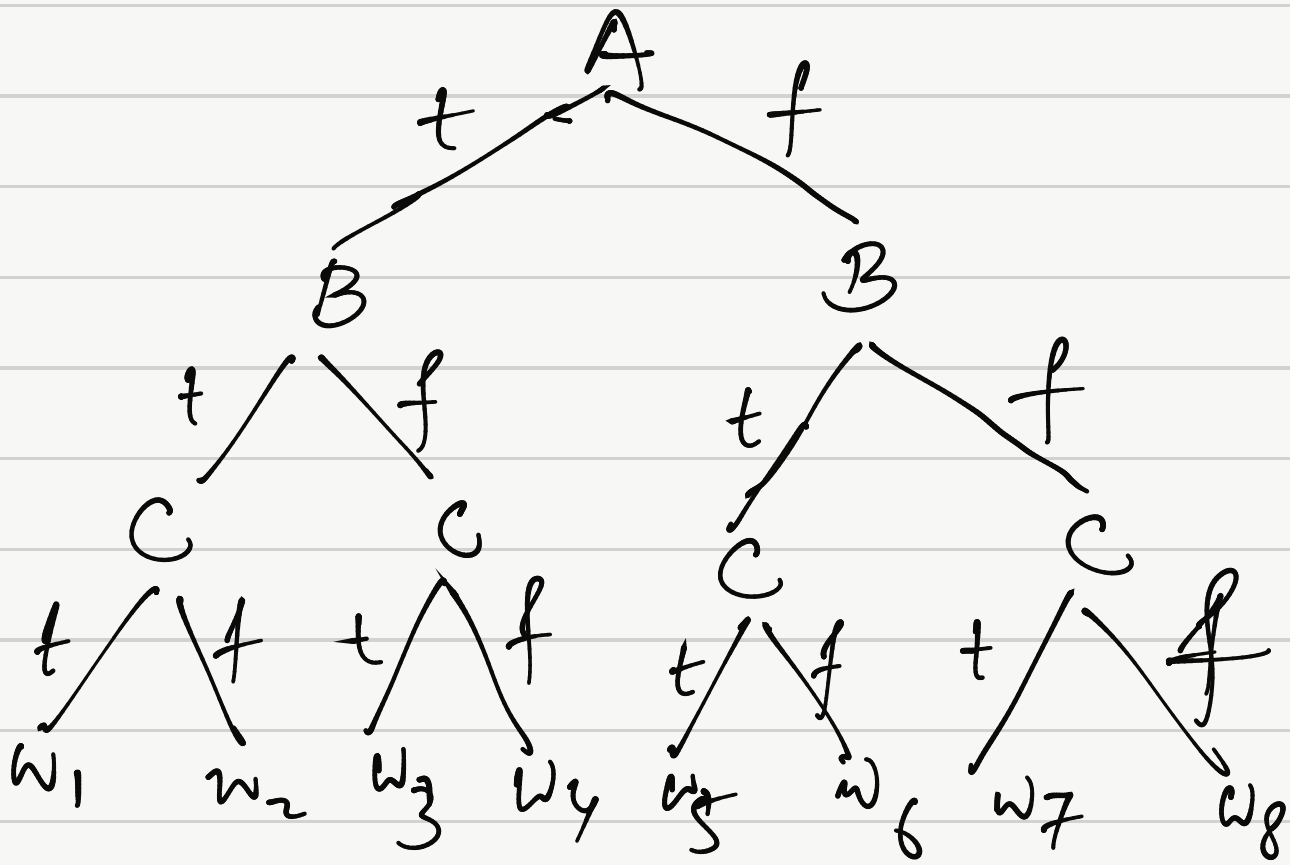
Tree width (w) of the connectivity graph
→ How far the graph is from a tree?

Time and space complexity = $O(n \exp(w))$

SAT by Search:

→ DP algorithm is not space efficient.

Search in the space of truth assignment



Each leaf is a complete truth assignment

Satisfiability ⇒ Process of finding a leaf node that satisfies the given CNF

DFS Search bounded by depth
n ⇒ n number of variables.

Take left from a node (P) in the assignment tree
$\Delta \mid P$

Take right $\Delta \mid \neg P$

As we proceed through depth, we condition over multiple variables.

Example:

$$\Delta = \{\{\neg A, B\}, \{\neg B, C\}\}$$

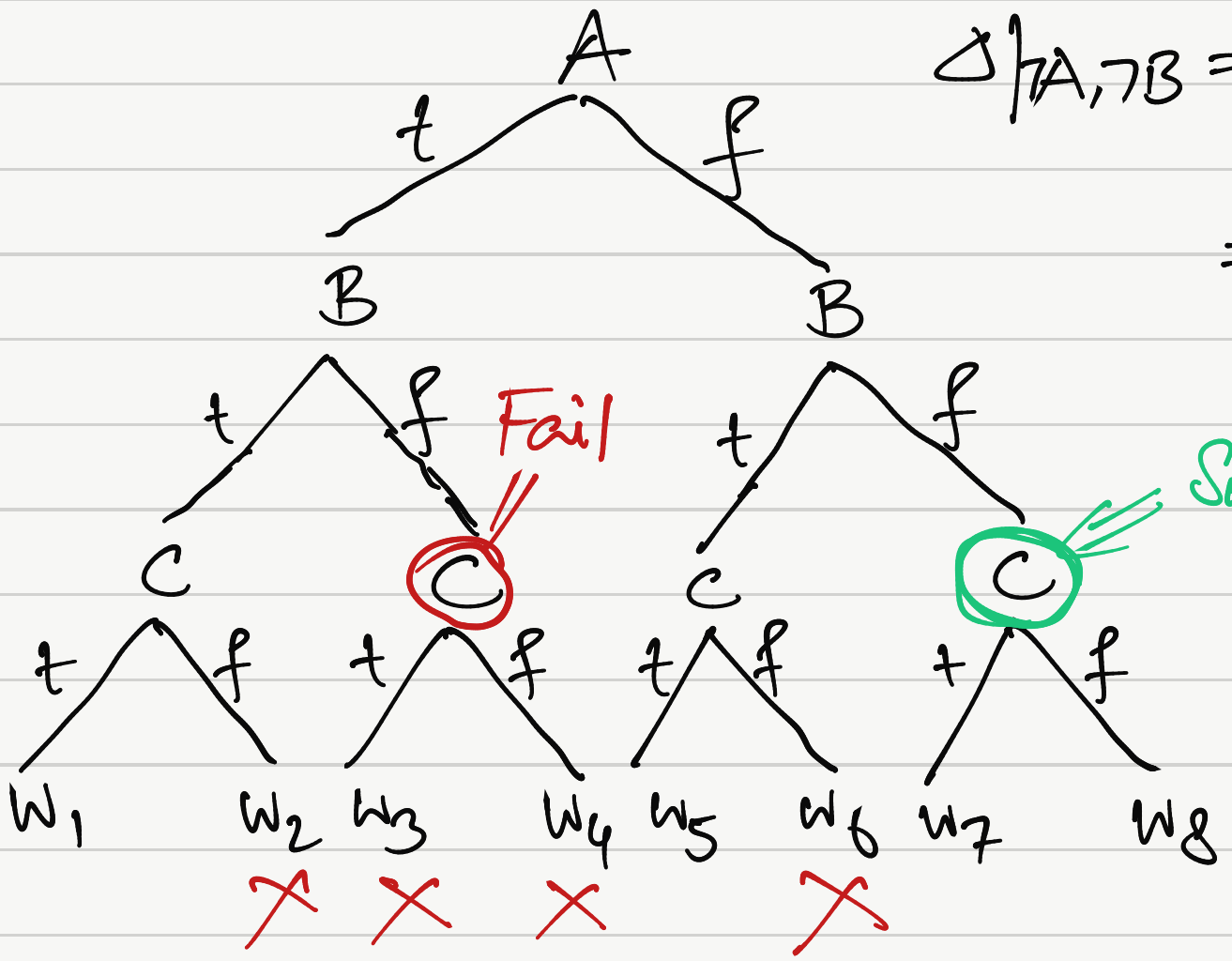
$$\Delta|A = \{\{false, B\}, \{\neg B, C\}\}$$

$$\Delta|A, \neg B = \{\{false, false\}, \{true, C\}\}$$

$$\Delta|A, \neg B, C = \{\{\}, true\} = \{\{\}\}$$

$$\Delta|A, \neg B, \neg C = \{\{\}, true\} = \{\{\}\}$$

w_3 and w_4 are unsatisfying assignments.



$$\Delta|\neg A, \neg B = \{\{true, false\}, \{true, C\}\} = \{\}$$

DPLL – (CNF Δ, depth d)
returns a set of literals or UNSAT

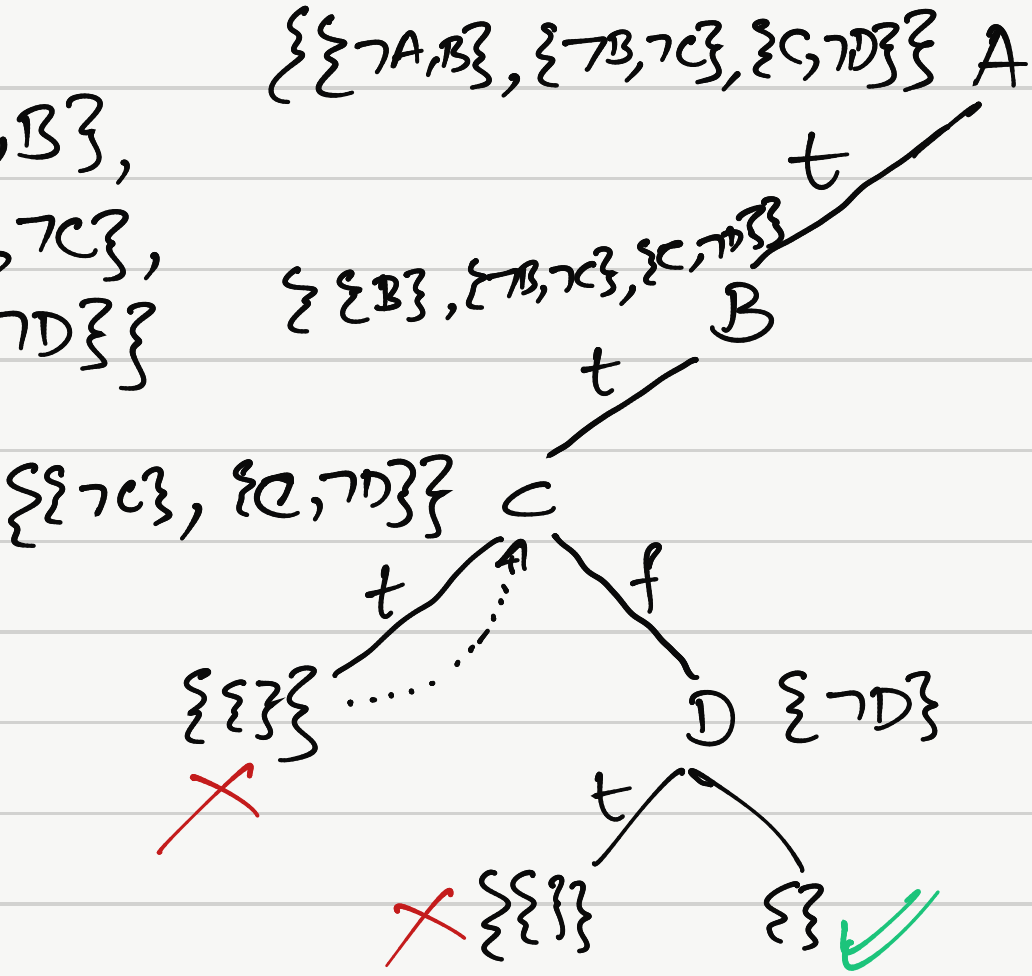
if Δ = {} then
    return {}
elif {} ∈ Δ then
    return UNSAT
elif L = DPLL–(Δ|P_{d+1}, d+1) ≠ UNSAT then
    return L ∪ {P_{d+1}}

elif L = DPLL–(Δ|$\neg P_{d+1}$, d+1) ≠ UNSAT then
    return L ∪ {$\neg P_{d+1}$}

else
    return UNSAT.

Amount of work done by DPLL –
↳ Termination Free

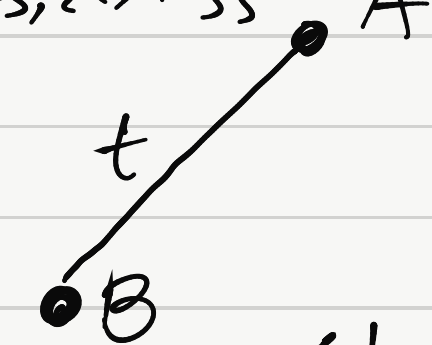
$\Delta = \{\{\neg A, B\}, \{\neg B, \neg C\}, \{C, \neg D\}\}$



Termination tree is useful if characterizing
UNSAT problems ⇒ Difficulty is related
to size and depth.

Unit Resolution:

$\{\{\neg A, B\}, \{\neg B, \neg C\}, \{C, \neg D\}\}$



$\{\{B\}, \{\neg B, \neg C\}, \{C, \neg D\}\}$

$\Delta|_A = \{\{B\}, \{\neg B, \neg C\}, \{C, \neg D\}\}$

→ neither $\{\}$ nor $\{\} \in \Delta|_A$

↳ Cannot declare success or failure.

Can we detect success or failure at this node (B)?

Unit Resolution or unit propagation:

→ Close the CNF under unit resolution and collect all unit clauses.

↳ Try to satisfy the unit clauses

For $\{P\}$ → set P = True

For $\{\neg P\}$ → set P = False

Propagate these decisions to other clauses
→ check success or failure

UNIT-RESOLUTION (Δ)
→ I : Set of literals that were either present as unit clauses in Δ or derived from Δ by unit resolution.
→ Γ : $\Delta | I$

$\Delta = \{\{\neg A, \neg B\}, \{B, C\}, \{\neg C, D\}, \{A\}\}$

$I = \{A, \neg B, C, D\}$
$\Gamma = \Delta | I = \{\}$

$\Delta = \{\{\neg A, \neg B\}, \{B, C\}, \{\neg C, D\}, \{C\}\}$

$I = \{C, D\}$
$\Gamma = \Delta | I = \{\{\neg A, \neg B\}\}$

DPLL (CNF Δ) returns a set of literals or UNSAT

$(I, \Gamma) = \text{UNIT\_RESOLUTION}(\Delta)$

```
if Γ = {} then
    return I
elif {} ∈ Γ then
    return UNSAT
else
    choose a literal x from Γ
    if L = DPLL(Γ|x) ≠ UNSAT then
        return L ∪ I ∪ {x}
    elif L = DPLL(Γ|¬x) ≠ UNSAT then
        return L ∪ I ∪ {¬x}
    else
        return UNSAT
```

SAT by Search + Inference

Non-chronological backtracing

ZChaff, MiniSAT

Incomplete SAT algo

WalkSAT.

Convert SAT problem into CSP:

Input: SAT Problem

output: Equivalent CSP

Dual Encoding:

Variables: dual variable D_i for each clause C_i

Domain: $dom(D_i) \Rightarrow$ set of tuples that satisfy C_i

Constraint: Binary constraint

$C_i \rightarrow C_j$ if they share at least one variable

$D_1 = x_1 \vee x_3 \qquad D_2 = \neg x_3 \vee x_2$

$Dom(D_1) = \{(t,f),(f,t),(t,t)$

Constraint $(D_1 - D_2)$:
second element of D_1 must be complement of first element of D_2

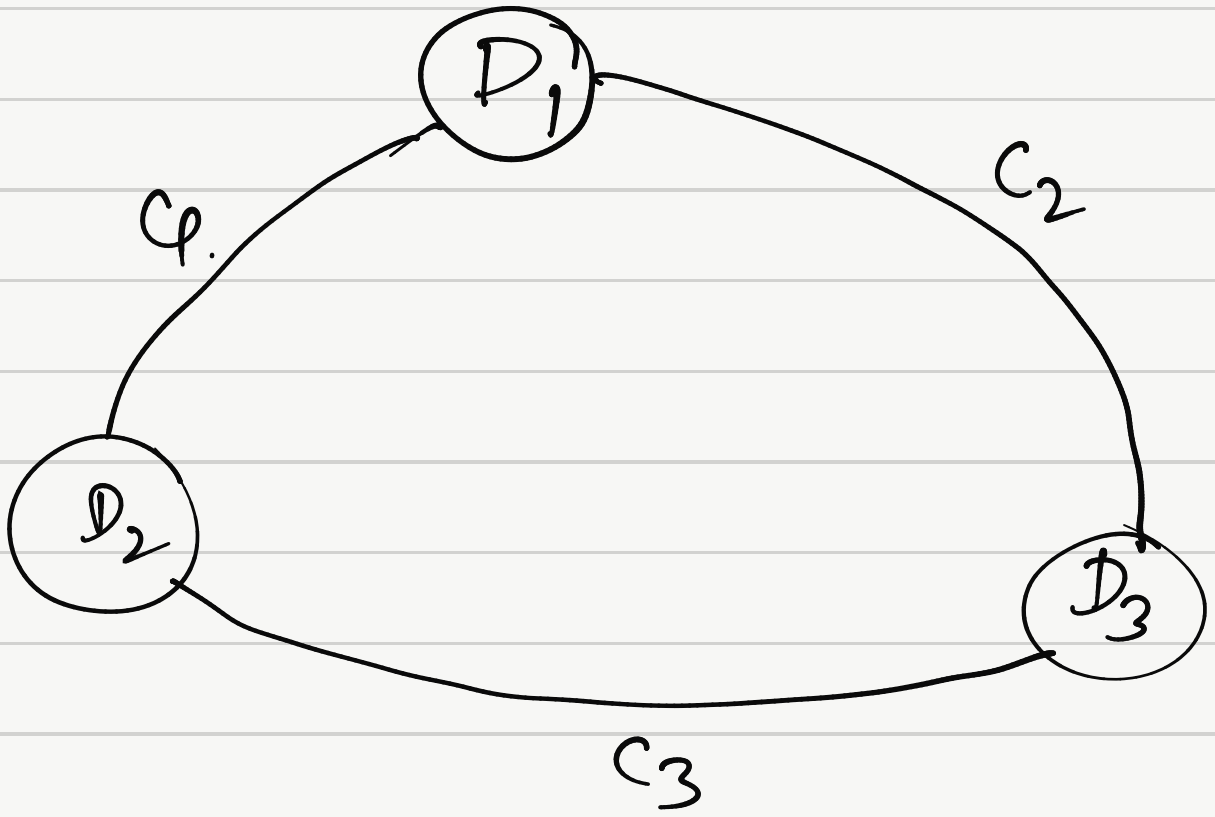
$$(x_1 \vee x_2 \vee \neg x_3) \wedge (x_2 \vee x_3) \wedge (\neg x_1 \vee x_3)$$

D_1 D_2 D_3

$$Dom(D_1) = \{\langle t, -, - \rangle, \langle -, t, - \rangle, \langle -, -, f \rangle\}$$

$$Dom(D_2) = \{(t, -), (-, t)\}$$

$$Dom(D_3) = \{(f, t), (f, f), (t, t)\}$$



C_1 : Ithird element of D_1 should be complement of 2nd element of D_2

C_2 : first element of D_1 should be complement of first element of D_3 and third of D_1 should be complement of 2nd of D_3

C_3 : 2nd element of D_2 and D_3 are same