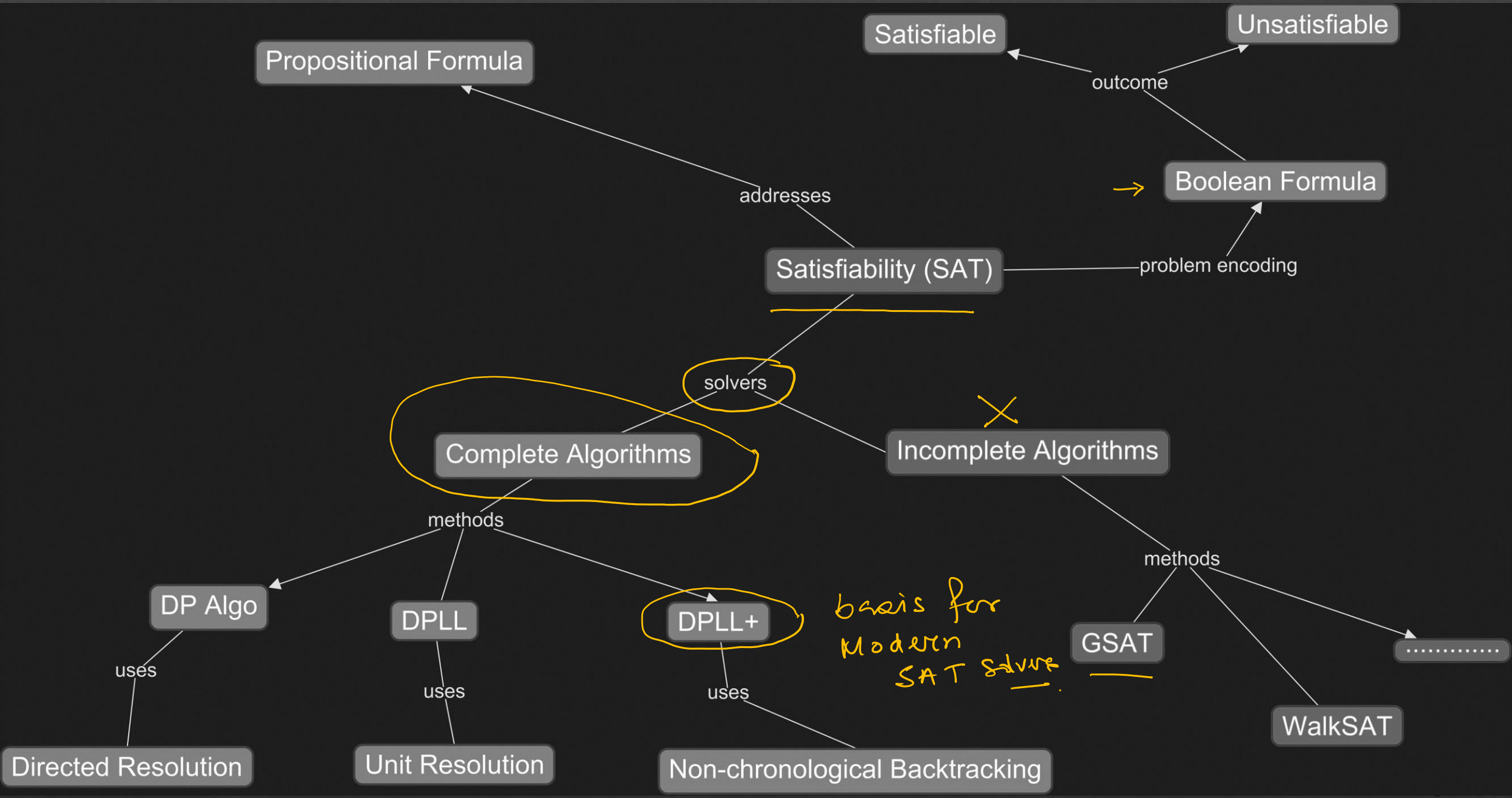
Satisfiability Problem

AIFA (AI61005)
2021 Autumn

Plaban Kumar Bhowmick


Propositional Formula
Satisfiable
Unsatisfiable
outcome
addresses
Boolean Formula
Satisfiability (SAT)
problem encoding
solvers
Complete Algorithms
Incomplete Algorithms
methods
methods
DP Algo
DPLL
DPLL+
GSAT
.....
uses
uses
uses
WalkSAT
Directed Resolution
Unit Resolution
Non-chronological Backtracking
basis for Modern SAT solvers

SAT: Why and What

Propositional Logic

- A *literal* is a Boolean variable x or its negation $\bar{x}$
- A *clause* is a disjunction ($\vee$) of literals A ∨ B
- A *CNF formula* is a conjunction ($\wedge$) of clauses

→ $F = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_2) \wedge (x_2 \vee \overline{x_3})$ CNF

$Vars(F) = \{x_1, x_2, x_3\}$

$Literals(F) = \{x_1, x_2, \overline{x_2}, x_3, \overline{x_3}\}$

$Clauses(F) = \{(x_1 \vee \overline{x_2} \vee \overline{x_3}), (x_2), (x_2 \vee \overline{x_3})\}$

Satisfiability

- Truth assignment ϕ (variables)
  - $\phi(x)$ = True or $\phi(x)$ = False
- ϕ satisfies → assignment
  - x if $\phi(x)$ = True
  - $\bar{x}$ if $\phi(x)$ = False
  - A clause if it satisfies at least one of its literals
  - A CNF formula if it satisfies all of its clauses

ϕ = {A = true or B = true or C = true}

A ∨ B ∨ C

Satisfiability

- ϕ is a *satisfying assignment* of a CNF F if it satisfies F
- F is *satisfiable* if there exists a ϕ that satisfies F
- The *Satisfiability Problem* determines whether a given F is satisfiable or not.

ϕ for which F is true

Satisfiability: Example

Satisfiable Formula

(x_1)

Unsatisfiable Formula

$(x_1) \wedge (\overline{x_1})$

$(x_3) \vee (\overline{x_2}) \vee (x_7)$

$(\overline{x_1} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1})$

$(x_2) \wedge (\overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3})$

$(x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_2})$

Satisfiability: Scheduling Meeting

- A can only meet on Monday or Wednesday
- B cannot meet on Wednesday
- C cannot meet on Friday
- D can only meet on Thursday or Friday

→ Boolean Formula

SAT problem instance

boolean

$$F = (x_1 \vee x_3) \wedge (\overline{x_3}) \wedge (\overline{x_5}) \wedge (x_4 \vee x_5) \wedge AtMostOne(x_1, x_2, x_3, x_4, x_5)$$

Satisfiability: Scheduling Meeting

A can only meet on Monday or Wednesday

B cannot meet on Wednesday

C cannot meet on Friday

D can only meet on Thursday or Friday

SAT problem instance

$$F = (x_1 \vee x_3) \wedge (\overline{x_3}) \wedge (\overline{x_5}) \wedge (x_4 \vee x_5)$$
$$\wedge\ (\overline{x_1} \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_5})$$
$$\wedge\ (\overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_2} \vee \overline{x_4}) \wedge (\overline{x_2} \vee \overline{x_5})$$
$$\wedge\ (\overline{x_3} \vee \overline{x_4}) \wedge (\overline{x_3} \vee \overline{x_5})$$
$$\wedge\ (\overline{x_4} \vee \overline{x_5})$$

SAT Solver → Unsatisfiable

Hardness of SAT

- SAT is NP-Complete
- No polynomial time algorithm for SAT yet
- Known complete algorithms have exponential runtime in worst case

Applications of SAT

- Hardware Model Checking
  - Does a circuit exhibit intended behavior?
  - Verify chip design (e.g., intel)
- Software Verification
  - SMT solvers to verify Microsoft products
  - Verification of embedded software in cars, airplanes etc.
- One of the best approaches in planning and scheduling
- Number theoretic problems
- Connection with NP-hard problems

SAT Encoding: Pythagorean Triple

Can the set of integers {1,2,3,….,} be partitioned into two sets such that no part contains a Pythagorean triple ($a, b, c \in \mathbb{N}$ st $a^2 + b^2 = c^2$)?

Is it possible to assign each integer one of two colors such that if $a^2 + b^2 = c^2$ then a, b, c do not all have the same color?

Not possible for 7825

Proof obtained by a SAT solver has 200 Terrabytes

Largest math proof ever

SAT Encoding: Pythagorean Triple

Is it possible to assign each integer one of two colors such that if $a^2 + b^2 = c^2$ then a, b, c do not all have the same color?

For each integer i, a Boolean variable x_i:

$x_i = 1$ if color(i)= B else $x_i = 0$

For each $\langle a, b, c \rangle$ st $a^2 + b^2 = c^2$:

R R B

$(x_a \vee x_b \vee x_c) \wedge (\overline{x_a} \vee \overline{x_b} \vee \overline{x_c})$

0 0 1 — True

1 1 0 — True

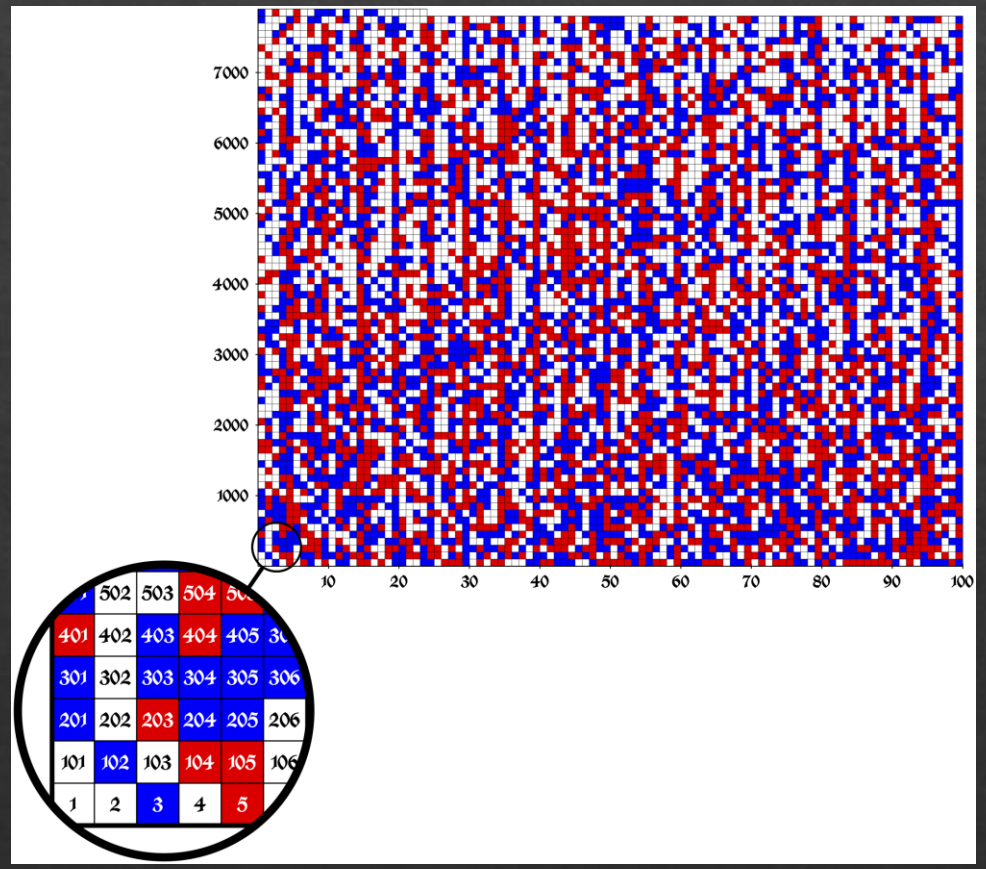
⇒ True

B, B, B

1 1 1 — True

0 0 0 — False

⇒ False



https://www.nature.com/articles/nature.2016.19990

SAT Encoding: Pythagorean Triple

Is it possible to assign each integer one of two colors such that if $a^2 + b^2 = c^2$ then a, b, c do not all have the same color?

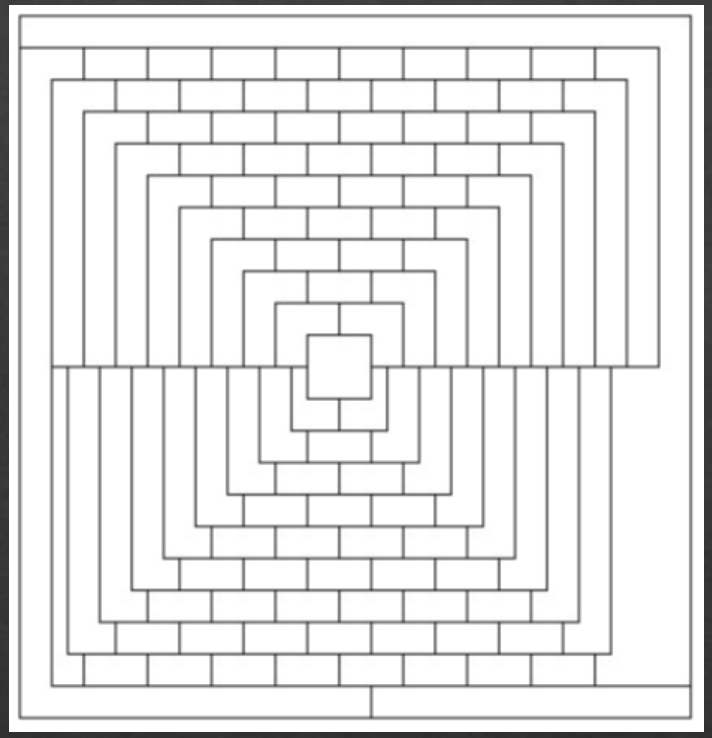
$$
\begin{aligned}
&(x_3 \vee x_4 \vee x_5) \wedge (\bar{x}_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge (x_5 \vee x_{12} \vee x_{13}) \wedge (\bar{x}_5 \vee \bar{x}_{12} \vee \bar{x}_{13}) \wedge \\
&(x_7 \vee x_{24} \vee x_{25}) \wedge (\bar{x}_7 \vee \bar{x}_{24} \vee \bar{x}_{25}) \wedge (x_9 \vee x_{40} \vee x_{41}) \wedge (\bar{x}_9 \vee \bar{x}_{40} \vee \bar{x}_{41}) \wedge \\
&(x_6 \vee x_8 \vee x_{10}) \wedge (\bar{x}_6 \vee \bar{x}_8 \vee \bar{x}_{10}) \wedge (x_8 \vee x_{15} \vee x_{17}) \wedge (\bar{x}_8 \vee \bar{x}_{15} \vee \bar{x}_{17}) \wedge \\
&(x_{10} \vee x_{24} \vee x_{26}) \wedge (\bar{x}_{10} \vee \bar{x}_{24} \vee \bar{x}_{26}) \wedge (x_{12} \vee x_{35} \vee x_{37}) \wedge (\bar{x}_{12} \vee \bar{x}_{35} \vee \bar{x}_{37}) \wedge \\
&(x_{14} \vee x_{48} \vee x_{50}) \wedge (\bar{x}_{14} \vee \bar{x}_{48} \vee \bar{x}_{50}) \wedge (x_9 \vee x_{12} \vee x_{15}) \wedge (\bar{x}_9 \vee \bar{x}_{12} \vee \bar{x}_{15}) \wedge \\
&(x_{15} \vee x_{36} \vee x_{39}) \wedge (\bar{x}_{15} \vee \bar{x}_{36} \vee \bar{x}_{39}) \wedge (x_{12} \vee x_{16} \vee x_{20}) \wedge (\bar{x}_{12} \vee \bar{x}_{16} \vee \bar{x}_{20}) \wedge \\
&(x_{16} \vee x_{30} \vee x_{34}) \wedge (\bar{x}_{16} \vee \bar{x}_{30} \vee \bar{x}_{34}) \wedge (x_{20} \vee x_{48} \vee x_{52}) \wedge (\bar{x}_{20} \vee \bar{x}_{48} \vee \bar{x}_{52}) \wedge \\
&(x_{15} \vee x_{20} \vee x_{25}) \wedge (\bar{x}_{15} \vee \bar{x}_{20} \vee \bar{x}_{25}) \wedge (x_{18} \vee x_{24} \vee x_{30}) \wedge (\bar{x}_{18} \vee \bar{x}_{24} \vee \bar{x}_{30}) \wedge \\
&(x_{24} \vee x_{45} \vee x_{51}) \wedge (\bar{x}_{24} \vee \bar{x}_{45} \vee \bar{x}_{51}) \wedge (x_{21} \vee x_{28} \vee x_{35}) \wedge (\bar{x}_{21} \vee \bar{x}_{28} \vee \bar{x}_{35}) \wedge \\
&(x_{20} \vee x_{21} \vee x_{29}) \wedge (\bar{x}_{20} \vee \bar{x}_{21} \vee \bar{x}_{29}) \wedge (x_{24} \vee x_{32} \vee x_{40}) \wedge (\bar{x}_{24} \vee \bar{x}_{32} \vee \bar{x}_{40}) \wedge \\
&(x_{28} \vee x_{45} \vee x_{53}) \wedge (\bar{x}_{28} \vee \bar{x}_{45} \vee \bar{x}_{53}) \wedge (x_{27} \vee x_{36} \vee x_{45}) \wedge (\bar{x}_{27} \vee \bar{x}_{36} \vee \bar{x}_{45}) \wedge \\
&(x_{30} \vee x_{40} \vee x_{50}) \wedge (\bar{x}_{30} \vee \bar{x}_{40} \vee \bar{x}_{50}) \wedge (x_{33} \vee x_{44} \vee x_{55}) \wedge (\bar{x}_{33} \vee \bar{x}_{44} \vee \bar{x}_{55})
\end{aligned}
$$

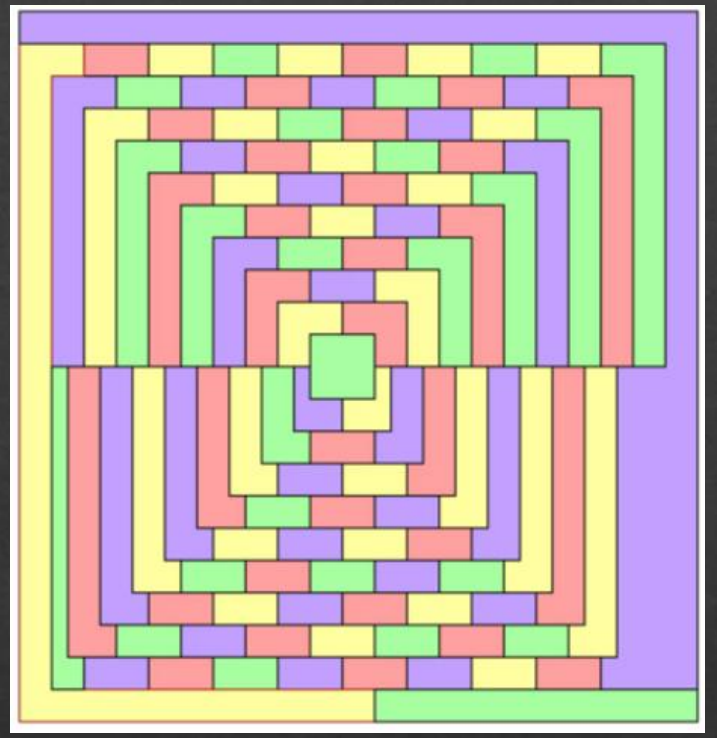
$i = 55$

N=55

7825

SAT Encoding: Graph Coloring





McGregor graph (of order 10): planar, 110 nodes

Cannot be colored with less than 5 colors

K- Coloring

SAT Encoding: Graph Coloring

A graph coloring using at most k colors is called k-coloring. The Graph Coloring Problem seeks whether k-coloring for a graph $G = \langle V, E \rangle$ exists.

5 v_1 v_2 v_3 v_4 v_5

Variables:

$k.|V|$ Boolean variables v_j for $v \in V, 1 \leq j \leq k, v_j = \text{True}$ if node v gets color j

j = Color

Clauses:

→ Every node gets a color: $(v_1 \vee v_2 \vee \cdots \vee v_k)$ for $v \in V$

→ Adjacent nodes have diff color: $(\overline{u_j} \vee \overline{v_j})$ for $(u, v) \in E, 1 \leq j \leq k$

$\overline{u_j}$ = False

j th Color u u_j = True

v_j = False

SAT Encoding: Graph Coloring

$V = \{u, v, w, x, y\}$ $k = 1,2,3$

$Variables = \{u_1, u_2, u_3, \dots, y_1, y_2, y_3\}$

Clauses:

Every node gets a color:

$(u_1 \vee u_2 \vee u_3) \wedge$

$\dots \dots \dots \dots \wedge$

$(y_1 \vee y_2 \vee y_3)$

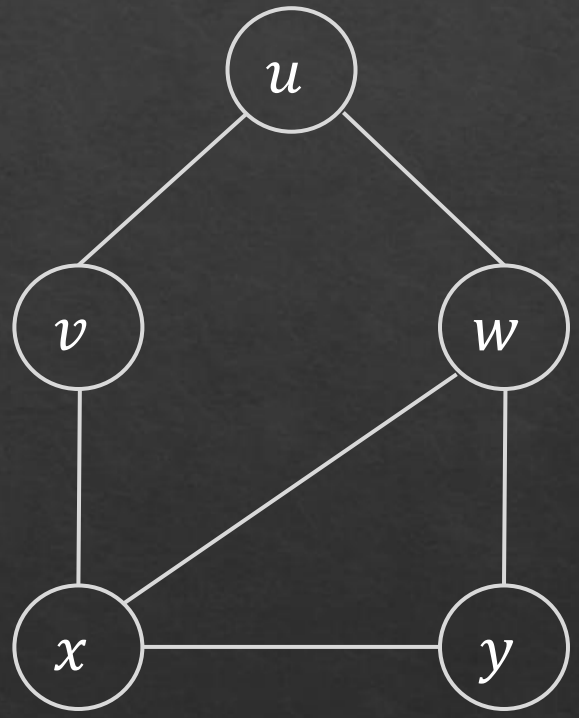
Adjacent nodes have diff color:

$(\overline{u_1} \vee \overline{v_1}) \wedge \cdots \wedge (\overline{u_3} \vee \overline{v_3})$

$\dots \dots \dots \dots$

$(\overline{x_1} \vee \overline{y_1}) \wedge \cdots \wedge (\overline{x_3} \vee \overline{y_3})$

For every edge



SAT Solvers

SAT Solver with inference

Resolution: $P = x$ $\quad \neg P = \neg x$

$(a_1 \vee a_2 \vee x \vee a_3 \vee a_4) \quad (b_1 \vee b_2 \vee \neg x \vee b_3 \vee b_4)$

$\Rightarrow (a_1 \vee a_2 \vee a_3 \vee a_4 \vee b_1 \vee b_2 \vee b_3 \vee b_4)$

$P \rightarrow$ Boolean variable

$\Delta \rightarrow$ CNF with clauses C_i and C_j

$P \in C_i \quad \neg P \in C_j$

Derived clause: $(C_i - \{P\}) \vee (C_j - \{\neg P\}) \rightarrow$ P_resolvant

$\Delta = \{(A, B, \neg C), (\neg B, D)\} \Rightarrow (A, \neg C, D) \rightarrow$ B-resolvant

SATISFIABILITY We derive empty clause

$\Delta = \{ \underset{\wedge}{\_\_}, \underset{\wedge}{\_\_}, \underset{\wedge}{\_\_}, \underset{\text{False}}{\Phi} \} = \{ \underset{\text{False}}{\Phi} \} \rightarrow$ UNSATISFIABILITY

→ Resolution is sound but Incomplete

not guaranteed to derive every clause that is implied by Δ.

COMPLETE SAT

→ Resolution is refutation Complete

guaranteed to derive an empty clause if Δ is unsatisfiable

⇓

Resolution based complete Algo for SAT testing

1. $\{\neg P, R\}$
2. $\{\neg Q, R\}$
3. $\{\neg R\}$ ⇒
4. $\{P, Q\}$

---

5. $\{\neg P\}$ (1,3) → Keep applying resolution until we derive empty clause ⇒ UNSAT
6. $\{\neg Q\}$ (2,3)
7. $\{Q\}$ (4,5)
8. $\{\phi\}$ (6,7) → until we have no option to apply resolution rule ⇒ SAT

↓ UNSAT

Unit Resolution: At least one of resolved clauses has only one literal

$$\frac{C_1 \quad C_2}{}$$

CONDITIONING

$\Delta|L$ ⇒ Replacing every occurrence of L by 'True' and $\neg L$ by 'False'

$$\Delta|L = \{\alpha - \{\neg L\} \mid \alpha \in \Delta, L \notin \alpha\}$$

$C = \alpha \wedge \beta \wedge \gamma$

↳ true

clauses in Δ can be partition into:

$\alpha = (A \vee L \vee \neg B)$

$\alpha = (A \vee \text{True} \vee \neg B)$

$\alpha = (A \vee \neg L \vee \neg B)$

↓ False

$(A \vee \neg B)$

1. $L \in \alpha$ ⇒ α does not appear in $\Delta|L$
2. $\neg L \in \alpha$ ⇒ occurrence of $\neg L$ in α does not have any effect. ⇒ $\{\alpha - \{\neg L\}\}$
3. L or $\neg L \notin \alpha$ ⇒ α will appear in $\Delta|L$ as it is $\alpha = (A \vee B \vee C)$

Example

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

$\neg C \in \alpha \rightarrow \Delta|_C = \{\{A, B\}, \{\neg A, D\}\}$

$\alpha = \{\neg C\}$ → $\{A, B\}$

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

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

$\Delta|_{C,A,\neg D} = \{\square\} \rightarrow$ Contradiction.

$\{\square\}$: Contains an empty clause = False

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

$\{\}$

$\Delta|_{\neg C,D} = \{\} \rightarrow$ Consistent Satisfiable.

DP Algo (Davis-Putnam)

Directed Resolution → Repeated application of resolution
↳ P-resolvant
Bucket Elimination

→ A bucket for each var.
→ Choose a variable ordering (π)
→ Construct and fill the buckets.
→ $\alpha \in \Delta$ - add α to the first bucket P from the top such that P appears in α

$(P, \neg P)$

Bucket Elimination Example

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

① $\pi = C, B, A, D, E$

C-resolution → $\{\neg A, \neg D\}$ (True) → True

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

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

A:

D:

E:

SATISFACTION

$\{\neg A, \neg D\}, \{\neg A, D\}$ →

→ No new resolution

② $\pi = E, A, B, C, D$

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

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

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

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

D:

SATISFACTION

DP(Δ, π) returns UNSAT or SAT

For each variable V:
    create empty bucket B_V

For each clause $C \in \Delta$:
    V ← first var. of C as per π
    $B_V \leftarrow B_V \cup \{C\}$

For each var V as per π:
    If B_V is not empty:
        for each V-resolvant of $C \in B_V$:
            if C is empty ($\{\}$):
                return UNSAT
        → U ← first var of C as per π
        $B_U \leftarrow B_U \{C\}$

return SAT

Forgetting (Existential Quantification)

$B \Rightarrow C,\ C \Rightarrow D \Rightarrow B \Rightarrow D$

$\exists C\ \Delta \models B \Rightarrow D$

$\Delta = \begin{matrix} A \Rightarrow B \\ B \Rightarrow C \\ C \Rightarrow D \\ D \Rightarrow E \end{matrix} \models A \Rightarrow E$

$A \Rightarrow D$
$D \Rightarrow E$
$A \Rightarrow E$

$\exists C, B, D\ \Delta \models A \Rightarrow E$

$\exists x \Delta = (\Delta|x) \vee (\Delta|\neg x)$

forgetting x

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

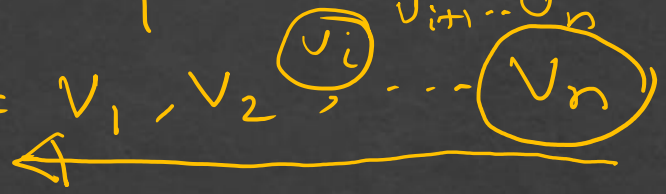
↳ fewer variables

$\{ A \Rightarrow B,\ B \Rightarrow C \mid A \Rightarrow C \}$

Extract Satisfying Assignment (ϕ)

→ Process vars. from bottom to top.

$\pi = v_1, v_2, \ldots, v_i, v_{i+1} \ldots v_n$



→ B_{v_i} is empty ⇒ Assign any value.

✓ else ⇒ Assign v_i such a way

$i < j < n$ v_j

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

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

True True

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

$E = T$, $D = F$ $A = F$

$B = F$, $C = T$

→ A :

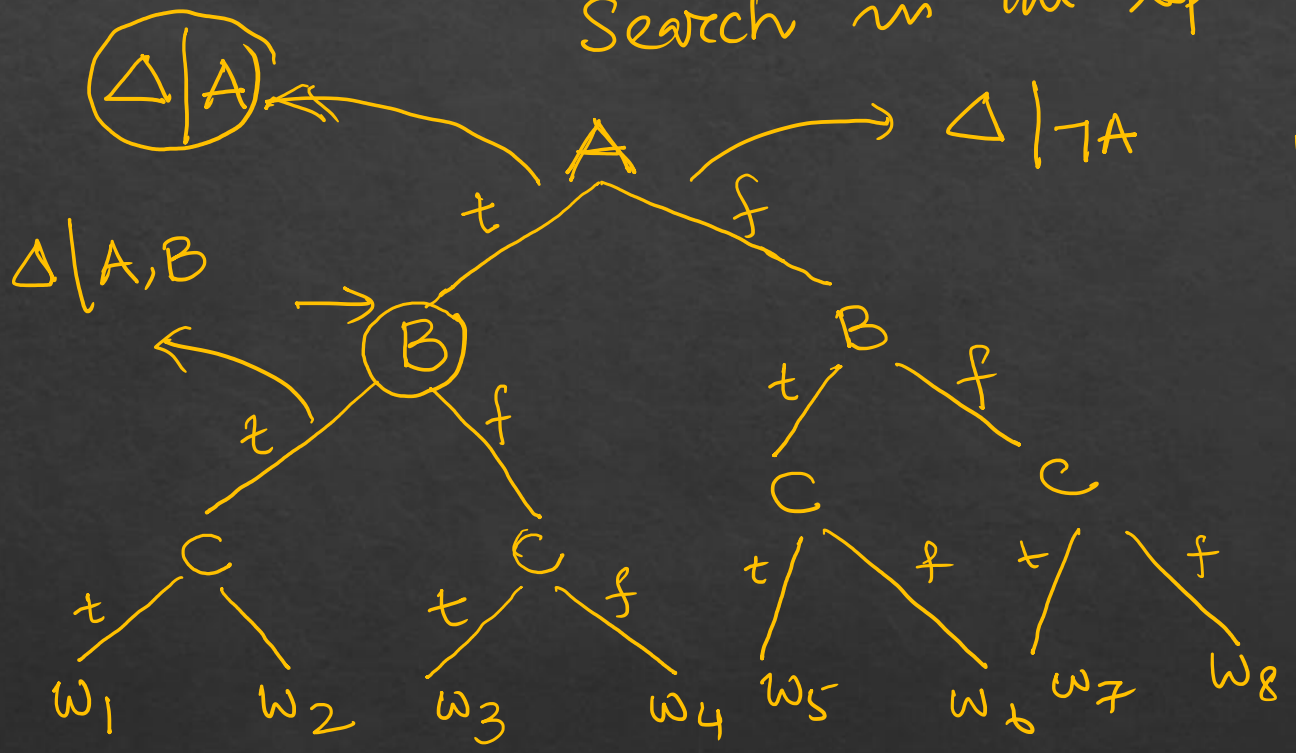
→ D :

E :

SAT by Search

DPLL (Davis-Putnam, Logemann-Loveland)

Search in the space of truth assignment



$w_1 \ldots w_8$

$w_3 = \{A=t, B=f, C=t\}$

leaf → a complete truth assignment

↳ Satisfiability

↳ finding a leaf node that satisfies the CNF

A tree that has depth n.

→ DFS to explore the search space

DPLL- (Δ, d) return SAT + variable instantiations or UNSAT

If $\Delta = \{\}$ then [// No Contradiction]
return $\{\}$

DPLL- $(\Delta, 0)$ else if $\{\} \in \Delta$ then [// Contradiction]
return UNSAT

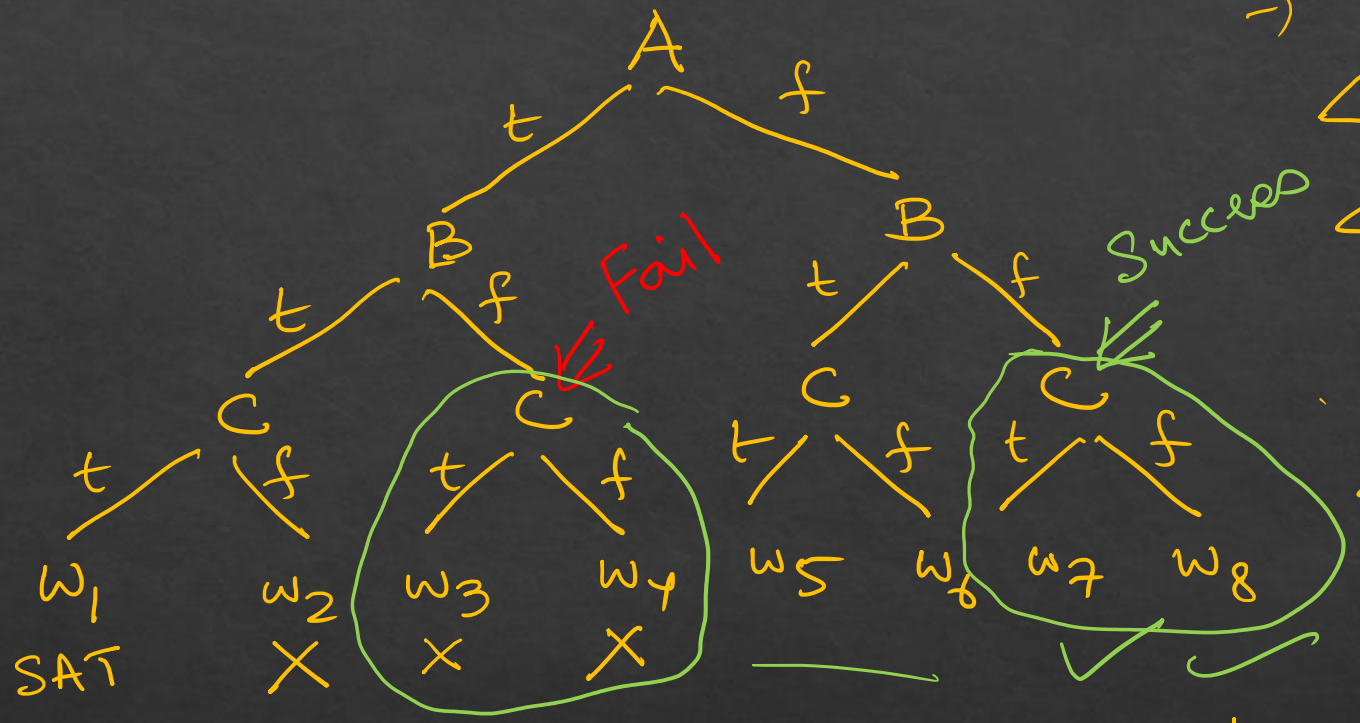
else if $L = \text{DPLL-}(\Delta | P_{d+1}, d+1) \neq \text{UNSAT}$ then
return $L \cup \{P_{d+1}\}$

else if $L = \text{DPLL-}(\Delta | \neg P_{d+1}, d+1) \neq \text{UNSAT}$ then
return $L \cup \{\neg P_{d+1}\}$

else
return UNSAT

Example DPLL-

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



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

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

$\Delta|_{A,B,C} = \{\} \rightarrow$ SAT

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

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

$\{\} \in \Delta|_{A,\neg B}$

↳ UNSAT

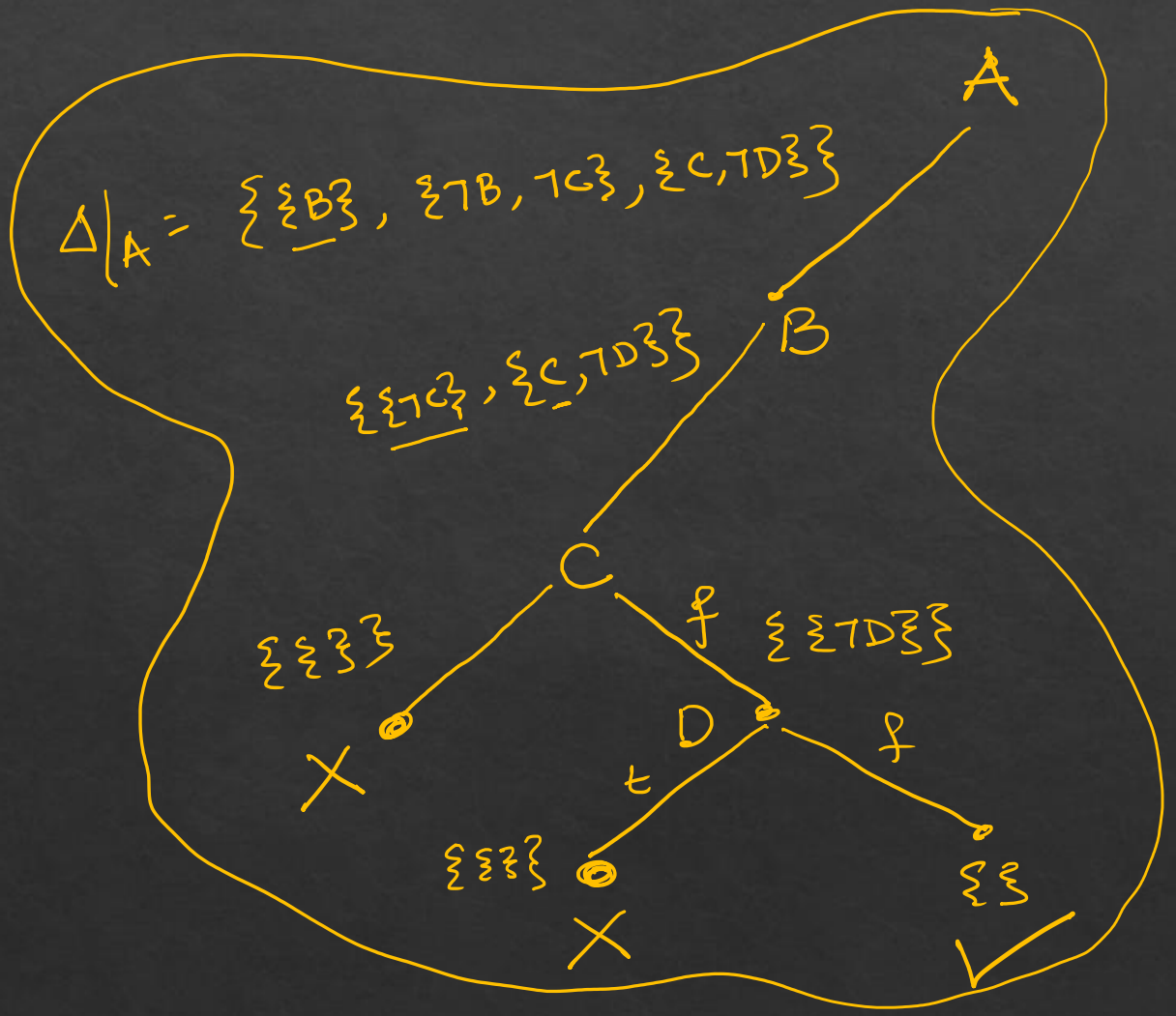
$\Delta|_{\neg A} = \{\{\neg B, C\}\}$

$\Delta|_{\neg A,\neg B} = \{\}$ → SAT

DPLL- may detect success before reaching the leaf.

Amount of work by DPLL-

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



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

$\Delta|_A = \{\}$ SAT

$\{\} \in \Delta|_A$ UNSAT

DPLL- cannot decide failure or success at this pt.

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

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

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

$\Delta|_A \rightarrow \{A\}$ $\{\neg A, B\}$

$\{B\}$

First: Closing Δ under unit resolution collecting all the unit clauses.

Second: Set the variables to satisfy unit clauses.

$\{P\} \Rightarrow$ Set P = True

$\{\neg P\} \rightarrow$ Set P = False

Third: Checking for success or fail.

Unit Propagation → Applying unit resolution in chain

I: A set of literals either present in unit clauses or were derived from Δ using unit resolution

Γ : $\Delta \mid I$

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

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

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

```
DPLL(Δ)
    (I, Γ) = UNIT-PROPAGATION(Δ)
    if Γ = {} then
        return I
    else if {} ∈ Γ then
        return UNSAT
    else
        Choose a literal X in Γ
        if L = DPLL(Γ|X) ≠ UNSAT then
            return L ∪ I ∪ {X}
        else L = DPLL(Γ|¬X) ≠ UNSAT then
            return L ∪ I ∪ {¬X}
    else
        return UNSAT
```

SAT by Search + Inference

GRASP, MiniSAT, zChaff

Limitation of DPLL

Chronological Backtracking:

→ try both values of V at level l.

→ trace contradiction?

↳ move to l-1, undo intermediate assignments.

→ try a new value at l-1

→ for l-2 ... So on

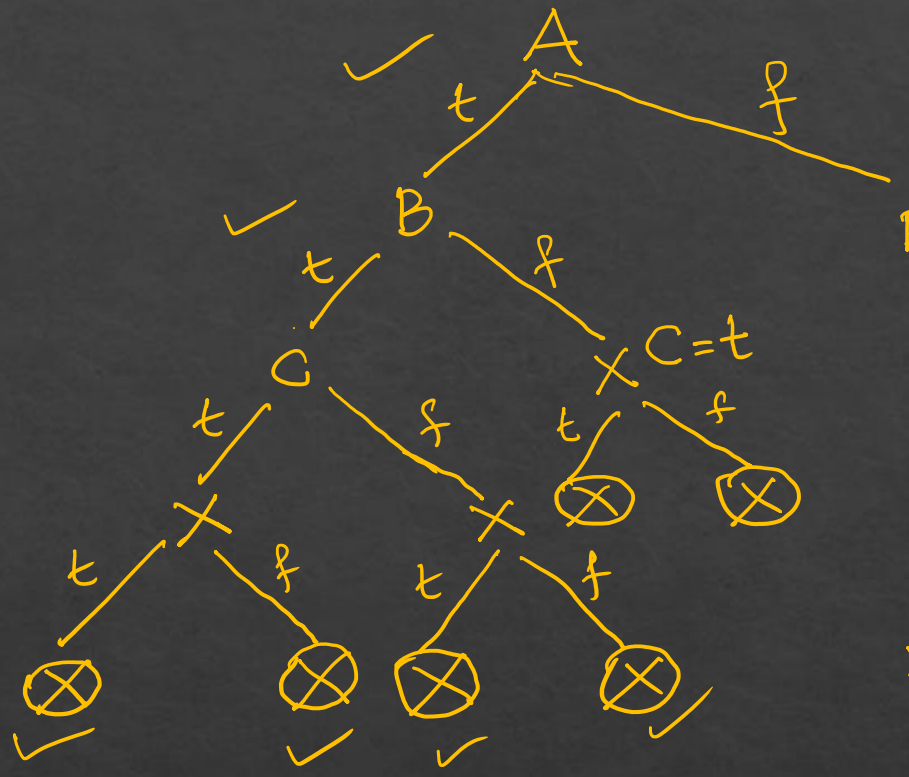
Final: move to level 0, always a contradiction

→ inconsistent

Not keeping track of the reason behind contradiction

Backtracking at level l is done after trying out all the possibilities at l+1

$$\Delta = \begin{cases} 1.\ \{A, B\} \\ 2.\ \{B, C\} \\ 3.\ \{\neg A, \neg X, Y\} \\ 4.\ \{\neg A, X, Z\} \\ 5.\ \{\neg A, \neg Y, Z\} \\ 6.\ \{\neg A, X, \neg Z\} \\ 7.\ \{\neg A, \neg Y, \neg Z\} \end{cases}$$



A=t

B=t { }

6

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

$\Delta|_{A,B}$

x=f : $\neg X$ –

4′ → {z} → 4″ →

6′4″ → {x} –

{ { } }

A=t : 3 → {¬x, y} → 3′

4 → {x, z} → 4′

5 → {¬y, z} → 5′ –

6 → {x, ¬z} – 6′ →

7 → {¬y, ¬z} – 7′

B=t, C=t

x=t : 3′ → {x} → 3″

5′3″ → {z} → 53″

7′3″ → {¬z} → 73″

53″ 73″ → { { } }

$\Delta|_{A,B,C,X}$ and $\Delta|_{A,B,C,\neg X}$ $\Delta|_{A,B,C}$

DPLL cannot detect

→ All the contradictions are at level 0 A = true

→ Assignment of B, C does not any effect → irrelevant

→ Chronological BT does not see it

→ Realization that A = T is a bad decision is achieved after 6 contradiction

Non-chronological BT

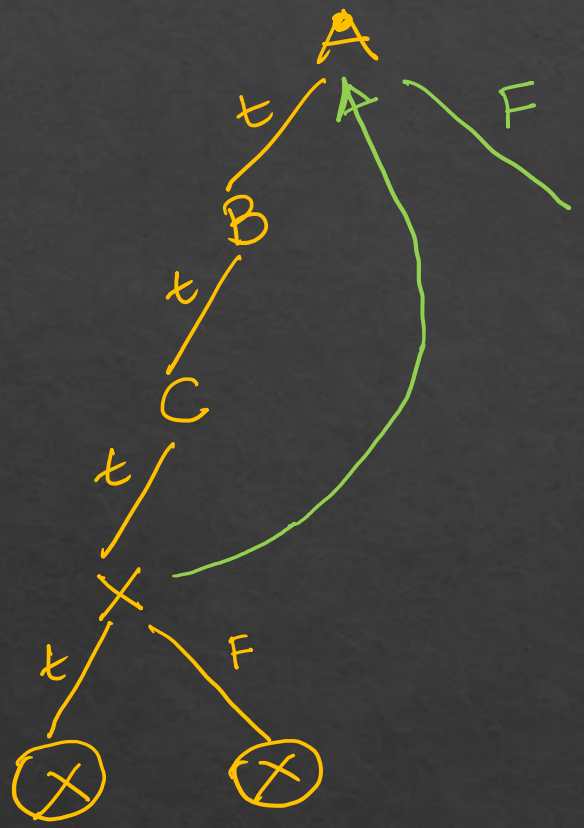
→ Probe the set of vars that actually are involved in contradiction (A)

→ technique for CSP (Backjumping)

Philosophy: Backtracking to a lower level l, without trying out every possibility between the current level and l

Intuitions: → identify every assignment that lead to contradiction
↳ set of assignments (conflict set)

→ backtrack to the most recent decision var in the conflict set



SEARCH

$A=T, B=T, C=T, X=T \Rightarrow Y=T, Z=T$

$CS1 = \{A=T, X=T, Y=T, Z=T\}$

$A=T, B=T, C=T, X=F \Rightarrow Z=T$

$CS2 = \{A=T, X=F, Z=T\}$

$A=F$

$A=F$

After 2 contradictions $\rightarrow$ $A=T$ is a bad decision

Empowering Clause: → unit Resolution DPLL

— $\alpha : \Delta \models \alpha$

— If we add α to Δ explicitly, unit resolution will have additional info

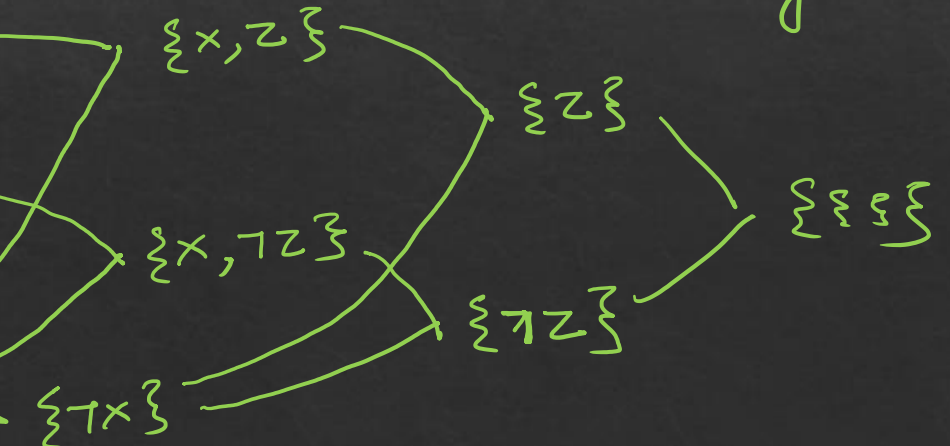
— $\Delta \models \{\neg A, \neg x\} \rightarrow$.

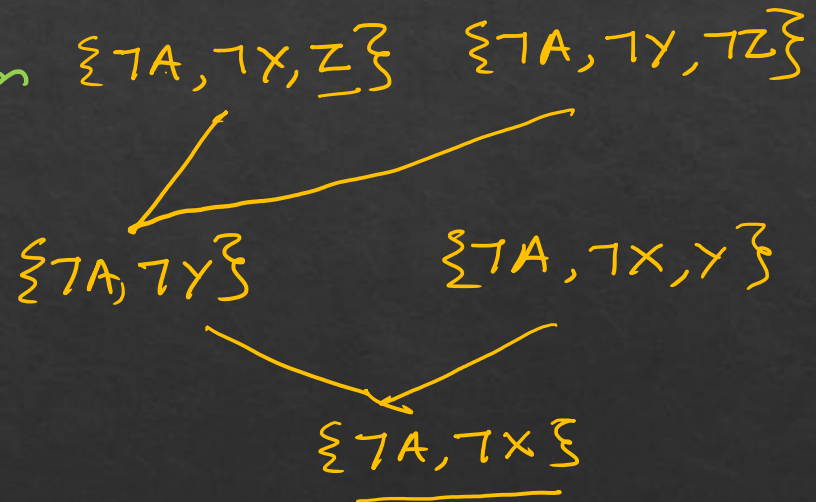
Empowering ⇒ unit resolution to detect contradiction early

1. $\{A, B\}$
2. $\{B, C\}$
3. $\{\neg A, \neg x, y\}$
4. $\{\neg A, x, z\}$
5. $\{\neg A, \neg x, z\}$
6. $\{\neg A, x, \neg z\}$
7. $\{\neg A, \neg y, \neg z\}$

---

8. $\{\neg A, \neg x\}$
9. $\{A\}$





How to identify empowering clauses?

→ Whenever UR finds a contradiction
  ↳ There is an opportunity to discover an empowering clause (new)
    ↳ allow new implications.

→ Conflict driven clauses or Conflict clause

Add these clauses → allow UR to detect contradictions earlier.

NCB and Conflic driven clauses (CDC)

→ Derive CDCs

→ Use these clauses in performing NCB.

Conflict Analysis

Δ

1. {A, B}
2. {C, D}
3. {¬A, ¬X, Y}
4. {¬A, X, Z}
5. {¬A, ¬Y, Z}
6. {¬A, X, ¬Z}
7. {¬A, ¬Y, ¬Z}

8. {¬A, ¬X}

Trace of unit resolution over Δ (Implication Graph)

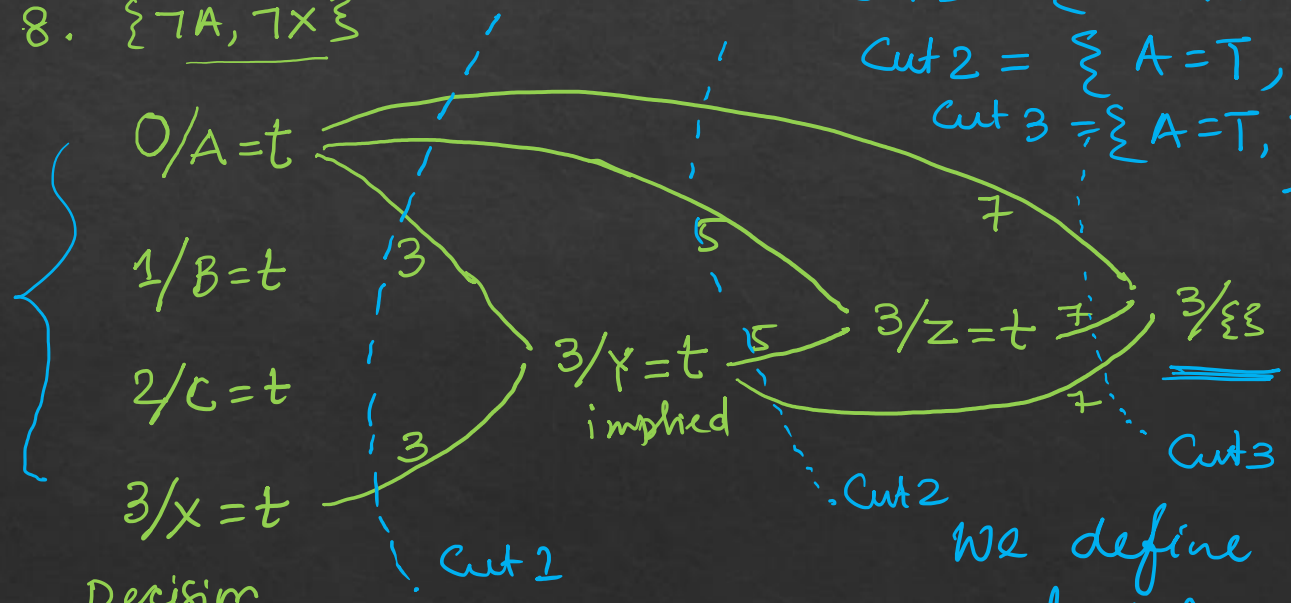
[Δ | A, B, C, X]

Decision node | implied node

Cut 1 = {A = T, X = T}

Cut 2 = {A = T, Y = T}

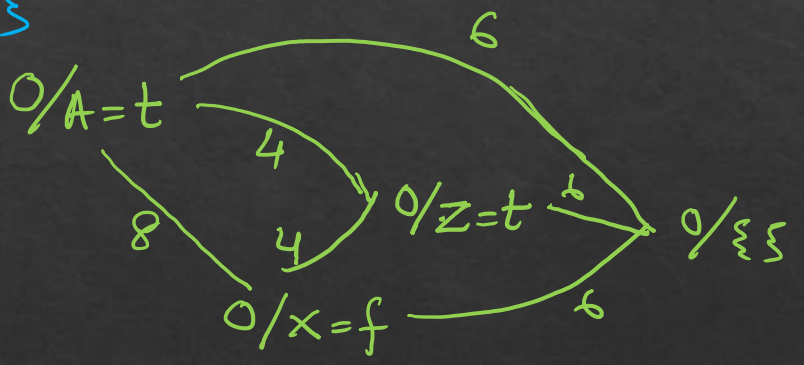
Cut 3 = {A = T, Y = T, Z = T}



We define cuts that seperates decision vars. from the conflict node

Power of the empowerment clauses

[Δ | A]



$$\left\{\begin{array}{l} Cut_1 = \{A=T,\ X=T\} \\ Cut_2 = \{A=T,\ Y=T\} \\ Cut_3 = \{A=T,\ Y=T,\ Z=T\} \end{array}\right\} \quad \text{Contradiction} \rightarrow \text{Conflict sets}$$

$\underline{A=T \wedge X=T}$ → Conflict set

↳ Adding this to Δ will lead to a contradiction.

$\alpha = \neg(A=T \wedge X=T)$ $\quad \Delta \models \alpha$

↳ $(\neg A \vee \neg X)$ → Add this to Δ

↳ learned Clauses

↳ conflict driven Clauses

Multiplicity of CDC

How to choose?

Asserting clause: Includes only one variable assignment at the last decision level

✓ $\{A^0 = T, X^3 = T\}$ ✓ $\{A^0 = T, Y^3 = T\}$

$\{A^0 = T, Y^3 = T, Z^3 = T\}$ ✗

Assertion level: second highest level in the clause

$\{A^0 = T, X^3 = T\}$ $\{A^0 = T, Y^3 = T\}$

⇓ ⇓

0 0

Backtracking to the assertion level

Modern SAT Solver

1. Identify Conflict driven clauses (asserting)
2. Backtrack to the assertion level
3. Add the CDC to Δ
4. Apply unit Resolution.

DPLL+(CNF Δ) returns SAT or UNSAT

$D \leftarrow ()$ // empty decision seq.

$\Gamma \leftarrow \{\}$ // learned clauses

While true do

If UR detects a contra in $(\Delta + \Gamma|_D)$:

Contradiction in original Δ ⟵ if $D = ()$: // contradiction w/o decision

return UNSAT

else // backtracking to AL

$\alpha \leftarrow$ asserting clause

$m \leftarrow$ AL of α

$D \leftarrow$ first m decision in D // reusing decisions l_{m+1} onward.

adding α to Γ

else // unit resolution does not lead to contra

if L is literal L or $\neg L$ ~~⊢~~ (Δ, Γ, D):
unit resolution

$D \leftarrow D; L$ ⟶ Decision.

else

return SAT

decide on variables that are still unassigned

1: {A, B}

2. {B, C} ✓

3. {¬A, ¬X, Y} ✓

→ 4. {¬A, X, Z} ✓

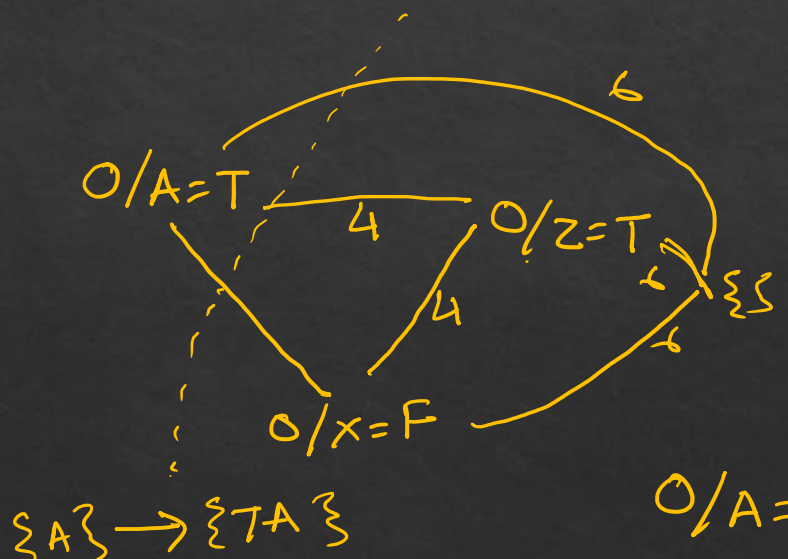
5. {¬A, ¬Y, Z} ✓

6. {¬A, X, ¬Z} ✓

7. {¬A, ¬Y, ¬Z} ✓

---

8. {¬A, ¬X}



Iteration 1:

$D = ( )$, $\Gamma = \{\ \}$

$D = (A=T, B=T, C=T, X=T)$

$\alpha = \{\neg A, \neg X\}$

$m = 0$

$D = (A=T)$, $\Gamma = \{\{\neg A, \neg X\}\}$ $\Delta' = \Delta + \Gamma$

Iteration 2

Δ, $D = (A=F)$ $\Gamma = \{\{\neg A, \neg X$

$\alpha = \{\neg A\}$ (0)

$m = -1$

$D = ( )$

$\Gamma = \{\{\neg A, \neg X\}, \{\neg A\}\}$ ↓

$A = F$

Iteration

$O/A=F$

1

$O/B=T$

SAT

$\boxed{A=F,\ B=T}$