

# CS 228 : Logic in Computer Science

Krishna. S

# CNF Explosion

---

Consider the formula  $\varphi = (p_1 \wedge p_2 \cdots \wedge p_n) \vee (q_1 \wedge q_2 \cdots q_m)$

- ▶ What is the equivalent CNF formula?
- ▶  $\bigwedge_{i \in \{1, \dots, n\}, j \in \{1, \dots, m\}} (p_i \vee q_j)$  has  $mn$  clauses
- ▶ Distributivity explodes the formula

# Tseitin Encoding : The Idea

---

- ▶ Introducing fresh variables, Tseitin encoding can give an equisatisfiable formula without exponential explosion.
- ▶  $\varphi = p \vee (q \wedge r)$
- ▶ Replace  $q \wedge r$  with a fresh variable  $x$  and add a clause which asserts that  $x$  simulates  $q \wedge r$
- ▶  $(p \vee x) \wedge (x \leftrightarrow (q \wedge r))$
- ▶ It is enough to consider (Why?)  $(p \vee x) \wedge (x \rightarrow (q \wedge r))$  which is  $(p \vee x) \wedge (\neg x \vee q) \wedge (\neg x \vee r)$

# Tseitin Encoding

---

- ▶ Assume the input formula is in NNF (all negations attached only to literals) and has only  $\wedge, \vee$
- ▶ Replace each  $G_1 \wedge \cdots \wedge G_n$  just below a  $\vee$  with a fresh variable  $p$  and conjunct  $(\neg p \vee G_1) \wedge \cdots \wedge (\neg p \vee G_n)$  (same as  $p \rightarrow G_1 \wedge \cdots \wedge G_n$ ).

# Tseitin Encoding

---

- ▶  $\varphi = (p_1 \wedge p_2 \cdots \wedge p_n) \vee (q_1 \wedge q_2 \cdots \wedge q_m)$
- ▶ Choose fresh variables  $x, y$
- ▶  $\psi = (x \vee y) \wedge \bigwedge_{i \in \{1, \dots, n\}} (\neg x \vee p_i) \wedge \bigwedge_{j \in \{1, \dots, m\}} (\neg y \vee q_j)$  has  $m + n + 1$  clauses
- ▶  $\varphi$  and  $\psi$  are equisatisfiable. Prove.

(DPLL) Davis-Putnam-Loveland-Logemann Method

# DPLL

---

- ▶ DPLL combines search and deduction to decide CNF satisfiability
- ▶ Underlies most modern SAT solvers

# Partial Assignments

---

An assignment is a function  $\alpha : V \rightarrow \{0, 1\}$  which maps each variable to true(1) or false (0). A partial assignment  $\alpha$  is one under which some variables are unassigned.



# Partial Assignments

---

An assignment is a function  $\alpha : V \rightarrow \{0, 1\}$  which maps each variable to true(1) or false (0). A partial assignment  $\alpha$  is one under which some variables are unassigned.

- ▶ Under a partial assignment  $\alpha$ , the **state** of a variable  $v$  is true if  $\alpha(v) = 1$ , false if  $\alpha(v) = 0$ , and unassigned otherwise.
- ▶ Let  $V = \{x, y, z\}$  and let  $\alpha(x) = 1, \alpha(y) = 0$ . Then the state of  $x$  under  $\alpha$  is true, state of  $y$  is false, and the state of  $z$  is unassigned.

# State of a Clause

---

Assume we have a formula in CNF. Under a partial assignment  $\alpha$ ,

- ▶ a clause  $C$  is true if there exists some literal  $\ell$  in  $C$  whose state is true
- ▶ a clause  $C$  is false if the state of all literals in  $C$  is false
- ▶ Otherwise, the state of  $C$  is unassigned

# State of a Clause

---

Assume we have a formula in CNF. Under a partial assignment  $\alpha$ ,

- ▶ a clause  $C$  is true if there exists some literal  $\ell$  in  $C$  whose state is true
- ▶ a clause  $C$  is false if the state of all literals in  $C$  is false
- ▶ Otherwise, the state of  $C$  is unassigned

Consider the partial assignment  $\alpha(x) = 0, \alpha(y) = 1$ .

# State of a Clause

---

Assume we have a formula in CNF. Under a partial assignment  $\alpha$ ,

- ▶ a clause  $C$  is true if there exists some literal  $\ell$  in  $C$  whose state is true
- ▶ a clause  $C$  is false if the state of all literals in  $C$  is false
- ▶ Otherwise, the state of  $C$  is unassigned

Consider the partial assignment  $\alpha(x) = 0, \alpha(y) = 1$ .

- ▶ The state of  $C = x \vee y \vee z$  is true

# State of a Clause

---

Assume we have a formula in CNF. Under a partial assignment  $\alpha$ ,

- ▶ a clause  $C$  is true if there exists some literal  $\ell$  in  $C$  whose state is true
- ▶ a clause  $C$  is false if the state of all literals in  $C$  is false
- ▶ Otherwise, the state of  $C$  is unassigned

Consider the partial assignment  $\alpha(x) = 0, \alpha(y) = 1$ .

- ▶ The state of  $C = x \vee y \vee z$  is true
- ▶ The state of  $C = x \vee \neg y \vee z$  is unassigned

# State of a Clause

---

Assume we have a formula in CNF. Under a partial assignment  $\alpha$ ,

- ▶ a clause  $C$  is true if there exists some literal  $\ell$  in  $C$  whose state is true
- ▶ a clause  $C$  is false if the state of all literals in  $C$  is false
- ▶ Otherwise, the state of  $C$  is unassigned

Consider the partial assignment  $\alpha(x) = 0, \alpha(y) = 1$ .

- ▶ The state of  $C = x \vee y \vee z$  is true
- ▶ The state of  $C = x \vee \neg y \vee z$  is unassigned
- ▶ The state of  $C = x \vee \neg y$  is false

# State of a Formula

---

Under a partial assignment  $\alpha$ ,

- ▶ A CNF formula  $F$  is true if for each  $C \in F$ ,  $C$  is true
- ▶ A CNF formula  $F$  is false if there  $C \in F$  such that  $C$  is false
- ▶ Otherwise,  $F$  is unassigned.

# State of a Formula

---

Under a partial assignment  $\alpha$ ,

- ▶ A CNF formula  $F$  is true if for each  $C \in F$ ,  $C$  is true
- ▶ A CNF formula  $F$  is false if there  $C \in F$  such that  $C$  is false
- ▶ Otherwise,  $F$  is unassigned.

Consider the partial assignment  $\alpha(x) = 0, \alpha(y) = 1$ .



# State of a Formula

---

Under a partial assignment  $\alpha$ ,

- ▶ A CNF formula  $F$  is true if for each  $C \in F$ ,  $C$  is true
- ▶ A CNF formula  $F$  is false if there  $C \in F$  such that  $C$  is false
- ▶ Otherwise,  $F$  is unassigned.

Consider the partial assignment  $\alpha(x) = 0, \alpha(y) = 1$ .

- ▶ The state of  $F = (x \vee y \vee z) \wedge (x \vee \neg y \vee z)$  is unassigned

# State of a Formula

---

Under a partial assignment  $\alpha$ ,

- ▶ A CNF formula  $F$  is true if for each  $C \in F$ ,  $C$  is true
- ▶ A CNF formula  $F$  is false if there  $C \in F$  such that  $C$  is false
- ▶ Otherwise,  $F$  is unassigned.

Consider the partial assignment  $\alpha(x) = 0, \alpha(y) = 1$ .

- ▶ The state of  $F = (x \vee y \vee z) \wedge (x \vee \neg y \vee z)$  is unassigned
- ▶ The state of  $F = (x \vee y \vee z) \wedge (x \vee \neg y)$  is false

# Unit Clause and Unit Literal

---

Let  $C$  be a clause and  $\alpha$  a partial assignment. Then

- ▶  $C$  is a unit clause under  $\alpha$  if there is a literal  $\ell \in C$  which is unassigned, and the rest are false.
- ▶ Then  $\ell$  is a unit literal under  $\alpha$ .

# Unit Clause and Unit Literal

---

Let  $C$  be a clause and  $\alpha$  a partial assignment. Then

- ▶  $C$  is a unit clause under  $\alpha$  if there is a literal  $\ell \in C$  which is unassigned, and the rest are false.
- ▶ Then  $\ell$  is a unit literal under  $\alpha$ .

Let  $\alpha(x) = 0, \alpha(y) = 1$  be a partial assignment.

- ▶  $C = x \vee \neg y \vee \neg z$  is a unit clause and  $\neg z$  is a unit literal

# Unit Clause and Unit Literal

---

Let  $C$  be a clause and  $\alpha$  a partial assignment. Then

- ▶  $C$  is a unit clause under  $\alpha$  if there is a literal  $\ell \in C$  which is unassigned, and the rest are false.
- ▶ Then  $\ell$  is a unit literal under  $\alpha$ .

Let  $\alpha(x) = 0, \alpha(y) = 1$  be a partial assignment.

- ▶  $C = x \vee \neg y \vee \neg z$  is a unit clause and  $\neg z$  is a unit literal
- ▶  $C = x \vee \neg y \vee \neg z \vee w$  is not a unit clause

# Unit Clause and Unit Literal

---

Let  $C$  be a clause and  $\alpha$  a partial assignment. Then

- ▶  $C$  is a unit clause under  $\alpha$  if there is a literal  $\ell \in C$  which is unassigned, and the rest are false.
- ▶ Then  $\ell$  is a unit literal under  $\alpha$ .

Let  $\alpha(x) = 0, \alpha(y) = 1$  be a partial assignment.

- ▶  $C = x \vee \neg y \vee \neg z$  is a unit clause and  $\neg z$  is a unit literal
- ▶  $C = x \vee \neg y \vee \neg z \vee w$  is not a unit clause
- ▶  $C = x \vee \neg y$  is not a unit clause

# DPLL

---

DPLL maintains a partial assignment, to begin with the empty assignment.

- ▶ Assigns unassigned variables 0 or 1 randomly
- ▶ Sometimes, forced to assign 0 or 1 to unit literals

# DPLL Actions

---

- ▶ DPLL has 3 actions : decisions, unit propagation and backtracking
- ▶ Decisions : Decide an assignment for a variable (random choice)
- ▶ Implied assignments or unit propagation : to deal with unit literals
- ▶ Backtrack when in a conflict



# DPLL Algorithm

---

- ▶ At any time, the state of the algorithm is a pair  $(F, \alpha)$  where  $F$  is the CNF and  $\alpha$  is a partial assignment
- ▶ A state  $(F, \alpha)$  is **successful** if  $\alpha$  sets some literal in each clause of  $F$  to be true
- ▶ A **conflict** state is one where  $\alpha$  sets all literals in some clause of  $F$  to be false

# DPLL Algorithm

---

- ▶ Let  $F|_{\alpha}$  denote the set of clauses obtained by deleting from  $F$ , any clause containing a true literal from  $\alpha$ , and deleting from each remaining clause, all literals false under  $\alpha$ . Let  $\alpha(x) = 0, \alpha(y) = 1$ .
- ▶ For  $F = (x \vee y \vee z) \wedge (x \vee \neg y \vee \neg z)$ ,  $F|_{\alpha} = \{\neg z\}$
- ▶ For  $F = (x \vee y) \wedge (\neg x \vee \neg y)$ ,  $F|_{\alpha} = \{\}$ .
- ▶ For  $F = (x \vee \neg y)$ ,  $\perp \in F|_{\alpha}$
- ▶ If  $(F, \alpha)$  is successful, then  $F|_{\alpha} = \{\}$
- ▶ If  $(F, \alpha)$  is in conflict, then the empty clause  $\perp$  is in  $F|_{\alpha}$ .

# The DPLL Algorithm

---

Input : CNF formula  $F$ .

1. Initialise  $\alpha$  as the empty assignment
2. While there is a unit clause  $L$  in  $F|_{\alpha}$ , add  $L = 1$  to  $\alpha$  (unit propagation)
3. If  $F|_{\alpha}$  contains no clauses, then stop and output  $\alpha$
4. If  $F|_{\alpha}$  contains the empty clause, then apply the learning procedure to add a new clause  $C$  to  $F$ . If it is the empty clause, output UNSAT. Otherwise, backtrack to the highest level at which  $C$  is a unit clause, go to line 2.
5. Decide on a new assignment  $p = b$  to be added to  $\alpha$ , goto line 2.

# DPLL Example

$$c_1 = \neg p_1 \vee p_2$$

$$c_2 = \neg p_1 \vee p_3 \vee p_5$$

$$c_3 = \neg p_2 \vee p_4$$

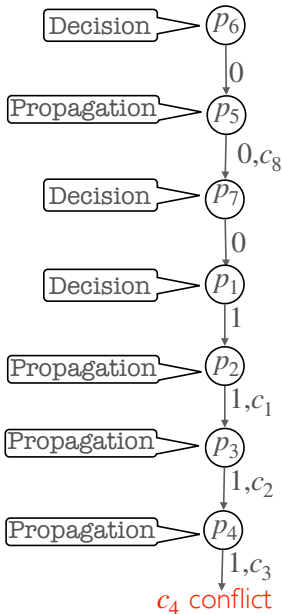
$$c_4 = \neg p_3 \vee \neg p_4$$

$$c_5 = p_1 \vee p_5 \vee \neg p_2$$

$$c_6 = p_2 \vee p_3$$

$$c_7 = p_2 \vee \neg p_3 \vee p_7$$

$$c_8 = p_6 \vee \neg p_5$$



## Clause Learning

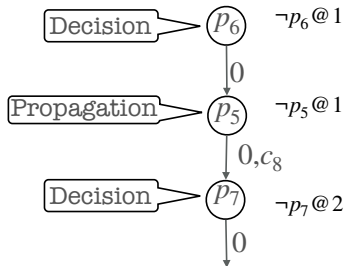
# Run of DPLL

---

The partial assignment in construction is called a **a run of DPLL**. In the previous slide, the run ended in a conflict.

# Decision Level

During a run, the decision level of a true literal is the number of decisions after which the literal was made true.



# Implication Graphs

---

During a DPLL run, we maintain a data structure called an implication graph.

Under a partial assignment  $\alpha$ , the implication graph  $G = (V, E)$ ,

- ▶  $V$  is the set of true literals under  $\alpha$ , and the conflict node
- ▶  $E = \{(\ell_1, \ell_2) \mid \neg \ell_1 \text{ belongs to the clause due to which unit propagation made } \ell_2 \text{ true}\}$

Each node is annotated with the decision level.



$$c_1 = \neg p_1 \vee p_2$$

$$c_2 = \neg p_1 \vee p_3 \vee p_5$$

$$c_3 = \neg p_2 \vee p_4$$

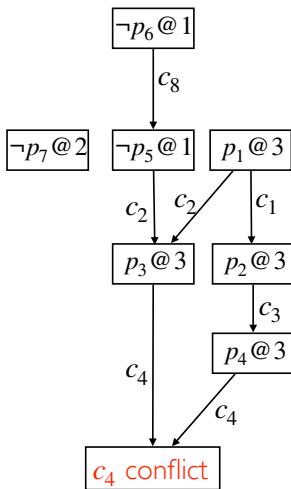
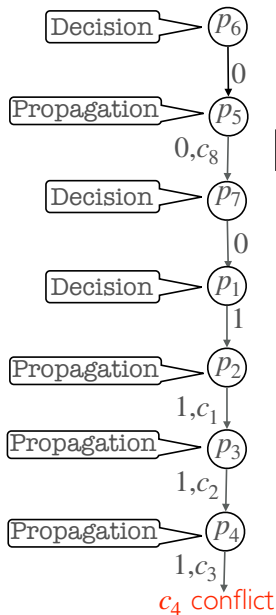
$$c_4 = \neg p_3 \vee \neg p_4$$

$$c_5 = p_1 \vee p_5 \vee \neg p_2$$

$$c_6 = p_2 \vee p_3$$

$$c_7 = p_2 \vee \neg p_3 \vee p_7$$

$$c_8 = p_6 \vee \neg p_5$$



# Conflict Clause

Traverse the implication graph backwards to find the set of decisions that created a conflict. The negations of the causing decisions is the **conflict clause**.

$$c_1 = \neg p_1 \vee p_2$$

$$c_2 = \neg p_1 \vee p_3 \vee p_5$$

$$c_3 = \neg p_2 \vee p_4$$

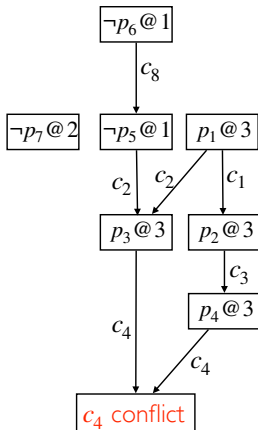
$$c_4 = \neg p_3 \vee \neg p_4$$

$$c_5 = p_1 \vee p_5 \vee \neg p_2$$

$$c_6 = p_2 \vee p_3$$

$$c_7 = p_2 \vee \neg p_3 \vee p_7$$

$$c_8 = p_6 \vee \neg p_5$$



Conflict clause :  $p_6 \vee \neg p_1$  is added : resolve  $c_4$  with  $c_3, c_1, c_2, c_8$

# Clause Learning

---

- ▶ We add the conflict clause to the input set of clauses
- ▶ backtrack to the second last conflicting decision, and proceed like DPLL

## Adding the conflict clause

- ▶ does not affect satisfiability of the original formula (think of resolution)
- ▶ ensures that the conflicting partial assignment will not be tried again

$$c_1 = \neg p_1 \vee p_2$$

$$c_2 = \neg p_1 \vee p_3 \vee p_5$$

$$c_3 = \neg p_2 \vee p_4$$

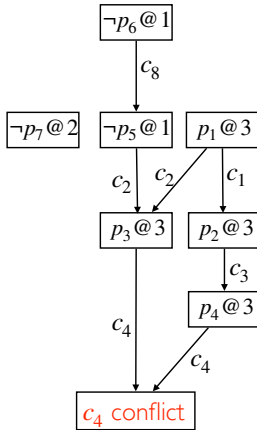
$$c_4 = \neg p_3 \vee \neg p_4$$

$$c_5 = p_1 \vee p_5 \vee \neg p_2$$

$$c_6 = p_2 \vee p_3$$

$$c_7 = p_2 \vee \neg p_3 \vee p_7$$

$$c_8 = p_6 \vee \neg p_5$$



The second last decision is  $p_6 = 0$ . Unit propagation will force  $p_1 = 0$ .

The combination  $p_6 = 0, p_1 = 1$  will not be tried again.

$$c_1 = \neg p_1 \vee p_2$$

$$c_2 = \neg p_1 \vee p_3 \vee p_5$$

$$c_3 = \neg p_2 \vee p_4$$

$$c_4 = \neg p_3 \vee \neg p_4$$

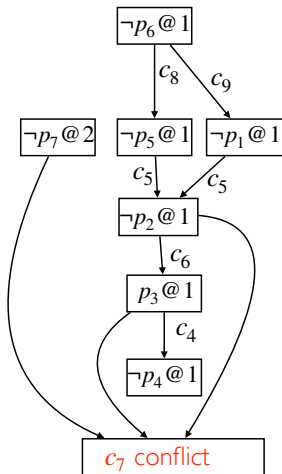
$$c_5 = p_1 \vee p_5 \vee \neg p_2$$

$$c_6 = p_2 \vee p_3$$

$$c_7 = p_2 \vee \neg p_3 \vee p_7$$

$$c_8 = p_6 \vee \neg p_5$$

$$c_9 = p_6 \vee \neg p_1$$



Conflict clause :  $p_7 \vee p_6$  is added and backtrack

Set  $p_7 = 1$  by unit propagation.

$$c_1 = \neg p_1 \vee p_2$$

$$c_2 = \neg p_1 \vee p_3 \vee p_5$$

$$c_3 = \neg p_2 \vee p_4$$

$$c_4 = \neg p_3 \vee \neg p_4$$

$$c_5 = p_1 \vee p_5 \vee \neg p_2$$

$$c_6 = p_2 \vee p_3$$

$$c_7 = p_2 \vee \neg p_3 \vee p_7$$

$$c_8 = p_6 \vee \neg p_5$$

$$c_9 = p_6 \vee \neg p_1$$

$$c_{10} = p_6 \vee p_7$$

