CSC410
SAT Solving
Inverted Lecture

Azadeh Farzan

Fall 2026

How do we decide if a propositional logic formula is satisfiable or unsatisfiable?

Example

$$p \wedge q \wedge (\neg p \vee \neg q)$$

Satisfiable or unsatisfiable?

A Complete Proof System

Inference Rules

Premises

Conclusion

$$\frac{\alpha_1 \quad \alpha_2 \quad \dots \quad \alpha_n}{\beta}$$

Examples:

$$\frac{\alpha \quad \beta}{(\alpha \wedge \beta)}$$

$$\frac{(\alpha \wedge \beta)}{(\alpha \vee \beta)}$$

What is a proof?

Start with some premises and using the rules, derive a conclusion!

Binary Resolution Rule

$$\frac{(\alpha \vee p) \quad ((\neg p) \vee \beta)}{(\alpha \vee \beta)}$$

- It works with formulas in CNF forms.
- It is used to prove contradictions.

Important Properties

Theorem. If from a set of clauses as a premise, we derive *false*, then the set of clauses is unsatisfiable.

Theorem. If there is no proof leading to *false*, then the set of clauses is satisfiable.

Conjunctive Normal Form

- $\big((p \vee (\neg q)) \wedge r \wedge ((\neg r) \vee p \vee q)\big)$
- $\big((\neg r) \vee p \vee q\big)$ (Only one clause.)
- $\big((\neg r) \wedge p \wedge q\big)$ (Three singleton clauses.)

Converting to CNF

1. Eliminate implication and equivalence.
   Replace $(\alpha \to \beta)$ by $((\neg\alpha) \vee \beta)$
   Replace $(\alpha \leftrightarrow \beta)$ by $((\neg\alpha) \vee \beta) \wedge (\alpha \vee (\neg\beta))$.
   (*Now only* $\wedge$, $\vee$ *and* $\neg$ *appear as connectives.*)
2. Apply De Morgan's and double-negation laws as often as possible.
   Replace $(\neg(\alpha \vee \beta))$ by $((\neg\alpha) \wedge (\neg\beta))$.
   Replace $(\neg(\alpha \wedge \beta))$ by $((\neg\alpha) \vee (\neg\beta))$.
   Replace $(\neg(\neg\alpha))$ by α.
   (*Now negation only occurs in literals.*)
3. Transform into a conjunction of clauses using distributivity.
   Replace $(\alpha \vee (\beta \wedge \gamma))$ by $((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$.
   (*One could stop here, but....*)
4. Simplify using idempotence, contradiction, excluded middle and Simplification I & II.

Example

$$p \wedge q \wedge (\neg p \vee \neg q)$$

Example

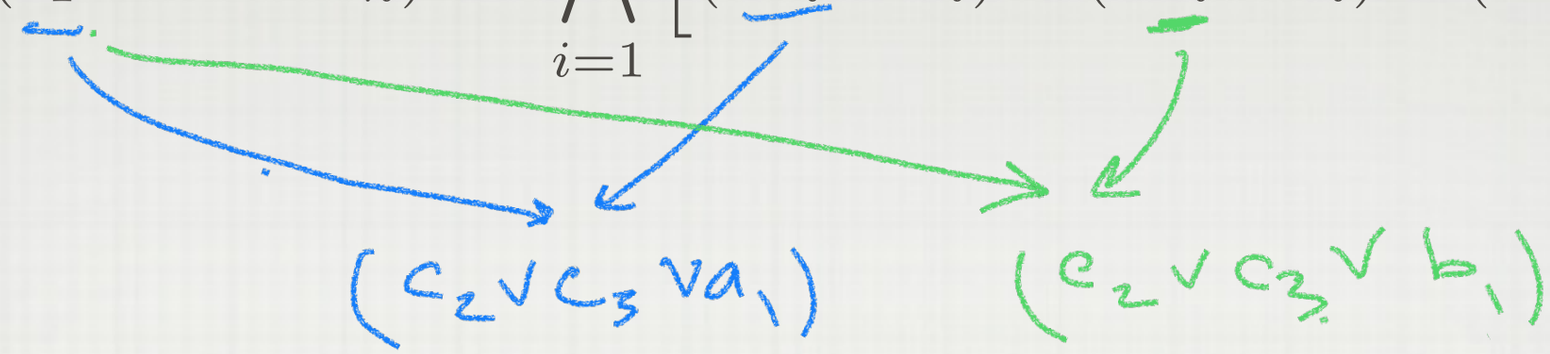
$$(p \vee q \vee r) \wedge (\neg p \vee q) \wedge (\neg q \vee \neg r)$$

Davis-Putnam Algorithm

- One can have an algorithm by exploring all applicable rules.
- The catch: it explodes and runs out of memory easily!

$$F_n \;=\; (c_1 \vee \cdots \vee c_n) \;\wedge\; \bigwedge_{i=1}^{n} \Big[(\neg c_i \vee a_i) \wedge (\neg c_i \vee b_i) \wedge (\neg a_i \vee \neg b_i) \Big]$$

$$F_n \;=\; (c_1 \vee \cdots \vee c_n) \;\wedge\; \bigwedge_{i=1}^{n} \Big[(\neg c_i \vee a_i) \wedge (\neg c_i \vee b_i) \wedge (\neg a_i \vee \neg b_i) \Big]$$



Consider F_3:

1. Eliminate c_1

$$(a_1 \vee c_2 \vee c_3) \;\wedge\; (b_1 \vee c_2 \vee c_3) \;\rightarrow\; 2$$

2. Eliminate c_2

$$4 \;\leftarrow\; (a_1 \vee a_2 \vee c_3) \wedge (a_1 \vee b_2 \vee c_3) \wedge (b_1 \vee a_2 \vee c_3) \wedge (b_1 \vee b_2 \vee c_3)$$

3. Eliminate c_3

$$\bigwedge_{x_i \in \{a_i, b_i\}} (x_1 \vee x_2 \vee x_3) \;\rightarrow\; 8$$

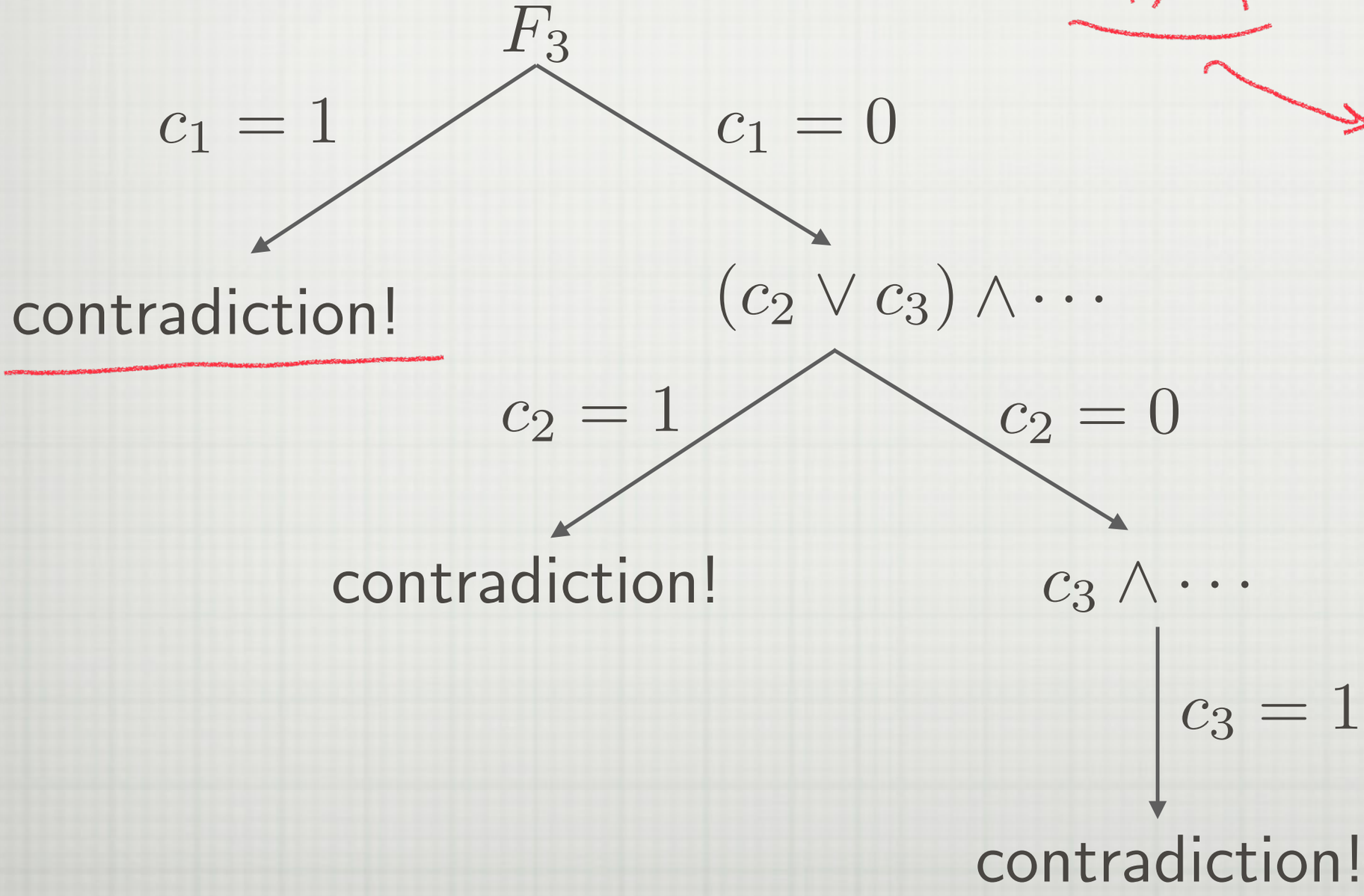
2^n clauses after eliminating c_n, from a linear-sized input.

Davis, Logemann and Loveland
replaced elimination with
splitting on a variable,
to avoid this memory explosion!

$$F_n \;=\; (c_1 \vee \cdots \vee c_n) \;\wedge\; \bigwedge_{i=1}^{n} \Big[(\neg c_i \vee a_i) \wedge (\neg c_i \vee b_i) \wedge (\neg a_i \vee \neg b_i) \Big]$$

a_1, b_1

$\bot$



Splitting saves on space, but not necessarily on time!

$$F_n \;=\; (c_1 \vee \cdots \vee c_n) \;\wedge\; \bigwedge_{i=1}^{n} \Big[(\neg c_i \vee a_i) \wedge (\neg c_i \vee b_i) \wedge (\neg a_i \vee \neg b_i)\Big]$$

1. Eliminate a_i

$$(\neg c_i \vee a_i),\ (\neg a_i \vee \neg b_i) \implies (\neg c_i \vee \neg b_i)$$

2. Eliminate b_i

$$(\neg c_i \vee b_i),\ (\neg c_i \vee \neg b_i) \implies (\neg c_i)$$

3. Eliminate c_1, c_2, c_3

$$(c_1 \vee c_2 \vee c_3), (\neg c_1) \Rightarrow (c_2 \vee c_3); \quad (c_2 \vee c_3), (\neg c_2) \Rightarrow (c_3); \quad (c_3), (\neg c_3) \Rightarrow \bot$$

With this order, elimination works well!

But splitting has exponential time!

Which led to CDCL
as a hybrid solution ...

Do splitting!

But retain memory of your past
through conflict clauses!

Binary Resolution Rule in CDCL

$$\frac{(\alpha \vee p) \quad ((\neg p) \vee \beta)}{(\alpha \vee \beta)}$$

Unit resolution:

$$\frac{(\alpha \vee p) \quad (\neg p)}{\alpha}$$

Contradiction:

$$\frac{p \quad (\neg p)}{\bot}$$

And, the standard rule used in conflict clause learning.