

COL:750/7250

Foundations of Automatic Verification

Instructor: Priyanka Golia

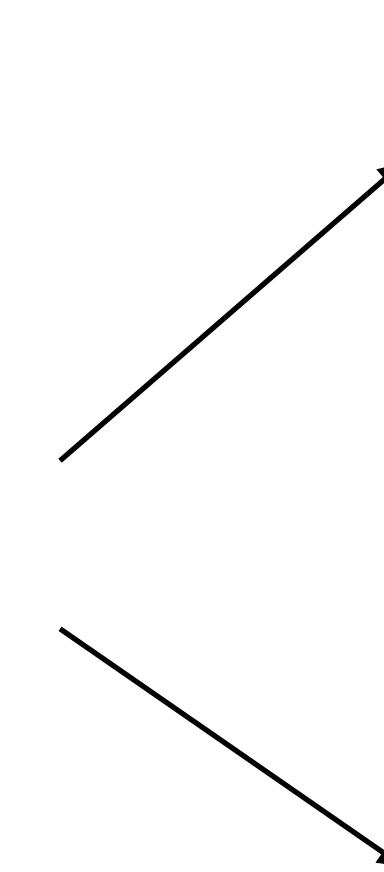
Course Webpage



<https://priyanka-golia.github.io/teaching/COL-750-COL7250/index.html>

Boolean
/propositional
formulas

---> SAT Solvers



If formula is **SAT**isfiable, gives an satisfying
assignment

UNSAT

Resolution Refutation

List of clauses C_1, C_2, \dots, C_t is a **resolution refutation** of formula F_{CNF} if:

1. C_t is empty \square
2. $C_k \in F_{CNF}$ or C_k is derived using **resolution** from C_i and C_j , where $i, j < k$

$Models(F) = \emptyset$
F is UNSAT

\downarrow

$$C_i = p \vee \alpha$$
$$C_j = \neg p \vee \beta$$

Then,

$$C_k = \alpha \vee \beta$$

C_k is derived from C_i, C_j

Resolution Refutation

$$F = (\neg p \vee \neg q \vee r) \wedge (\neg p \vee q) \wedge (p) \wedge (\neg r)$$
$$\frac{}{C_1} \quad \frac{}{C_2} \quad \frac{}{C_3} \quad \frac{}{C_4}$$

Resolution on C_1, C_3 $\frac{(\neg p \vee \neg q \vee r) \wedge (p)}{C_5 : (\neg q \vee r)}$

Resolution on C_2, C_3 $\frac{(\neg p \vee q) \wedge (p)}{C_6 : (q)}$

Resolution on C_5, C_4 $\frac{(\neg q \vee r) \wedge (\neg r)}{C_7 : (\neg q)}$

Resolution on C_6, C_7 $\frac{(q) \wedge (\neg q)}{C_8 : \square}$

List of clauses C_1, C_2, \dots, C_8 is a resolution refutation of F

SAT Solving using Resolution

1. Start with F_{CNF}
2. Perform Resolution until
 1. Empty clause is derived \rightarrow return UNSAT
 2. No further resolution is possible \rightarrow return SAT

One of these two cases will occur — resolution is sound and complete.

Resolution Refutation

Thm: A formula F_{CNF} is refutable if and only if F_{CNF} is unsatisfiable

→ direction is easy to see: if F_{CNF} is refutable then F_{CNF} is unsatisfiable.

HW:

← direction: if F_{CNF} is unsatisfiable then F_{CNF} is refutable

Hint: Induction on # of propositional variables.

Bottleneck of Resolution Refutation

Space required to perform Resolutions:

1. At every resolution step: $\binom{m}{2}$ new clauses are added to the formula, where m is number of clauses.
2. This is done linear many times ($O(\text{Vars}(F))$ many), hence over growth can be exponential.
3. Resolution is EXPSPACE.

DP algorithm for SAT Solving (Martin Davis - Hilary Putnam 1960)

1. Start with F_{CNF}
2. Pick a literal l that occurs with both polarities in F_{CNF} .
3. For every clause C in F_{CNF} containing l and every clause C' in F_{CNF} containing its negation $\neg l$ perform resolution
 1. $r = (C \setminus \{l\}) \cup (C' \setminus \{\neg l\})$
 2. $F_{CNF} \leftarrow add_to_formula(r, F_{CNF})$
4. For every clause C that contains l or $\neg l$ do
 1. $F_{CNF} \leftarrow remove_from_formula(C, F_{CNF})$

DP algorithm for SAT Solving (Martin Davis - Hilary Putnam 1960)

1. Start with F_{CNF}
2. Pick a literal l that occurs with both polarities in F_{CNF} .
$$F_{CNF} \leftarrow \text{Resolution}(C, l, F_{CNF})$$
3. For every clause C in F_{CNF} containing l and every clause C' in F_{CNF} containing its negation $\neg l$ perform resolution
 1. $r = (C \setminus \{l\}) \cup (C' \setminus \{\neg l\})$
 2. $F_{CNF} \leftarrow \text{add_to_formula}(r, F_{CNF})$
4. For every clause C that contains l or $\neg l$ do
 1. $F_{CNF} \leftarrow \text{remove_from_formula}(C, F_{CNF})$

DP algorithm for SAT Solving (Martin Davis - Hilary Putnam 1960)

1. Start with F_{CNF}
2. Pick a literal l that occurs with both polarities in F_{CNF} in different clauses :
 1. $F_{CNF} \leftarrow \text{Resolution}(C, l, F_{CNF})$
4. For every clause C that contains l or $\neg l$ do
 1. $F_{CNF} \leftarrow \text{remove_from_formula}(C, F_{CNF})$

DP algorithm for SAT Solving (Martin Davis - Hilary Putnam 1960)

1. Start with F_{CNF}
2. If F_{CNF} has empty clause then
 1. Return UNSAT
3. If $\exists l$ that occurs with both polarities in different clauses in F_{CNF}
 1. Return SAT
3. Pick a literal l that occurs with both polarities in F_{CNF} in different clauses
 1. $F_{CNF} \leftarrow \text{Resolution}(C, l, F_{CNF})$
4. For every clause C that contains l or $\neg l$ do :
 1. $F_{CNF} \leftarrow \text{remove_from_formula}(C, F_{CNF})$

Is this correct?

How about $(p \vee \neg p)$

DP algorithm for SAT Solving (Martin Davis - Hilary Putnam 1960)

1. Start with F_{CNF}
2. For every clause C in F_{CNF} that contains both l and $\neg l$ do:
 1. $F_{CNF} \leftarrow \text{remove_from_formula}(C, F_{CNF})$
3. If F_{CNF} is empty
 1. Return SAT
4. If F_{CNF} has empty clause then
 1. Return UNSAT
5. If $\exists l$ that occurs with both polarities in different clauses in F_{CNF}
 1. Return SAT
6. Pick a literal l that occurs with both polarities in F_{CNF} in different clauses:
 1. $F_{CNF} \leftarrow \text{Resolution}(C, l, F_{CNF})$
7. For every clause C that contains l or $\neg l$ do :
 1. $F_{CNF} \leftarrow \text{remove_from_formula}(C, F_{CNF})$

DP algorithm for SAT Solving (Martin Davis - Hilary Putnam 1960)

1. Start with F_{CNF}
2. For every clause C in F_{CNF} that contains both l and $\neg l$ do:

1. $F_{CNF} \leftarrow \text{remove_from_formula}(C, F_{CNF})$

3. If F_{CNF} is empty
 1. Return SAT
4. If F_{CNF} has empty clause then
 1. Return UNSAT
5. If $\exists l$ that occurs with both polarities in different clauses in F_{CNF}
 1. Return SAT

Can we do better?

6. Pick a literal l that occurs with both polarities in F_{CNF} .
 1. $F_{CNF} \leftarrow \text{Resolution}(C, l, F_{CNF})$
7. For every clause C that contains l or $\neg l$ do :
 1. $F_{CNF} \leftarrow \text{remove_from_formula}(C, F_{CNF})$

Pure Literal Elimination

Pure literal: a literal l all of which occurrences in F have the same polarity.

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

$(\boxed{p} \vee q \vee r) \wedge (\neg q \vee r) \wedge (\boxed{p} \vee \neg r) \wedge (\boxed{p} \vee \neg q)$

Literal p has positive polarity in all occurrence in F . P is pure literal.

$(p \vee \neg q \vee r) \wedge (\neg q \vee r) \wedge (\neg p \vee \neg r) \wedge (p \vee \neg q) - \neg q$ is pure literal

Pure Literal Elimination

Pure literal: a literal l all of which occurrences in F have the same polarity.

For every clause that contains a pure literal:

$$F_{CNF} \leftarrow \text{remove_from_formula}(C, F_{CNF})$$

DP algorithm for SAT Solving (Martin Davis - Hilary Putnam 1960)

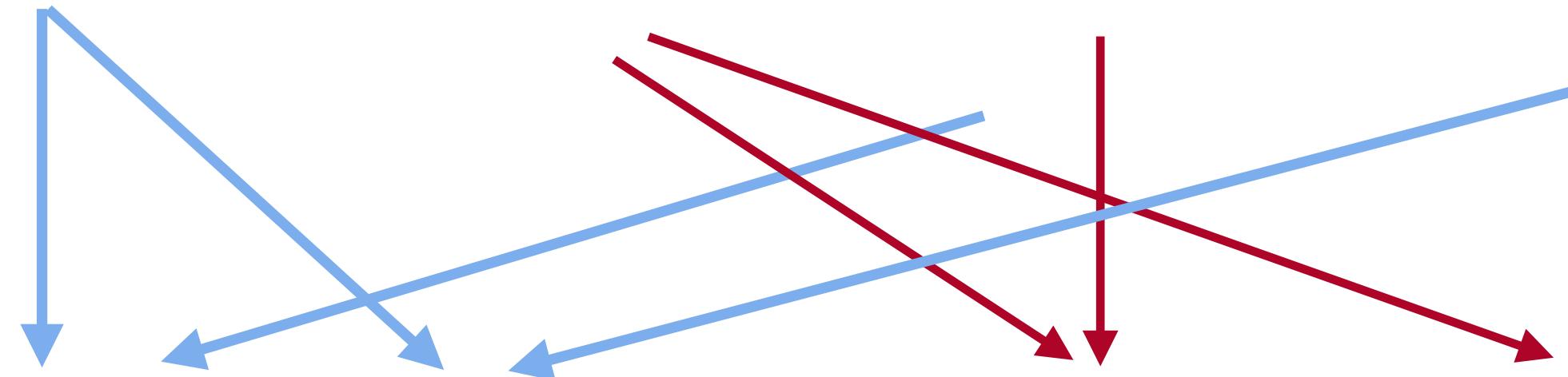
1. Start with F_{CNF}
2. For every clause C in F_{CNF} that either contains both l and $\neg l$ or has pure literal do:
 1. $F_{CNF} \leftarrow \text{remove_from_formula}(C, F_{CNF})$
3. If F_{CNF} is empty
 1. Return SAT
4. If F_{CNF} has empty clause then
 1. Return UNSAT
5. If $\exists l$ that occurs with both polarities in different clauses in F_{CNF}
 1. Return SAT
6. Pick a literal l that occurs with both polarities in F_{CNF} .
 1. $F_{CNF} \leftarrow \text{Resolution}(C, l, F_{CNF})$
7. For every clause C that contains l or $\neg l$ do :
 1. $F_{CNF} \leftarrow \text{remove_from_formula}(C, F_{CNF})$

DP algorithm for SAT Solving (Martin Davis - Hilary Putnam 1960)

1. Start with F_{CNF}
2. For every clause C in F_{CNF} that either contains both l and $\neg l$ or has pure literal do:
 1. $F_{CNF} \leftarrow \text{remove_from_formula}(C, F_{CNF})$
3. If F_{CNF} is empty
 1. Return SAT
4. If F_{CNF} has empty clause then
 1. Return UNSAT
5. Pick a literal l that occurs with both polarities in F_{CNF} .
 1. $F_{CNF} \leftarrow \text{Resolution}(C, l, F_{CNF})$
6. For every clause C that contains l or $\neg l$ do :
 1. $F_{CNF} \leftarrow \text{remove_from_formula}(C, F_{CNF})$

DP algorithm

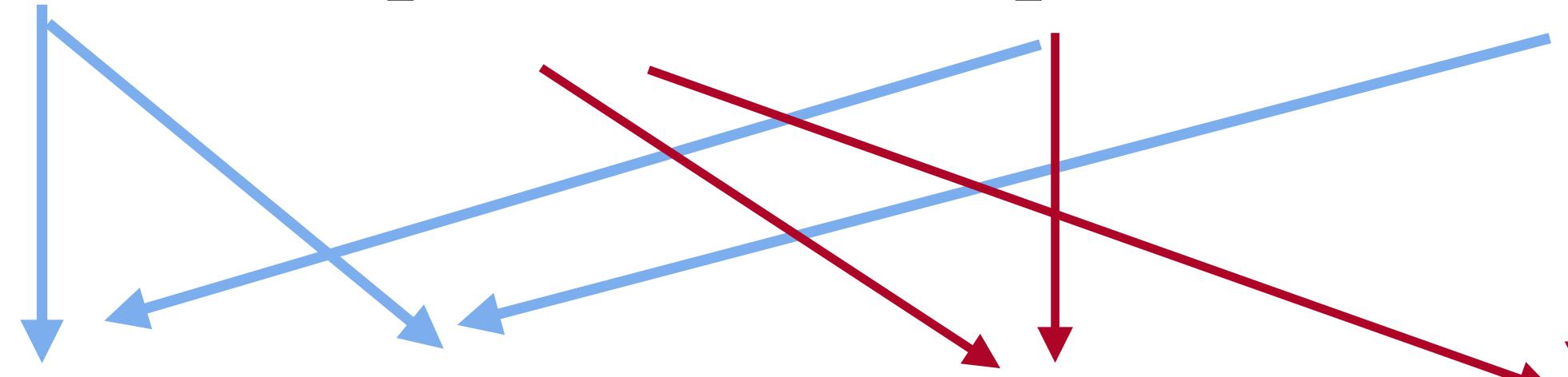
$$F = (p \vee q) \wedge (p \vee \neg q) \wedge (\neg p \vee r) \wedge (\neg p \vee \neg r)$$



* No pure literal, no clause with $l \vee \neg l$

Pick literal p

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



* No pure literal, no clause with $l \vee \neg l$

Pick literal q

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

$$(r) \wedge (\neg r)$$



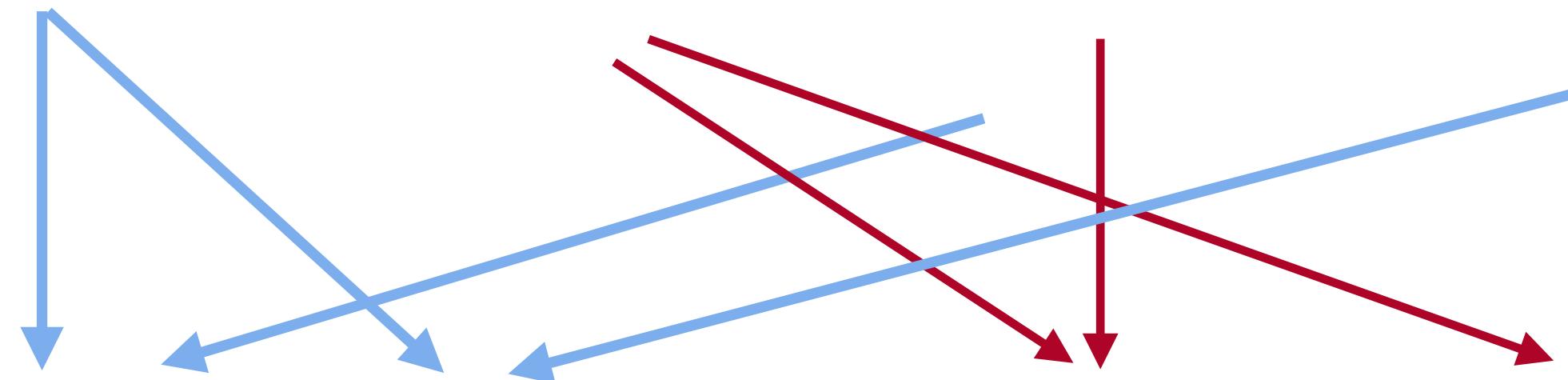
*remove clauses with $l \vee \neg l$

Pick literal r

F has empty clause – UNSAT

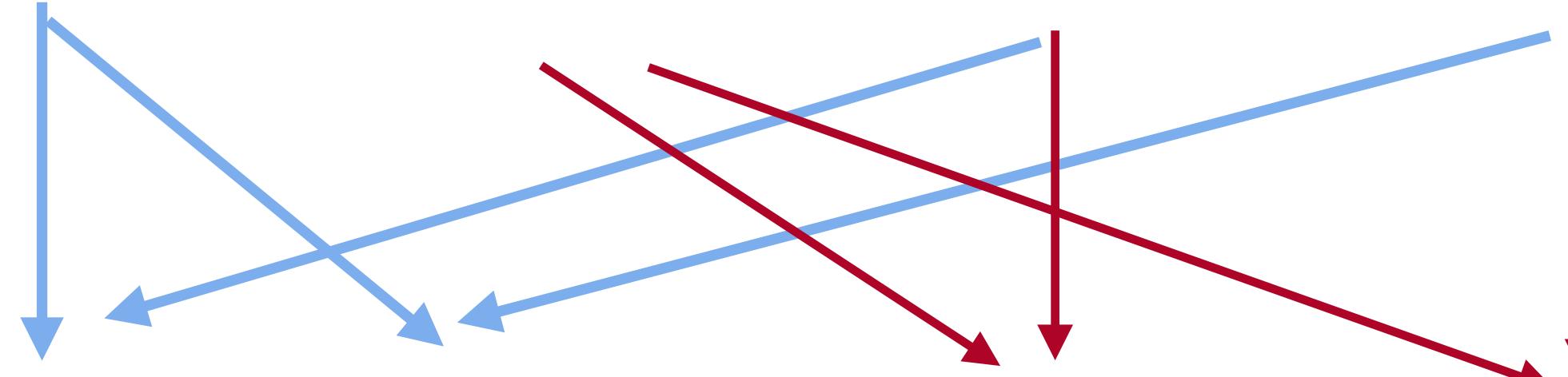
DP algorithm

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



* No pure literal, no clause with $l \vee \neg l$
Pick literal p

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



* No pure literal, no clause with $l \vee \neg l$
Pick literal q

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

$$(r) \wedge (\neg r)$$



*remove clauses with $l \vee \neg l$

Pick literal r

F has empty clause – UNSAT

DP algorithm

$$F = (p \vee q \vee r) \wedge (q \vee \neg r \vee \neg s) \wedge (\neg q \vee s) \wedge (\neg p \vee \neg s)$$

$$F = (p \vee q) \wedge (p \vee \neg q) \wedge (\neg p \vee r) \wedge (\neg p \vee \neg r) \quad \text{Unit clause}$$

p has to take value 0, $(\neg p \vee r) \wedge (\neg p \vee \neg r)$ are True

Can we remove all clauses that have $\neg p$?

$$(\neg p) \wedge (p \vee q) \equiv_{SAT} q$$

$$(\neg p) \wedge (p \vee \neg q) \equiv_{SAT} \neg q$$

Unit Propagation

While F contains a unit clause (l) do:

For every clause C in F that has l do:

$$F_{CNF} \leftarrow \text{remove_from_formula}(C, F_{CNF})$$

For every clause C in F that has $\neg l$ do:

$$F_{CNF} \leftarrow \text{remove_from_formula}(C, F_{CNF})$$

$$F_{CNF} \leftarrow \text{add_to_formula}(C \setminus \neg l, F_{CNF})$$

DP algorithm for SAT Solving (Martin Davis - Hilary Putnam 1960)

1. Start with F_{CNF}
2. For every clause C in F_{CNF} that either contains both l and $\neg l$ or has pure literal do:
 1. $F_{CNF} \leftarrow \text{remove_from_formula}(C, F_{CNF})$
3. $F_{CNF} \leftarrow \text{UnitPropagation}(F_{CNF})$
4. If F_{CNF} is empty
 1. Return SAT
5. If F_{CNF} has empty clause then
 1. Return UNSAT
6. Pick a literal l that occurs with both polarities in F_{CNF} .
 1. $F_{CNF} \leftarrow \text{Resolution}(C, l, F_{CNF})$
7. For every clause C that contains l or $\neg l$ do :
 1. $F_{CNF} \leftarrow \text{remove_from_formula}(C, F_{CNF})$

DPLL algorithm (Davis -Putnam-Logemann-Loveland 1960)

Complete and Sound algorithm & takes linear space in worst case.

Still the basis of SAT solver

zChaff Solver – efficient implementation of DPLL.

Won test of time award at CAV 2001.