C06 – SAT solvers

Program Verification

FMI · Denisa Diaconescu · Spring 2022

CNF - Conjunctive normal form

The SAT problem

The SAT problem:

Given a propositional formula with n variables,
can we find an interpretation to make the formula true?

A SAT solver is a program that automatically decides whether a propositional formula is satisfiable (i.e, answers the SAT problem).

If it is satisfiable, a SAT solver will produce an example of an interpretation that satisfies the formula.

CNF - Conjunctive normal form

All current fast SAT solvers work on CNF (or slightly generalized CNF).

- A literal is a propositional variable or its negation
  - example: p, $\neg q$
  - For a literal l we write $\sim l$ for the negation of l cancelling double negations
- A clause is a disjunction of literals
  - example: $p \vee \neg q \vee r$
  - Since $\vee$ is associative, we can represent clauses as lists of literals.
  - The empty clause (0 disjuncts) is defined to be $\perp$
  - A unit clause is a clause consisting of exactly one literal.
- A formula is in CNF if it is a conjuction of clauses
  - example: $(p \vee \neg q \vee r) \wedge (\neg p \vee s \vee t \vee \neg u)$
  - Since $\wedge$ is associative, we can represent formulas in CNF as lists of clauses.
  - The empty conjunction is defined to be $\top$

Conversion to CNF

Any propositional formula can be transformed into an equivalent formula in CNF (need not be unique!).

Two formulas are equivalent if they are satisfied by the same interpretations.

Conversion to CNF

Any propositional formula can be transformed into an equivalent formula in CNF (need not be unique!).

Two formulas are equivalent if they are satisfied by the same interpretations.

Example
The formula p is equivalent with the following formulas in CNF:

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

Conversion to CNF

We can rewrite the formula directly via the following equivalences:

- Remove implications: rewrite $A \rightarrow B$ to $\neg A \vee B$
- Push all negations inwards:
  - rewrite $\neg(A \vee B)$ to $\neg A \wedge \neg B$
  - rewrite $\neg(A \wedge B)$ to $\neg A \vee \neg B$
- Remove double negations: rewrite $\neg\neg A$ to A
- Eliminate $\top$ and $\bot$:
  - rewrite $A \vee \bot$ to A
  - remove clauses containing $\top$
- Distribute disjunctions over conjunctions: rewrite $A \vee (B \wedge C)$ to $(A \vee B) \wedge (A \vee C)$

Conversion to CNF

Example

Applying the above rules to the formula

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

we obtain the equivalent formula in CNF:

Conversion to CNF

Example

Applying the above rules to the formula

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

we obtain the equivalent formula in CNF:

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

Conversion to CNF

Example

Applying the above rules to the formula

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

we obtain the equivalent formula in CNF:

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

We can further simplify:

- Remove duplicate clauses, duplicate literals from clauses
- Remove clauses in which a literal is both positive and negative
- In fact, each variable need only occur in each clause at most once!

Example

If we simply the CNF formula from the above example we get

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

CNF and Validity

Theorem

A clause $L_1 \vee \ldots \vee L_m$ is valid iff there are $1 \leq i, j \leq m$ such that L_i is $\neg L_j$.

CNF and Validity

Theorem

A clause $L_1 \vee \ldots \vee L_m$ is valid iff there are $1 \leq i, j \leq m$ such that L_i is $\neg L_j$.

Checking validity for formulas in CNF is very easy! For each clause of the formula, check if it contains a literal and its negation.

Example

- $(\neg q \vee p \vee r) \wedge (\neg p \vee r) \wedge q$ is not valid
- $(\neg q \vee p \vee q) \wedge (\neg p \vee p)$ is valid

CNF and Validity

Theorem

A clause $L_1 \vee \ldots \vee L_m$ is valid iff there are $1 \leq i, j \leq m$ such that L_i is $\neg L_j$.

Checking validity for formulas in CNF is very easy! For each clause of the formula, check if it contains a literal and its negation.

Example

- $(\neg q \vee p \vee r) \wedge (\neg p \vee r) \wedge q$ is not valid
- $(\neg q \vee p \vee q) \wedge (\neg p \vee p)$ is valid

Satisfiability is not so easy!

$\boxed{\varphi \text{ satisfiable} \quad \text{iff} \quad \neg\varphi \text{ is not valid}}$

Conversion to CNF

- The previous method to transform a formula into an equivalent one in CNF can blow-up exponentially!
- There exist transformations into CNF that avoid an exponential increase in size by preserving satisfiability rather than equivalence.
- These transformations are guaranteed to only linearly increase the size of the formula, but introduce new variables (e.g., Tseitin transformation).
- Two formulas are equisatisfiable if either both formulas are satisfiable or both are not
  - Equisatisfiable formulas may disagree for a particular choice of variables.

SAT solvers algorithms

Davis-Putnam algorithm

- First attempt at a better-than-brute-force SAT algorithm (1960)
  - Original algorithm tackles first-order logic
  - We present the propositional case
- We assume as input a formula A in CNF
  - a set of clauses
  - a set of sets of literals
- The DP algorithm rewrites the set of clauses until
  - A is $\top$ (the set is empty) then returns sat, or
  - A contains an empty clause $\bot$ return unsat

Resolution rule

$$\boxed{\dfrac{p \vee \alpha \quad \neg p \vee \beta}{\alpha \vee \beta} \quad \textit{resolution}}$$

$$\boxed{\dfrac{p \vee p \vee \alpha}{p \vee \alpha} \quad \textit{merging}}$$

Resolution rule

$$\boxed{\frac{p \vee \alpha \quad \neg p \vee \beta}{\alpha \vee \beta} \quad \textit{resolution}}$$

$$\boxed{\frac{p \vee p \vee \alpha}{p \vee \alpha} \quad \textit{merging}}$$

Example

$$\frac{x_1 \vee x_2 \vee x_3 \quad x_1 \vee \neg x_2 \vee x_4}{x_1 \vee x_1 \vee x_3 \vee x_4} \qquad \frac{x_1 \vee x_1 \vee x_3 \vee x_4}{x_1 \vee x_3 \vee x_4}$$

Resolution rule

If a variable p occurs both positively and negatively in clauses of A:

- Let $C_{pos} = \{A_1 \vee p, A_2 \vee p, \ldots\}$ be the clauses in A in which p occurs positively
- Let $C_{neg} = \{B_1 \vee \neg p, B_2 \vee \neg p, \ldots\}$ be the clauses in A in which p occurs negatively
- Remove these two sets of clauses from A, and replace them with the new set

$$\{A_i \vee B_j \mid A_i \vee p \in C_{pos}, B_j \vee \neg p \in C_{neg}\}$$

Davis-Putnam algorithm

- Iteratively apply the following steps:
  - Select variable x
  - Apply resolution rule between every pair of clauses of the form $(x \vee \alpha)$ and $(\neg x \vee \beta)$
  - Remove all clauses containing either x or $\neg x$
- Terminate if
  - The empty formula is derived ($\top$) and then return sat, or
  - An empty clause is derived ($\bot$) and then return unsat

Davis-Putnam algorithm

Example

1. $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3) \wedge (x_3 \vee x_4) \wedge (x_3 \vee \neg x_4)$

Davis-Putnam algorithm

Example

1. $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3) \wedge (x_3 \vee x_4) \wedge (x_3 \vee \neg x_4)$
2. $(\neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3) \wedge (x_3 \vee x_4) \wedge (x_3 \vee \neg x_4)$

Davis-Putnam algorithm

Example

1. $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3) \wedge (x_3 \vee x_4) \wedge (x_3 \vee \neg x_4)$
2. $(\neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3) \wedge (x_3 \vee x_4) \wedge (x_3 \vee \neg x_4)$
3. $(\neg x_3 \vee x_3) \wedge (x_3 \vee x_4) \wedge (x_3 \vee \neg x_4)$

Davis-Putnam algorithm

Example

1. $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3) \wedge (x_3 \vee x_4) \wedge (x_3 \vee \neg x_4)$
2. $(\neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3) \wedge (x_3 \vee x_4) \wedge (x_3 \vee \neg x_4)$
3. $(\neg x_3 \vee x_3) \wedge (x_3 \vee x_4) \wedge (x_3 \vee \neg x_4)$
4. $(\neg x_3 \vee x_3) \wedge (x_3)$

Example

1. $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3) \wedge (x_3 \vee x_4) \wedge (x_3 \vee \neg x_4)$
2. $(\neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3) \wedge (x_3 \vee x_4) \wedge (x_3 \vee \neg x_4)$
3. $(\neg x_3 \vee x_3) \wedge (x_3 \vee x_4) \wedge (x_3 \vee \neg x_4)$
4. $(\neg x_3 \vee x_3) \wedge (x_3)$
5. $\top$

Formula is SAT

Davis-Putnam algorithm

Example

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

Davis-Putnam algorithm

Example

1. $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (x_2) \wedge (\neg x_2 \vee x_3)$
2. $(\neg x_2 \vee \neg x_3) \wedge (x_2) \wedge (\neg x_2 \vee x_3)$

Davis-Putnam algorithm

Example

1. $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (x_2) \wedge (\neg x_2 \vee x_3)$
2. $(\neg x_2 \vee \neg x_3) \wedge (x_2) \wedge (\neg x_2 \vee x_3)$
3. $(\neg x_3) \wedge (x_3)$

Example

1. $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (x_2) \wedge (\neg x_2 \vee x_3)$
2. $(\neg x_2 \vee \neg x_3) \wedge (x_2) \wedge (\neg x_2 \vee x_3)$
3. $(\neg x_3) \wedge (x_3)$
4. $\perp$

Formula is UNSAT

Davis-Putnam algorithm

Main issues of the approach:

- In which order should the resolution steps be performed?
- In which order the variables should be selected? (variable elimination)
- Worst-case exponential in memory consumption!

Faster algorithms

- Davis-Putnam algorithm: the refinements
  - Add specific cases to order variable elimination steps.
  - The pure literal rule and the unit propagation rule

Faster algorithms

- Davis-Putnam algorithm: the refinements
  - Add specific cases to order variable elimination steps.
  - The pure literal rule and the unit propagation rule
- Davis-Putnam-Logemann-Loveland algorithm
  - Standard backtrack search
  - space efficient DP

Faster algorithms

- Davis-Putnam algorithm: the refinements
  - Add specific cases to order variable elimination steps.
  - The pure literal rule and the unit propagation rule
- Davis-Putnam-Logemann-Loveland algorithm
  - Standard backtrack search
  - space efficient DP
- Conflict-Driven Clause Learning algorithm
  - An extension of DPLL with:
    - Clause learning
    - Non-chronological backtracking
  - Clause learning can be performed with various strategies
  - CDCL algorithms are use in almost all modern SAT solvers

Davis-Putnam algorithm: the refinements

Add specific cases to order variable elimination steps.

- Iteratively apply the following steps:
  - Apply the pure literal rule and unit propagation
  - Select variable x
  - Apply resolution rule between every pair of clauses of the form $(x \vee \alpha)$ and $(\neg x \vee \beta)$
  - Remove all clauses containing either x or $\neg x$
- Terminate if
  - The empty formula is derived $(\top)$ and then return sat, or
  - An empty clause is derived $(\bot)$ and then return unsat

Pure literal rule

If a variable p occurs either only positively or only negatively in A,
delete all clauses of A in which p occurs.

- A literal is pure if occurs only positively or negatively in a CNF formula
- Pure literal rule: eliminate first pure literals since no resolvant are produced!
- Applying a variable elimination step on a pure literal strictly reduced the number of clauses!
- Preserves satisfiability, not logical equivalence!

Pure literal rule

If a variable p occurs either only positively or only negatively in A,
delete all clauses of A in which p occurs.

- A literal is pure if occurs only positively or negatively in a CNF formula
- Pure literal rule: eliminate first pure literals since no resolvant are produced!
- Applying a variable elimination step on a pure literal strictly reduced the number of clauses!
- Preserves satisfiability, not logical equivalence!

Example
In $(\neg x_1 \vee x_2) \wedge (x_3 \vee \neg x_2) \wedge (x_4 \vee \neg x_5) \wedge (x_5 \vee \neg x_4)$ the pure literals are $\neg x_1$ and x_3.

Unit propagation rule

If l is a unit clause in A, then update A by:

- removing all clauses which have l as a disjunct, and
- updating all clauses in A containing $\sim l$ as a disjunct by removing that disjunct

- Specific case of resolution
- Only shorten clauses!
- a.k.a. Boolean constraint propagation or BCP
- Is arguably the key component to fast SAT solving
- Since clauses are shortened, new unit clauses may appear. Empty clauses also!
- Apply unit propagation while new unit clauses are produced.
- Preserves logical equivalence!

DP algorithm

Example

1. $p \land (\neg p \lor q) \land (\neg q \lor r) \land (\neg r \lor s \lor t)$

DP algorithm

Example

1. $p \wedge (\neg p \vee q) \wedge (\neg q \vee r) \wedge (\neg r \vee s \vee t)$

We apply the Pure literal rule for s and t and we delete the last clause.

DP algorithm

Example

1. $p \wedge (\neg p \vee q) \wedge (\neg q \vee r) \wedge (\neg r \vee s \vee t)$

   We apply the Pure literal rule for s and t and we delete the last clause.

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

DP algorithm

Example

1. $p \wedge (\neg p \vee q) \wedge (\neg q \vee r) \wedge (\neg r \vee s \vee t)$

   We apply the Pure literal rule for s and t and we delete the last clause.

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

   We apply the Unit propagation rule for p.

DP algorithm

Example

1. $p \wedge (\neg p \vee q) \wedge (\neg q \vee r) \wedge (\neg r \vee s \vee t)$

   We apply the Pure literal rule for s and t and we delete the last clause.

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

   We apply the Unit propagation rule for p.

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

DP algorithm

Example

1. $p \wedge (\neg p \vee q) \wedge (\neg q \vee r) \wedge (\neg r \vee s \vee t)$

   We apply the Pure literal rule for s and t and we delete the last clause.

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

   We apply the Unit propagation rule for p.

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

   We apply the Unit propagation rule for q.

DP algorithm

Example

1. $p \wedge (\neg p \vee q) \wedge (\neg q \vee r) \wedge (\neg r \vee s \vee t)$

   We apply the Pure literal rule for s and t and we delete the last clause.

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

   We apply the Unit propagation rule for p.

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

   We apply the Unit propagation rule for q.

4. r

DP algorithm

Example

1. $p \wedge (\neg p \vee q) \wedge (\neg q \vee r) \wedge (\neg r \vee s \vee t)$

   We apply the Pure literal rule for s and t and we delete the last clause.

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

   We apply the Unit propagation rule for p.

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

   We apply the Unit propagation rule for q.

4. r

   We apply the Unit propagation rule for r.

DP algorithm

Example

1. $p \wedge (\neg p \vee q) \wedge (\neg q \vee r) \wedge (\neg r \vee s \vee t)$

   We apply the Pure literal rule for s and t and we delete the last clause.

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

   We apply the Unit propagation rule for p.

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

   We apply the Unit propagation rule for q.

4. r

   We apply the Unit propagation rule for r.

5. $\top$

The formula is SAT

DP: The limits

- The approach runs easily out of memory
- The solution: using backtrack search!

Davis-Putnam-Logemann-Loveland algorithm

Preliminary definitions

- Propositional variable can be assigned value False or True.
  - In some contexts variables may be unassigned
- A clause is satisfied is at least one of its literals is assigned value true
  - $(x_1 \vee \neg x_2 \vee \neg x_3)$
- A clause is unsatisfied if all of its literals are assigned value false
  - $(x_1 \vee \neg x_2 \vee \neg x_3)$
- A clause is unit if it contains one single unassigned literal and all other literals are assigned value false
  - $(x_1 \vee \neg x_2 \vee \neg x_3)$
- A formula is satisfied if all its clauses are satisfied
- A formula is unsatisfied if at least one of its clauses is unsatisfied

Davis-Putnam-Logemann-Loveland algorithm

- DPLL algorithm
- Standard backtrack search
- space efficient DP

DPLL

DPLL(F, $\mathcal{I}$):

- Apply unit propagation
- If conflict identified, return UNSAT
- Apply the pure literal rule
- If F is satisfied (empty), return SAT
- Select decision variable x
    - If DPLL(F, $\mathcal{I} \cup$ x) = SAT return SAT
    - return DPLL(F, $\mathcal{I} \cup$ x)

Notes:

x We use red to denote that a variable / literal is false

x We use green to denote that a variable / literal is true

Conflict all disjuncts of a clause are assigned false.

Pure literals in backtrack search

As before.

Example
$(\neg x_1 \vee x_2) \wedge (x_3 \vee \neg x_2) \wedge (x_4 \vee \neg x_5) \wedge (x_5 \vee \neg x_4)$

becomes

$(x_4 \vee \neg x_5) \wedge (x_5 \vee \neg x_4)$

Unit propagation in backtrack search

Unit clause rule in backtrack search:

> Given a unit clause, its only unassigned literal must be assigned value true for the clause to be satisfied.

Example
For unit clause $(x_1 \vee \neg x_2 \vee \neg x_3)$, the variable x_3 must be assigned value false.

Unit propogation rule: Iterated application of the unit clause rule.

Unit propagation in backtrack search

Example (1)

- $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$

Unit propagation in backtrack search

Example (1)

- $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$
- $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$

Unit propagation in backtrack search

Example (1)

- $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$
- $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$
- $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$

Unit propagation in backtrack search

Example (2)

- $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_4)$

Unit propagation in backtrack search

Example (2)

- $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_4)$
- $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_4)$

Unit propagation in backtrack search

Example (2)

- $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_4)$
- $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_4)$
- $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_4)$

Conflict!

Unit propagation can satisfy clauses but can also unsatisfy clauses (i.e. conflicts)

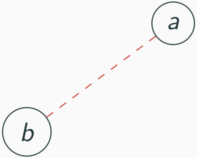
An example of DPLL

$(a \vee \neg b \vee d) \wedge$
$(a \vee \neg b \vee e) \wedge$
$(\neg b \vee \neg d \vee \neg e) \wedge$
$(a \vee b \vee c \vee d) \wedge$
$(a \vee b \vee c \vee \neg d) \wedge$
$(a \vee b \vee \neg c \vee e) \wedge$
$(a \vee b \vee \neg c \vee \neg e)$

An example of DPLL

Select decision variable: a

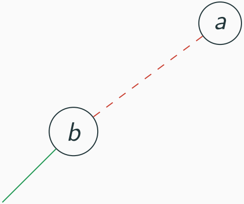
$(a \vee \neg b \vee d) \wedge$
$(a \vee \neg b \vee e) \wedge$
$(\neg b \vee \neg d \vee \neg e) \wedge$
$(a \vee b \vee c \vee d) \wedge$
$(a \vee b \vee c \vee \neg d) \wedge$
$(a \vee b \vee \neg c \vee e) \wedge$
$(a \vee b \vee \neg c \vee \neg e)$



An example of DPLL

Select decision variable: b

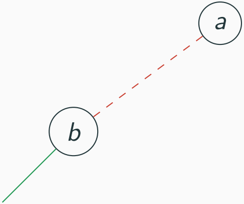
$(a \vee \neg b \vee d) \wedge$
$(a \vee \neg b \vee e) \wedge$
$(\neg b \vee \neg d \vee \neg e) \wedge$
$(a \vee b \vee c \vee d) \wedge$
$(a \vee b \vee c \vee \neg d) \wedge$
$(a \vee b \vee \neg c \vee e) \wedge$
$(a \vee b \vee \neg c \vee \neg e)$



An example of DPLL

Unit propagation: d

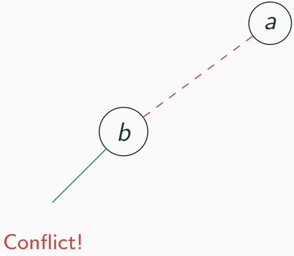
$(a \vee \neg b \vee d) \wedge$
$(a \vee \neg b \vee e) \wedge$
$(\neg b \vee \neg d \vee \neg e) \wedge$
$(a \vee b \vee c \vee d) \wedge$
$(a \vee b \vee c \vee \neg d) \wedge$
$(a \vee b \vee \neg c \vee e) \wedge$
$(a \vee b \vee \neg c \vee \neg e)$



An example of DPLL

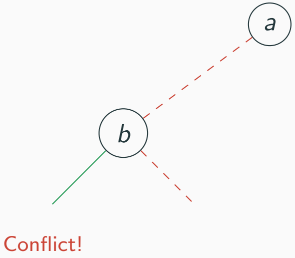
Unit propagation: e

$(a \vee \neg b \vee d) \wedge$
$(a \vee \neg b \vee e) \wedge$
$(\neg b \vee \neg d \vee \neg e) \wedge$
$(a \vee b \vee c \vee d) \wedge$
$(a \vee b \vee c \vee \neg d) \wedge$
$(a \vee b \vee \neg c \vee e) \wedge$
$(a \vee b \vee \neg c \vee \neg e)$



Select decision variable: b

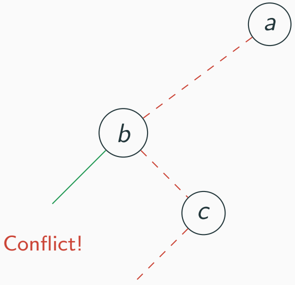
$(a \vee \neg b \vee d) \wedge$
$(a \vee \neg b \vee e) \wedge$
$(\neg b \vee \neg d \vee \neg e) \wedge$
$(a \vee b \vee c \vee d) \wedge$
$(a \vee b \vee c \vee \neg d) \wedge$
$(a \vee b \vee \neg c \vee e) \wedge$
$(a \vee b \vee \neg c \vee \neg e)$



An example of DPLL

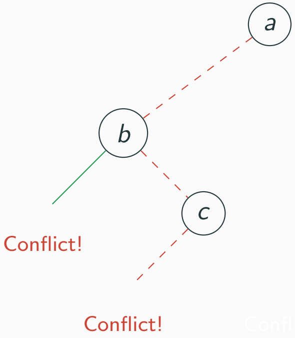
Select decision variable: c

$(a \lor \neg b \lor d) \land$
$(a \lor \neg b \lor e) \land$
$(\neg b \lor \neg d \lor \neg e) \land$
$(a \lor b \lor c \lor d) \land$
$(a \lor b \lor c \lor \neg d) \land$
$(a \lor b \lor \neg c \lor e) \land$
$(a \lor b \lor \neg c \lor \neg e)$



Unit propagation: d

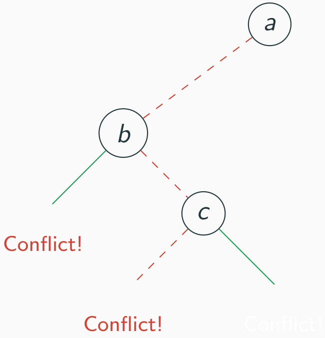
$(a \lor \neg b \lor d) \land$
$(a \lor \neg b \lor e) \land$
$(\neg b \lor \neg d \lor \neg e) \land$
$(a \lor b \lor c \lor d) \land$
$(a \lor b \lor c \lor \neg d) \land$
$(a \lor b \lor \neg c \lor e) \land$
$(a \lor b \lor \neg c \lor \neg e)$



An example of DPLL

Select decision variable: c

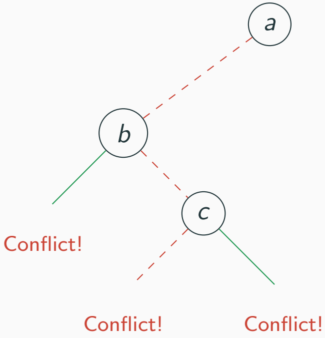
$(a \vee \neg b \vee d) \wedge$
$(a \vee \neg b \vee e) \wedge$
$(\neg b \vee \neg d \vee \neg e) \wedge$
$(a \vee b \vee c \vee d) \wedge$
$(a \vee b \vee c \vee \neg d) \wedge$
$(a \vee b \vee \neg c \vee e) \wedge$
$(a \vee b \vee \neg c \vee \neg e)$



An example of DPLL

Unit propagation: e

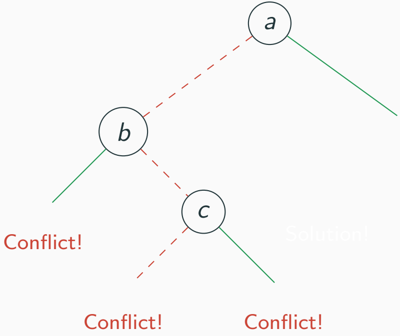
$(a \vee \neg b \vee d) \wedge$
$(a \vee \neg b \vee e) \wedge$
$(\neg b \vee \neg d \vee \neg e) \wedge$
$(a \vee b \vee c \vee d) \wedge$
$(a \vee b \vee c \vee \neg d) \wedge$
$(a \vee b \vee \neg c \vee e) \wedge$
$(a \vee b \vee \neg c \vee \neg e)$



An example of DPLL

Select decision variable: a

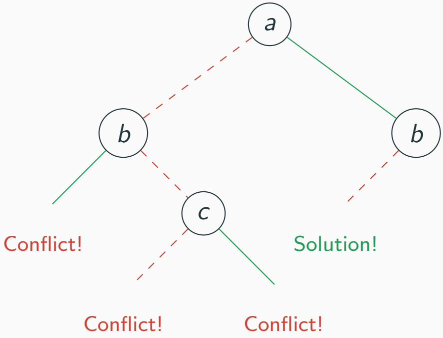
$(a \lor \neg b \lor d) \land$
$(a \lor \neg b \lor e) \land$
$(\neg b \lor \neg d \lor \neg e) \land$
$(a \lor b \lor c \lor d) \land$
$(a \lor b \lor c \lor \neg d) \land$
$(a \lor b \lor \neg c \lor e) \land$
$(a \lor b \lor \neg c \lor \neg e)$



An example of DPLL

Select decision variable: b

$(a \vee \neg b \vee d) \wedge$
$(a \vee \neg b \vee e) \wedge$
$(\neg b \vee \neg d \vee \neg e) \wedge$
$(a \vee b \vee c \vee d) \wedge$
$(a \vee b \vee c \vee \neg d) \wedge$
$(a \vee b \vee \neg c \vee e) \wedge$
$(a \vee b \vee \neg c \vee \neg e)$



Conflict-Driven Clause Learning algorithm

Conflict-Driven Clause Learning algorithm

- CDCL
- An extension of DPLL with:
  - Clause learning
  - Non-chronological backtracking
- Clause learning can be performed with various strategies
- CDCL algorithms are use in almost all modern SAT solvers

Clause learning

During backtrack search, for each conflict learn new clause, which explains and prevents repetition of the same conflict.

Example

$$\varphi = (a \vee b) \wedge (\neg b \vee c \vee d) \wedge (\neg b \vee e) \wedge (\neg d \vee \neg e \vee f) \ldots$$

Clause learning

During backtrack search, for each conflict learn new clause,
which explains and prevents repetition of the same conflict.

Example

$$\varphi \;=\; (a \vee b) \wedge (\neg b \vee c \vee d) \wedge (\neg b \vee e) \wedge (\neg d \vee \neg e \vee f) \ldots$$

- Assume decision $c = \mathit{False}$ and $f = \mathit{False}$

Clause learning

During backtrack search, for each conflict learn new clause, which explains and prevents repetition of the same conflict.

Example

$$\varphi = (a \vee b) \wedge (\neg b \vee c \vee d) \wedge (\neg b \vee e) \wedge (\neg d \vee \neg e \vee f) \ldots$$

- Assume decision $c = \mathit{False}$ and $f = \mathit{False}$
- Assign $a = \mathit{False}$

Clause learning

During backtrack search, for each conflict learn new clause,
which explains and prevents repetition of the same conflict.

Example

$$\varphi \;=\; (a \vee b) \wedge (\neg b \vee c \vee d) \wedge (\neg b \vee e) \wedge (\neg d \vee \neg e \vee f) \ldots$$

- Assume decision $c = \mathit{False}$ and $f = \mathit{False}$
- Assign $a = \mathit{False}$
- Unit propagation b

Clause learning

During backtrack search, for each conflict learn new clause, which explains and prevents repetition of the same conflict.

Example

$$\varphi \;=\; (a \vee b) \wedge (\neg b \vee c \vee d) \wedge (\neg b \vee e) \wedge (\neg d \vee \neg e \vee f) \ldots$$

- Assume decision $c = \mathit{False}$ and $f = \mathit{False}$
- Assign $a = \mathit{False}$
- Unit propagation b
- Unit propagation d

Clause learning

During backtrack search, for each conflict learn new clause,
which explains and prevents repetition of the same conflict.

Example

$$\varphi \;=\; (a \vee b) \wedge (\neg b \vee c \vee d) \wedge (\neg b \vee e) \wedge (\neg d \vee \neg e \vee f) \ldots$$

- Assume decision $c = \mathit{False}$ and $f = \mathit{False}$
- Assign $a = \mathit{False}$
- Unit propagation b
- Unit propagation d and e

Clause learning

During backtrack search, for each conflict learn new clause,
which explains and prevents repetition of the same conflict.

Example

$$\varphi = (a \vee b) \wedge (\neg b \vee c \vee d) \wedge (\neg b \vee e) \wedge (\neg d \vee \neg e \vee f) \ldots$$

- Assume decision $c = \mathit{False}$ and $f = \mathit{False}$
- Assign $a = \mathit{False}$
- Unit propagation b
- Unit propagation d and e
- A conflict is reached: $(\neg d \vee \neg e \vee f)$ is unsatisfied
- $\varphi \wedge \neg a \wedge \neg c \wedge \neg f \rightarrow \bot$, therefore $\varphi \rightarrow a \vee c \vee f$
- Learn new clause $(a \vee c \vee f)$

Non-chronological backtracking

- aka conflict directed backjumping
- During backtrack search, for each conflict backtrack to one of the causes of conflict.

Non-chronological backtracking

Example

$$\begin{aligned}\varphi \;=\; & (a \vee b) \wedge (\neg b \vee c \vee d) \wedge (\neg b \vee e) \wedge (\neg d \vee \neg e \vee f) \wedge \\ & (a \vee c \vee f) \wedge (\neg a \vee g) \wedge (\neg g \vee b) \wedge (\neg h \vee j) \wedge (\neg i \vee k)\end{aligned}$$

Non-chronological backtracking

Example

$$\varphi = (a \lor b) \land (\neg b \lor c \lor d) \land (\neg b \lor e) \land (\neg d \lor \neg e \lor f) \land$$
$$(a \lor c \lor f) \land (\neg a \lor g) \land (\neg g \lor b) \land (\neg h \lor j) \land (\neg i \lor k)$$

- Assume decision $c = False$, $f = False$, $h = False$ and $i = False$

Non-chronological backtracking

Example

$$\varphi = (a \lor b) \land (\neg b \lor c \lor d) \land (\neg b \lor e) \land (\neg d \lor \neg e \lor f) \land$$
$$(a \lor c \lor f) \land (\neg a \lor g) \land (\neg g \lor b) \land (\neg h \lor j) \land (\neg i \lor k)$$

- Assume decision $c = False$, $f = False$, $h = False$ and $i = False$
- Learnt clause $(a \lor c \lor f)$ unit-propagates a

Non-chronological backtracking

Example

$$\varphi = (a \vee b) \wedge (\neg b \vee c \vee d) \wedge (\neg b \vee e) \wedge (\neg d \vee \neg e \vee f) \wedge$$
$$(a \vee c \vee f) \wedge (\neg a \vee g) \wedge (\neg g \vee b) \wedge (\neg h \vee j) \wedge (\neg i \vee k)$$

- Assume decision $c = False$, $f = False$, $h = False$ and $i = False$
- Assignment $a = False$ cause conflict. Learnt clause $(a \vee c \vee f)$ implies a
- Unit propagation g

Non-chronological backtracking

Example

$$\varphi = (a \vee b) \wedge (\neg b \vee c \vee d) \wedge (\neg b \vee e) \wedge (\neg d \vee \neg e \vee f) \wedge$$
$$(a \vee c \vee f) \wedge (\neg a \vee g) \wedge (\neg g \vee b) \wedge (\neg h \vee j) \wedge (\neg i \vee k)$$

- Assume decision $c = False$, $f = False$, $h = False$ and $i = False$
- Assignment $a = False$ cause conflict. Learnt clause $(a \vee c \vee f)$ implies a
- Unit propagation g, b

Non-chronological backtracking

Example

$$\varphi = (a \vee b) \wedge (\neg b \vee c \vee d) \wedge (\neg b \vee e) \wedge (\neg d \vee \neg e \vee f) \wedge$$
$$(a \vee c \vee f) \wedge (\neg a \vee g) \wedge (\neg g \vee b) \wedge (\neg h \vee j) \wedge (\neg i \vee k)$$

- Assume decision $c = \mathit{False}$, $f = \mathit{False}$, $h = \mathit{False}$ and $i = \mathit{False}$
- Assignment $a = \mathit{False}$ cause conflict. Learnt clause $(a \vee c \vee f)$ implies a
- Unit propagation g, b, d

Non-chronological backtracking

Example

$$\varphi = (a \lor b) \land (\neg b \lor c \lor d) \land (\neg b \lor e) \land (\neg d \lor \neg e \lor f) \land$$
$$(a \lor c \lor f) \land (\neg a \lor g) \land (\neg g \lor b) \land (\neg h \lor j) \land (\neg i \lor k)$$

- Assume decision $c = False$, $f = False$, $h = False$ and $i = False$
- Assignment $a = False$ cause conflict. Learnt clause $(a \lor c \lor f)$ implies a
- Unit propagation g, b, d, and e

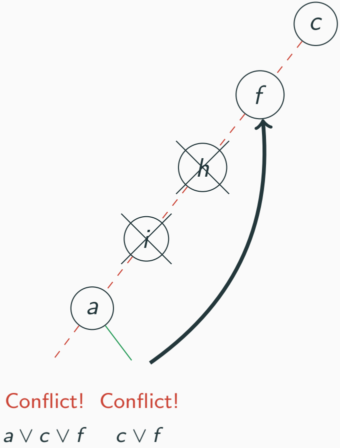
Non-chronological backtracking

Example

$$\varphi = (a \vee b) \wedge (\neg b \vee c \vee d) \wedge (\neg b \vee e) \wedge (\neg d \vee \neg e \vee f) \wedge$$
$$(a \vee c \vee f) \wedge (\neg a \vee g) \wedge (\neg g \vee b) \wedge (\neg h \vee j) \wedge (\neg i \vee k)$$

- Assume decision $c = False$, $f = False$, $h = False$ and $i = False$
- Assignment $a = False$ cause conflict. Learnt clause $(a \vee c \vee f)$ implies a
- Unit propagation g, b, d, and e
- A conflict is again reached: $(\neg d \vee \neg e \vee f)$ is unsatisfied
- Learn new clause $(c \vee f)$

Non-chronological backtracking



- Learnt clause: $c \lor f$
- Need to backtrack, given new clause
- Backtrack to most recent decision $f = false$

- Clause learning and non-chronological backtracking are hallmarks of modern SAT solvers