

Outline

DPLL

Unit Propagation

DPLL

Pure Atom

Atom p is **pure in a formula** A , if either all occurrences of p in A are positive or all occurrences of p in A are negative.

Example: $p \wedge r \rightarrow (q \rightarrow (p \wedge \neg r))$.

- Both occurrences of p are positive, so p is pure.
- The only occurrence of q is negative, so q is pure.
- r is not pure, since it has both negative and positive occurrences.

Pure Atom

Atom p is **pure in a formula** A , if either all occurrences of p in A are positive or all occurrences of p in A are negative.

Example: $p \wedge r \rightarrow (q \rightarrow (p \wedge \neg r))$.

- ▶ Both occurrences of p are positive, so p is pure.
- ▶ The only occurrence of q is negative, so q is pure.
- ▶ r is not pure, since it has both negative and positive occurrences.

Pure Atom

Atom p is **pure in a formula** A , if either all occurrences of p in A are positive or all occurrences of p in A are negative.

Example: $p \wedge r \rightarrow (q \rightarrow (p \wedge \neg r))$.

- ▶ Both occurrences of p are positive, so p is pure.
- ▶ The only occurrence of q is negative, so q is pure.
- ▶ r is not pure, since it has both negative and positive occurrences.

Pure Atom

Atom p is **pure in a formula** A , if either all occurrences of p in A are positive or all occurrences of p in A are negative.

Example: $p \wedge r \rightarrow (q \rightarrow (p \wedge \neg r))$.

- ▶ Both occurrences of p are positive, so p is pure.
- ▶ The only occurrence of q is negative, so q is pure.
- ▶ r is not pure, since it has both negative and positive occurrences.

Pure Atom

Atom p is **pure in a formula** A , if either all occurrences of p in A are positive or all occurrences of p in A are negative.

Example: $p \wedge r \rightarrow (q \rightarrow (p \wedge \neg r))$.

- ▶ Both occurrences of p are positive, so p is pure.
- ▶ The only occurrence of q is negative, so q is pure.
- ▶ r is not pure, since it has both negative and positive occurrences.

Properties of Pure Atoms

Lemma (Pure Atom)

Let p has only positive occurrences in A and $I \models A$. Define

$$I'(q) \stackrel{\text{def}}{\Leftrightarrow} \begin{cases} 1, & \text{if } p = q; \\ I(q), & \text{otherwise,} \end{cases}$$

for all q . Then $I' \models A$.

Likewise, let p has only negative occurrences in A and $I \models A$. Define

$$I'(q) \stackrel{\text{def}}{\Leftrightarrow} \begin{cases} 0, & \text{if } p = q; \\ I(q), & \text{otherwise,} \end{cases}$$

for all q . Then $I' \models A$.

Theorem (Pure Atom)

Let an atom p has only *positive* (respectively, only *negative*) occurrences in A . Then A is satisfiable if and only if so is A_p^{\top} (respectively, A_p^{\perp}).

Properties of Pure Atoms

Lemma (Pure Atom)

Let p has only positive occurrences in A and $I \models A$. Define

$$I'(q) \stackrel{\text{def}}{\Leftrightarrow} \begin{cases} 1, & \text{if } p = q; \\ I(q), & \text{otherwise,} \end{cases}$$

for all q . Then $I' \models A$.

Likewise, let p has only negative occurrences in A and $I \models A$. Define

$$I'(q) \stackrel{\text{def}}{\Leftrightarrow} \begin{cases} 0, & \text{if } p = q; \\ I(q), & \text{otherwise,} \end{cases}$$

for all q . Then $I' \models A$.

Theorem (Pure Atom)

Let an atom p has only positive (respectively, only negative) occurrences in A . Then A is satisfiable if and only if so is A_p^{\top} (respectively, A_p^{\perp}).

Properties of Pure Atoms

Lemma (Pure Atom)

Let p has only positive occurrences in A and $I \models A$. Define

$$I'(q) \stackrel{\text{def}}{\Leftrightarrow} \begin{cases} 1, & \text{if } p = q; \\ I(q), & \text{otherwise,} \end{cases}$$

for all q . Then $I' \models A$.

Likewise, let p has only negative occurrences in A and $I \models A$. Define

$$I'(q) \stackrel{\text{def}}{\Leftrightarrow} \begin{cases} 0, & \text{if } p = q; \\ I(q), & \text{otherwise,} \end{cases}$$

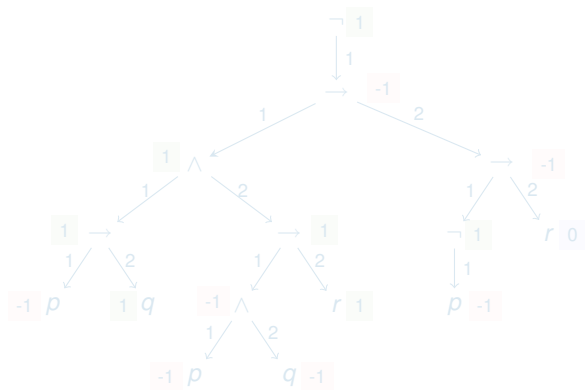
for all q . Then $I' \models A$.

Theorem (Pure Atom)

Let an atom p has only *positive* (respectively, only *negative*) occurrences in A . Then A is satisfiable if and only if so is A_p^T (respectively, A_p^\perp).

Pure atom, example

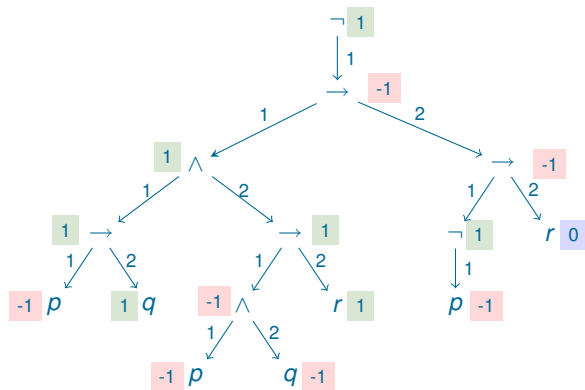
Consider $\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (\neg p \rightarrow r))$.



All occurrences of p are negative, so, for the purpose of checking satisfiability we can replace p by \perp .

Pure atom, example

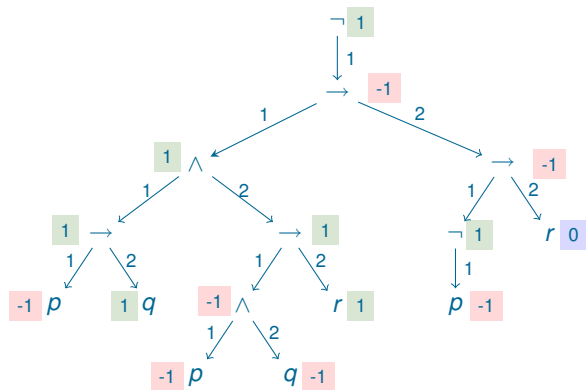
Consider $\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (\neg p \rightarrow r))$.



All occurrences of p are negative, so, for the purpose of checking satisfiability we can replace p by \perp .

Pure atom, example

Consider $\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (\neg p \rightarrow r))$.



All occurrences of p are negative, so, for the purpose of checking satisfiability we can replace p by \perp .

Example, continued

$$\begin{aligned} & \neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (\neg p \rightarrow r)) && \Rightarrow \\ \neg & ((\perp \rightarrow q) \wedge (\perp \wedge q \rightarrow r) \rightarrow (\neg \perp \rightarrow r)) && \Rightarrow \\ & \neg(\top \wedge (\perp \wedge q \rightarrow r) \rightarrow (\neg \perp \rightarrow r)) && \Rightarrow \\ & \neg((\perp \wedge q \rightarrow r) \rightarrow (\neg \perp \rightarrow r)) && \Rightarrow \\ & \neg((\perp \rightarrow r) \rightarrow (\neg \perp \rightarrow r)) && \Rightarrow \\ & \neg(\top \rightarrow (\neg \perp \rightarrow r)) && \Rightarrow \\ & \neg(\neg \perp \rightarrow r) && \Rightarrow \\ & \neg(\top \rightarrow r) && \Rightarrow \\ & \neg r && \Rightarrow \\ & \neg \perp && \Rightarrow \\ & \top && \Rightarrow \end{aligned}$$

Satisfiability-checking for sets of clauses

The CNF transformation of

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

gives the set of four clauses:

$$\begin{aligned} &\neg p \vee q \\ &\neg p \vee \neg q \vee r \\ &p \\ &\neg r \end{aligned}$$

Every interpretation that satisfies this set of clauses **must** assign 1 to p and 0 to r , so we do not have to guess values of these variables.

In fact, we can do even better and establish unsatisfiability **without** any guessing.

Satisfiability-checking for sets of clauses

The CNF transformation of

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

gives the set of four clauses:

$$\begin{aligned} &\neg p \vee q \\ &\neg p \vee \neg q \vee r \\ &p \\ &\neg r \end{aligned}$$

Every interpretation that satisfies this set of clauses **must** assign **1** to p and **0** to r , so **we do not have to guess values of these variables**.

In fact, we can do even better and establish unsatisfiability **without any guessing**.

Satisfiability-checking for sets of clauses

The CNF transformation of

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

gives the set of four clauses:

$$\begin{aligned} &\neg p \vee q \\ &\neg p \vee \neg q \vee r \\ &p \\ &\neg r \end{aligned}$$

Every interpretation that satisfies this set of clauses **must** assign **1** to p and **0** to r , so **we do not have to guess values of these variables**.

In fact, we can do even better and establish unsatisfiability **without any guessing**.

Searching for satisfiability

$$\begin{array}{l|l} \neg p \vee q & q \\ \neg p \vee \neg q \vee r & \neg q \vee r \\ p & \\ \neg r & \neg r \end{array}$$

Searching for satisfiability

$$\begin{array}{l|l} q & q \\ \neg q \vee r & \neg q \\ \neg r & \end{array}$$

Unit propagation

Let S be a set of clauses. **Unit propagation**: repeatedly performing the following transformation: if S contains a unit clause, i.e. a clause consisting of one literal L , then

1. remove from S every clause of the form $L \vee C'$;
2. replace in S every clause of the form $\bar{L} \vee C'$ by the clause C' .

Searching for satisfiability

n_1	$\neg q \vee n_4$
$\neg n_1 \vee \neg n_2$	$\neg n_5 \vee \neg n_6 \vee r$
$n_1 \vee n_2$	$n_6 \vee n_5$
$\neg n_2 \vee \neg n_3 \vee n_7$	$\neg r \vee n_5$
$n_3 \vee n_2$	$\neg n_6 \vee p$
$\neg n_7 \vee n_2$	$\neg n_6 \vee q$
$\neg n_3 \vee n_4$	$\neg p \vee \neg q \vee n_6$
$\neg n_3 \vee n_5$	$\neg n_7 \vee \neg p \vee r$
$\neg n_4 \vee \neg n_5 \vee n_3$	$p \vee n_7$
$\neg n_4 \vee \neg p \vee q$	$\neg r \vee n_7$
$p \vee n_4$	

	$\neg q \vee n_4$
$\neg n_2$	$\neg n_5 \vee \neg n_6 \vee r$
	$n_6 \vee n_5$
$\neg n_2 \vee \neg n_3 \vee n_7$	$\neg r \vee n_5$
$n_3 \vee n_2$	$\neg n_6 \vee p$
$\neg n_7 \vee n_2$	$\neg n_6 \vee q$
$\neg n_3 \vee n_4$	$\neg p \vee \neg q \vee n_6$
$\neg n_3 \vee n_5$	$\neg n_7 \vee \neg p \vee r$
$\neg n_4 \vee \neg n_5 \vee n_3$	$p \vee n_7$
$\neg n_4 \vee \neg p \vee q$	$\neg r \vee n_7$
$p \vee n_4$	

Unit Propagation

$\neg n_2$	$\neg q \vee n_4$
$\neg n_2 \vee \neg n_3 \vee n_7$	$\neg n_5 \vee \neg n_6 \vee r$
$n_3 \vee n_2$	$n_6 \vee n_5$
$\neg n_7 \vee n_2$	$\neg r \vee n_5$
$\neg n_3 \vee n_4$	$\neg n_6 \vee p$
$\neg n_3 \vee n_5$	$\neg n_6 \vee q$
$\neg n_4 \vee \neg n_5 \vee n_3$	$\neg p \vee \neg q \vee n_6$
$\neg n_4 \vee \neg p \vee q$	$\neg n_7 \vee \neg p \vee r$
$p \vee n_4$	$p \vee n_7$
	$\neg r \vee n_7$

	$\neg q \vee n_4$
	$\neg n_5 \vee \neg n_6 \vee r$
n_3	$n_6 \vee n_5$
$\neg n_7$	$\neg r \vee n_5$
$\neg n_3 \vee n_4$	$\neg n_6 \vee p$
$\neg n_3 \vee n_5$	$\neg n_6 \vee q$
$\neg n_4 \vee \neg n_5 \vee n_3$	$\neg p \vee \neg q \vee n_6$
$\neg n_4 \vee \neg p \vee q$	$\neg n_7 \vee \neg p \vee r$
$p \vee n_4$	$p \vee n_7$
	$\neg r \vee n_7$

Unit Propagation

n_3	$\neg q \vee n_4$
$\neg n_7$	$\neg n_5 \vee \neg n_6 \vee r$
$\neg n_3 \vee n_4$	$n_6 \vee n_5$
$\neg n_3 \vee n_5$	$\neg r \vee n_5$
$\neg n_4 \vee \neg n_5 \vee n_3$	$\neg n_6 \vee p$
$\neg n_4 \vee \neg p \vee q$	$\neg n_6 \vee q$
$p \vee n_4$	$\neg p \vee \neg q \vee n_6$
	$\neg n_7 \vee \neg p \vee r$
	$p \vee n_7$
	$\neg r \vee n_7$

	$\neg q \vee n_4$
	$\neg n_5 \vee \neg n_6 \vee r$
n_4	$n_6 \vee n_5$
n_5	$\neg r \vee n_5$
	$\neg n_6 \vee p$
$\neg n_4 \vee \neg p \vee q$	$\neg n_6 \vee q$
$p \vee n_4$	$\neg p \vee \neg q \vee n_6$
	p
	$\neg r$

Unit Propagation

n_4	$\neg q \vee n_4$
n_5	$\neg n_5 \vee \neg n_6 \vee r$
$\neg n_4 \vee \neg p \vee q$	$n_6 \vee n_5$
$p \vee n_4$	$\neg r \vee n_5$
	$\neg n_6 \vee p$
	$\neg n_6 \vee q$
	$\neg p \vee \neg q \vee n_6$
	p
	$\neg r$

	$\neg n_6$
q	
	$\neg n_6 \vee q$
	$\neg q \vee n_6$

Unit Propagation

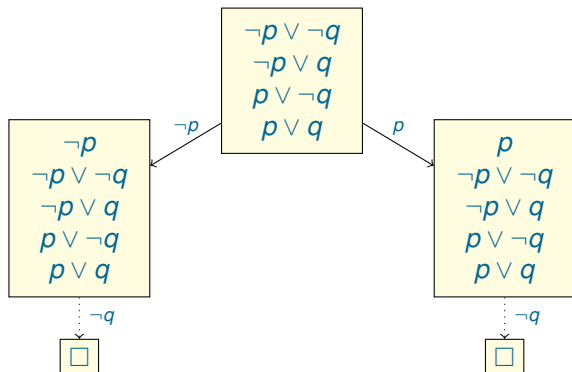
$$\begin{array}{l|l} q & \\ \neg n_6 & \\ \neg n_6 \vee q & \\ \neg q \vee n_6 & \square \end{array}$$

DPLL = splitting + unit propagation

```
procedure DPLL(S)  
input: set of clauses S  
output: satisfiable or unsatisfiable  
parameters: function select_literal  
begin  
  S := propagate(S)  
  if S is empty then return satisfiable  
  if S contains  $\square$  then return unsatisfiable  
  L := select_literal(S)  
  if DPLL( $S \cup \{L\}$ ) = satisfiable  
    then return satisfiable  
    else return DPLL( $S \cup \{\bar{L}\}$ )  
end
```

DPLL. Example 1

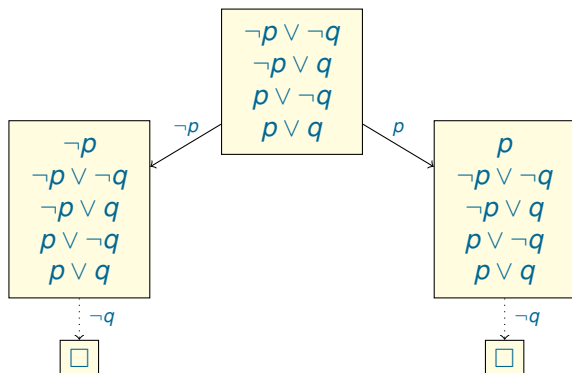
Can be illustrated using **DPLL trees** (similar to splitting trees).



Since all branches end up in a set containing the **empty clause**, the initial set of clauses is **unsatisfiable**.

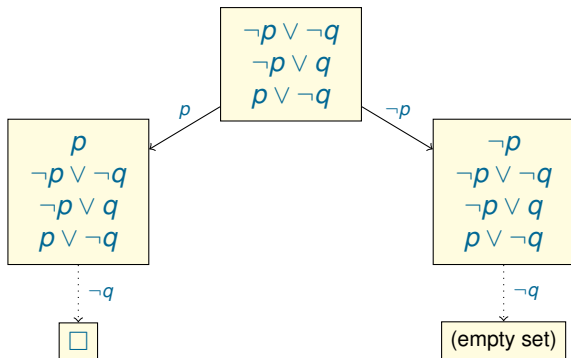
DPLL. Example 1

Can be illustrated using DPLL trees (similar to splitting trees).



Since all branches end up in a set containing the **empty clause**, the initial set of clauses is **unsatisfiable**.

DPLL. Example 2

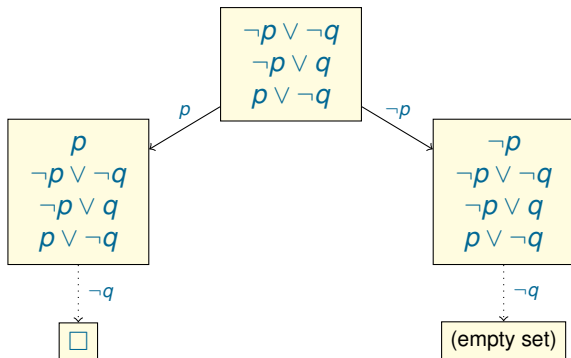


The set of clauses is **satisfiable**.

The model can be obtained by collecting all selected literals and literals used in unit propagation on the branch resulting in the empty set.

This DPLL tree gives us the model $\{p \mapsto 0, q \mapsto 0\}$.

DPLL. Example 2

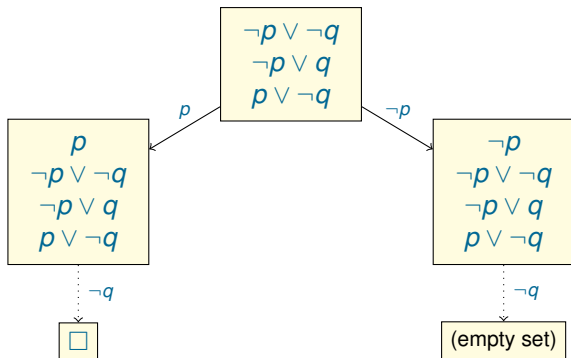


The set of clauses is **satisfiable**.

The model can be obtained by collecting all **selected literals** and **literals used in unit propagation** on the branch resulting in the empty set.

This DPLL tree gives us the model $\{p \mapsto 0, q \mapsto 0\}$.

DPLL. Example 2



The set of clauses is **satisfiable**.

The model can be obtained by collecting all **selected literals** and **literals used in unit propagation** on the branch resulting in the empty set.

This DPLL tree gives us the model $\{p \mapsto 0, q \mapsto 0\}$.

Two optimisations

Tautology: a clause $p \vee \neg p \vee C$.

Every tautology is **valid**, so tautologies can be removed.

A literal L in S is called **pure** if S contains no clauses of the form $\bar{L} \vee C$

All clauses containing a pure literal **can be satisfied** by assigning a suitable truth value to the variable of this literal.

Hence, clauses containing pure literals can be removed, too.

Two optimisations

Tautology: a clause $p \vee \neg p \vee C$.

Every tautology is **valid**, so tautologies can be removed.

A literal L in S is called **pure** if S contains no clauses of the form $\bar{L} \vee C$

All clauses containing a pure literal **can be satisfied** by assigning a suitable truth value to the variable of this literal.

Hence, clauses containing pure literals can be removed, too.

Two optimisations

Tautology: a clause $p \vee \neg p \vee C$.

Every tautology is **valid**, so tautologies can be removed.

A literal L in S is called **pure** if S contains no clauses of the form $\bar{L} \vee C$

All clauses containing a pure literal **can be satisfied** by assigning a suitable truth value to the variable of this literal.

Hence, clauses containing pure literals can be removed, too.

Two optimisations

Tautology: a clause $p \vee \neg p \vee C$.

Every tautology is **valid**, so tautologies can be removed.

A literal L in S is called **pure** if S contains no clauses of the form $\bar{L} \vee C$

All clauses containing a pure literal **can be satisfied** by assigning a suitable truth value to the variable of this literal.

Hence, clauses containing pure literals can be removed, too.

Two optimisations

Tautology: a clause $p \vee \neg p \vee C$.

Every tautology is **valid**, so tautologies can be removed.

A literal L in S is called **pure** if S contains no clauses of the form $\bar{L} \vee C$

All clauses containing a pure literal **can be satisfied** by assigning a suitable truth value to the variable of this literal.

Hence, clauses containing pure literals can be removed, too.