

DPLL (1962)

Davis-Putnam-Logemann-Loveland algorithm

- for satisfiability checking
- algorithm runs basic backtracking
- each iteration, run the *splitting rule*:
 - choosing a literal,
 - assigning a truth value to it,
 - simplifying the formula and
 - then recursively checking if the simplified formula is satisfiable.
 - simplification 1: removing all clauses which become true under the assignment from the formula, and
 - simplification 2: removing all literals that become false from the remaining clauses.

DPLL (1962)

Davis-Putnam-Logemann-Loveland algorithm

```
function DPLL( $\Phi$ ) {  
    if  $\Phi$  is a consistent set of literals then return T;  
    if  $\Phi$  contains an empty clause then return F;  
    for every unit clause  $l$  in  $\Phi$   
         $\Phi$ =unit-propagate( $l$ ,  $\Phi$ );  
    for every literal  $l$  that occurs pure in  $\Phi$ ,  
         $\Phi$ =pure-literal-assign( $l$ ,  $\Phi$ );  
     $l$  := choose-literal( $\Phi$ );  
    return DPLL( $\Phi \wedge l$ ) OR DPLL( $\Phi \wedge \text{not}(l)$ );  
}
```

DPLL (1962)

Davis-Putnam-Logemann-Loveland algorithm

Enhancement by the eager use of the following rules :

■ Unit propagation

- If a clause is a *unit clause*, i.e. it contains only a single unassigned literal, this clause can only be satisfied by assigning the necessary value to make this literal true.
- In practice, this often leads to deterministic cascades of units, thus avoiding a large part of the naive search space.

■ Pure literal elimination

- If a propositional variable occurs with only one polarity in the formula, it is called *pure*.
- Pure literals can always be assigned in a way that makes all clauses containing them true.
- Most current implementations omit it, as the effect for efficient implementations now is negligible or, due to the overhead for detecting purity, even negative.

DPLL (1962)

Davis-Putnam-Logemann-Loveland algorithm

An example: Prove $p, (p \rightarrow q), (q \rightarrow r) \models r$

Conversion to clauses:

$$\Rightarrow p, (\neg p \vee q), (\neg q \vee r), \neg r$$

Unit propagation with $p=\text{true}$, $r = \text{false}$:

$$\Rightarrow \text{true}, (\text{false} \vee q), (\neg q \vee \text{false}), \text{true}$$

$$\Rightarrow q, \neg q$$

Pure literal elimination :

$$\Rightarrow q, \neg q$$

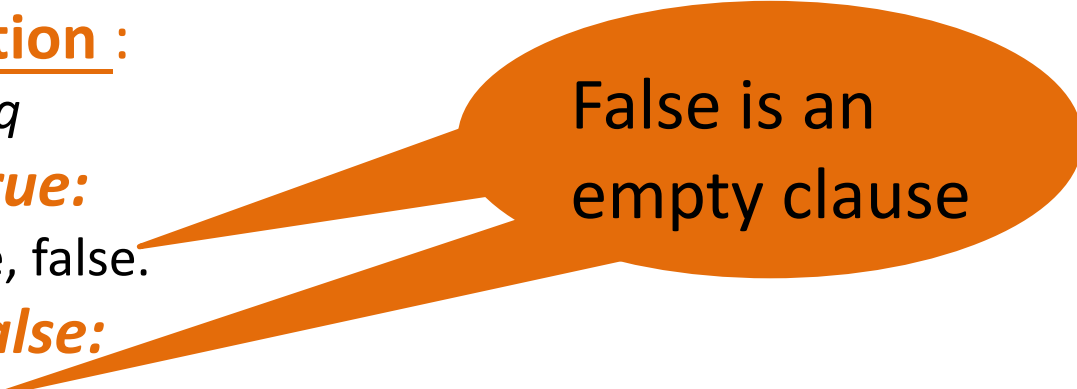
Choose literal $q = \text{true}$:

$$\Rightarrow \text{true}, \text{false}.$$

Choose literal $q = \text{false}$:

$$\Rightarrow \text{false}, \text{true}.$$

Thus the lemma is proven by refutation with DPLL.



False is an
empty clause

DPLL (1962)

Davis-Putnam-Logemann-Loveland algorithm

An example (another presentation):

Prove $p, (p \rightarrow q), (q \rightarrow r) \models r$

Conversion to clauses as sets of literals:

$\Rightarrow \{p\}, \{\neg p, q\}, \{\neg q, r\}, \{\neg r\}$

Unit propagation with $p=\text{true}$, $r = \text{false}$:

$\Rightarrow \{\text{true}\}, \{\text{false}, q\}, \{\neg q, \text{false}\}, \{\text{true}\}$

$\Rightarrow \{q\}, \{\neg q\} : \text{elimination of true clause and false literal}$

Pure literal elimination :

$\Rightarrow \{q\}, \{\neg q\}$

Choosing literal $q=\text{true}$:

$\Rightarrow \{\text{true}\}, \{\text{false}\}$

$\Rightarrow \{\} : \text{elimination of true clause and false literal}$