

Decision Procedures

Jochen Hoenicke



Software Engineering
Albert-Ludwigs-University Freiburg

Summer 2012

Foundations: Propositional Logic

<u>Atom</u>	<u>truth symbols</u> \top (“true”) and \perp (“false”) <u>propositional variables</u> $P, Q, R, P_1, Q_1, R_1, \dots$
<u>Literal</u>	atom α or its negation $\neg\alpha$
<u>Formula</u>	literal or application of a <u>logical connective</u> to formulae F, F_1, F_2
	$\neg F$ “not” (negation)
	$(F_1 \wedge F_2)$ “and” (conjunction)
	$(F_1 \vee F_2)$ “or” (disjunction)
	$(F_1 \rightarrow F_2)$ “implies” (implication)
	$(F_1 \leftrightarrow F_2)$ “if and only if” (iff)

formula $F : ((P \wedge Q) \rightarrow (T \vee \neg Q))$

atoms: P, Q, T

literal: $\neg Q$

subformulas: $(P \wedge Q), (T \vee \neg Q)$

abbreviation

$$F : P \wedge Q \rightarrow T \vee \neg Q$$

Formula F and Interpretation I is evaluated to a truth value 0/1
where 0 corresponds to value false
1 true

Interpretation $I : \{P \mapsto 1, Q \mapsto 0, \dots\}$

Evaluation of logical operators:

F_1	F_2	$\neg F_1$	$F_1 \wedge F_2$	$F_1 \vee F_2$	$F_1 \rightarrow F_2$	$F_1 \leftrightarrow F_2$
0	0	1	0	0	1	1
0	1		0	1	1	0
1	0	0	0	1	0	0
1	1		1	1	1	1

$$F : P \wedge Q \rightarrow P \vee \neg Q$$

$$I : \{P \mapsto 1, Q \mapsto 0\}$$

P	Q	$\neg Q$	$P \wedge Q$	$P \vee \neg Q$	F
1	0	1	0	1	1

1 = true

0 = false

F evaluates to true under I

$I \models F$ if F evaluates to 1 / true under I
 $I \not\models F$ 0 / false

Base Case:

$I \models \top$

$I \not\models \perp$

$I \models P$ iff $I[P] = 1$

$I \not\models P$ iff $I[P] = 0$

Inductive Case:

$I \models \neg F$ iff $I \not\models F$

$I \models F_1 \wedge F_2$ iff $I \models F_1$ and $I \models F_2$

$I \models F_1 \vee F_2$ iff $I \models F_1$ or $I \models F_2$

$I \models F_1 \rightarrow F_2$ iff, if $I \models F_1$ then $I \models F_2$

$I \models F_1 \leftrightarrow F_2$ iff, $I \models F_1$ and $I \models F_2$,
or $I \not\models F_1$ and $I \not\models F_2$

$$F : P \wedge Q \rightarrow P \vee \neg Q$$

$$I : \{P \mapsto 1, Q \mapsto 0\}$$

1. $I \models P$ since $I[P] = 1$
2. $I \not\models Q$ since $I[Q] = 0$
3. $I \models \neg Q$ by 2, \neg
4. $I \not\models P \wedge Q$ by 2, \wedge
5. $I \models P \vee \neg Q$ by 1, \vee
6. $I \models F$ by 4, \rightarrow Why?

Thus, F is true under I .

Definition (Satisfiability)

F is **satisfiable** iff there exists an interpretation I such that $I \models F$.

Definition (Validity)

F is **valid** iff for all interpretations I , $I \models F$.

Note

F is valid iff $\neg F$ is unsatisfiable

Proof.

F is valid iff $\forall I : I \models F$ iff $\neg \exists I : I \not\models F$ iff $\neg F$ is unsatisfiable. \square

Decision Procedure: An algorithm for deciding validity or satisfiability.

Now assume, you are a decision procedure.

Which of the following formulae is satisfiable, which is valid?

- $F_1 : P \wedge Q$
satisfiable, not valid
- $F_2 : \neg(P \wedge Q)$
satisfiable, not valid
- $F_3 : P \vee \neg P$
satisfiable, valid
- $F_4 : \neg(P \vee \neg P)$
unsatisfiable, not valid
- $F_5 : (P \rightarrow Q) \wedge (P \vee Q) \wedge \neg Q$
unsatisfiable, not valid

Is there a formula that is unsatisfiable and valid?

$$F : P \wedge Q \rightarrow P \vee \neg Q$$

P	Q	$P \wedge Q$	$\neg Q$	$P \vee \neg Q$	F
0	0	0	1	1	1
0	1	0	0	0	1
1	0	0	1	1	1
1	1	1	0	1	1

Thus F is valid.

$$F : P \vee Q \rightarrow P \wedge Q$$

P	Q	$P \vee Q$	$P \wedge Q$	F
0	0	0	0	1
0	1	1	0	0
1	0	1	0	0
1	1	1	1	1

← satisfying I

← falsifying I

Thus F is satisfiable, but invalid.

- Assume F is not valid and I a falsifying interpretation: $I \not\models F$
- Apply proof rules.
- If no contradiction reached and no more rules applicable, F is invalid.
- If in every branch of proof a contradiction reached, F is valid.

$$\frac{I \models \neg F}{I \not\models F}$$

$$\frac{I \not\models \neg F}{I \models F}$$

$$\frac{I \models F \wedge G}{\begin{array}{l} I \models F \\ I \models G \end{array}} \leftarrow \text{and}$$

$$\frac{I \not\models F \wedge G}{\begin{array}{l} I \not\models F \\ I \not\models G \end{array}} \leftarrow \text{or}$$

$$\frac{I \models F \vee G}{I \models F \mid I \models G}$$

$$\frac{I \not\models F \vee G}{\begin{array}{l} I \not\models F \\ I \not\models G \end{array}}$$

$$\frac{I \models F \rightarrow G}{I \not\models F \mid I \models G}$$

$$\frac{I \not\models F \rightarrow G}{\begin{array}{l} I \models F \\ I \not\models G \end{array}}$$

$$\frac{I \models F \leftrightarrow G}{I \models F \wedge G \mid I \not\models F \vee G}$$

$$\frac{I \not\models F \leftrightarrow G}{I \models F \wedge \neg G \mid I \models \neg F \wedge G}$$

$$\frac{\begin{array}{l} I \models F \\ I \not\models F \end{array}}{I \models \perp}$$

Prove $F : P \wedge Q \rightarrow P \vee \neg Q$ is valid.

Let's assume that F is not valid and that I is a falsifying interpretation.

- | | | |
|----|--|---------------------------|
| 1. | $I \not\models P \wedge Q \rightarrow P \vee \neg Q$ | assumption |
| 2. | $I \models P \wedge Q$ | 1, Rule \rightarrow |
| 3. | $I \not\models P \vee \neg Q$ | 1, Rule \rightarrow |
| 4. | $I \models P$ | 2, Rule \wedge |
| 5. | $I \not\models P$ | 3, Rule \vee |
| 6. | $I \models \perp$ | 4 and 5 are contradictory |

Thus F is valid.

Example 2

Prove $F : (P \rightarrow Q) \wedge (Q \rightarrow R) \rightarrow (P \rightarrow R)$ is valid.

Let's assume that F is not valid.

	1. $I \not\models F$	assumption
	2. $I \models (P \rightarrow Q) \wedge (Q \rightarrow R)$	1, Rule \rightarrow
	3. $I \not\models P \rightarrow R$	1, Rule \rightarrow
	4. $I \models P$	3, Rule \rightarrow
	5. $I \not\models R$	3, Rule \rightarrow
	6. $I \models P \rightarrow Q$	2, Rule \wedge
	7. $I \models Q \rightarrow R$	2, Rule \wedge
8a.	$I \not\models P$	8b. $I \models Q$ 6 \rightarrow
9a.	$I \models \perp$	9ba. $I \not\models Q$ 9bb. $I \models R$
		10ba. $I \models \perp$ 10bb. $I \models \perp$

Our assumption is incorrect in all cases — F is valid.

Example 3

Is $F : P \vee Q \rightarrow P \wedge Q$ valid?

Let's assume that F is not valid.

1.	$I \not\models P \vee Q \rightarrow P \wedge Q$	assumption										
2.	$I \models P \vee Q$	1 and \rightarrow										
3.	$I \not\models P \wedge Q$	1 and \rightarrow										
4a.	$I \models P$	2 and \vee	4b.	$I \models Q$	2 and \vee							
5aa.	$I \not\models P$		5ab.	$I \not\models Q$		5ba.	$I \not\models P$		5bb.	$I \not\models Q$		
6aa.	$I \models \perp$										6bb.	$I \models \perp$

We cannot always derive a contradiction. F is not valid.

Falsifying interpretation:

$I_1 : \{P \mapsto \text{true}, Q \mapsto \text{false}\}$ $I_2 : \{Q \mapsto \text{true}, P \mapsto \text{false}\}$

We have to derive a contradiction in **all** cases for F to be valid.

Idea: Simplify decision procedure, by simplifying the formula first.
Convert it into a simpler normal form, e.g.:

- **Negation Normal Form:** No \rightarrow and no \leftrightarrow ; negation only before atoms.
- **Conjunctive Normal Form:** Negation normal form, where conjunction is outside, disjunction is inside.
- **Disjunctive Normal Form:** Negation normal form, where disjunction is outside, conjunction is inside.

The formula in normal form should be equivalent to the original input.

F_1 and F_2 are equivalent ($F_1 \Leftrightarrow F_2$)

iff for all interpretations I , $I \models F_1 \leftrightarrow F_2$

To prove $F_1 \Leftrightarrow F_2$ show $F_1 \leftrightarrow F_2$ is valid.

F_1 implies F_2 ($F_1 \Rightarrow F_2$)

iff for all interpretations I , $I \models F_1 \rightarrow F_2$

$F_1 \Leftrightarrow F_2$ and $F_1 \Rightarrow F_2$ are not formulae!

If $F_1 \Leftrightarrow F'_1$ and $F_2 \Leftrightarrow F'_2$, then

- $\neg F_1 \Leftrightarrow \neg F'_1$
- $F_1 \vee F_2 \Leftrightarrow F'_1 \vee F'_2$
- $F_1 \wedge F_2 \Leftrightarrow F'_1 \wedge F'_2$
- $F_1 \rightarrow F_2 \Leftrightarrow F'_1 \rightarrow F'_2$
- $F_1 \leftrightarrow F_2 \Leftrightarrow F'_1 \leftrightarrow F'_2$

- if we replace in a formula F a subformula F_1 by F'_1 and obtain F' , then $F \Leftrightarrow F'$.

Negations appear only in literals. (only \neg, \wedge, \vee)

To transform F to equivalent F' in NNF use recursively the following template equivalences (left-to-right):

$$\begin{array}{l} \neg\neg F_1 \Leftrightarrow F_1 \quad \neg\top \Leftrightarrow \perp \quad \neg\perp \Leftrightarrow \top \\ \neg(F_1 \wedge F_2) \Leftrightarrow \neg F_1 \vee \neg F_2 \\ \neg(F_1 \vee F_2) \Leftrightarrow \neg F_1 \wedge \neg F_2 \end{array} \left. \vphantom{\begin{array}{l} \neg\neg F_1 \Leftrightarrow F_1 \\ \neg(F_1 \wedge F_2) \Leftrightarrow \neg F_1 \vee \neg F_2 \\ \neg(F_1 \vee F_2) \Leftrightarrow \neg F_1 \wedge \neg F_2 \end{array}} \right\} \text{De Morgan's Law}$$
$$F_1 \rightarrow F_2 \Leftrightarrow \neg F_1 \vee F_2$$
$$F_1 \leftrightarrow F_2 \Leftrightarrow (F_1 \rightarrow F_2) \wedge (F_2 \rightarrow F_1)$$

Convert $F : (Q_1 \vee \neg\neg R_1) \wedge (\neg Q_2 \rightarrow R_2)$ into NNF

$$\begin{aligned} & (Q_1 \vee \neg\neg R_1) \wedge (\neg Q_2 \rightarrow R_2) \\ \Leftrightarrow & (Q_1 \vee R_1) \wedge (\neg Q_2 \rightarrow R_2) \\ \Leftrightarrow & (Q_1 \vee R_1) \wedge (\neg\neg Q_2 \vee R_2) \\ \Leftrightarrow & (Q_1 \vee R_1) \wedge (Q_2 \vee R_2) \end{aligned}$$

The last formula is equivalent to F and is in NNF.

Disjunction of conjunctions of literals

$$\bigvee_i \bigwedge_j \ell_{i,j} \quad \text{for literals } \ell_{i,j}$$

To convert F into equivalent F' in DNF,
transform F into NNF and then
use the following template equivalences (left-to-right):

$$\left. \begin{aligned} (F_1 \vee F_2) \wedge F_3 &\Leftrightarrow (F_1 \wedge F_3) \vee (F_2 \wedge F_3) \\ F_1 \wedge (F_2 \vee F_3) &\Leftrightarrow (F_1 \wedge F_2) \vee (F_1 \wedge F_3) \end{aligned} \right\} \textit{dist}$$

Convert $F : (Q_1 \vee \neg\neg R_1) \wedge (\neg Q_2 \rightarrow R_2)$ into DNF

$$\begin{aligned}
 & (Q_1 \vee \neg\neg R_1) \wedge (\neg Q_2 \rightarrow R_2) \\
 \Leftrightarrow & (Q_1 \vee R_1) \wedge (Q_2 \vee R_2) && \text{in NNF} \\
 \Leftrightarrow & (Q_1 \wedge (Q_2 \vee R_2)) \vee (R_1 \wedge (Q_2 \vee R_2)) && \text{dist} \\
 \Leftrightarrow & (Q_1 \wedge Q_2) \vee (Q_1 \wedge R_2) \vee (R_1 \wedge Q_2) \vee (R_1 \wedge R_2) && \text{dist}
 \end{aligned}$$

The last formula is equivalent to F and is in DNF. Note that formulas can grow exponentially.

Conjunction of disjunctions of literals

$$\bigwedge_i \bigvee_j l_{i,j} \quad \text{for literals } l_{i,j}$$

To convert F into equivalent F' in CNF,
transform F into NNF and then
use the following template equivalences (left-to-right):

$$\begin{aligned}(F_1 \wedge F_2) \vee F_3 &\Leftrightarrow (F_1 \vee F_3) \wedge (F_2 \vee F_3) \\ F_1 \vee (F_2 \wedge F_3) &\Leftrightarrow (F_1 \vee F_2) \wedge (F_1 \vee F_3)\end{aligned}$$

A disjunction of literals $P_1 \vee P_2 \vee \neg P_3$ is called a **clause**.

For brevity we write it as set: $\{P_1, P_2, \overline{P_3}\}$.

A formula in CNF is a set of clauses (a set of sets of literals).

Definition (Equisatisfiability)

F and F' are **equisatisfiable**, iff

F is satisfiable if and only if F' is satisfiable

Every formula is equisatisfiable to either \top or \perp .

There is a **efficient conversion** of F to F' where

- F' is in CNF and
- F and F' are equisatisfiable

Note: efficient means polynomial in the size of F .

Basic Idea:

- Introduce a new variable P_G for every subformula G ; unless G is already an atom.
- For each subformula $G : G_1 \circ G_2$ produce a small formula $P_G \leftrightarrow P_{G_1} \circ P_{G_2}$.
- encode each of these (small) formulae separately to CNF.

The formula

$$P_F \wedge \bigwedge_G \text{CNF}(P_G \leftrightarrow P_{G_1} \circ P_{G_2})$$

is equisatisfiable to F .

The number of subformulae is linear in the size of F .

The time to convert one small formula is constant!

Convert $F : P \vee Q \rightarrow P \wedge \neg R$ to CNF.

Introduce new variables: $P_F, P_{P \vee Q}, P_{P \wedge \neg R}, P_{\neg R}$. Create new formulae and convert them to CNF separately:

- $P_F \leftrightarrow (P_{P \vee Q} \rightarrow P_{P \wedge \neg R})$ in CNF:

$$F_1 : \{ \{ \overline{P_F}, \overline{P_{P \vee Q}}, P_{P \wedge \neg R} \}, \{ P_F, P_{P \vee Q} \}, \{ P_F, \overline{P_{P \wedge \neg R}} \} \}$$

- $P_{P \vee Q} \leftrightarrow P \vee Q$ in CNF:

$$F_2 : \{ \{ \overline{P_{P \vee Q}}, P \vee Q \}, \{ P_{P \vee Q}, \overline{P} \}, \{ P_{P \vee Q}, \overline{Q} \} \}$$

- $P_{P \wedge \neg R} \leftrightarrow P \wedge P_{\neg R}$ in CNF:

$$F_3 : \{ \{ \overline{P_{P \wedge \neg R}} \vee P \}, \{ \overline{P_{P \wedge \neg R}}, P_{\neg R} \}, \{ P_{P \wedge \neg R}, \overline{P}, \overline{P_{\neg R}} \} \}$$

- $P_{\neg R} \leftrightarrow \neg R$ in CNF: $F_4 : \{ \{ \overline{P_{\neg R}}, \overline{R} \}, \{ P_{\neg R}, R \} \}$

$\{ \{ P_F \} \} \cup F_1 \cup F_2 \cup F_3 \cup F_4$ is in CNF and equisatisfiable to F .

- Algorithm to decide PL formulae in CNF.
- Published by Davis, Logemann, Loveland (1962).
- Often miscited as Davis, Putnam (1960), which describes a different algorithm.

Decides the satisfiability of PL formulae in CNF

Decision Procedure DPLL: Given F in CNF

```
let rec DPLL  $F$  =  
  let  $F'$  = PROP  $F$  in  
  let  $F''$  = PLP  $F'$  in  
  if  $F'' = \top$  then true  
  else if  $F'' = \perp$  then false  
  else  
    let  $P$  = CHOOSE vars( $F''$ ) in  
    (DPLL  $F''\{P \mapsto \top\}$ )  $\vee$  (DPLL  $F''\{P \mapsto \perp\}$ )
```

Unit Propagation (PROP)

If a clause contains one literal l ,

- Set l to \top .
- Remove all clauses containing l .
- Remove $\neg l$ in all clauses.

Based on resolution

$$\frac{l \quad \neg l \vee C}{C} \leftarrow \text{clause}$$

Pure Literal Propagation (PLP)

If P occurs only positive (without negation), set it to \top .

If P occurs only negative set it to \perp .

$$F : (\neg P \vee Q \vee R) \wedge (\neg Q \vee R) \wedge (\neg Q \vee \neg R) \wedge (P \vee \neg Q \vee \neg R)$$

Branching on Q

$$F\{Q \mapsto \top\} : (R) \wedge (\neg R) \wedge (P \vee \neg R)$$

By unit resolution

$$\frac{R \quad (\neg R)}{\perp}$$

$$F\{Q \mapsto \top\} = \perp \Rightarrow \text{false}$$

On the other branch

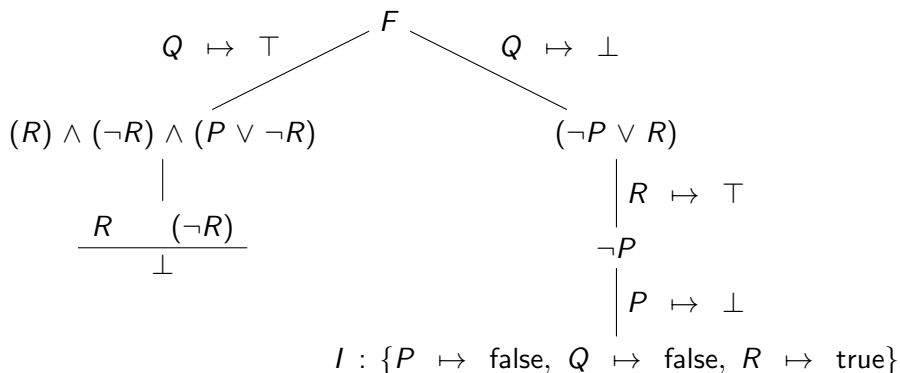
$$F\{Q \mapsto \perp\} : (\neg P \vee R)$$

$$F\{Q \mapsto \perp, R \mapsto \top, P \mapsto \perp\} = \top \Rightarrow \text{true}$$

F is satisfiable with satisfying interpretation

$$I : \{P \mapsto \text{false}, Q \mapsto \text{false}, R \mapsto \text{true}\}$$

$$F : (\neg P \vee Q \vee R) \wedge (\neg Q \vee R) \wedge (\neg Q \vee \neg R) \wedge (P \vee \neg Q \vee \neg R)$$



A island is inhabited only by knights and knaves. Knights always tell the truth, and knaves always lie. You meet four inhabitants: Alice, Bob, Charles and Doris.

- Alice says that Doris is a knave.
- Bob tells you that Alice is a knave.
- Charles claims that Alice is a knave.
- Doris tells you, 'Of Charles and Bob, exactly one is a knight.'

Let A denote that Alice is a Knight, etc. Then:

- $A \leftrightarrow \neg D$
- $B \leftrightarrow \neg A$
- $C \leftrightarrow \neg A$
- $D \leftrightarrow \neg(C \leftrightarrow B)$

In CNF:

- $\{\bar{A}, \bar{D}\}, \{A, D\}$
- $\{\bar{B}, \bar{A}\}, \{B, A\}$
- $\{\bar{C}, \bar{A}\}, \{C, A\}$
- $\{\bar{D}, \bar{C}, \bar{B}\}, \{\bar{D}, C, B\}, \{D, \bar{C}, B\}, \{D, C, \bar{B}\}$

$$F : \{ \{ \bar{A}, \bar{D} \}, \{ A, D \}, \{ \bar{B}, \bar{A} \}, \{ B, A \}, \{ \bar{C}, \bar{A} \}, \{ C, A \}, \\ \{ \bar{D}, \bar{C}, \bar{B} \}, \{ \bar{D}, C, B \}, \{ D, \bar{C}, B \}, \{ D, C, \bar{B} \} \}$$

PROP and PLP are not applicable. Decide on A:

$$F\{A \mapsto \perp\} : \{ \{ D \}, \{ B \}, \{ C \}, \{ \bar{D}, \bar{C}, \bar{B} \}, \{ \bar{D}, C, B \}, \{ D, \bar{C}, B \}, \{ D, C, \bar{B} \} \}$$

By PROP we get:

$$F\{A \mapsto \perp, D \mapsto \top, B \mapsto \top, C \mapsto \top\} : \perp$$

Unsatisfiable! Now set A to \top :

$$F\{A \mapsto \top\} : \{ \{ \bar{D} \}, \{ \bar{B} \}, \{ \bar{C} \}, \{ \bar{D}, \bar{C}, \bar{B} \}, \{ \bar{D}, C, B \}, \{ D, \bar{C}, B \}, \{ D, C, \bar{B} \} \}$$

By PROP we get:

$$F\{A \mapsto \top, D \mapsto \perp, B \mapsto \perp, C \mapsto \perp\} : \top$$

Satisfying assignment!

Consider the following problem:

$$\{\{A_1, B_1\}, \{\overline{P_0}, \overline{A_1}, P_1\}, \{\overline{P_0}, \overline{B_1}, P_1\}, \{A_2, B_2\}, \{\overline{P_1}, \overline{A_2}, P_2\}, \{\overline{P_1}, \overline{B_2}, P_2\}, \\ \dots, \{A_n, B_n\}, \{\overline{P_{n-1}}, \overline{A_n}, P_n\}, \{\overline{P_{n-1}}, \overline{B_n}, P_n\}, \{P_0\}, \{\overline{P_n}\}\}$$

For some literal orderings, we need exponentially many steps.

Note, that

$$\{\{A_i, B_i\}, \{\overline{P_{i-1}}, \overline{A_i}, P_i\}, \{\overline{P_{i-1}}, \overline{B_i}, P_i\}\} \Rightarrow \{\{\overline{P_{i-1}}, P_i\}\}$$

If we **learn** the right clauses, unit propagation will immediately give unsatisfiable.

Do not change the clause set, but only assign literals (as global variables).
When you assign true to a literal ℓ , also assign false to $\bar{\ell}$.

For a partial assignment

- A clause is true if one of its literals is assigned true.
- A clause is a **conflict clause** if all its literals are assigned false.
- A clause is a **unit clause** if all but one literals are assigned false and the last literal is unassigned.

If the assignment of a literal from a conflict clause is removed we get a unit clause.

Explain unsatisfiability of partial assignment by conflict clause and learn it!

Idea: Explain unsatisfiability of partial assignment by conflict clause and learn it!

- If a conflict is found we return the conflict clause.
- If variable in conflict were derived by unit propagation use resolution rule to generate a new conflict clause.
- If variable in conflict was derived by decision, use learned conflict as unit clause

The functions DPLL and PROP return a **conflict clause** or **satisfiable**.

```
let rec DPLL =
  let PROP U =
    ...
    if conflictclauses  $\neq \emptyset$ 
      CHOOSE conflictclauses
    else if unitclauses  $\neq \emptyset$ 
      PROP (CHOOSE unitclauses)
    else if coreclauses  $\neq \emptyset$ 
      let  $\ell = \text{CHOOSE} (\bigcup \text{coreclauses}) \cap \text{unassigned}$  in
      val[ $\ell$ ] :=  $\top$ 
      let C = DPLL in
      if (C = satisfiable) satisfiable
      else
        val[ $\ell$ ] := undef
        if ( $\ell \notin C$ ) C
        else LEARN C; PROP C
    else satisfiable
```


The function PROP takes a unit clause and does unit propagation. It calls DPLL recursively and returns a **conflict clause** or satisfiable. recursively:

```
let PROP U =  
  let l = CHOOSE U ∩ unassigned in  
  val[l] := ⊤  
  let C = DPLL in  
  if (C = satisfiable)  
    satisfiable  
  else  
    val[l] := undef  
    if ( $\bar{l} \notin C$ ) C  
    else  $U \setminus \{l\} \cup C \setminus \{\bar{l}\}$ 
```

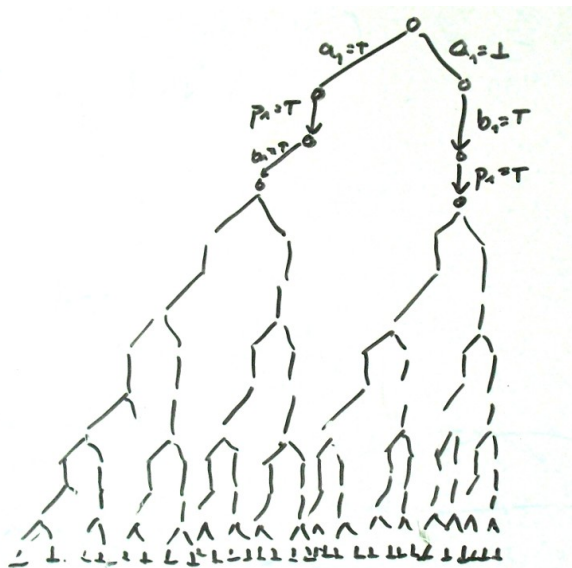
The last line does resolution:

$$\frac{l \vee C_1 \quad \neg l \vee C_2}{C_1 \vee C_2}$$

$$\{\{A_1, B_1\}, \{\overline{P_0}, \overline{A_1}, P_1\}, \{\overline{P_0}, \overline{B_1}, P_1\}, \{A_2, B_2\}, \{\overline{P_1}, \overline{A_2}, P_2\}, \{\overline{P_1}, \overline{B_2}, P_2\}, \\ \dots, \{A_n, B_n\}, \{\overline{P_{n-1}}, \overline{A_n}, P_n\}, \{\overline{P_{n-1}}, \overline{B_n}, P_n\}, \{P_0\}, \{\overline{P_n}\}\}$$

- Unit propagation (PROP) sets P_0 and $\overline{P_n}$ to true.
- Decide, e.g. A_1 , PROP sets $\overline{P_1}$
- Continue until A_{n-1} , PROP sets $\overline{P_{n-1}}, \overline{A_n}$ and $\overline{B_n}$
- Conflict clause computed: $\{\overline{A_{n-1}}, \overline{P_{n-2}}, P_n\}$.
- Conflict clause does not depend on A_1, \dots, A_{n-2} and can be used again.

DPLL (without Learning)





- Pure Literal Propagation is unnecessary:
A pure literal is always chosen right and never causes a conflict.
- Modern SAT-solvers use this procedure but differ in
 - heuristics to choose literals/clauses.
 - efficient data structures to find unit clauses.
 - better conflict resolution to minimize learned clauses.
 - restarts (without forgetting learned clauses).
- Even with the optimal heuristics DPLL is still exponential:
The Pidgeon-Hole problem requires exponential resolution proofs.

- Syntax and Semantics of Propositional Logic
- Methods to decide satisfiability/validity of formulae:
 - Truth table
 - Semantic Tableaux
 - DPLL
- Run-time of all algorithm is worst-case exponential in length of formula.
- Deciding satisfiability is NP-complete.