Decision Procedures

Jochen Hoenicke

Software Engineering
Albert-Ludwigs-University Freiburg

Summer 2013

Foundations: Propositional Logic

Syntax of Propositional Logic

Atom	truth symbols $\top$ ("true") and $\bot$ ("false")
	propositional variables $P, Q, R, P_1, Q_1, R_1, \cdots$
Literal	atom α or its negation $\neg\alpha$
Formula	literal or application of a
	logical connective 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)

Example: Syntax

formula F : $((P \wedge Q) \rightarrow (\top \vee \neg Q))$
atoms: $P, Q, \top$
literal: $\neg Q$
subformulas: $(P \wedge Q)$, $\quad (\top \vee \neg Q)$
abbreviation

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

Semantics (meaning) of PL

Formula F and Interpretation I is evaluated to a truth value $0/1$

where 0 corresponds to value false
1 true

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

Evaluation of logical operators:

F_1	F_2	$\neg F_1$	$F_1 \land F_2$	$F_1 \lor F_2$	$F_1 \rightarrow F_2$	$F_1 \leftrightarrow F_2$
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

Example: Semantics

$F : P \wedge Q \rightarrow P \vee \neg Q$
$I : \{P \mapsto 1, Q \mapsto 0\}$

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

$1 = \text{true}$ $\qquad$ $0 = \text{false}$

F evaluates to true under I

Inductive Definition of PL's Semantics

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

Base Case:

$I \models \top$
$I \not\models \bot$

$I \models P$ 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 $I \not\models F_2$

Example: Inductive Reasoning

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

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

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.

Satisfiability and Validity

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.

Examples: Satisfiability and Validity

Now assume, you are a decision procedure.

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

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

Is there a formula that is unsatisfiable and valid?

Method 1: Truth Tables

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

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	$\leftarrow$ satisfying I
0	1	1	0	0	$\leftarrow$ falsifying I
1	0	1	0	0	
1	1	1	1	1	

Thus F is satisfiable, but invalid.

Method 2: Semantic Argument (Semantic Tableaux)

- 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.

Semantic Argument: Proof rules

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

$$\frac{I \models F \land G}{\begin{array}{l} I \models F \\ I \models G \end{array}} \leftarrow \text{and} \qquad\qquad \frac{I \not\models F \land G}{I \not\models F \quad \mid \quad I \not\models G}$$

$$\nwarrow \text{or}$$

$$\frac{I \models F \lor G}{I \models F \quad \mid \quad I \models G} \qquad\qquad \frac{I \not\models F \lor G}{\begin{array}{l} I \not\models F \\ I \not\models G \end{array}} \qquad\qquad \frac{\begin{array}{l} I \models F \\ I \not\models F \end{array}}{I \models \bot}$$

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

$$\frac{I \models F \leftrightarrow G}{I \models F \land G \quad \mid \quad I \not\models F \lor G} \qquad\qquad \frac{I \not\models F \leftrightarrow G}{I \models F \land \neg G \quad \mid \quad I \models \neg F \land G}$$

Example

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

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

1.	$I \not\models P \wedge Q \to P \vee \neg Q$	assumption
2.	$I \models P \wedge Q$	1, Rule $\to$
3.	$I \not\models P \vee \neg Q$	1, Rule $\to$
4.	$I \models P$	2, Rule $\wedge$
5.	$I \not\models P$	3, Rule $\vee$
6.	$I \models \bot$	4 and 5 are contradictory

Thus F is valid.

Example 2

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

Let's assume that F is not valid.

$$
\begin{array}{lll}
1. & I \not\models F & \text{assumption} \\
2. & I \models (P \rightarrow Q) \land (Q \rightarrow R) & 1, \text{Rule} \rightarrow \\
3. & I \not\models P \rightarrow R & 1, \text{Rule} \rightarrow \\
4. & I \models P & 3, \text{Rule} \rightarrow \\
5. & I \not\models R & 3, \text{Rule} \rightarrow \\
6. & I \models P \rightarrow Q & 2, \text{Rule} \land \\
7. & I \models Q \rightarrow R & 2, \text{Rule} \land
\end{array}
$$

$$
\begin{array}{ll|l}
8a. & I \not\models P & 8b. \quad I \models Q \qquad 6 \rightarrow \\
9a. & I \models \bot & \begin{array}{ll|ll} 9ba. & I \not\models Q & 9bb. & I \models R \\ 10ba. & I \models \bot & 10bb. & I \models \bot \end{array}
\end{array}
$$

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

Example 3

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

Let's assume that F is not valid.

$$
\begin{array}{lll}
1. & I \not\models P \vee Q \rightarrow P \wedge Q & \text{assumption} \\
2. & I \models P \vee Q & 1 \text{ and } \rightarrow \\
3. & I \not\models P \wedge Q & 1 \text{ and } \rightarrow
\end{array}
$$

$$
\begin{array}{cc|cc}
\multicolumn{2}{c|}{4a. \quad I \models P \qquad 2 \text{ and } \vee} & \multicolumn{2}{c}{4b. \quad I \models Q \qquad 2 \text{ and } \vee} \\
\begin{array}{l} 5aa. \ I \not\models P \\ 6aa. \ I \models \bot \end{array} & \begin{array}{|l} 5ab. \ I \not\models Q \\ \ \end{array} & \begin{array}{l} 5ba. \ I \not\models P \\ \ \end{array} & \begin{array}{|l} 5bb. \ I \not\models Q \\ 6bb. \ I \models \bot \end{array}
\end{array}
$$

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

Falsifying interpretation:

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

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

Normal Forms

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.

Equivalence

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!

Equivalence is a Congruence relation

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 $F \Leftrightarrow F'$.

Negation Normal Form (NNF)

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):

$$\neg\neg F_1 \Leftrightarrow F_1 \qquad \neg\top \Leftrightarrow \bot \qquad \neg\bot \Leftrightarrow \top$$

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

$$F_1 \rightarrow F_2 \Leftrightarrow \neg F_1 \vee F_2$$

$$F_1 \leftrightarrow F_2 \Leftrightarrow (F_1 \rightarrow F_2) \wedge (F_2 \rightarrow F_1)$$

Example: Negation Normal Form

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

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

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

Disjunctive Normal Form (DNF)

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):

$$\left.\begin{array}{l} (F_1 \vee F_2) \wedge F_3 \Leftrightarrow (F_1 \wedge F_3) \vee (F_2 \wedge F_3) \\ F_1 \wedge (F_2 \vee F_3) \Leftrightarrow (F_1 \wedge F_2) \vee (F_1 \wedge F_3) \end{array}\right\} \mathit{dist}$$

Example

Convert $F : (Q_1 \lor \neg\neg R_1) \land (\neg Q_2 \to R_2)$ into DNF

$$\begin{array}{ll}
\ (Q_1 \lor \neg\neg R_1) \land (\neg Q_2 \to R_2) & \\
\Leftrightarrow (Q_1 \lor R_1) \land (Q_2 \lor R_2) & \text{in NNF} \\
\Leftrightarrow (Q_1 \land (Q_2 \lor R_2)) \lor (R_1 \land (Q_2 \lor R_2)) & \text{dist} \\
\Leftrightarrow (Q_1 \land Q_2) \lor (Q_1 \land R_2) \lor (R_1 \land Q_2) \lor (R_1 \land R_2) & \text{dist}
\end{array}$$

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

Conjunctive Normal Form (CNF)

Conjunction of disjunctions of literals

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

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

$$(F_1 \land F_2) \lor F_3 \Leftrightarrow (F_1 \lor F_3) \land (F_2 \lor F_3)$$
$$F_1 \lor (F_2 \land F_3) \Leftrightarrow (F_1 \lor F_2) \land (F_1 \lor F_3)$$

A disjunction of literals $P_1 \lor P_2 \lor \neg P_3$ 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).

Equisatisfiability

Definition (Equisatisfiability)

F and F' are equisatisfiable, iff

$$F \text{ is satisfiable if and only if } F' \text{ is satisfiable}$$

Every formula is equisatifiable to either $\top$ or $\bot$.

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.

Conversion to CNF

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

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

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.

Davis-Putnam-Logemann-Loveland (DPLL) Algorithm

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

Davis-Putnam-Logemann-Loveland (DPLL) Algorithm

Decides the satisfiability of PL formulae in CNF

<u>Decision Procedure DPLL</u>: Given F in CNF

$$
\begin{array}{l}
\texttt{let rec } \textsc{dpll}\ F = \\
\quad \texttt{let } F' = \textsc{prop}\ F \texttt{ in} \\
\quad \texttt{let } F'' = \textsc{plp}\ F' \texttt{ in} \\
\quad \texttt{if } F'' = \top \texttt{ then true} \\
\quad \texttt{else if } F'' = \bot \texttt{ then false} \\
\quad \texttt{else} \\
\qquad \texttt{let } P = \textsc{choose}\ \mathsf{vars}(F'') \texttt{ in} \\
\qquad (\textsc{dpll}\ F''\{P \mapsto \top\}) \lor (\textsc{dpll}\ F''\{P \mapsto \bot\})
\end{array}
$$

Unit Propagation (PROP)

If a clause contains one literal ℓ,

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

Based on resolution

$$\frac{\ell \qquad \neg\ell \vee C}{C} \leftarrow \text{clause}$$

Pure Literal Propagagion

Pure Literal Propagation (PLP)

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

Example

$$F : (\neg P \lor Q \lor R) \land (\neg Q \lor R) \land (\neg Q \lor \neg R) \land (P \lor \neg Q \lor \neg R)$$

Branching on Q

$$F\{Q \mapsto \top\} : (R) \land (\neg R) \land (P \lor \neg R)$$

By unit resolution

$$\frac{R \qquad (\neg R)}{\bot}$$

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

On the other branch

$$F\{Q \mapsto \bot\} : (\neg P \lor R)$$
$$F\{Q \mapsto \bot,\ R \mapsto \top,\ P \mapsto \bot\} = \top \Rightarrow \text{true}$$

F is satisfiable with satisfying interpretation

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

Example

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

$$F$$

$$Q \mapsto \top \qquad\qquad Q \mapsto \bot$$

$$(R) \wedge (\neg R) \wedge (P \vee \neg R) \qquad\qquad (\neg P \vee R)$$

$$\frac{R \qquad (\neg R)}{\bot}$$

$$R \mapsto \top$$

$$\neg P$$

$$P \mapsto \bot$$

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

Knight and Knaves

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.'

Knight and Knaves

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:

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

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

PROP and PLP are not applicable. Decide on A:

$$F\{A \mapsto \bot\} : \{\{D\}, \{B\}, \{C\}, \{\overline{D}, \overline{C}, \overline{B}\}, \{\overline{D}, C, B\}, \{D, \overline{C}, B\}, \{D, C, \overline{B}\}\}$$

PROP yields $F\{A \mapsto \bot, D \mapsto \top, B \mapsto \top, C \mapsto \top\} : \bot$
Unsatisfiable! Now set A to $\top$:

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

PROP yields $F\{A \mapsto \top, D \mapsto \bot, B \mapsto \bot, C \mapsto \bot\} : \top$
Satisfying assignment!

Learning is Useful

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.

Partial Assignments and Unit/Conflict Clauses

Instead of changing clause set, only remember the literal assignment. When you assign true to a literal ℓ,also assign false to $\overline{\ell}$.
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!

Conflict Driven Clause Learning (CDCL)

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

DPLL with Learning (CDCL)

We describe DPLL by a set of rules modifying a configuration.
A configuration is a triple

$$\langle M, F, C \rangle ,$$

where

- M (model) is a sequence of literals (that are currently set to true) interspersed with backtracking points denoted by $\square$.
- F (formula) is a formula in CNF,
  i. e., a set of clauses where each clause is a set of literals.
- C (conflict) is either $\top$ or a conflict clause (a set of literals).
  A conflict clause C is a clause with $F \Rightarrow C$ and $M \not\models C$.
  Thus, a conflict clause shows $M \not\models F$.

Rule Based Description

We describe the algorithm by a set of rules, which each describe a set of transitions between configurations, e. g.,

Explain $\dfrac{\langle M, F, C \cup \{\ell\}\rangle}{\langle M, F, C \cup \{\ell_1, \dots, \ell_k\}\rangle}$ where $\ell \notin C$, $\{\ell_1, \dots, \ell_k, \bar{\ell}\} \in F$, and $\bar{\ell_1}, \dots, \bar{\ell_k} \prec \bar{\ell}$ in M.

Here, $\bar{\ell_1}, \dots, \bar{\ell_k} \prec \ell$ in M means the literals $\bar{\ell_1}, \dots, \bar{\ell_k}$ occur in the sequence M before the literal ℓ (and all literals appear in M).

Example: for $M = P_1\bar{P}_3\bar{P}_2\bar{P}_4$, $F = \{\{P_1\}, \{P_3, \bar{P}_4\}\}$, and $C = \{P_2\}$ the transition

$$\langle M, F, \{P_2, P_4\}\rangle \longrightarrow \langle M, F, \{P_2, P_3\}\rangle$$

is possible.

Rules for CDCL (Conflict Driven Clause Learning)

Decide	$\dfrac{\langle M, F, \top\rangle}{\langle M \cdot \Box \cdot \ell, F, \top\rangle}$	where $\ell \in lit(F)$, $\ell, \overline{\ell}$ $\not\text{in}$ M
Propagate	$\dfrac{\langle M, F, \top\rangle}{\langle M \cdot \ell, F, \top\rangle}$	where $\{\ell_1, \dots, \ell_k, \ell\} \in F$ and $\overline{\ell_1}, \dots, \overline{\ell_k}$ in M, $\ell, \overline{\ell}$ $\not\text{in}$ M.
Conflict	$\dfrac{\langle M, F, \top\rangle}{\langle M, F, \{\ell_1, \dots, \ell_k\}\rangle}$	where $\{\ell_1, \dots, \ell_k\} \in F$ and $\overline{\ell_1}, \dots, \overline{\ell_k}$ in M.
Explain	$\dfrac{\langle M, F, C \cup \{\ell\}\rangle}{\langle M, F, C \cup \{\ell_1, \dots, \ell_k\}\rangle}$	where $\ell \notin C$, $\{\ell_1, \dots, \ell_k, \overline{\ell}\} \in F$, and $\overline{\ell_1}, \dots, \overline{\ell_k} \prec \overline{\ell}$ in M.
Learn	$\dfrac{\langle M, F, C\rangle}{\langle M, F \cup \{C\}, C\rangle}$	where $C \neq \top$, $C \notin F$.
Back	$\dfrac{\langle M, F, \{\ell_1, \dots, \ell_k, \ell\}\rangle}{\langle M' \cdot \ell, F, \top\rangle}$	where $\{\ell_1, \dots, \ell_k, \ell\} \in F$, $M = M' \cdot \Box \cdots \overline{\ell} \cdots$, and $\overline{\ell_1}, \dots, \overline{\ell_k}$ in M'.

Example: DPLL with Learning

$$P_1 \wedge (\neg P_2 \vee P_3) \wedge (\neg P_4 \vee P_3) \wedge (P_2 \vee P_4) \wedge (\neg P_1 \vee \neg P_4 \vee \neg P_3) \wedge (P_4 \vee \neg P_3)$$

The algorithm starts with $M = \epsilon$, $C = \top$ and
$F = \{\{P_1\}, \{\bar{P}_2, P_3\}, \{\bar{P}_4, P_3\}, \{P_2, P_4\}, \{\bar{P}_1, \bar{P}_4, \bar{P}_3\}, \{P_4, \bar{P}_3\}\}$.

$$\begin{aligned}
&\langle \epsilon, F, \top \rangle \xrightarrow{\text{Propagate}} \langle P_1, F, \top \rangle \xrightarrow{\text{Decide}} \langle P_1 \Box \bar{P}_2, F, \top \rangle \xrightarrow{\text{Propagate}} \\
&\langle P_1 \Box \bar{P}_2 P_4, F, \top \rangle \xrightarrow{\text{Propagate}} \langle P_1 \Box \bar{P}_2 P_4 P_3, F, \top \rangle \xrightarrow{\text{Conflict}} \\
&\langle P_1 \Box \bar{P}_2 P_4 P_3, F, \{\bar{P}_1, \bar{P}_4, \bar{P}_3\} \rangle \xrightarrow{\text{Explain}} \langle P_1 \Box \bar{P}_2 P_4 P_3, F, \{\bar{P}_1, \bar{P}_4\} \rangle \xrightarrow{\text{Learn}} \\
&\langle P_1 \Box \bar{P}_2 P_4 P_3, F', \{\bar{P}_1, \bar{P}_4\} \rangle \xrightarrow{\text{Back}} \langle P_1 \bar{P}_4, F', \top \rangle \xrightarrow{\text{Propagate}} \langle P_1 \bar{P}_4 P_2 P_3, F', \top \rangle \xrightarrow{\text{Conflict}} \\
&\langle P_1 \bar{P}_4 P_2 P_3, F', \{P_4, \bar{P}_3\} \rangle \xrightarrow{\text{Explain}} \langle P_1 \bar{P}_4 P_2 P_3, F', \{P_4, \bar{P}_2\} \rangle \xrightarrow{\text{Explain}} \\
&\langle P_1 \bar{P}_4 P_2 P_3, F', \{P_4\} \rangle \xrightarrow{\text{Explain}} \langle P_1 \bar{P}_4 P_2 P_3, F', \{\bar{P}_1\} \rangle \xrightarrow{\text{Explain}} \langle P_1 \bar{P}_4 P_2 P_3, F', \emptyset \rangle \xrightarrow{\text{Learn}} \\
&\langle P_1 \bar{P}_4 P_2 P_3, F' \cup \{\emptyset\}, \emptyset \rangle
\end{aligned}$$

where $F' = F \cup \{\{\bar{P}_1, \bar{P}_4\}\}$.

Example

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

- Unit propagation sets P_0 and $\overline{P_n}$: $\langle \epsilon, F, \top\rangle \xrightarrow{\text{Propagate}} \langle P_0 \overline{P_n}, F, \top\rangle$
- $\xrightarrow{\text{Decide } A_1} \xrightarrow{\text{Propagate } P_1} \langle P_0 \overline{P_n} \square A_1 P_1, F, \top\rangle$
- Continue until $\langle P_0 \overline{P_n} \square A_1 P_1 \dots \square A_{n-2} P_{n-1}, F, \top\rangle$.
- $\xrightarrow{\text{Propagate } \overline{A_n} \text{ and } \overline{B_n}} \langle \dots A_{n-2} P_{n-1} \overline{A_n B_n}, F, \top\rangle$.
- $\xrightarrow{\text{Conflict}} \langle \dots \square A_{n-2} P_{n-1} \overline{A_n B_n}, F, \{A_n, B_n\}\rangle$
  $\xrightarrow{\text{Explain}^*} \langle \dots \square A_{n-2} P_{n-1} \overline{A_n B_n}, F, \{\overline{P_{n-1}}, P_n\}\rangle$.
- The explained clause can be learned. One can now backtrack to the first decision point: $\xrightarrow{\text{Learn}} \xrightarrow{\text{Back}} \langle P_0 \overline{P_n P_{n-1}}, F', \top\rangle$.

Correctness of CDCL

Theorem (Correctness of CDCL)

Let F be a propositional formula in CNF. Let

$$\langle \epsilon, F, \top \rangle = \langle M_0, F_0, C_0 \rangle \longrightarrow \ldots \longrightarrow \langle M_n, F_n, C_n \rangle$$

be a maximal sequence of rule application of CDCL. Then F is satisfiable iff C_n is $\top$.

Before proving the theorem, we note some important invariants:

- M_i never contains a literal more than once.
- M_i never contains ℓ and $\overline{\ell}$.
- Every $\square$ in M_i is followed immediately by a literal.
- If $C_i = \{\ell_1, \ldots, \ell_k\}$ then $\overline{\ell_1}, \ldots, \overline{\ell_k}$ in M.
- C_i is always logically implied by F_i.
- F is equivalent to F_i for all steps i of the computation.
- If a literal ℓ in M is not immediately preceded by $\square$, then F contains a clause $\{\ell, \ell_1, \ldots, \ell_k\}$ and $\overline{\ell_1}, \ldots, \overline{\ell_k} \prec \ell$ in M.

Correctness proof

Proof: If the sequence ends with $\langle M_n, F_n, \top\rangle$ and there is no rule applicable, then:

- Since Decide is not applicable, all literals of F_n appear in M_n either positively or negatively.
- Since Conflict is not applicable, for each clause at least one literal appears in M_n positively.

Thus, M_n is a model for F_n, which is equivalent to F.

If the sequence ends with $\langle M_n, F_n, C_n\rangle$ with $C_n \neq \top$.
Assume $C_n = \{\ell_1, \dots, \ell_k, \ell\} \neq \emptyset$. W.l.o.g., $\bar{\ell}_1, \dots, \bar{\ell}_k \prec \bar{\ell}$. Then:

- Since Learn is not applicable, $C_n \in F_n$.
- Since Explain is not applicable $\bar{\ell}$ must be immediately preceded by $\square$.
- However, then Back is applicable, contradiction!

Therefore, the assumption was wrong and $C_n = \emptyset (= \bot)$.
Since F_n implies C_n and F is equivalent to F_n, F is not satisfiable.

Functional Implementation of CDCL

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

```
let rec DPLL =
  let PROP U =
    ...
  if conflictclauses ≠ ∅
    CHOOSE conflictclauses
  else if unitclauses ≠ ∅
    PROP (CHOOSE unitclauses)
  else if coreclauses ≠ ∅
     let ℓ = CHOOSE (⋃coreclauses) ∩ unassigned in
     val[ℓ] := ⊤
     let C = DPLL in
     if (C = satisfiable) satisfiable
     else
        val[ℓ] := undef
        if (ℓ̄ ∉ C) C
        else LEARN C; PROP C
  else satisfiable
```

Unit propagation

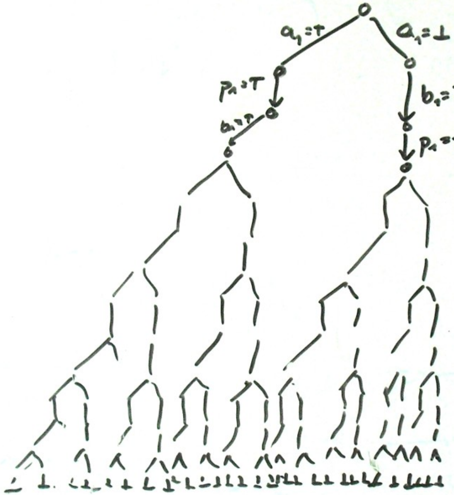
The function PROP takes a unit clause and does unit propagation. It calls DPLL recursively and returns a conflict clause or satisfiable. recursively:

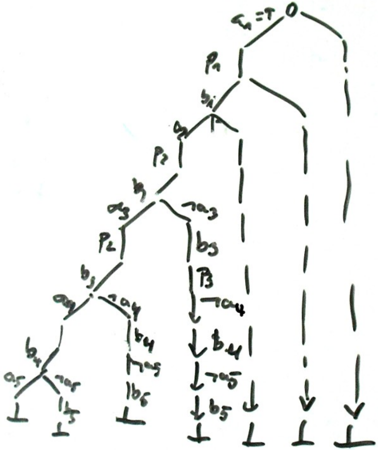
```
let PROP U =
  let ℓ = CHOOSE U ∩ unassigned in
  val[ℓ] := ⊤
  let C = DPLL in
  if (C = satisfiable)
    satisfiable
  else
    val[ℓ] := undef
    if (ℓ̄ ∉ C) C
    else  U \ {ℓ} ∪ C \ {ℓ̄}
```

The last line does resolution:

$$\frac{\ell \lor C_1 \qquad \neg\ell \lor C_2}{C_1 \lor C_2}$$

DPLL versus CDCL





Some Notes about DPLL with 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.

Summary

- 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.