Decision Procedures

Jochen Hoenicke

Software Engineering
Albert-Ludwigs-University Freiburg

Winter Term 2019/2020

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: $(P \wedge Q)$, $(\top \vee \neg Q)$

Parentheses can be omitted: $F : P \wedge Q \rightarrow \top \vee \neg Q$

- $\neg$ binds stronger than
- $\wedge$ binds stronger than
- $\vee$ binds stronger than
- $\rightarrow, \leftrightarrow$.

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

$F : P \land Q \to P \lor \neg Q$
$I : \{P \mapsto 1, Q \mapsto 0\}$

P	Q	$\neg Q$	$P \land Q$	$P \lor \neg Q$	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.

Remark: Functional Programming

Formulas can be embedded in functional languages, e.g.

$$\texttt{datatype}\ \textbf{fml} = \textsc{Var}\ \texttt{of}\ \textbf{int} \mid \textsc{False} \mid \textsc{True} \mid \textsc{Not}\ \texttt{of}\ \textbf{fml} \mid \textsc{And}\ \texttt{of}\ \textbf{fml} * \textbf{fml} \mid \textsc{Or}\ \texttt{of}\ \textbf{fml} * \textbf{fml} \mid \textsc{Impl}\ \texttt{of}\ \textbf{fml} * \textbf{fml} \mid \textsc{Iff}\ \texttt{of}\ \textbf{fml} * \textbf{fml}$$

The evaluation operator $\models$ can be implemented by a recursive function:

```
let rec EVAL (I : int → bool) (F : fml) =
  match F with
  | VAR x        →  (I x)
  | TRUE         →  true
  | FALSE        →  false
  | NOT F1       →  (not (EVAL I F1))
  | AND F1 F2    →  (EVAL I F1) & (EVAL I F2)
  | OR F1 F2     →  (EVAL I F1) | (EVAL I F2)
  | IMPL F1 F2   →  (not (EVAL I F1)) | (EVAL I F2)
  | IFF F1 F2    →  (EVAL I (IMPL F1 F2)) & (EVAL I (IMPL F2 F1))
```

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?

- $F_1 \; : \; P \wedge Q$
  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?

We will present three Decision Procedures for propositional logic

- Truth Tables
- Semantic Argument
- DPLL/CDCL

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

- 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

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

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

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

$$\frac{I \not\models F \land G}{I \not\models F \quad \mid \quad I \not\models G} \quad \nwarrow \text{or}$$

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

$$\frac{I \models F \lor G}{I \models F \quad \mid \quad I \models G}$$

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

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

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

$$\frac{I \models F \leftrightarrow G}{\begin{array}{l} I \models F \\ I \models G \end{array} \quad \Bigg| \quad \begin{array}{l} I \not\models F \\ I \not\models G \end{array}}$$

$$\frac{I \not\models F \leftrightarrow G}{\begin{array}{l} I \models F \\ I \not\models G \end{array} \quad \Bigg| \quad \begin{array}{l} I \not\models F \\ I \models G \end{array}}$$

Example

Prove $F : P \land Q \rightarrow P \lor \neg Q$ is valid.

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

1.	$I \not\models P \land Q \rightarrow P \lor \neg Q$	assumption
2.	$I \models P \land Q$	1, Rule $\rightarrow$
3.	$I \not\models P \lor \neg Q$	1, Rule $\rightarrow$
4.	$I \models P$	2, Rule $\land$
5.	$I \not\models P$	3, Rule $\lor$
6.	$I \models \bot$	4 and 5 are contradictory

Thus F is valid.

Example 2

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

Let's assume that F is not valid.

1.	$I \not\models F$	assumption
2.	$I \models (P \rightarrow Q) \land (Q \rightarrow R)$	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 $\land$
7.	$I \models Q \rightarrow R$	2, Rule $\land$

$8a.\ I \not\models P$	$8b.\ I \models Q$ $6 \rightarrow$	
$9a.\ I \models \bot$	$9ba.\ I \not\models Q$	$9bb.\ I \models R$
	$10ba.\ I \models \bot$	$10bb.\ I \models \bot$

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

Example 3

Is $F : P \lor Q \rightarrow P \land Q$ valid?

Let's assume that F is not valid.

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

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

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

Falsifying interpretation:

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

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

Method 3: DPLL/CDCL

DPLL/CDCL is a efficient decision procedure for propositional logic.
History:

- 1960s: Davis, Putnam, Logemann, and Loveland presented DPLL.
- 1990s: Conflict Driven Clause Learning (CDCL).
- Today, very efficient solvers using specialized data structures and improved heuristics.

DPLL/CDCL doesn't work on arbitrary formulas, but only on a certain normal form.

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):

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

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

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

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

Example: Negation Normal Form

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

$$
\begin{aligned}
& (Q_1 \lor \neg\neg R_1) \land (\neg Q_2 \to R_2) \\
\Leftrightarrow\ & (Q_1 \lor R_1) \land (\neg Q_2 \to R_2) \\
\Leftrightarrow\ & (Q_1 \lor R_1) \land (\neg\neg Q_2 \lor R_2) \\
\Leftrightarrow\ & (Q_1 \lor R_1) \land (Q_2 \lor R_2)
\end{aligned}
$$

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

Is this a (deterministic) algorithm?

- static finiteness: Can the algorithm be described in finite space?
- dynamic finiteness: Does the algorithm use finite space?
- termination: Does the algorithm run in finite time?
- deterministic: the order of steps determined?
- deterministic result: is the result always the same?

termination: Yes, but not obvious.
deterministic: No
deterministic result: Yes (not obvious)

NNF in ML

```
let rec NNF (F : fml) =
  match F with
  | Not True             →  False        | Not False → True
  | Not (Not F1)         →  NNF F1
  | Not (And F1 F2)      →  Or (NNF (Not F1)) (NNF (Not F2))
  | Not (Or F1 F2)       →  And (NNF (Not F1)) (NNF (Not F2))
  | Not (Impl F1 F2)     →  And (NNF F1) (NNF (Not F2))
  | Not (Iff F1 F2)      →  Or  (And (NNF F1) (NNF (Not F2)))
                                (And (NNF (Not F1)) (NNF F2))
  | And F1 F2            →  And (NNF F1) (NNF F2)
  | Or F1 F2             →  Or (NNF F1) (NNF F2)
  | Impl F1 F2           →  Or (NNF (Not F1)) (NNF F2)
  | Iff F1 F2            →  And  (Or (NNF (Not F1)) (NNF F2))
                                 (Or (NNF F1) (NNF (Not F2)))
  | _                    →  F
```

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):

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

Example

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

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

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

Conjunction of disjunctions of literals

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

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

$$(F_1 \wedge F_2) \vee F_3 \Leftrightarrow (F_1 \vee F_3) \wedge (F_2 \vee F_3)$$
$$F_1 \vee (F_2 \wedge F_3) \Leftrightarrow (F_1 \vee F_2) \wedge (F_1 \vee F_3)$$

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).

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 equisatisfiable 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!

Example: CNF

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

- $P_F \leftrightarrow (P_{P\vee Q} \to P_{P\wedge\neg R})$ 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

Decision Procedure DPLL: Given F in CNF

```
let rec DPLL F =
  let F' = PROP F in
  let F'' = PLP F' in
  if F'' = ⊤ then true
  else if F'' = ⊥ then false
  else
     let P = CHOOSE vars(F'') in
     (DPLL F''{P ↦ ⊤}) ∨ (DPLL F''{P ↦ ⊥})
```

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

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

Branching on Q

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

By unit resolution

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

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

On the other branch

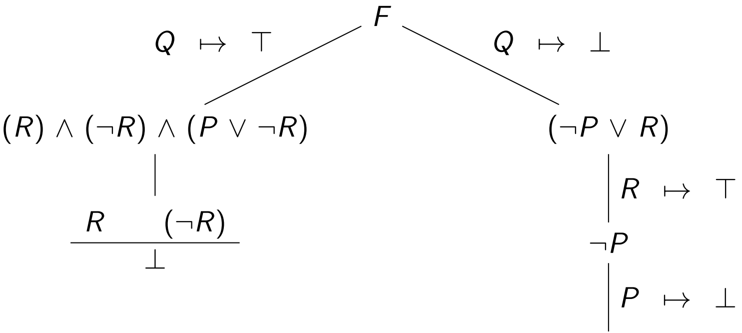
$F\{Q \mapsto \bot\} : (\neg P \vee R)$

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

F is satisfiable with satisfying interpretation

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

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



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

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:

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

Solving Knights and Knaves

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

PROP and PLP are not applicable. Decide on A:

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

By PROP we get:

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

Unsatisfiable! Now set A to $\top$:

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

By PROP we get:

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

Satisfying assignment!

Learning is Useful

Consider the following problem:

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

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

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!

Conflict Driven Clause Learning (CDCL)

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

- If a conflict is found we remember the conflict clause.
- If variable in conflict was derived by unit propagation use the resolution rule to generate a new conflict clause.

$$\frac{\ell \lor C_1 \qquad \neg\ell \lor C_2}{C_1 \lor C_2} \qquad \text{(resolution rule)}$$

- If variable in conflict was derived by decision, use learned conflict as unit clause

DPLL with Learning (CDCL)

We describe DPLL 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) annotated with $\square$ for decisions or a clause for unit propagation.
- 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 $\dfrac{\langle M, F, C \cup \{\bar{\ell}\}\rangle}{\langle M, F, C \cup \{\ell_1, \ldots, \ell_k\}\rangle}$ where $\bar{\ell} \notin C$, ℓ^{C_ℓ} in M, and $C_\ell = \{\ell_1, \ldots, \ell_k, \ell\}$.

Here, ℓ^{C_ℓ} in M means that the literal ℓ occurs in M annotated with the clause C_ℓ.

Example: for $C_1 = \{P_1\}, C_2 = \{P_3, \overline{P_4}\}$, $M = P_1^{C_1}\overline{P_3}^{\square}\overline{P_2}^{\square}\overline{P_4}^{C_2}$, $F = \{C_1, C_2\}$, 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	$\dfrac{\langle M, F, \top\rangle}{\langle M \cdot \ell^{\Box}, F, \top\rangle}$	where $\ell \in \mathit{lit}(F)$, $\ell, \overline{\ell}$ $\not\in$ M
Propagate	$\dfrac{\langle M, F, \top\rangle}{\langle M \cdot \ell^{C_\ell}, F, \top\rangle}$	where $C_\ell = \{\ell_1, \dots, \ell_k, \ell\} \in F$ with $\overline{\ell_1}, \dots, \overline{\ell_k}$ in M, $\ell, \overline{\ell}$ $\not\in$ 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	$\dfrac{\langle M, F, C \cup \{\overline{\ell}\}\rangle}{\langle M, F, C \cup \{\ell_1, \dots, \ell_k\}\rangle}$	where $\overline{\ell} \notin C$, ℓ^{C_ℓ} in M, and $C_\ell = \{\ell_1, \dots, \ell_k, \ell\}$.
Learn	$\dfrac{\langle M, F, C\rangle}{\langle M, F \cup \{C\}, C\rangle}$	where $C \neq \top$, $C \notin F$.
Back	$\dfrac{\langle M, F, C_\ell\rangle}{\langle M' \cdot \ell^{C_\ell}, F, \top\rangle}$	where $C_\ell = \{\ell_1, \dots, \ell_k, \ell\} \in F$, $M = M' \cdot \ell'^{\Box} \cdots$, and $\overline{\ell_1}, \dots, \overline{\ell_k}$ in M', $\overline{\ell}$ $\not\in$ M'.

A run of DPLL is a maximal sequence of configurations

$$\langle M_0, F_0, C_0\rangle \rightarrow \langle M_1, F_1, C_1\rangle \rightarrow \dots$$

starting with $M_0 = \epsilon$, F the input formula in CNF, and $C_0 = \top$, and where each transition follows one of the six rules.
If the run ends with $\emptyset \in F$, the formula is unsatisfiable. Otherwise it is satisfiable and the last M gives an interpretation for the input formula F.

Example: Knights and Knaves

$F = \{C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, C_9, C_{10}\}$ with $C_1 = \{\overline{A}, \overline{D}\}$,
$C_2 = \{A, D\}$, $C_3 = \{\overline{B}, \overline{A}\}$, $C_4 = \{B, A\}$, $C_5 = \{\overline{C}, \overline{A}\}$, $C_6 = \{C, A\}$,
$C_7 = \{\overline{D}, \overline{C}, \overline{B}\}$, $C_8 = \{\overline{D}, C, B\}$, $C_9 = \{D, \overline{C}, B\}$, $C_{10} = \{D, C, \overline{B}\}$.

$\langle \epsilon, F, \top \rangle \xrightarrow{\text{Decide}} \langle \overline{A}^{\Box}, F, \top \rangle \xrightarrow{\text{Propagate}} \langle \overline{A}^{\Box} D^{C_2}, F, \top \rangle \xrightarrow{\text{Propagate}}$
$\langle \overline{A}^{\Box} D^{C_2} B^{C_4}, F, \top \rangle \xrightarrow{\text{Propagate}} \langle \overline{A}^{\Box} D^{C_2} B^{C_4} C^{C_6}, F, \top \rangle \xrightarrow{\text{Conflict}}$
$\langle \overline{A}^{\Box} D^{C_2} B^{C_4} C^{C_6}, F, \{\overline{D}, \overline{C}, \overline{B}\} \rangle \xrightarrow{\text{Explain}}$
$\langle \overline{A}^{\Box} D^{C_2} B^{C_4} C^{C_6}, F, \{A, \overline{D}, \overline{B}\} \rangle \xrightarrow{\text{Explain}} \langle \overline{A}^{\Box} D^{C_2} B^{C_4} C^{C_6}, F, \{A, \overline{B}\} \rangle \xrightarrow{\text{Explain}}$
$\langle \overline{A}^{\Box} D^{C_2} B^{C_4} C^{C_6}, F, \{A\} \rangle \xrightarrow{\text{Learn}} \langle \overline{A}^{\Box} D^{C_2} B^{C_4} C^{C_6}, F', \{A\} \rangle \xrightarrow{\text{Back}}$
$\langle A^{\{A\}}, F', \top \rangle \xrightarrow{\text{Propagate}} \langle A^{\{A\}} \overline{D}^{C_1}, F', \top \rangle \xrightarrow{\text{Propagate}}$
$\langle A^{\{A\}} \overline{D}^{C_1} \overline{B}^{C_3}, F', \top \rangle \xrightarrow{\text{Propagate}} \langle A^{\{A\}} \overline{D}^{C_1} \overline{B}^{C_3} \overline{C}^{C_5}, F', \top \rangle$

where $F' = F \cup \{A\}$.

Example: DPLL with Learning

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

$F = \{C_1, C_2, C_3, C_4, C_5, C_6\}$ with $C_1 = \{P_1\}$, $C_2 = \{\overline{P_2}, P_3\}$, $C_3 = \{\overline{P_4}, P_3\}$, $C_4 = \{P_2, P_4\}$, $C_5 = \{\overline{P_1}, \overline{P_4}, \overline{P_3}\}$, $C_6 = \{P_4, \overline{P_3}\}$.

$\langle \epsilon, F, \top \rangle \xrightarrow{\text{Propagate}} \langle P_1^{C_1}, F, \top \rangle \xrightarrow{\text{Decide}} \langle P_1^{C_1} \overline{P_2}^{\Box}, F, \top \rangle \xrightarrow{\text{Propagate}}$
$\langle P_1^{C_1} \overline{P_2}^{\Box} P_4^{C_4}, F, \top \rangle \xrightarrow{\text{Propagate}} \langle P_1^{C_1} \overline{P_2}^{\Box} P_4^{C_4} P_3^{C_3}, F, \top \rangle \xrightarrow{\text{Conflict}}$
$\langle P_1^{C_1} \overline{P_2}^{\Box} P_4^{C_4} P_3^{C_3}, F, \{\overline{P_1}, \overline{P_4}, \overline{P_3}\} \rangle \xrightarrow{\text{Explain}}$
$\langle P_1^{C_1} \overline{P_2}^{\Box} P_4^{C_4} P_3^{C_3}, F, \{\overline{P_1}, \overline{P_4}\} \rangle \xrightarrow{\text{Learn}} \langle P_1^{C_1} \overline{P_2}^{\Box} P_4^{C_4} P_3^{C_3}, F', \{\overline{P_1}, \overline{P_4}\} \rangle \xrightarrow{\text{Back}}$
$\langle P_1^{C_1} \overline{P_4}^{C_7}, F', \top \rangle \xrightarrow{\text{Propagate}} \langle P_1^{C_1} \overline{P_4}^{C_7} P_2^{C_4}, F', \top \rangle \xrightarrow{\text{Propagate}}$
$\langle P_1^{C_1} \overline{P_4}^{C_7} P_2^{C_4} P_3^{C_2}, F', \top \rangle \xrightarrow{\text{Conflict}} \langle P_1^{C_1} \overline{P_4}^{C_7} P_2^{C_4} P_3^{C_2}, F', \{P_4, \overline{P_3}\} \rangle \xrightarrow{\text{Explain}}$
$\langle P_1^{C_1} \overline{P_4}^{C_7} P_2^{C_4} P_3^{C_2}, F', \{P_4, \overline{P_2}\} \rangle \xrightarrow{\text{Explain}} \langle P_1^{C_1} \overline{P_4}^{C_7} P_2^{C_4} P_3^{C_2}, F', \{P_4\} \rangle \xrightarrow{\text{Explain}}$
$\langle P_1^{C_1} \overline{P_4}^{C_7} P_2^{C_4} P_3^{C_2}, F', \{\overline{P_1}\} \rangle \xrightarrow{\text{Explain}} \langle P_1^{C_1} \overline{P_4}^{C_7} P_2^{C_4} P_3^{C_2}, F', \emptyset \rangle \xrightarrow{\text{Learn}}$
$\langle P_1^{C_1} \overline{P_4}^{C_7} P_2^{C_4} P_3^{C_2}, F' \cup \{\emptyset\}, \emptyset \rangle$ where $C_7 = \{\overline{P_1}, \overline{P_4}\}$, $F' = F \cup \{C_7\}$.

Correctness of DPLL (with Learning)

Theorem (Correctness of DPLL)

Let F be a Σ-formula and F' its propositional core. 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 DPLL.

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 $\overline{\ell}$.
- If $M_i = M' \ell^{C_\ell} \cdots$, then $C_\ell = \{\ell_1, \ldots, \ell_k, \ell\}$ with $\overline{\ell_1}, \ldots, \overline{\ell_k}$ in M' and $C_\ell \in F_i$.
- Every $\ell \in C_i$ occurs negated in M_i.
- C_i is always implied by F_i.
- F is equivalent to F_i for all steps i of the computation.

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 $C_n = \{\ell_1, \dots, \ell_k, \ell\} \neq \emptyset$. Note that $\overline{\ell_1}, \dots, \overline{\ell_k}, \overline{\ell}$ in M.
W.l.o.g., $\overline{\ell}$ is the last one that occurs in M. Then:

- Since Learn is not applicable, $C_n \in F_n$.
- Since Explain is not applicable $\overline{\ell}$ must be annotated with $\Box$.
- However, then Back is applicable, contradiction!

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

Theorem (Termination of DPLL)

Let F be a propositional formula. Then every sequence

$$\langle \epsilon, F, \top \rangle = \langle M_0, F_0, C_0 \rangle \longrightarrow \langle M_1, F_1, C_1 \rangle \longrightarrow \dots$$

terminates.

Proof of Termination

There are finitely many literals, therefore,

- finitely many clauses C,
- finitely many sequences M of literals annotated with clauses
- finitely many sets of clauses F.

Since everything is finite, it is sufficient to show that there is no cycle, by defining a partial ordering.

- We define $M \prec M'$ if $M\$$ comes lexicographically before $M'\$$, where ℓ^C is smaller than $\ell'^{\square}$ and $\$$ is considered to be the largest symbol.
  Example: $\ell_1^{C_1}\ell_2^{C_2}\$ \prec \ell_1^{C_1}\ell_3^{\square}\ell_4^{C_4}\$ \prec \ell_1^{C_1}\ell_3^{\square}\$ \prec \ell_1^{C_1}\$$
- For a sequence $M = \overline{\ell_1} \ldots \overline{\ell_n}$, the conflict clauses are ordered by their weight w: $w(\top) = 2^{n+1}$, $w(C) = \sum_{\ell_i \in C} 2^i$, $w(\emptyset) = 0$.
  The weight depends on the order in which the literals occur in M.
  Example: $\emptyset \prec_{\overline{\ell_1\ell_2\ell_3}} \{\ell_1, \ell_2\} \prec_{\overline{\ell_1\ell_2\ell_3}} \{\ell_3\} \prec_{\overline{\ell_1\ell_2\ell_3}} \{\ell_2, \ell_3\} \prec_{\overline{\ell_1\ell_2\ell_3}} \top$

These are well-orderings, because the domains are finite.

Proof of Termination (cont.)

Termination Proof: Every rule application decreases the value of $\langle M_i, F_i, C_i \rangle$ according to the well-ordering:

$$\langle M, F, C \rangle \prec \langle M', F', C' \rangle, \text{ iff } \begin{cases} M \prec M', \\ \text{or } M = M', C \prec_M C', \\ \text{or } M = M', C = C', F \supsetneq F'. \end{cases}$$

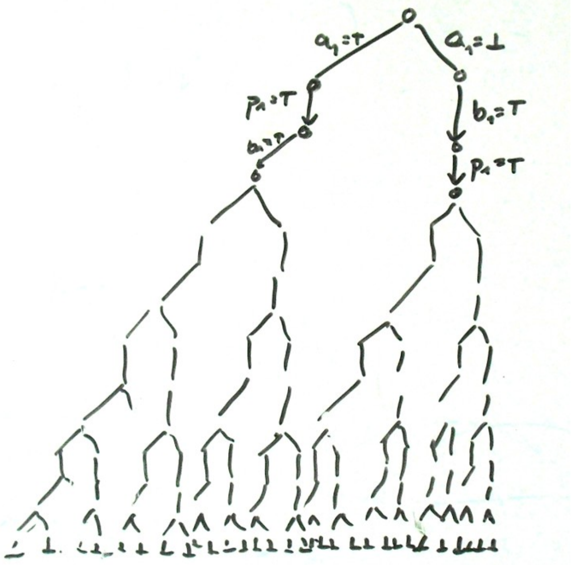
Hence there is no cycle and the DPLL algorithm terminates.

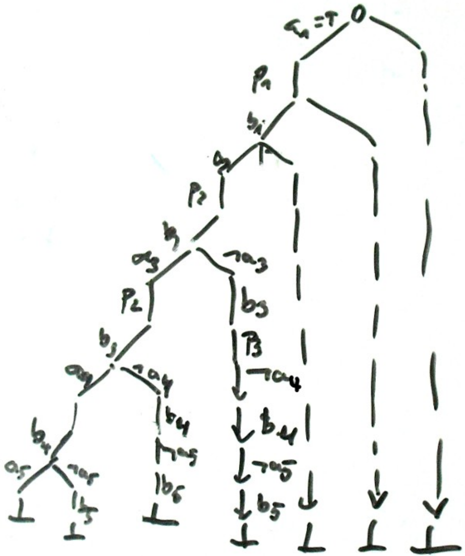
Example

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

- Unit propagation sets P_0 and $\overline{P_n}$ to true.
- Decide, e.g. A_1, then propagate $\overline{P_1}$
- Continue until A_{n-1}, then propagate $\overline{P_{n-1}}$,$\overline{A_n}$ and $\overline{B_n}$
- Conflict: $\{A_n, B_n\}$.
- Explain computes new conflict clause: $\{\overline{P_{n-1}}, P_n\}$.
- Conflict clause does not depend on $A_1, \dots, A_{n-1}$ and can be used again.

DPLL (without Learning)





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 Argument
  - DPLL
- Run-time of all presented algorithms is worst-case exponential in length of formula.
- Deciding satisfiability is NP-complete.