EC4219: Software Engineering

Lecture 4 — Propositional Logic (2)

Normal Forms and DPLL

Sunbeom So

2024 Spring

Overview

- Goal: An algorithm called *DPLL* for determining satisfiability.
  - Many SAT solvers used today are based on DPLL.
  - SAT solver: software that solves the boolean satisfiability problem (theorem prover for propositional logic)
- DPLL requires converting formulas to a representation called *normal forms*.
- Thus, we will study normal forms first and then DPLL.

Normal Forms

- A normal form of formulas is a certain syntactic restriction such that there is an equivalent formula F' for every formula F of the logic.
- Three normal forms are particularly important for propositional logic.
  - Negation Normal Form (NNF)
  - Disjunctive Normal Form (DNF)
  - Conjunctive Normal Form (CNF)

Negation Normal Form (NNF)

- NNF requires that $\neg$, $\land$, and $\lor$ are the only connectives (i.e., no $\rightarrow$ and $\leftrightarrow$) and that negations appear only in literals (i.e., negations appear only in front of atoms).
  - Is $P \land Q \land (R \lor \neg S)$ in NNF?

Negation Normal Form (NNF)

- NNF requires that $\neg$, $\wedge$, and $\vee$ are the only connectives (i.e., no $\rightarrow$ and $\leftrightarrow$) and that negations appear only in literals (i.e., negations appear only in front of atoms).
  - Is $P \wedge Q \wedge (R \vee \neg S)$ in NNF? **O**
  - Is $\neg P \vee \neg(P \wedge Q))$ in NNF?

Negation Normal Form (NNF)

- NNF requires that $\neg$, $\wedge$, and $\vee$ are the only connectives (i.e., no $\rightarrow$ and $\leftrightarrow$) and that negations appear only in literals (i.e., negations appear only in front of atoms).
  - Is $P \wedge Q \wedge (R \vee \neg S)$ in NNF? **O**
  - Is $\neg P \vee \neg(P \wedge Q))$ in NNF? **X**
  - Is $\neg\neg P \wedge Q$ in NNF?

Negation Normal Form (NNF)

- NNF requires that $\neg$, $\wedge$, and $\vee$ are the only connectives (i.e., no $\rightarrow$ and $\leftrightarrow$) and that negations appear only in literals (i.e., negations appear only in front of atoms).
  - Is $P \wedge Q \wedge (R \vee \neg S)$ in NNF? **O**
  - Is $\neg P \vee \neg(P \wedge Q))$ in NNF? **X**
  - Is $\neg\neg P \wedge Q$ in NNF? **X**
- Transforming a formula F to an equivalent formula F' in NNF can be done by repeatedly applying the list of template equivalences below:

$$
\begin{array}{rcl}
\neg\neg F_1 & \iff & F_1 \\
\neg\top & \iff & \bot \\
\neg\bot & \iff & \top \\
\neg(F_1 \wedge F_2) & \iff & \neg F_1 \vee \neg F_2 \\
\neg(F_1 \vee F_2) & \iff & \neg F_1 \wedge \neg F_2 \\
F_1 \rightarrow F_2 & \iff & \neg F_1 \vee F_2 \\
F_1 \leftrightarrow F_2 & \iff & (F_1 \rightarrow F_2) \wedge (F_2 \rightarrow F_1)
\end{array}
$$

Example: NNF

Convert $F : \neg(P \rightarrow \neg(P \wedge Q))$ into NNF.

- By the template equivalence $F_1 \rightarrow F_2 \iff \neg F_1 \vee F_2$, we produce

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

- Applying the template $\neg(F_1 \vee F_2) \iff \neg F_1 \wedge \neg F_2$, we produce

$$F'' : \neg\neg P \wedge \neg\neg(P \wedge Q)$$

- Finally, applying the template equivalence $\neg\neg F_1 \iff F_1$ to each conjunct produces

$$F''' : P \wedge P \wedge Q$$

F''' is in NNF and is equivalent to F.

Disjunctive Normal Form (DNF)

- A formula is in disjunctive normal form (DNF) if it is a disjunction of conjunctive clauses (conjunctions of literals):

$$\bigvee_i \bigwedge_j l_{i,j}$$

- To convert a formula F into an equivalent formula in DNF,
  (1) transform F into NNF, and then
  (2) distribute conjunctions over disjunctions:

$$\begin{array}{rcl} (F_1 \lor F_2) \land F_3 & \iff & (F_1 \land F_3) \lor (F_2 \land F_3) \\ F_1 \land (F_2 \lor F_3) & \iff & (F_1 \land F_2) \lor (F_1 \land F_3) \end{array}$$

Example: DNF

To convert

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

into DNF,

- Transform F into NNF.

$$F' : (Q_1 \lor Q_2) \land (R_1 \lor R_2)$$

- Apply distributivity.

$$F'' : (Q_1 \land (R_1 \lor R_2)) \lor (Q_2 \land (R_1 \lor R_2)).$$

  Apply distributivity again.

$$F''' : (Q_1 \land R_1) \lor (Q_1 \land R_2) \lor (Q_2 \land R_1) \lor (Q_2 \land R_2)$$

F''' is in DNF and is equivalent to F.

Conjunctive Normal Form (CNF)

- A formula is in conjunctive normal form (CNF) if it is a conjunction of disjunctive clauses (disjunctions of literals):

$$\bigwedge_i \bigvee_j l_{i,j}$$

- To convert a formula F into an equivalent formula in DNF,
  (1) transform F into NNF, and then
  (2) distribute disjunctions over conjunctions:

$$\begin{array}{ccc} (F_1 \land F_2) \lor F_3 & \iff & (F_1 \lor F_3) \land (F_2 \lor F_3) \\ F_1 \lor (F_2 \land F_3) & \iff & (F_1 \lor F_2) \land (F_1 \lor F_3) \end{array}$$

Example: CNF

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

- Transform F into NNF.

$$F' : (Q_1 \land Q_2) \lor (R_1 \lor R_2)$$

- Apply distributivity.

$$F'' : (Q_1 \lor R_1 \lor R_2) \land (Q_2 \lor R_1 \lor R_2)$$

F'' is in CNF and is equivalent to F.

Dicision Procedures

- A *decision procedure* decides whether F is satisfiable after some finite steps of computation.
- Approaches for deciding satisfiability:
  - **Search**: exhaustively search for all possible assignments.
  - **Deduction**: deduce facts from known facts by iteratively applying proof rules.
  - **Combination**: Modern SAT solvers are based on *DPLL* that combines search and deduction in an effective way.
- Plan: we will first define the naive approach and extend it to DPLL, the basis for modern SAT solvers.

Exhaustive Search (Truth Table Method)

- The naive, recursive algorithm for deciding satisfiability.

Algorithm 1 SAT

Input: A PL formula F
Output: Satisfiability ($true$: SAT, $false$: UNSAT)

1: **if** $F = \top$ **then return** $\top$
2: **else if** $F = \bot$ **then return** $\bot$
3: **else**
4: $\quad P \leftarrow \mathbf{ChooseVar}(F)$
5: $\quad$ **return** $\mathbf{SAT}(F\{P \mapsto \top\}) \lor \mathbf{SAT}(F\{P \mapsto \bot\})$

- When applying $F\{P \mapsto \top\}$ and $F\{P \mapsto \bot\}$, the resulting formulas should be simplified using template equivalences on PL.

$$
\begin{array}{rclrclrcl}
\top & \iff & \neg\bot & \bot & \iff & \neg\top & \neg\neg F & \iff & F \\
F \land \top & \iff & F & F \land \bot & \iff & \bot & F \land F & \iff & F \\
F \lor \top & \iff & \top & F \lor \bot & \iff & F & F \lor F & \iff & F \\
& \cdots & & & \cdots & & & \cdots &
\end{array}
$$

Example 1: Exhaustive Search

Consider the formula $F : (P \to Q) \wedge P \wedge \neg Q$.

- Choose variable P and

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

  which simplifies to $F_1 : Q \wedge \neg Q$.
  - $F_1\{Q \mapsto \top\} : \bot$
  - $F_1\{Q \mapsto \bot\} : \bot$

- Recurse on the other branch for P in F.

$$F\{P \mapsto \bot\} : (\bot \to Q) \wedge \bot \wedge \neg Q$$

  which simplifies to $\bot$.

Since all branches end without finding a satisfying assignment, we conclude F is UNSAT.

Example 2: Exhaustive Search

Determine the satisfiability of $\boldsymbol{F : (P \rightarrow Q) \land \neg P}$.

Example 2: Exhaustive Search

Determine the satisfiability of $F : (P \to Q) \land \neg P$.

- Choose P and recurse on the first case:

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

  which is equivalent to $\bot$.

- Try the other case:

$$F\{P \mapsto \bot\} : (\bot \to Q) \land \neg\bot$$

  which is equivalent to $\top$.

We conclude F is SAT by the second case. Assigning any value to Q produces a satisfying interpretation:

$$I : \{P \mapsto \mathit{false}, Q \to \mathit{true}\}$$

Equisatisfiability

- SAT solvers convert a given formula F to CNF.
- Conversion to an *equivalent* CNF incurs exponential blow-up in worst-case.
- F is converted to an *equisatisfiable* CNF formula, which increases the size by only a constant factor.
- F and F' are equisatisfiable when F is satisfiable iff F' is satisfiable.
- Equisatisfiability is a weaker notion of equivalence, which is still useful when deciding satisfiability.

Idea: Introduce new variables to represent the subformulas of F with extra clauses that assert that these new variables are equivalent to the subformulas that they represent.

Example: Conversion to an Equisatisfiable Formula in CNF

Consider $F : x_1 \to (x_2 \land x_3)$

- Introduce two variables a_1 and a_2 with two equivalences:

$$\begin{array}{ll} G_1 : & a_1 \leftrightarrow (x_1 \to a_2) \\ G_2 : & a_2 \leftrightarrow (x_2 \land x_3) \end{array}$$

  We need to satisfy all the equivalences.

- Convert the equivalences to CNF:

$$\begin{array}{lll} G_1 & \iff & (a_1 \to (x_1 \to a_2)) \land ((x_1 \to a_2) \to a_1) \\ & \iff & (\neg a_1 \lor (\neg x_1 \lor a_2)) \land (\neg(\neg x_1 \lor a_2) \lor a_1) \\ & \iff & (\neg a_1 \lor \neg x_1 \lor a_2) \land ((x_1 \land \neg a_2) \lor a_1) \\ & \iff & (\neg a_1 \lor \neg x_1 \lor a_2) \land (a_1 \lor x_1) \land (a_1 \lor \neg a_2) \\ G_2 & \iff & (a_2 \to (x_2 \land x_3)) \land ((x_2 \land x_3) \to a_2) \\ & \iff & (\neg a_2 \lor (x_2 \land x_3)) \land (\neg(x_2 \land x_3) \lor a_2) \\ & \iff & (\neg a_2 \lor x_2) \land (\neg a_2 \lor x_3) \land (a_2 \lor \neg x_2 \lor \neg x_3) \end{array}$$

- The final, equisatisfiable CNF formula F':

$$\begin{array}{lll} F' & = & a_1 \land (a_1 \lor x_1) \land (a_1 \lor \neg a_2) \land (\neg a_1 \lor \neg x_1 \lor a_2) \land \\ & & (\neg a_2 \lor x_2) \land (\neg a_2 \lor x_3) \land (a_2 \lor \neg x_2 \lor \neg x_3) \end{array}$$

The Resolution Procedure

- Applicable only to CNF formulas.
- Observation: to satisfy clauses $C_1[P]$ and $C_2[\neg P]$ that share the variable P but disagree on its value, either the rest of C_1 or the rest of C_2 must be satisfied. Why?
- The clause $C_1[\bot] \vee C_2[\bot]$ (with simplification) can be added as a conjunction to F to produce an equivalent formula still in CNF.
- The proof rule for *clausal resolution*:

$$\frac{C_1[P] \quad C_2[\neg P]}{C_1[\bot] \vee C_2[\bot]}$$

  The new clause $C_1[\bot] \vee C_2[\bot]$ is called the *resolvent*.
- If ever $\bot$ is deduced via resolution, F must be unsatisfiable. Otherwise, if no further resolutions are possible, F must be satisfiable.

Example 1: Resolution

Consider $F : (\neg P \vee Q) \wedge P \wedge \neg Q$.

- From the resolution

$$\frac{(\neg P \vee Q) \quad P}{Q}$$

  we can construct $F' : (\neg P \vee Q) \wedge P \wedge \neg Q \wedge Q$. From the resolution

$$\frac{\neg Q \quad Q}{\perp}$$

  we can deduce that F is unsatisfiable.

Example 2: Resolution

Consider $F : (\neg P \vee Q) \wedge \neg Q$.

- The resolution

$$\frac{(\neg P \vee Q) \quad \neg Q}{\neg P}$$

  yields $F' : (\neg P \vee Q) \wedge \neg Q \wedge \neg P$.

- Since no further resolutions are possible, F is satisfiable. Indeed, we have a satisfying interpretation $I : \{P \mapsto \mathit{false}, Q \mapsto \mathit{false}\}$.
- A CNF formula, which does not contain the clause $\perp$ and to which no more resolutions are applicable, represents all possible satisfying interpretations.

DPLL

- The Davis-Putnam-Logemann-Loveland algorithm (DPLL) combines the enumerative search and a restricted form of resolution, called *unit resolution*:

$$\frac{l \quad C[\neg l]}{C[\bot]}$$

  where l is a literal (i.e., $l = P$ or $l = \neg P$ for some propositional variable P).
- The process of applying this resolution as much as possible is called *Boolean constraint propagation (BCP)*.
- Like the resolution procedure, DPLL operates on PL formulas in CNF.

Example: BCP

Consider $F : P \wedge (\neg P \vee Q) \wedge (R \vee \neg Q \vee S)$ where P is a unit clause.

- Applying the unit resolution

$$\frac{P \quad (\neg P \vee Q)}{Q}$$

yields $F' : Q \wedge (R \vee \neg Q \vee S)$.

- Again, applying the unit resolution

$$\frac{Q \quad R \vee \neg Q \vee S}{R \vee S}$$

to F' produces $F'' : R \vee S$, ending the current round of BCP.

DPLL with BCP

DPLL is similar to SAT, except that it begins by applying BCP.

Algorithm 2 DPLL

Input: A PL formula $\boldsymbol{F}$
Output: Satisfiability ($\boldsymbol{true}$: SAT, $\boldsymbol{false}$: UNSAT)

1: $\boldsymbol{F} \leftarrow \mathbf{BCP}(\boldsymbol{F})$
2: **if** $\boldsymbol{F} = \top$ **then return** $\top$
3: **else if** $\boldsymbol{F} = \bot$ **then return** $\bot$
4: **else**
5: $\quad \boldsymbol{P} \leftarrow \mathbf{ChooseVar}(\boldsymbol{F})$
6: $\quad$ **return** $\mathbf{DPLL}(\boldsymbol{F}\{\boldsymbol{P} \mapsto \top\}) \vee \mathbf{DPLL}(\boldsymbol{F}\{\boldsymbol{P} \mapsto \bot\})$

Pure Literal Propagation (PLP)

- If variable P appears only positively or only negatively in F, remove all clauses containing an instance of P.
  - If P appears only positively (i.e., no $\neg P$ in F), replace P by $\top$.
  - If P appears only negatively (i.e., no P in F), replace P by $\bot$.
- The original formula F and the resulting formula F' are equisatisfiable.
- When only such pure variables remain, the formula must be satisfiable. A full interpretation can be constructed by setting each variable's value based on whether it appears only positively ($true$) or only negatively ($false$).

DPLL with BCP and PLP

Algorithm 3 DPLL

Input: A PL formula F
Output: Satisfiability ($true$: SAT, $false$: UNSAT)

1: $F \leftarrow \mathrm{BCP}(F)$
2: $F \leftarrow \mathrm{PLP}(F)$
3: **if** $F = \top$ **then return** $\top$
4: **else if** $F = \bot$ **then return** $\bot$
5: **else**
6: $\quad P \leftarrow \mathrm{ChooseVar}(F)$
7: $\quad$ **return** $\mathrm{DPLL}(F\{P \mapsto \top\}) \lor \mathrm{DPLL}(F\{P \mapsto \bot\})$

Example 1: Final DPLL

Consider $F : P \wedge (\neg P \vee Q) \wedge (R \vee \neg Q \vee S)$ in our previous BCP example.

- Recall that applying BCP yields $F'' : R \vee S$, where the unit resolutions correspond to the partial interpretation $\{P \mapsto true, Q \mapsto true\}$.
- All variables occur positively, so F is satisfiable:

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

- Branching (lines 6 and 7 in Algorithm 3) is not required in this example.

Example 2: Final DPLL

Consider

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

- No BCP and PLP are applicable.
- Choose Q on the true branch:

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

  We finish this branch, as the unit resolution with R and $\neg R$ deduces $\perp$.
- On the other branch for Q:

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

  P and R are pure, and thus F is satisfiable with the satisfying interpretation:

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

Summary

- Q. Why computational logic in software engineering?
  A. Mathematical basis for systemically analyzing software.
- Syntax and semantics of propositional logic
- Satisfiability and validity
- Equivalence, implications, and equisatisfiability
- Substitution
- Normal forms: NNF, DNF, CNF
- Decision procedures for satisfiability