# COSE419: Software Verification

# Lecture 4 — Propositional Logic

Hakjoo Oh
2024 Spring

# Syntax

- **Atom**: basic elements
  - truth symbols $\bot$ ("false") and $\top$ ("true")
  - propositional variables $P, Q, R, \ldots$
- **Literal**: an atom $\alpha$ or its negation $\neg\alpha$.
- **Formula**: a literal or the application of a logical connective (boolean connective) to formulas

$$
\begin{array}{rcll}
F & \rightarrow & \bot & \\
& | & \top & \\
& | & P & \\
& | & \neg F & \text{negation ("not")} \\
& | & F_1 \wedge F_2 & \text{conjunction ("and")} \\
& | & F_1 \vee F_2 & \text{disjunction ("or")} \\
& | & F_1 \rightarrow F_2 & \text{implication ("implies")} \\
& | & F_1 \leftrightarrow F_2 & \text{iff ("if and only if")}
\end{array}
$$

# Syntax

- Formula $G$ is a **subformula** of formula $F$ if it occurs syntactically within $G$.

$$
\begin{array}{rcl}
\textbf{sub}(\bot) & = & \{\bot\} \\
\textbf{sub}(\top) & = & \{\top\} \\
\textbf{sub}(P) & = & \{P\} \\
\textbf{sub}(\neg F) & = & \{\neg F\} \cup \textbf{sub}(F) \\
\textbf{sub}(F_1 \wedge F_2) & = & \{F_1 \wedge F_2\} \cup \textbf{sub}(F_1) \cup \textbf{sub}(F_2) \\
 & \vdots &
\end{array}
$$

- $F : (P \wedge Q) \rightarrow (P \vee \neg Q)$
  - $\textbf{sub}(F) =$
- The strict subformulas of a formula are all its subformulas except itself.

# Syntax

- To minimally use parentheses, we define the relative precedence of the logical connectives from highest to lowest as follows:

$$\neg \quad \land \quad \lor \quad \rightarrow \quad \leftrightarrow$$

- Additionally, $\rightarrow$ and $\leftrightarrow$ associate to the right, e.g.,

$$P \rightarrow Q \rightarrow R \iff P \rightarrow (Q \rightarrow R)$$

- Examples:
  - $(P \land Q) \rightarrow (P \lor \neg Q) \iff P \land Q \rightarrow P \lor \neg Q$
  - $(P_1 \land ((\neg P_2) \land \top)) \lor ((\neg P_1) \land P_2) \iff P_1 \land \neg P_2 \land \top \lor \neg P_1 \land P_2$

# Semantics

- The semantics of a logic provides its meaning. The meaning of a PL formula is either true or false.
- The semantics of a formula is defined with an **interpretation** (or assignment) that assigns truth values to propositional variables.
- For example, $F : P \land Q \to P \lor \neg Q$ evaluates to true under the interpretation $I : \{P \mapsto \textbf{true}, Q \mapsto \textbf{false}\}$:

| $P$ | $Q$ | $\neg Q$ | $P \land Q$ | $P \lor \neg Q$ | $F$ |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 1 |

- The tabular notation is unsuitable for predicate logic. Instead, we define the semantics inductively.

# Inductive Definition of Semantics

In an inductive definition, the meaning of basic elements is defined first. The meaning of complex elements is defined in terms of subcomponents.

- We write $I \vDash F$ if $F$ evaluates to **true** under $I$.
- We write $I \nvDash F$ if $F$ evaluates to **false** under $I$.

$$
\begin{array}{lll}
I \vDash \top, \quad I \nvDash \bot, & & \\
I \vDash P & \text{iff} & I[P] = \textbf{true} \\
I \nvDash P & \text{iff} & I[P] = \textbf{false} \\
I \vDash \neg F & \text{iff} & I \nvDash F \\
I \vDash F_1 \wedge F_2 & \text{iff} & I \vDash F_1 \text{ and } I \vDash F_2 \\
I \vDash F_1 \vee F_2 & \text{iff} & I \vDash F_1 \text{ or } I \vDash F_2 \\
I \vDash F_1 \rightarrow F_2 & \text{iff} & I \nvDash F_1 \text{ or } I \vDash F_2 \\
I \vDash F_1 \leftrightarrow F_2 & \text{iff} & (I \vDash F_1 \text{ and } I \vDash F_2) \text{ or } (I \nvDash F_1 \text{ and } I \nvDash F_2)
\end{array}
$$

# Example

Consider the formula

$$F : P \land Q \to P \lor \neg Q$$

and the interpretation

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

The truth value of $F$ is computed as follows:

| | | |
|---|---|---|
| 1. | $I \models P$ | since $I[P] = \textbf{true}$ |
| 2. | $I \not\models Q$ | since $I[Q] = \textbf{false}$ |
| 3. | $I \models \neg Q$ | by 2 and semantics of $\neg$ |
| 4. | $I \not\models P \land Q$ | by 2 and semantics of $\land$ |
| 5. | $I \models P \lor \neg Q$ | by 1 and semantics of $\lor$ |
| 6. | $I \models F$ | by 4 and semantics of $\to$ |

# Satisfiability and Validity

- A formula $F$ is **satisfiable** iff there exists an interpretation $I$ such that $I \models F$.
- A formula $F$ is **valid** iff for all interpretations $I$, $I \models F$.
- Satisfiability and validity are dual[1]:

$$F \text{ is valid iff } \neg F \text{ is unsatisfiable}$$

  - Proof: exercise
- We can check satisfiability by deciding validity, and vice versa.

[1] In logic, functions (or relations) $A$ and $B$ are dual if $A(x) = \neg B(\neg x)$

# Deciding Validity and Satisfiability

Two approaches to show $F$ is valid:

- Truth table method performs exhaustive search: e.g., $F: P \land Q \rightarrow P \lor \neg Q$.

| $P$ | $Q$ | $P \land Q$ | $\neg Q$ | $P \lor \neg Q$ | $F$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 |

  Non-applicable to logic with infinite domain (e.g., first-order logic).

- Semantic argument method uses deduction:
  - Assume $F$ is invalid: $I \not\models F$ for some $I$ (falsifying interpretation).
  - Apply deduction rules (proof rules) to derive a contradiction.
  - If every branch of the proof derives a contradiction, then $F$ is valid.
  - If some branch of the proof never derives a contradiction, then $F$ is invalid. This branch describes a falsifying interpretation of $F$.

# Deduction Rules for Propositional Logic

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

$$\frac{I \models F \land G}{I \models F, I \models G} \qquad \frac{I \not\models F \land G}{I \not\models F \mid I \not\models G}$$

$$\frac{I \models F \lor G}{I \models F \mid I \models G} \qquad \frac{I \not\models F \lor G}{I \not\models F, I \not\models G}$$

$$\frac{I \models F \to G}{I \not\models F \mid I \models G} \qquad \frac{I \not\models F \to G}{I \models F, I \not\models G}$$

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

$$\frac{I \models F \quad I \not\models F}{I \models \bot}$$

# Example 1

To prove that the formula

$$F : P \land Q \to P \lor \neg Q$$

is valid, assume that it is invalid and derive a contradiction:

| | | |
|---|---|---|
| **1.** | $I \not\models P \land Q \to P \lor \neg Q$ | assumption |
| **2.** | $I \models P \land Q$ | by 1 and semantics of $\to$ |
| **3.** | $I \not\models P \lor \neg Q$ | by 1 and semantics of $\to$ |
| **4.** | $I \models P$ | by 2 and semantics of $\land$ |
| **5.** | $I \not\models P$ | by 3 and semantics of $\lor$ |
| **6.** | $I \models \bot$ | 4 and 5 are contradictory |

# Example 2

To prove that the formula

$$F : (P \to Q) \land (Q \to R) \to (P \to R)$$

is valid, assume that it is invalid and derive a contradiction:

| | | |
|---|---|---|
| 1. | $I \not\models F$ | assumption |
| 2. | $I \models (P \to Q) \land (Q \to R)$ | by 1 and semantics of $\to$ |
| 3. | $I \not\models P \to R$ | by 1 and semantics of $\to$ |
| 4. | $I \models P$ | by 3 and semantics of $\to$ |
| 5. | $I \not\models R$ | by 3 and semantics of $\to$ |
| 6. | $I \models P \to Q$ | 2 and semantics of $\land$ |
| 7. | $I \models Q \to R$ | 2 and semantics of $\land$ |

Two cases to consider from 6:

1. $I \not\models P$: contradiction with 4.
2. $I \models Q$: two cases to consider from 7:
    1. $I \not\models Q$: contradiction
    2. $I \models R$: contradiction with 5.

# Proof Tree

A proof evolves as a tree.

- A *branch* is a sequence descending from the root.
- A branch is *closed* if it contains a contradiction. Otherwise, the branch is *open*.
- It is a proof of the validity of $F$ if every branch is closed; otherwise, each open branch describes a falsifying interpretation of $F$.

# Exercise

Use the semantic argument method to prove that the following $F$ is valid.

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

# Derived Rules

The proof rules are sufficient, but **derived rules** can make proofs more concise. E.g., the rule of modus ponens:

$$\frac{I \models F \qquad I \models F \rightarrow G}{I \models G}$$

The proof of the validity of the formula:

$$F : (P \rightarrow Q) \land (Q \rightarrow R) \rightarrow (P \rightarrow R)$$

| | | |
|---|---|---|
| 1. | $I \not\models F$ | assumption |
| 2. | $I \models (P \rightarrow Q) \land (Q \rightarrow R)$ | by 1 and semantics of $\rightarrow$ |
| 3. | $I \not\models P \rightarrow R$ | by 1 and semantics of $\rightarrow$ |
| 4. | $I \models P$ | by 3 and semantics of $\rightarrow$ |
| 5. | $I \not\models R$ | by 3 and semantics of $\rightarrow$ |
| 6. | $I \models P \rightarrow Q$ | 2 and semantics of $\land$ |
| 7. | $I \models Q \rightarrow R$ | 2 and semantics of $\land$ |
| 8. | $I \models Q$ | by 4, 6, and modus ponens |
| 9. | $I \models R$ | by 8, 7, and modus ponens |
| 10. | $I \models \bot$ | 5 and 9 are contradictory |

# Equivalence and Implication

- Two formulas $F_1$ and $F_2$ are equivalent

$$F_1 \iff F_2$$

  iff $F_1 \leftrightarrow F_2$ is valid, i.e., for all interpretations $I$, $I \models F_1 \leftrightarrow F_2$.
- Formula $F_1$ implies formula $F_2$

$$F_1 \implies F_2$$

  iff $F_1 \rightarrow F_2$ is valid, i.e., for all interpretations $I$, $I \models F_1 \rightarrow F_2$.
- $F_1 \iff F_2$ and $F_1 \implies F_2$ are not formulas. They are semantic assertions.
- We can check equivalence and implication by checking satisfiability.

# Examples

- $P \iff \neg\neg P$
- $P \to Q \iff \neg P \lor Q$

# Exercise

Prove that

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

# Substitution

- A substitution $\boldsymbol{\sigma}$ is a mapping from formulas to formulas:

$$\boldsymbol{\sigma : \{F_1 \mapsto G_2, \ldots, F_n \mapsto G_n\}}$$

- The domain of $\boldsymbol{\sigma}$, **dom**$(\boldsymbol{\sigma})$, is

$$\textbf{dom}(\boldsymbol{\sigma}) : \{\boldsymbol{F_1, \ldots, F_n}\}$$

  while the range **range**$(\boldsymbol{\sigma})$ is

$$\textbf{range}(\boldsymbol{\sigma}) : \{\boldsymbol{G_1, \ldots, G_n}\}$$

- The application of a substitution $\boldsymbol{\sigma}$ to a formula $\boldsymbol{F}$, $\boldsymbol{F\sigma}$, replaces each occurence of $\boldsymbol{F_i}$ with $\boldsymbol{G_i}$. Replacements occur all at once.
- When two subformulas $\boldsymbol{F_j}$ and $\boldsymbol{F_k}$ are in **dom**$(\boldsymbol{\sigma})$ and $\boldsymbol{F_k}$ is a strict subformula of $\boldsymbol{F_j}$, then $\boldsymbol{F_j}$ is replaced first.

# Example

Consider formula

$$F : P \land Q \to P \lor \neg Q$$

and substitution

$$\sigma : \{P \mapsto R, P \land Q \mapsto P \to Q\}$$

Then,

$$F\sigma : (P \to Q) \to R \lor \neg Q$$

Note that $F\sigma \neq (R \to Q) \to R \lor \neg Q$.

# Substitution

- A variable substitution is a substitution in which the domain consists only of propositional variables.
- When we write $F[F_1, \ldots, F_n]$, we mean that formula $F$ can have formulas $F_1, \ldots, F_n$ as subformulas.
- If $\sigma$ is $\{F_1 \mapsto G_1, \ldots, F_n \mapsto G_n\}$, then

$$F[F_1, \ldots, F_n]\sigma : F[G_1, \ldots, G_n]$$

- For example, in the previous example, writing

$$F[P, P \land Q]\sigma : F[R, P \to Q]$$

  emphasizes that $P$ and $P \land Q$ are replaced by $R$ and $P \to Q$, respectively.

# Semantic Consequences of Substitution

**Proposition (Substitution of Equivalent Formulas)**

*Consider substitution* $\sigma : \{F_1 \mapsto G_1, \ldots, F_n \mapsto G_n\}$ *such that for each* $i$, $F_i \iff G_i$. *Then,* $F \iff F\sigma$.

For example, applying $\sigma : \{P \to Q \mapsto \neg P \lor Q\}$ to $F : (P \to Q) \to R$ produces $(\neg P \lor Q) \to R$ that is equivalent to $F$.

**Proposition (Valid Template)**

*If* $F$ *is valid and* $G = F\sigma$ *for some variable substitution* $\sigma$, *then* $G$ *is valid.*

For example, because $F : (P \to Q) \leftrightarrow (\neg P \lor Q)$ is valid, every formula of the form $F_1 \to F_2$ is equivalent to $\neg F_1 \lor F_2$, for arbitrary formulas $F_1$ and $F_2$.

Proving the validity of $F$ proves the validity of an infinite set of formulas

# Composition of Substitutions

Given substitutions $\sigma_1$ and $\sigma_2$, their composition $\sigma = \sigma_1\sigma_2$ ("apply $\sigma_1$ and then $\sigma_2$") is computed as follows:

1. Apply $\sigma_2$ to each formula of the range of $\sigma_1$, and add the results to $\sigma$.
2. If $F_i$ of $F_i \mapsto G_i$ appears in the domain of $\sigma_2$ but not in the domain of $\sigma_1$, then add $F_i \mapsto G_i$ to $\sigma$.

For example,

$$\begin{aligned}
\sigma_1\sigma_2 : &\{P \mapsto R, P \wedge Q \mapsto P \rightarrow Q\}\{P \mapsto S, S \mapsto Q\} \\
&= \{P \mapsto R\sigma_2, P \wedge Q \mapsto (P \rightarrow Q)\sigma_2, S \mapsto Q\} \\
&= \{P \mapsto R, P \wedge Q \mapsto S \rightarrow Q, S \mapsto Q\}
\end{aligned}$$

# Normal Forms

A normal form of formulas is a syntactic restriction such that for every formula of the logic, there is an equivalent formula in the normal form.

Three useful normal forms in logic:

- **Negation Normal Form (NNF)**
- **Disjunctive Normal Form (DNF)**
- **Conjunctive Normal Form (CNF)**

# Negation Normal Form (NNF)

- NNF requires that $\neg$, $\wedge$, and $\vee$ are the only connectives (i.e., no $\rightarrow$ and $\leftrightarrow$) and that negations are only applied to variables.
  - $P \wedge Q \wedge (R \vee \neg S)$
  - $\neg P \vee \neg(P \wedge Q)$
  - $\neg\neg P \wedge Q$
- Transforming a formula $F$ to equivalent formula $F'$ in NNF can be done by repeatedly applying (left-to-right) the following template equivalences:

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

# Exercise

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

# Disjunctive Normal Form (DNF)

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

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

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

$$\begin{array}{lcl} (F_1 \vee F_2) \wedge F_3 & \iff & (F_1 \wedge F_3) \vee (F_2 \wedge F_3) \\ F_1 \wedge (F_2 \vee F_3) & \iff & (F_1 \wedge F_2) \vee (F_1 \wedge F_3) \end{array}$$

# Exercise

To convert

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

into DNF,

- first transform it into NNF:
- then apply distributivity:

# Conjunctive Normal Form (CNF)

- A formula is in conjunctive normal form (CNF) if it is a conjunction of disjunctions of literals:
$$\bigwedge_i \bigvee_j l_{i,j}$$
where each disjunction of literals is called a **clause**.
- To convert a formula $F$ into an equivalent formula in DNF, transform $F$ into NNF and distribute disjunctions over conjunctions:
$$\begin{array}{lcl} (F_1 \land F_2) \lor F_3 & \iff & (F_1 \lor F_3) \land (F_2 \lor F_3) \\ F_1 \lor (F_2 \land F_3) & \iff & (F_1 \lor F_2) \land (F_1 \lor F_3) \end{array}$$
- Exercise) Convert $F : (Q_1 \land \neg\neg Q_2) \lor (\neg R_1 \rightarrow R_2)$ into CNF

# Decision Procedures

- A **decision procedure** decides whether $F$ is satisfiable after some finite steps of computation.
- Approaches for deciding satisfiability:
  - **Search**: exhaustively search through 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

# Exhaustive Search

- The recursive algorithm for deciding satisfiability:

$$\begin{array}{l} \text{let rec } \textbf{SAT } F = \\ \quad \text{if } F = \top \text{ then true} \\ \quad \text{else if } F = \bot \text{ then false} \\ \quad \text{else} \\ \quad\quad \text{let } P = \textbf{Choose}(\textbf{vars}(F)) \text{ in} \\ \quad\quad (\textbf{SAT } F\{P \mapsto \top\}) \vee (\textbf{SAT } F\{P \mapsto \bot\}) \end{array}$$

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

$$\begin{array}{rcl} \top & \iff & \neg\bot \\ F \wedge \top & \iff & F \\ F \vee \top & \iff & \top \end{array} \quad \begin{array}{rcl} \bot & \iff & \neg\top \\ F \wedge \bot & \iff & \bot \\ F \vee \bot & \iff & F \end{array} \quad \begin{array}{rcl} \neg\neg F & \iff & F \\ F \wedge F & \iff & F \\ F \vee F & \iff & F \end{array}$$

$$\cdots$$

# Example

$$F : (P \to Q) \land P \land \neg Q$$

- Choose variable $P$ and

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

which simplifies to

$$F_1 : Q \land \neg Q$$

  - $F_1\{Q \mapsto \top\} : \bot$
  - $F_1\{Q \mapsto \bot\} : \bot$

- Recurse on the other branch for $P$ in $F$:

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

which simplifies to $\bot$.

- All branches end without finding a satisfying assignment.

# Example

$$F : (P \to Q) \land \neg P$$

- Choose $P$ and recurse on the first case:

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

which is equivalent to $\bot$.

- Try the other case:

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

which is equivalent to $\top$.

- Arbitrarily assigning a value to $Q$ produces the satisfying interpretation:

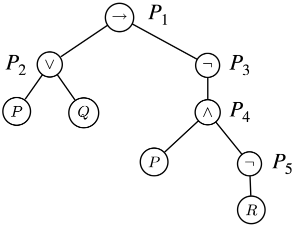
$$I : \{P \mapsto \textbf{false}, Q \mapsto \textbf{true}\}.$$

# 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.

# Conversion to an Equisatisfiable Formula in CNF

- Introduce new variables for each subformula of $F$ with extra clauses to assert that these variables are equivalent to the subformulas that they represent.
- Example: $F : P \lor Q \to \neg(P \land \neg R)$




  - $F$ is equisatisfiable to

$$P_1 \land P_1 \leftrightarrow (P_2 \to P_3) \land P_2 \leftrightarrow (P \lor Q) \land$$
$$P_3 \leftrightarrow \neg P_4 \land P_4 \leftrightarrow (P \land P_5) \land P_5 \leftrightarrow \neg R$$

  - In CNF:

$$P_1 \land (\neg P_1 \lor \neg P_2 \lor P_3) \land (P_2 \lor P_1) \land (\neg P_3 \lor P_1) \land \ldots$$

# Conversion to an Equisatisfiable Formula in CNF

Convert $F$ into

$$F' : \textbf{Rep}(F) \wedge \bigwedge_{G \in \textbf{sub}(F)} \textbf{En}(G)$$

- $\textbf{Rep} : \textbf{PL} \to \textbf{V} \cup \{\top, \bot\}$
  - The representative function that maps PL formulas to propositional variables $\textbf{V}$, $\top$, and $\bot$.
  - In the general case, it maps $F$ to its representative propositional variable $P_F$ such that the truth value of $P_F$ is the same as that of $F$.
- $\textbf{En} : \textbf{PL} \to \textbf{PL}$
  - The encoding function that maps PL formulas to PL formulas.
  - It maps a PL formula $F$ to a PL formula $F'$ in CNF that asserts that $F$'s representative, $P_F$, is equivalent to $F$: "$\textbf{Rep}(F) \leftrightarrow F$".

# Conversion to an Equisatisfiable Formula in CNF

$$\mathbf{Rep}(\top) = \top \qquad \mathbf{Rep}(\bot) = \bot \qquad \mathbf{Rep}(P) = P \qquad \mathbf{Rep}(F) = P_F$$
$$\mathbf{En}(\top) = \top \qquad \mathbf{En}(\bot) = \top \qquad \mathbf{En}(P) = \top$$

$\mathbf{En}(F_1 \wedge F_2) =$
let $P = \mathbf{Rep}(F_1 \wedge F_2)$ in
$(\neg P \vee \mathbf{Rep}(F_1)) \wedge (\neg P \vee \mathbf{Rep}(F_2)) \wedge (\neg \mathbf{Rep}(F_1) \vee \neg \mathbf{Rep}(F_2) \vee P)$

$\mathbf{En}(\neg F) =$
let $P = \mathbf{Rep}(\neg F)$ in
$(\neg P \vee \neg \mathbf{Rep}(F_1)) \wedge (P \vee \mathbf{Rep}(F))$

$\mathbf{En}(F_1 \vee F_2) =$
let $P = \mathbf{Rep}(F_1 \vee F_2)$ in
$(\neg P \vee \mathbf{Rep}(F_1) \vee \mathbf{Rep}(F_2)) \wedge (\neg \mathbf{Rep}(F_1) \vee P) \wedge (\neg \mathbf{Rep}(F_2) \vee P)$

$\mathbf{En}(F_1 \rightarrow F_2) =$
let $P = \mathbf{Rep}(F_1 \rightarrow F_2)$ in
$(\neg P \vee \neg \mathbf{Rep}(F_1) \vee \mathbf{Rep}(F_2)) \wedge (\mathbf{Rep}(F_1) \vee P) \wedge (\neg \mathbf{Rep}(F_2) \vee P)$

$\mathbf{En}(F_1 \leftrightarrow F_2) =$
let $P = \mathbf{Rep}(F_1 \leftrightarrow F_2)$ in
$(\neg P \vee \neg \mathbf{Rep}(F_1) \vee \mathbf{Rep}(F_2)) \wedge (\neg P \vee \mathbf{Rep}(F_1) \vee \neg \mathbf{Rep}(F_2))$
$\wedge (P \vee \neg \mathbf{Rep}(F_1) \vee \neg \mathbf{Rep}(F_2)) \wedge (P \vee \mathbf{Rep}(F_1) \vee \mathbf{Rep}(F_2))$

# Example

$$F : (Q_1 \wedge Q_2) \vee (R_1 \wedge R_2)$$

is converted into

$$F' : P_F \wedge \bigwedge_{G \in \textbf{sub}(F)} \textbf{En}(G)$$

where $\textbf{sub}(F) = \{Q_1, Q_2, R_1, R_2, Q_1 \wedge Q_2, R_1 \wedge R_2, F\}$ and

$$\begin{aligned}
\textbf{En}(Q_1) = \textbf{En}(Q_2) = \textbf{En}(R_1) = \textbf{En}(R_2) &= \top \\
\textbf{En}(Q_1 \wedge Q_2) &= (\neg P_{(Q_1 \wedge Q_2)} \vee Q_1) \wedge (\neg P_{(Q_1 \wedge Q_2)} \vee Q_2) \\
&\quad \wedge (\neg Q_1 \vee \neg Q_2 \vee P_{(Q_1 \wedge Q_2)}) \\
\textbf{En}(R_1 \wedge R_2) &= (\neg P_{(R_1 \wedge R_2)} \vee R_1) \wedge (\neg P_{(R_1 \wedge R_2)} \vee R_2) \\
&\quad \wedge (\neg R_1 \vee \neg R_2 \vee P_{(R_1 \wedge R_2)}) \\
\textbf{En}(F) &= (\neg P_F \vee P_{(Q_1 \wedge Q_2)} \vee P_{(R_1 \wedge R_2)}) \\
&\quad \wedge (\neg P_{(Q_1 \wedge Q_2)} \vee P_F) \wedge (\neg P_{(R_1 \wedge R_2)} \vee P_F)
\end{aligned}$$

# The Resolution Procedure

- Applicable only to CNF formulas.
- Observation: to satisfy clauses $C_1[P]$ and $C_2[\neg P]$ that share 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.

# Examples

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

- From resolution
  $$\frac{(\neg P \vee Q) \qquad P}{Q},$$
  construct $(\neg P \vee Q) \wedge P \wedge \neg Q \wedge Q$. From resolution
  $$\frac{\neg Q \qquad Q}{\bot}$$
  deduce that $F$ is unsatisfiable.

# Examples

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

- The resolution procedure yields

$$(\neg P \lor Q) \land \neg Q \land \neg P$$

  No further resolutions are possible. $F$ is satisfiable.

- A satisfying interpretation:

$$I : \{P \mapsto \textbf{false}, Q \mapsto \textbf{false}\}$$

- A CNF formula that does not contain the clause $\bot$ 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**:

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

  where $l$ is a literal ($l = P$ or $l = \neg P$).
- Because $C[\bot]$ is a subset of $C[\neg l]$, $C[\neg l]$ is replaced by $C[\bot]$. Also, $l$ is removed as it must be assigned true.
- Thus, performing unit resolution is identical to replacing $l$ by true in the original formula.
- The process of applying this resolution as much as possible is called **Boolean constraint propagation (BCP)**.

# BCP Example

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

- Apply unit resolution

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

to produce $F' : Q \wedge (R \vee \neg Q \vee S)$. Applying unit resolution

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

produces $F'' : R \vee S$, ending this round of BCP.

# DPLL

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

$$
\begin{array}{l}
\text{let rec } \textbf{DPLL } F = \\
\quad \text{let } F' = \textbf{BCP}(F) \text{ in} \\
\quad \text{if } F' = \top \text{ then true} \\
\quad \text{else if } F' = \bot \text{ then false} \\
\quad \text{else} \\
\quad\quad \text{let } P = \textbf{Choose}(\textbf{vars}(F')) \text{ in} \\
\quad\quad (\textbf{DPLL } F'\{P \mapsto \top\}) \vee (\textbf{DPLL } F'\{P \mapsto \bot\})
\end{array}
$$

# Pure Literal Elimination (PLE)

- 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 resulting formula $F'$ is equisatisfiable to $F$.
- 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).

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

- $P$ appears only negatively in $F$

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

- $R$ and $S$ appear only positively in $F$

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

# DPLL with PLP

let rec **DPLL** $F$ =
  let $F'$ = **PLE**(**BCP**($F$)) in
  if $F' = \top$ then true
  else if $F' = \bot$ then false
  else
    let $P$ = **Choose**(**vars**($F'$)) in
    (**DPLL** $F'\{P \mapsto \top\}$) $\vee$ (**DPLL** $F'\{P \mapsto \bot\}$)

# Example 1

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

1. Applying BCP produces
$$F'' : R \lor S$$
2. All variables occur positively, so $F$ is satisfiable.
3. A satisfying interpretation:

$$\{P \mapsto \textbf{true}, Q \mapsto \textbf{true}, R \mapsto \textbf{true}, S \mapsto \textbf{true}\}$$

# Example 2

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

- No BCP and PLP are applicable.
- Choose $Q$ to branch on:

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

  The unit resolution with $R$ and $\neg R$ deduces $\bot$, finishing this branch.
- On the other branch for $Q$:

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

  $P$ and $R$ are pure, so the formula is satisfiable. A satisfying interpretation:

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

# Summary

- Syntax and semantics of propositional logic
- Satisfiability and validity
- Equivalence, implications, and equisatisfiability
- Substitution
- Normal forms: NNF, DNF, CNF
- Decision procedures for satisfiability