

# AAA528: Computational Logic

## Lecture 1 — Propositional Logic

Hakjoo Oh  
2026 Spring

# Syntax

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

$F$	$\rightarrow$	$\perp$	
		$\top$	
		$P$	
		$\neg F$	negation (“not”)
		$F_1 \wedge F_2$	conjunction (“and”)
		$F_1 \vee F_2$	disjunction (“or”)
		$F_1 \rightarrow F_2$	implication (“implies”)
		$F_1 \leftrightarrow F_2$	iff (“if and only if”)

# Syntax

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

$$\begin{aligned}\mathbf{sub}(\perp) &= \{\perp\} \\ \mathbf{sub}(\top) &= \{\top\} \\ \mathbf{sub}(P) &= \{P\} \\ \mathbf{sub}(\neg F) &= \{\neg F\} \cup \mathbf{sub}(F) \\ \mathbf{sub}(F_1 \wedge F_2) &= \{F_1 \wedge F_2\} \cup \mathbf{sub}(F_1) \cup \mathbf{sub}(F_2) \\ &\vdots\end{aligned}$$

- $F : (P \wedge Q) \rightarrow (P \vee \neg Q)$ 
  - ▶  $\mathbf{sub}(F) =$
- 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:

$$\neg \quad \wedge \quad \vee \quad \rightarrow \quad \leftrightarrow$$

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

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

- Examples:

- ▶  $(P \wedge Q) \rightarrow (P \vee \neg Q) \iff P \wedge Q \rightarrow P \vee \neg Q$

- ▶  $(P_1 \wedge ((\neg P_2) \wedge \top)) \vee ((\neg P_1) \wedge P_2) \iff P_1 \wedge \neg P_2 \wedge \top \vee \neg P_1 \wedge P_2$

# 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,  $F : P \wedge Q \rightarrow P \vee \neg Q$  evaluates to true under the interpretation  $I : \{P \mapsto \mathbf{true}, Q \mapsto \mathbf{false}\}$ :

$P$	$Q$	$\neg Q$	$P \wedge Q$	$P \vee \neg Q$	$F$
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  $I \models F$  if  $F$  evaluates to **true** under  $I$ .
- We write  $I \not\models F$  if  $F$  evaluates to **false** under  $I$ .

$$I \models \top, \quad I \not\models \perp,$$

$$I \models P \quad \text{iff} \quad I[P] = \mathbf{true}$$

$$I \not\models P \quad \text{iff} \quad I[P] = \mathbf{false}$$

$$I \models \neg F \quad \text{iff} \quad I \not\models F$$

$$I \models F_1 \wedge F_2 \quad \text{iff} \quad I \models F_1 \text{ and } I \models F_2$$

$$I \models F_1 \vee F_2 \quad \text{iff} \quad I \models F_1 \text{ or } I \models F_2$$

$$I \models F_1 \rightarrow F_2 \quad \text{iff} \quad I \not\models F_1 \text{ or } I \models F_2$$

$$I \models F_1 \leftrightarrow F_2 \quad \text{iff} \quad (I \models F_1 \text{ and } I \models F_2) \text{ or } (I \not\models F_1 \text{ and } I \not\models F_2)$$

## Example

Consider the formula

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

and the interpretation

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

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

1.  $I \models P$             since  $I[P] = \mathbf{true}$
2.  $I \not\models Q$             since  $I[Q] = \mathbf{false}$
3.  $I \models \neg Q$         by 2 and semantics of  $\neg$
4.  $I \not\models P \wedge Q$     by 2 and semantics of  $\wedge$
5.  $I \models P \vee \neg Q$     by 1 and semantics of  $\vee$
6.  $I \models F$             by 4 and semantics of  $\rightarrow$

# 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<sup>1</sup>:

$F$  is valid iff  $\neg F$  is unsatisfiable

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

---

<sup>1</sup>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.,  
 $F : P \wedge Q \rightarrow P \vee \neg Q$ .

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

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

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

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

$$\frac{I \models F \wedge G}{I \models F, I \models G}$$

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

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

$$\frac{I \not\models F \vee G}{I \not\models F, I \not\models G}$$

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

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

$$\frac{I \models F \leftrightarrow G}{I \models F \wedge G \mid I \models \neg F \wedge \neg G}$$

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

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

## Example 1

To prove that the formula

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

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

1.  $I \not\models P \wedge Q \rightarrow P \vee \neg Q$  assumption
2.  $I \models P \wedge Q$  by 1 and semantics of  $\rightarrow$
3.  $I \not\models P \vee \neg Q$  by 1 and semantics of  $\rightarrow$
4.  $I \models P$  by 2 and semantics of  $\wedge$
5.  $I \not\models P$  by 3 and semantics of  $\vee$
6.  $I \models \perp$  4 and 5 are contradictory

## Example 2

To prove that the formula

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

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

- |    |  |                                     |
|----|--|-------------------------------------|
| 1. | $I \not\models F$                                      | assumption                          |
| 2. | $I \models (P \rightarrow Q) \wedge (Q \rightarrow R)$ | 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 $\wedge$         |
| 7. | $I \models Q \rightarrow R$                            | 2 and semantics of $\wedge$         |

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:

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

The proof of the validity of the formula:

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

1.  $I \not\models F$  assumption
2.  $I \models (P \rightarrow Q) \wedge (Q \rightarrow R)$  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  $\wedge$
7.  $I \models Q \rightarrow R$  2 and semantics of  $\wedge$
8.  $I \models Q$  by 4, 6, and modus ponens
9.  $I \models R$  by 8, 7, and modus ponens
10.  $I \models \perp$  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

- $P \iff \neg\neg P$
- $P \rightarrow Q \iff \neg P \vee Q$

## Exercise

Prove that

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

# Substitution

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

$$\sigma : \{F_1 \mapsto G_1, \dots, F_n \mapsto G_n\}$$

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

$$\mathbf{dom}(\sigma) : \{F_1, \dots, F_n\}$$

while the range  $\mathbf{range}(\sigma)$  is

$$\mathbf{range}(\sigma) : \{G_1, \dots, G_n\}$$

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

## Example

Consider formula

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

and substitution

$$\sigma : \{P \mapsto R, P \wedge Q \mapsto P \rightarrow Q\}$$

Then,

$$F\sigma : (P \rightarrow Q) \rightarrow R \vee \neg Q$$

Note that  $F\sigma \neq (R \rightarrow Q) \rightarrow R \vee \neg Q$ .

## Substitution

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

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

- For example, in the previous example, writing

$$F[P, P \wedge Q]\sigma : F[R, P \rightarrow Q]$$

emphasizes that  $P$  and  $P \wedge Q$  are replaced by  $R$  and  $P \rightarrow Q$ , respectively.

# Semantic Consequences of Substitution

## Proposition (Substitution of Equivalent Formulas)

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

For example, applying  $\sigma : \{P \rightarrow Q \mapsto \neg P \vee Q\}$  to  $F : (P \rightarrow Q) \rightarrow R$  produces  $(\neg P \vee Q) \rightarrow R$  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  $F : (P \rightarrow Q) \iff (\neg P \vee Q)$  is valid, every formula of the form  $F_1 \rightarrow F_2$  is equivalent to  $\neg F_1 \vee F_2$ , 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:

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

## Exercise

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

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

## Exercise

To convert

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

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:

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

- Exercise) Convert  $F : (Q_1 \wedge \neg\neg Q_2) \vee (\neg R_1 \rightarrow R_2)$  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:

let rec **SAT**  $F$  =  
  if  $F = \top$  then true  
  else if  $F = \perp$  then false  
  else  
    let  $P = \mathbf{Choose}(\mathbf{vars}(F))$  in  
    (**SAT**  $F\{P \mapsto \top\}$ )  $\vee$  (**SAT**  $F\{P \mapsto \perp\}$ )

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

$$\begin{array}{l} \top \iff \neg\perp \qquad \perp \iff \neg\top \qquad \neg\neg F \iff F \\ F \wedge \top \iff F \qquad F \wedge \perp \iff \perp \qquad F \wedge F \iff F \\ F \vee \top \iff \top \qquad F \vee \perp \iff F \qquad F \vee F \iff F \\ \dots \end{array}$$

## Example

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

- Choose variable  $P$  and

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

which simplifies to

$$F_1 : Q \wedge \neg Q$$

- ▶  $F_1\{Q \mapsto \top\} : \perp$
- ▶  $F_1\{Q \mapsto \perp\} : \perp$

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

$$F\{P \mapsto \perp\} : (\perp \rightarrow Q) \wedge \perp \wedge \neg Q$$

which simplifies to  $\perp$ .

- All branches end without finding a satisfying assignment.

## Example

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

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

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

which is equivalent to  $\perp$ .

- Try the other case:

$$F\{P \rightarrow \perp\} : (\perp \rightarrow Q) \wedge \neg \perp$$

which is equivalent to  $\top$ .

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

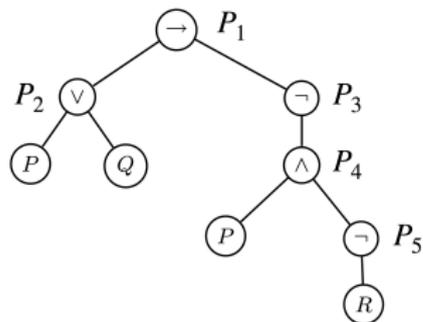
$$I : \{P \mapsto \text{false}, Q \mapsto \text{true}\}.$$

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



- ▶  $F$  is equisatisfiable to

$$P_1 \wedge P_1 \leftrightarrow (P_2 \rightarrow P_3) \wedge P_2 \leftrightarrow (P \vee Q) \wedge \\ P_3 \leftrightarrow \neg P_4 \wedge P_4 \leftrightarrow (P \wedge P_5) \wedge P_5 \leftrightarrow \neg R$$

- ▶ In CNF:

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

# Conversion to an Equisatisfiable Formula in CNF

Convert  $F$  into

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

- **Rep** :  $\mathbf{PL} \rightarrow \mathbf{V} \cup \{\top, \perp\}$

- ▶ The representative function that maps PL formulas to propositional variables  $\mathbf{V}$ ,  $\top$ , and  $\perp$ .
- ▶ 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  $F$ .

- **En** :  $\mathbf{PL} \rightarrow \mathbf{PL}$

- ▶ 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  $F$ : “ $\mathbf{Rep}(F) \leftrightarrow F'$ ”.

# Conversion to an Equisatisfiable Formula in CNF

$$\begin{array}{llll} \text{Rep}(\top) = \top & \text{Rep}(\perp) = \perp & \text{Rep}(P) = P & \text{Rep}(F) = P_F \\ \text{En}(\top) = \top & \text{En}(\perp) = \top & \text{En}(P) = \top & \end{array}$$

$$\begin{array}{l} \text{En}(F_1 \wedge F_2) = \\ \quad \text{let } P = \text{Rep}(F_1 \wedge F_2) \text{ in} \\ \quad (\neg P \vee \text{Rep}(F_1)) \wedge (\neg P \vee \text{Rep}(F_2)) \wedge (\neg \text{Rep}(F_1) \vee \neg \text{Rep}(F_2) \vee P) \end{array}$$

$$\begin{array}{l} \text{En}(\neg F) = \\ \quad \text{let } P = \text{Rep}(\neg F) \text{ in} \\ \quad (\neg P \vee \neg \text{Rep}(F)) \wedge (P \vee \text{Rep}(F)) \end{array}$$

$$\begin{array}{l} \text{En}(F_1 \vee F_2) = \\ \quad \text{let } P = \text{Rep}(F_1 \vee F_2) \text{ in} \\ \quad (\neg P \vee \text{Rep}(F_1) \vee \text{Rep}(F_2)) \wedge (\neg \text{Rep}(F_1) \vee P) \wedge (\neg \text{Rep}(F_2) \vee P) \end{array}$$

$$\begin{array}{l} \text{En}(F_1 \rightarrow F_2) = \\ \quad \text{let } P = \text{Rep}(F_1 \rightarrow F_2) \text{ in} \\ \quad (\neg P \vee \neg \text{Rep}(F_1) \vee \text{Rep}(F_2)) \wedge (\text{Rep}(F_1) \vee P) \wedge (\neg \text{Rep}(F_2) \vee P) \end{array}$$

$$\begin{array}{l} \text{En}(F_1 \leftrightarrow F_2) = \\ \quad \text{let } P = \text{Rep}(F_1 \leftrightarrow F_2) \text{ in} \\ \quad (\neg P \vee \neg \text{Rep}(F_1) \vee \text{Rep}(F_2)) \wedge (\neg P \vee \text{Rep}(F_1) \vee \neg \text{Rep}(F_2)) \\ \quad \wedge (P \vee \neg \text{Rep}(F_1) \vee \neg \text{Rep}(F_2)) \wedge (P \vee \text{Rep}(F_1) \vee \text{Rep}(F_2)) \end{array}$$

## Example

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

is converted into

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

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

$$\text{En}(Q_1) = \text{En}(Q_2) = \text{En}(R_1) = \text{En}(R_2) = \top$$

$$\begin{aligned} \text{En}(Q_1 \wedge Q_2) &= (\neg P_{(Q_1 \wedge Q_2)} \vee Q_1) \wedge (\neg P_{(Q_1 \wedge Q_2)} \vee Q_2) \\ &\quad \wedge (\neg Q_1 \vee \neg Q_2 \vee P_{(Q_1 \wedge Q_2)}) \end{aligned}$$

$$\begin{aligned} \text{En}(R_1 \wedge R_2) &= (\neg P_{(R_1 \wedge R_2)} \vee R_1) \wedge (\neg P_{(R_1 \wedge R_2)} \vee R_2) \\ &\quad \wedge (\neg R_1 \vee \neg R_2 \vee P_{(R_1 \wedge R_2)}) \end{aligned}$$

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

# 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  $C_1[\perp] \vee C_2[\perp]$  (with simplification) can be added as a conjunction to  $F$  to produce an equivalent formula still in CNF.
- The proof rule for **clausal resolution**:

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

The new clause  $C_1[\perp] \vee C_2[\perp]$  is called the **resolvent**.

- If ever  $\perp$  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

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

construct  $(\neg P \vee Q) \wedge P \wedge \neg Q \wedge Q$ . From resolution

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

deduce that  $F$  is unsatisfiable.

## Examples

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

- The resolution procedure yields

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

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

- A satisfying interpretation:

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

- A CNF formula that 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**:

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

where  $l$  is a literal ( $l = P$  or  $l = \neg P$ ).

- Because  $C[\perp]$  is a subset of  $C[\neg l]$ ,  $C[\neg l]$  is replaced by  $C[\perp]$ . 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

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

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

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

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

# DPLL

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

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

## 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  $\perp$ .
- 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 PLE

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

## Example 1

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

- 1 Applying BCP produces

$$F'' : R \vee S$$

- 2 All variables occur positively, so  $F$  is satisfiable.
- 3 A satisfying interpretation:

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

## 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 PLE 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  $\perp$ , finishing this branch.

- On the other branch for  $Q$ :

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

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

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

# Summary

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