1. Propositional Logic

Huixing Fang

School of Information Engineering
Yangzhou University

Outline

1 Syntax

Syntax of PL: a set of **symbols** and **rules** for combining them to form "sentences" (formulae)

Symbols

1. Truth symbols: $\top$("true"), $\bot$("false")
2. Propositional variables: P, Q, P_i, Q_i, ...
3. Logical connectives: $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$

Example 1

1. $\neg P$: negation, "not";
2. $P \wedge Q$: conjunction, "and";
3. $P \vee Q$: disjunction, "or";
4. $P \rightarrow Q$: implication, "implies";
5. $P \leftrightarrow Q$: iff, "if and only if".

1 Syntax

Arity of Logical connectives

1. unary: negation($\neg$) is unary (takes one argument)
2. binary: others($\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$) are binary (take two arguments)

Antecedent/Consequent

The left and right arguments of $\rightarrow$ are called the antecedent and consequent, respectively.

$$P \rightarrow Q,$$

in which, P is antecedent, and Q is consequent.

Terminology

Atom : truth symbol $\top$, $\bot$ or propositional variable P, Q, ...
Literal : an atom A or its negation $\neg A$.
Formula : a literal or the application of a logical connective to formulae.

1 Syntax

Formula S is a **subformula** of formula F if it occurs syntactically within F.

Example 2 (Subformula)

1. subformula of P is P;
2. subformulae of $\neg F$: $\neg F$ and the subformulae of F;
3. subformulae of $F_1 \wedge F_2$: $F_1 \wedge F_2$ and the subformulae of F_1 and F_2.

Notice that every formula is a subformula of itself. The strict subformulae of a formula are all its subformulae except itself.

Example 3

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

in which, P and Q are propositional variables. Each instance of P and Q is an atom and a literal. $\neg Q$ is a literal, but not an atom. F has six subformulae: F, $P \wedge Q$, $P \vee \neg Q$, $\neg Q$, P, Q.

1 Syntax

Relative Precedence

The relative precedence of the logical connectives from highest to lowest:

$$\neg \quad > \quad \wedge \quad > \quad \vee \quad > \quad \to \quad > \quad \leftrightarrow$$

and, $\to$, $\leftrightarrow$ associate to the right.

Example 4

$(P \wedge Q) \to (P \vee \neg Q)$ can be abbreviated to $P \wedge Q \to P \vee \neg Q$

Example 5

$$(P_1 \wedge ((\neg P_2) \wedge \top)) \vee ((\neg P_1) \wedge P_2)$$

can be abbreviated to

$$P_1 \wedge \neg P_2 \wedge \top \vee \neg P_1 \wedge P_2$$

Outline

The **semantics** of a logic provides its meaning, and is given by the truth values **true** and **false**.

Definition 6 (Interpretation)

An **interpretation** $\mathcal{I}$ assigns to every **propositional variable** exactly one **truth value**.

Example 7

$$\mathcal{I} : \{P \mapsto \texttt{true}, Q \mapsto \texttt{false}, \ldots\}$$

is an interpretation assigning **true** to P and **false** to Q, where ... elides the (countably infinitely many) assignments that are not relevant to us.

Given a PL formula F and an interpretation $\mathcal{I}$, the truth value of F can be computed.

2.1 Truth Table

The simplest manner of computing the truth value of a PL formula F is via a truth table. How to evaluate each logical connective in terms of its arguments?

1. $\neg F$.

F	$\neg F$
0	1
1	0

2. Binary connectives.

F_1	F_2	$F_1 \land F_2$	$F_1 \lor F_2$	$F_1 \rightarrow F_2$	$F_1 \leftrightarrow F_2$
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	0
1	1	1	1	1	1

Example 8

Consider the formula

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

and the interpretation

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

To evaluate the truth value of F under $\mathcal{I}$, construct the following table:

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

The top row is given by the subformulae of F. $\mathcal{I}$ provides values for the first two columns.

Inductive Definition:

1. defines the meaning of basic elements first, i.e. atoms;
2. then defines a more complex element in terms of these elements.

Two symbols:

1. We write $\mathcal{I} \models F$ if F evaluates to **true** under interpretation $\mathcal{I}$
2. and write $\mathcal{I} \not\models F$ if F evaluates to **false**.

The meaning of truth symbols

1. $\mathcal{I} \models \top$
2. $\mathcal{I} \not\models \bot$

Under any interpretation $\mathcal{I}$, $\top$ has value **true**, and $\bot$ has value **false**.

2.2 Inductive Definition

The meaning of propositional variables

1. $\mathcal{I} \models P$ iif $\mathcal{I}[P] = \texttt{true}$, P has value **true** iff the interpretation $\mathcal{I}$ assigns P to have value **true**;
2. $\mathcal{I} \not\models P$ iif $\mathcal{I}[P] = \texttt{false}$,

Assume that formulae F, F_1, and F_2 have truth values. From these formulae, evaluate the semantics of more complex formulae:

Semantics of more complex formulae

$$
\begin{array}{ll}
\mathcal{I} \models \neg F & \text{iff } \mathcal{I} \not\models F \\
\mathcal{I} \models F_1 \wedge F_2 & \text{iff } \mathcal{I} \models F_1 \text{ and } \mathcal{I} \models F_2 \\
\mathcal{I} \models F_1 \vee F_2 & \text{iff } \mathcal{I} \models F_1 \text{ or } \mathcal{I} \models F_2 \\
\mathcal{I} \models F_1 \rightarrow F_2 & \text{iff, if } \mathcal{I} \models F_1 \text{ then } \mathcal{I} \models F_2 \\
\mathcal{I} \models F_1 \leftrightarrow F_2 & \text{iff } \mathcal{I} \models F_1 \text{ and } \mathcal{I} \models F_2, \text{or } \mathcal{I} \not\models F_1 \text{ and } \mathcal{I} \not\models F_2
\end{array}
$$

2.2 Inductive Definition

Example 9

Consider the formula $F : P \wedge Q \rightarrow P \vee \neg Q$ and the interpretation

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

Compute the truth value of F as follows:

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

We considered the distinct subformulae of F according to the subformula ordering: F_1 precedes F_2 if F_1 is a subformula of F_2.

Outline

3. Satisfiability and Validity

Determining satisfiability and validity of formulae are important tasks in logic.

Definition 10 (Satisfiable)

A formula F is satisfiable iff there exists **an interpretation** $\mathcal{I}$ such that $\mathcal{I} \models F$.

Definition 11 (Valid)

A formula F is valid iff for **all interpretations** $\mathcal{I} \models F$.

Satisfiability and validity are dual concepts, and switching from one to the other is easy. F is valid iff $\neg F$ is unsatisfiable.

Example 12

Consider the formula $F : P \wedge Q \rightarrow P \vee \neg Q$. Is it valid?

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

Example 13

Consider the formula $F : P \vee Q \rightarrow P \wedge Q$. Is it valid?

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

3.2 Semantic Argument Method

Semantic Argument

1. begins by assuming that the given formula F is invalid: there is a falsifying interpretation $\mathcal{I}$ such that $\mathcal{I} \not\models F$;
2. The proof proceeds by applying the semantic definitions of the logical connectives in the form of proof rules.

Proof rule

A proof rule has one or more **premises** (assumed facts) and one or more **deductions** (deduced facts).

An application of a proof rule requires matching the premises to facts already existing in the semantic argument and then forming the deductions.

Proof rules:

- According to the semantics of negation:

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

- According to the semantics of conjunction:

$$\frac{\mathcal{I} \models F \wedge G}{\begin{array}{c}\mathcal{I} \models F \\ \mathcal{I} \models G\end{array}} \qquad\qquad \frac{\mathcal{I} \not\models F \wedge G}{\mathcal{I} \not\models F \;\mid\; \mathcal{I} \not\models G}$$

The latter deduction results in a fork in the proof; each case must be considered separately.

Proof rules:

- According to the semantics of disjunction:

$$\frac{\mathcal{I} \models F \vee G}{\mathcal{I} \models F \quad | \quad \mathcal{I} \models G} \qquad\qquad \frac{\mathcal{I} \not\models F \vee G}{\begin{array}{c}\mathcal{I} \not\models F \\ \mathcal{I} \not\models G\end{array}}$$

- According to the semantics of implication:

$$\frac{\mathcal{I} \models F \rightarrow G}{\mathcal{I} \not\models F \quad | \quad \mathcal{I} \models G} \qquad\qquad \frac{\mathcal{I} \not\models F \rightarrow G}{\begin{array}{c}\mathcal{I} \models F \\ \mathcal{I} \not\models G\end{array}}$$

Proof rules:

- According to the semantics of iff:

$$\frac{\mathcal{I} \models F \leftrightarrow G}{\mathcal{I} \models F \wedge G \;\mid\; \mathcal{I} \not\models F \vee G} \qquad \frac{\mathcal{I} \not\models F \leftrightarrow G}{\mathcal{I} \models F \wedge \neg G \;\mid\; \mathcal{I} \models \neg F \wedge G}$$

- Contradiction occurs when an interpretation $\mathcal{I}$ both satisfies F and does not satisfy F:

$$\frac{\begin{array}{c}\mathcal{I} \models F \\ \mathcal{I} \not\models F\end{array}}{\mathcal{I} \models \perp}$$

Example 14

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

The contradiction indicates that our assumption must be wrong.

Example 15

To prove that the formula

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

is valid, assume otherwise and derive a contradiction:

1. $\mathcal{I} \not\models F$	assumption
2. $\mathcal{I} \models (P \rightarrow Q) \wedge (Q \rightarrow R)$	by 1 and semantics of $\rightarrow$
3. $\mathcal{I} \not\models (P \rightarrow R)$	by 1 and semantics of $\rightarrow$
4. $\mathcal{I} \models P$	by 3 and semantics of $\rightarrow$
5. $\mathcal{I} \not\models R$	by 3 and semantics of $\rightarrow$
6. $\mathcal{I} \models P \rightarrow Q$	by 2 and semantics of $\wedge$
7. $\mathcal{I} \models Q \rightarrow R$	by 2 and semantics of $\wedge$

There are two cases to consider from 6. In the first case,

$8a.\ \mathcal{I} \not\models P$	by 6 and semantics of $\rightarrow$
$9a.\ \mathcal{I} \models \perp$	4 and 8a are contradictory

In the second case,

$8b.\ \mathcal{I} \models Q$	by 6 and semantics of $\rightarrow$

Now there are two more cases from 7. In the first case,

$9ba.\ \mathcal{I} \not\models Q$	by 7 and semantics of $\rightarrow$
$10ba.\ \mathcal{I} \models \perp$	8b and 9ba are contradictory

In the second case,

$9bb.\ \mathcal{I} \models R$	by 7 and semantics of $\rightarrow$
$10bb.\ \mathcal{I} \models \perp$	5 and 9bb are contradictory

Example 16

The derived rule of modus ponens simplifies the proof of Example 15.

MODUS PONENS

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

1.	$\mathcal{I} \not\models F$	assumption
2.	$\mathcal{I} \models (P \rightarrow Q) \wedge (Q \rightarrow R)$	by 1 and semantics of $\rightarrow$
3.	$\mathcal{I} \not\models (P \rightarrow R)$	by 1 and semantics of $\rightarrow$
4.	$\mathcal{I} \models P$	by 3 and semantics of $\rightarrow$
5.	$\mathcal{I} \not\models R$	by 3 and semantics of $\rightarrow$
6.	$\mathcal{I} \models P \rightarrow Q$	by 2 and semantics of $\wedge$
7.	$\mathcal{I} \models Q \rightarrow R$	by 2 and semantics of $\wedge$
8.	$\mathcal{I} \models Q$	by 4, 6, and *modus ponens*
9.	$\mathcal{I} \models R$	by 8, 7, and *modus ponens*
10.	$\mathcal{I} \models \perp$	5 and 9 are contradictory

Outline

- Two formulae F_1 and F_2 are **equivalent** if they evaluate to the same truth value under all interpretations $\mathcal{I}$.
- We write $F_1 \Leftrightarrow F_2$ when F_1 and F_2 are equivalent.
- $F_1 \Leftrightarrow F_2$ is **not** a formula; it simply abbreviates the statement "F_1 and F_2 are equivalent."

Example 17

To prove that

$$P \Leftrightarrow \neg\neg P,$$

we prove that

$$P \leftrightarrow \neg\neg P$$

is valid via a truth table:

P	$\neg P$	$\neg\neg P$	$P \leftrightarrow \neg\neg P$
0	1	0	1
1	0	1	1

- Formula F_1 **implies** formula F_2 if $\mathcal{I} \models F_2$ for every interpretation $\mathcal{I}$ such that $\mathcal{I} \models F_1$.
- Another way to state that F_1 implies F_2 is to assert the validity of the formula $F_1 \rightarrow F_2$.
- We write $F_1 \Rightarrow F_2$ when F_1 implies F_2. The implication $F_1 \Rightarrow F_2$ is **not** a formula.

Example 18

To prove that

$$R \wedge (\neg R \vee P) \Rightarrow P,$$

we prove that

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

is valid via a semantic argument.

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

Suppose F is not valid; then there exists an interpretation $\mathcal{I}$ such that $\mathcal{I} \not\models F$.

1. $\mathcal{I} \not\models F$	assumption
2. $\mathcal{I} \models R \land (\neg R \lor P)$	by 1 and semantics of $\rightarrow$
3. $\mathcal{I} \not\models P$	by 1 and semantics of $\rightarrow$
4. $\mathcal{I} \models R$	by 2 and semantics of $\land$
5. $\mathcal{I} \models \neg R \lor P$	by 2 and semantics of $\land$

There are two cases to consider. In the first case,

$6a.\ \mathcal{I} \models \neg R$	by 5 and semantics of $\lor$
$7a.\ \mathcal{I} \models \bot$	4 and 6a are contradictory

What is the second case?

Outline

- Substitution is a **syntactic operation** on formulae with significant semantic consequences.
- It allows us to **prove** the validity of entire sets of formulae via **formula templates**.

Definition 19 (5 Substitution)

A **substitution** σ is a mapping from formulae to formulae:

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

The **domain** of σ, $\mathtt{domain}(\sigma)$, is

$$\mathtt{domain}(\sigma) : \{F_1, ..., F_n\},$$

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

$$\mathtt{range}(\sigma) : \{G_1, ..., G_n\}.$$

5 Substitution

- The **application** of a substitution σ to a formula F, $F\sigma$, **replaces** each occurrence of a formula F_i in $\texttt{domain}(\sigma)$ with its corresponding formula G_i in $\texttt{range}(\sigma)$.
- when both subformulae F_j and F_k are in $\texttt{domain}(\sigma)$, and F_k is a strict subformula of F_j , then F_j is replaced by the corresponding formula G_j.

Example 20

Consider the formula

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

and the substitution

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

Then

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

5 Substitution

- A **variable substitution** is a substitution in which the **domain** consists **only of propositional variables**.
- Useful notation: $F[F_1, ..., F_n]$, we mean that formula F can have formulae F_i, $i = 1, ..., n$, as subformulae.
- Two semantic consequences can be derived from substitution.

Proposition 21 (Substitution of Equivalent Formulae)

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

Proposition 22 (Valid Template)

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

Example 23

Consider applying substitution

$$\sigma : \{P \to Q \mapsto \neg P \vee Q\}$$

to

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

Since $P \to Q \Leftrightarrow \neg P \vee Q$, the formula

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

is equivalent to F.

Composition of substitutions

Given substitutions σ_1 and σ_2, compute substitution σ such that $F\sigma_1\sigma_2 = F\sigma$ for any F:

1. apply σ_2 to each formula of the range of σ_1, and add the results to σ;
2. if F_i of $F_i \mapsto G_i$ appears in the domain of σ_2 but not in the domain of σ_1, then add $F_i \mapsto G_i$ to σ.

Example 24

Compute the composition of substitutions

$$\sigma_1\sigma_2 : \{P \mapsto R, P \wedge Q \mapsto P \to Q\}\{P \mapsto S, S \mapsto Q\}$$

as follows:

$$\begin{aligned}\sigma &= \{P \mapsto R\sigma_2, P \wedge Q \mapsto (P \to Q)\sigma_2, S \mapsto Q\} \\ &= \{P \mapsto R, P \wedge Q \mapsto S \to Q, S \mapsto Q\}\end{aligned}$$

Outline

6 Normal Forms

- A **normal form** of formulae is a **syntactic restriction** such that for every formula of the logic, there is an equivalent formula in the normal form.
- Three normal forms are particularly important for PL:
  1. **NNF**: Negation normal form;
  2. **DNF**: Disjunctive normal form;
  3. **CNF**: Conjunctive normal form.

Negation Normal Form (NNF)

NNF requires that $\neg$, $\wedge$, and $\vee$ be the only connectives and that **negations** appear **only** in **literals**.

Transforming a formula F to equivalent formula F' in NNF can be computed **recursively** using the following list of **template equivalences**:

$$\neg\neg F_1 \Leftrightarrow F_1 \tag{1}$$

$$\neg\top \Leftrightarrow \bot \tag{2}$$

$$\neg\bot \Leftrightarrow \top \tag{3}$$

$$\neg(F_1 \wedge F_2) \Leftrightarrow \neg F_1 \vee \neg F_2 \tag{4}$$

$$\neg(F_1 \vee F_2) \Leftrightarrow \neg F_1 \wedge \neg F_2 \tag{5}$$

$$F_1 \to F_2 \Leftrightarrow \neg F_1 \vee F_2 \tag{6}$$

$$F_1 \leftrightarrow F_2 \Leftrightarrow (F_1 \to F_2) \wedge (F_2 \to F_1) \tag{7}$$

The equivalences (4) and (5) are known as De Morgan's Law.

Example 25

To convert the formula $F : \neg(P \to \neg(P \wedge Q))$ into NNF, apply the template equivalence

$$F_1 \to F_2 \Leftrightarrow \neg F_1 \vee F_2$$

to produce $F' : \neg(\neg P \vee \neg(P \wedge Q))$, apply De Morgan's law

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

to produce $F'' : \neg\neg P \wedge \neg\neg(P \wedge Q)$, apply

$$\neg\neg F_1 \Leftrightarrow F_1$$

twice to produce

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

which is in NNF and equivalent to F.

6.2 DNF

Disjunctive Normal Form (DNF)

A formula is in disjunctive normal form if it is a disjunction of conjunctions of literals:

$$\bigvee_i \bigwedge_j \ell_{i,j} \text{ for literals } \ell_{i,j}.$$

To convert a formula F into an equivalent formula in DNF, **transform F into NNF** and then use the following table of **template equivalences**:

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

When implementing the transformation, the equivalences should be applied left-to-right. The equivalences simply say that **conjunction distributes over disjunction**.

6.2 DNF

Example 26

To convert

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

into DNF, first transform it into NNF

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

and then apply distributivity to obtain

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

and then distributivity twice again to produce

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

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

6.3 CNF

Conjunctive Normal Form (CNF)

A formula is in conjunctive normal form if it is a conjunction of disjunctions of literals:

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

Each inner block of disjunctions is called a **clause**.

To convert a formula F into an equivalent formula in CNF, **transform F into NNF** and then use the following table of **template equivalences**:

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

Example 27

To convert

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

into CNF, first transform F into NNF:

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

Then apply distributivity to obtain

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

which is in CNF and equivalent to F.

Outline

7 Decision Procedures for Satisfiability

Decision Procedure

A **decision procedure** for satisfiability of PL formulae reports, after some finite amount of **computation**, **whether** a given PL formula F is **satisfiable**.

Algorithm 1: Decision procedure based on the truth-table method

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

SAT is a recursive function that takes one argument. This algorithm returns **true** immediately upon finding a satisfying interpretation.

Example 28

Consider the formula

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

To compute **SAT** F, choose a variable, say P, and recurse on the first case,

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

which simplifies to

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

Now try each of

$$F_1\{Q \mapsto \top\} \text{ and } F_1\{Q \mapsto \bot\}.$$

Both simplifies to $\bot$, so this branch ends without finding a satisfying interpretation.

Now try the other branch for P in F:

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

which simplifies to $\bot$. This branch also ends without finding a satisfying interpretation. Thus, F is unsatisfiable.

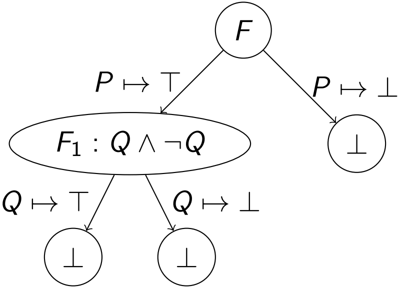

Figure: Visualizing runs of **SAT**

Example 29

Consider the formula

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

To compute **SAT** F, choose a variable, say P, and recurse on the first case,

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

which simplifies to $\bot$. Therefore, try

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

instead, which simplifies to $\top$. Arbitrarily assigning a value to Q produces the following satisfying interpretation:

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

7.1 PL formulae to Equisatisfiable CNF Formulae

- The transformation (Section 6) produces an equivalent formula that can be exponentially larger: e.g., DNF to CNF.
- To decide the satisfiability of F, we need only examine an equisatisfiable formula F'.
- F and F' are **equisatisfiable** when F is satisfiable iff F' is satisfiable.

Conversion(at most constant factor larger)

1. introduce new propositional variables to represent the subformulae of formula F;
2. add extra clauses (in F') that assert that these new variables are equivalent to the subformulae that they represent.

7.1 PL formulae to Equisatisfiable CNF Formulae

Representative Function (Rep)

$$\begin{aligned} \mathtt{Rep} : \mathtt{PL} &\to \mathcal{V} \cup \{\top, \bot\} \\ F &\mapsto P_F \end{aligned}$$

$\mathtt{PL}$ is the set of PL formulae, $\mathcal{V}$ represents the set of propositional variables. P_F provides a compact way of referring to F.

Encoding Function (En)

$$\begin{aligned} \mathtt{En} : \mathtt{PL} &\to \mathtt{PL} \\ F &\mapsto F' \end{aligned}$$

intended to map a PL formula F to a PL formula F' in CNF that asserts that F's representative, P_F, is equivalent to F: $\mathtt{Rep}(F) \leftrightarrow F$.

7.1 PL formulae to Equisatisfiable CNF Formulae

Base cases for defining Rep and En

On $\top$, $\perp$ and propositional variables P:

$$
\begin{aligned}
\mathtt{Rep}(\top) &= \top & \mathtt{En}(\top) &= \mathtt{Rep}(\top) \leftrightarrow \top = \top \leftrightarrow \top = \top \\
\mathtt{Rep}(\perp) &= \perp & \mathtt{En}(\perp) &= \mathtt{Rep}(\perp) \leftrightarrow \perp = \perp \leftrightarrow \perp = \perp \\
\mathtt{Rep}(P) &= P & \mathtt{En}(P) &= \mathtt{Rep}(P) \leftrightarrow P = P \leftrightarrow P = \top
\end{aligned}
$$

For the inductive case, F is a formula other than an atom:

$$\mathtt{Rep}(F) = P_F.$$

En then asserts the equivalence of F and P_F as a CNF formula.

7.1 PL formulae to Equisatisfiable CNF Formulae

On Conjunction

On conjunction, $F_1 \wedge F_2$:

$$\begin{aligned}
&\texttt{En}(F_1 \wedge F_2) = \\
&\quad \texttt{let } P = \texttt{Rep}(F_1 \wedge F_2) \texttt{ in} \\
&\quad (\neg P \vee \texttt{Rep}(F_1)) \wedge (\neg P \vee \texttt{Rep}(F_2)) \wedge (\neg \texttt{Rep}(F_1) \vee \neg \texttt{Rep}(F_2) \vee P)
\end{aligned}$$

The returned formula

$$(\neg P \vee \texttt{Rep}(F_1)) \wedge (\neg P \vee \texttt{Rep}(F_2)) \wedge (\neg \texttt{Rep}(F_1) \vee \neg \texttt{Rep}(F_2) \vee P)$$

is in CNF and is equivalent to

$$\texttt{Rep}(F_1 \wedge F_2) \leftrightarrow \texttt{Rep}(F_1) \wedge \texttt{Rep}(F_2).$$

In detail, the first two clauses together assert $P \leftrightarrow \texttt{Rep}(F_1) \wedge \texttt{Rep}(F_2)$, the last clause asserts $\texttt{Rep}(F_1) \wedge \texttt{Rep}(F_2) \to P$.

7.1 PL formulae to Equisatisfiable CNF Formulae

On Negation

$\texttt{En}(\neg F)$ returns a formula equivalent to $\texttt{Rep}(\neg F) \leftrightarrow \neg\texttt{Rep}(F)$:

$$
\begin{aligned}
&\texttt{En}(\neg F) = \\
&\quad \texttt{let } P = \texttt{Rep}(\neg F) \texttt{ in} \\
&\quad (\neg P \vee \neg\texttt{Rep}(F)) \wedge (P \vee \texttt{Rep}(F))
\end{aligned}
$$

On Disjunction

$$
\begin{aligned}
&\texttt{En}(F_1 \vee F_2) = \\
&\quad \texttt{let } P = \texttt{Rep}(F_1 \vee F_2) \texttt{ in} \\
&\quad (\neg P \vee \texttt{Rep}(F_1) \vee \texttt{Rep}(F_2)) \wedge (\neg\texttt{Rep}(F_1) \vee P) \\
&\quad\ \wedge (\neg\texttt{Rep}(F_2) \vee P)
\end{aligned}
$$

7.1 PL formulae to Equisatisfiable CNF Formulae

On Implication

$$
\begin{aligned}
&\texttt{Rep}(F_1 \to F_2) = \\
&\quad \texttt{let}\, P = \texttt{Rep}(F_1 \to F_2) \texttt{ in} \\
&\quad (\neg P \vee \neg\texttt{Rep}(F_1) \vee \texttt{Rep}(F_2)) \wedge (\texttt{Rep}(F_1) \vee P) \wedge (\neg\texttt{Rep}(F_2) \vee P)
\end{aligned}
$$

On IIF

$$
\begin{aligned}
&\texttt{En}(F_1 \leftrightarrow F_2) = \\
&\quad \texttt{let } P = \texttt{Rep}(F_1 \leftrightarrow F_2) \texttt{ in} \\
&\quad (\neg P \vee \neg\texttt{Rep}(F_1) \vee \texttt{Rep}(F_2)) \wedge (\neg P \vee \texttt{Rep}(F_1) \vee \neg\texttt{Rep}(F_2)) \\
&\quad\ \wedge (P \vee \neg\texttt{Rep}(F_1) \vee \neg\texttt{Rep}(F_2)) \wedge (P \vee \texttt{Rep}(F_1) \vee \texttt{Rep}(F_2))
\end{aligned}
$$

7.1 PL formulae to Equisatisfiable CNF Formulae

Full CNF formula equisatisfiable to F

If S_F is the set of all subformulae of F (including F itself), then

$$F' : \mathtt{Rep}(F) \wedge \bigwedge_{G \in S_F} \mathtt{En}(G)$$

is in CNF and is equisatisfiable to F.

Example 30

Consider formula

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

which is in DNF. To convert it to CNF, we collect its subformulae

$$S_F : \{Q_1, Q_2, Q_1 \wedge Q_2, R_1, R_2, R_1 \wedge R_2, F\}$$

Continue Example 30, compute

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

Then

$$F' = P_{(F)} \wedge \bigwedge_{G \in S_F} \mathtt{En}(G)$$

is equisatisfiable to F and is in CNF.

7.2 The Resolution Procedure

Resolution follows from the following observation of any PL formula F in CNF:

- 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;
- If P is **true**, then a literal other than $\neg P$ in C_2 must be satisfied;
- If P is **false**, then a literal other than P in C_1 must be satisfied.

Resolution

Clausal resolution is stated as the following proof rule:

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

In which, $P \mapsto \bot$ for C_1, and $\neg P \mapsto \bot$ in C_2. From the two clauses of the premise, deduce the new clause, called the **resolvent**. The resolvent represents that other literals (excludes P and $\neg P$) have to be satisfied.

Example 31

The CNF of $(P \to Q) \wedge P \wedge \neg Q$ is the following:

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

From resolution

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

construct

$$F_1 : (\neg P \vee Q) \wedge P \wedge \neg Q \wedge Q.$$

From resolution

$$\frac{\neg Q \qquad Q}{\perp},$$

deduce that F is unsatisfiable.

Example 32

Consider the formula

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

The one possible resolution

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

yields

$$F_1 : (\neg P \vee Q) \wedge \neg Q \wedge \neg P.$$

Since no further resolutions are possible, F is satisfiable. Indeed,

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

is a satisfying interpretation.

Modern satisfiability procedures for propositional logic are based on the **D**avis-**P**utnam-**L**ogemann-**L**oveland algorithm (DPLL)

- DPLL operates on PL formulae in CNF
- DPLL applies a restricted form of resolution: Boolean constraint propagation (BCP)
- BCP is based on unit resolution

Unit Resolution

Unit resolution operates on two clauses. One clause, called the **unit clause**, consists of a single literal ℓ. The second clause contains the negation of ℓ: $C[\neg\ell]$:

$$\frac{\ell \qquad C[\ell]}{C[\bot]}.$$

Example 33

In the formula $F : (P) \wedge (\neg P \vee Q) \wedge (R \vee \neg Q \vee S)$, (P) is a unit clause. Therefore applying unit resolution

$$\frac{P \qquad (\neg P \vee Q)}{Q} (\text{partial interpretation } \{P \mapsto \top\})$$

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

$$\frac{Q \qquad (R \vee \neg Q \vee S)}{R \vee S} (\text{partial interpretation } \{Q \mapsto \top\})$$

produces

$$F'' : (R \vee S),$$

ending this round of BCP.

The implementation of DPLL is structurally similar to **SAT**, except that it begins by applying BCP:

Algorithm 2: Basic DPLL

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

As in **SAT**, intermediate formulae are simplified according to the template equivalences.

Example 34

Consider the formula

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

On the first level of recursion, **DPLL** recognizes the unit clause (P) and applies the BCP steps from Example 33, resulting in the formula

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

The unit resolutions correspond to the partial interpretation

$$\{P \mapsto \texttt{true}, Q \mapsto \texttt{true}\}.$$

Only positively occurring variables remain, so F is satisfiable. In particular,

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

is a satisfyng interpretation of F.

Example 35

Consider the formula

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

Branching on Q or R will result in unit clauses; choose Q. Then

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

The unit resolution

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

finishes this branch. On the other branch,

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

P appears only negatively, and R appears only positively, so the formula is satisfiable. F is satisfied by $\mathcal{I} : \{P \mapsto \texttt{false}, Q \mapsto \texttt{false}, R \mapsto \texttt{true}\}$.

7.3 DPLL

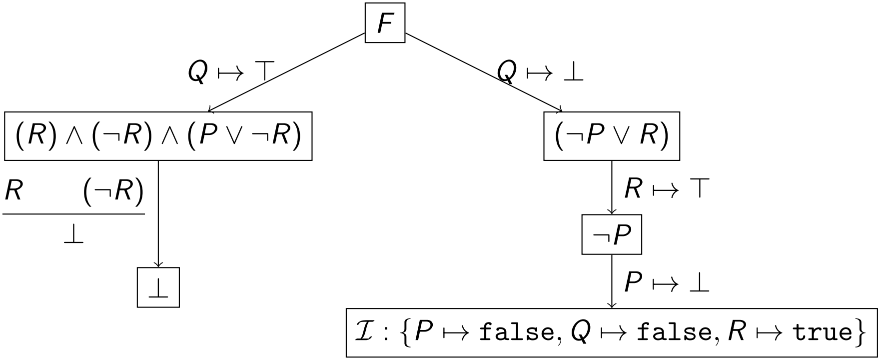

Figure: Visualizing of Example 35

Summary

1. **Syntax**. How one constructs a PL formula. Propositional variables, atoms, literals, logical connectives
2. **Semantics**. What a PL formula means. Truth values **true** and **false**. Interpretations. Truth-table definition, inductive definition.
3. **Satisfiability and validity**. Whether a PL formula evaluates to true under any or all interpretations. Duality of satisfiability and validity, truth-table method, semantic argument method.
4. **Equivalence and implication**. Whether two formulae always evaluate to the same truth value under every interpretation. Whether under any interpretation, if one formula evaluates to true, the other also evaluates to true. Reduction to validity.
5. **Substitution**, which is a tool for manipulating formulae and making general claims. Substitution of equivalent formulae. Valid templates.
6. **Normal forms and Decision procedures for satisfiability**.