Normal Forms of Propositional Logic

Bow-Yaw Wang

Institute of Information Science
Academia Sinica, Taiwan

September 12, 2017
Normal Forms

- Semantic equivalence, satisfiability, and validity
- Conjunctive normals forms and validity
- Horn clauses and satisfiability
Consider two formulae $\phi_1 \land \phi_2$ and $\phi_2 \land \phi_1$.

Intuitively, $\phi_1 \land \phi_2$ and $\phi_2 \land \phi_1$ should have the same “meaning.”

More formally, two formulae $\phi$ and $\psi$ have the same meaning if their truth tables coincide.

**Definition**

Let $\phi$ and $\psi$ be propositional logic formulae. $\phi$ and $\psi$ are **semantically equivalent** (written $\phi \equiv \psi$) if both $\phi \models \psi$ and $\psi \models \phi$ hold.

**Examples**

\[
\begin{align*}
p \implies q & \equiv \neg q \implies \neg p \quad & p \implies q & \equiv \neg p \lor q \\
p \land q \implies p & \equiv r \lor \neg r \quad & p \land q \implies r & \equiv p \implies (q \implies r)
\end{align*}
\]

A formula $\phi$ is valid if it is a tautology.

**Definition**

Let $\phi$ be a propositional logic formula. $\phi$ is **valid** if $\models \phi$. 
Lemma

Let $\phi_1, \phi_2, \ldots, \phi_n, \psi$ be propositional logic formulae. $\phi_1, \phi_2, \ldots, \phi_n \models \psi$ iff $\models \phi_1 \implies (\phi_2 \implies \cdots \implies (\phi_n \implies \psi))$.

Proof.

Suppose $\models \phi_1 \implies (\phi_2 \implies \cdots \implies (\phi_n \implies \psi))$. Consider any valuation. If $\phi_1, \phi_2, \ldots, \phi_n$ evaluate to $T$ under the valuation, $\phi$ must evaluate to $T$ since $\models \phi_1 \implies (\phi_2 \implies \cdots \implies (\phi_n \implies \psi))$. Hence $\phi_1, \phi_2, \ldots, \phi_n \models \psi$.

The other direction is proved in Step 1 of the completeness theorem.
Conjunctive Normal Form (CNF)

Definition

A literal $L$ is either an atom $p$ or its negation $\neg p$. A clause $D$ is a disjunction of literals. A formula $C$ is in conjunctive normal form (CNF) if it is a conjunction of clauses.

\[
L ::= p | \neg p \\
D ::= L | L \lor D \\
C ::= D | D \land C
\]

Examples: \((\neg q \lor p \lor r) \land (\neg p \lor r) \land q, (p \lor r) \land (\neg p \lor r) \land (p \lor \neg r)\)
Validity of CNF Formulae

Lemma

A clause \( L_1 \lor L_2 \lor \cdots \lor L_m \) is valid iff there is a propositional atom \( p \) such that \( L_i \) is \( p \) and \( L_j \) is \( \neg p \) for some \( 1 \leq i, j \leq m \).

Proof.

Without loss of generality, assume \( L_1 = p \) and \( L_2 = \neg p \). Then \( p \lor \neg p \lor L_3 \lor \cdots \lor L_m \) evaluates to \( T \) for any valuation. The clause is valid. Conversely, consider the valuation where all literals evaluate to \( F \). This is possible since every literal \( L_i \) has no negation in the clause. The clause evaluates to \( F \) under the valuation.

Examples:

- \( p \lor q \lor q \lor \neg p \lor r \) is valid;
- \( p \lor \neg q \lor r \lor \neg q \) is not valid (consider \( \{ p \leftrightarrow F, q \leftrightarrow T, r \leftrightarrow F \} \)).

For any propositional logic formula \( \phi \) in CNF, the validity of \( \phi \) can be checked in linear time.
Satisfiability of CNF Formulae

Definition

Let \( \phi \) be a propositional logic formula. \( \phi \) is **satisfiable** if it evaluates to T under some valuation.

Example: \( p \lor q \implies p \) is satisfiable (consider \( \{ p \mapsto T, q \mapsto T \} \) ); it is 
not valid (consider \( \{ p \mapsto F, q \mapsto T \} \) ).

Proposition

Let \( \phi \) be a propositional logic formula. \( \phi \) is satisfiable iff \( \neg \phi \) is not valid.

Proof.

Suppose \( \phi \) evaluates to T under a valuation. Then \( \neg \phi \) evaluates to F under the valuation. \( \neg \phi \) is not valid. 
Conversely, suppose \( \neg \phi \) is not valid. Hence \( \neg \phi \) evaluates to F under a valuation. Thus \( \phi \) evaluates to T under the valuation. \( \phi \) is satisfiable.
Suppose we have the truth table for a formula $\phi$ with propositional atoms $p_1, p_2, \ldots, p_n$.

For each line $l$ where $\phi$ evaluates to F, construct a clause $\psi_l$ as follows.

$$
\psi_l = L_{l,1} \lor L_{l,2} \lor \cdots \lor L_{l,n}
$$

where $L_{l,j} = \neg p_j$ if $p_j$ is T at line $l$; otherwise $L_{l,j} = p_j$.

Then $\phi \equiv \psi_1 \land \psi_2 \land \cdots \psi_m$ where $\psi_i$’s are constructed for every line evaluating $\phi$ to F.

Observe that $\psi_1 \land \psi_2 \land \cdots \psi_m$ is F iff $\psi_l$ is F for some $1 \leq l \leq m$.
\[
\psi_l = L_{l,1} \lor L_{l,2} \lor \cdots \lor L_{l,n}
\]

is F iff $L_{l,j}$ is F for every $1 \leq j \leq n$. $L_{l,j}$ is F iff $p_j$ has its truth value at line $l$.

In other words, $\psi_1 \land \psi_2 \land \cdots \psi_m$ is F under a valuation iff the valuation evaluates $\phi$ to F in $\phi$’s truth table.
Example

Translate $p \lor q \implies q \land \neg r$ into CNF.

Proof.

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$r$</th>
<th>$p \lor q \implies q \land \neg r$</th>
<th>$p$</th>
<th>$q$</th>
<th>$r$</th>
<th>$p \lor q \implies q \land \neg r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$r$</th>
<th>$\psi_1$</th>
<th>$p$</th>
<th>$q$</th>
<th>$r$</th>
<th>$\psi_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>$p \lor \neg q \lor \neg r$</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>$\neg p \lor q \lor r$</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>$\neg p \lor q \lor \neg r$</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>$\neg p \lor \neg q \lor \neg r$</td>
</tr>
</tbody>
</table>

$p \lor q \implies q \land \neg r \equiv (p \lor \neg q \lor \neg r) \land (\neg p \lor q \lor r) \land (\neg p \lor q \lor \neg r) \land (\neg p \lor \neg q \lor \neg r)$. ■
Normal Forms

- Semantic equivalence, satisfiability, and validity
- Conjunctive normals forms and validity
- Horn clauses and satisfiability
Validity Checking

- Given a propositional logic formula in conjunctive normal form, we can check the validity of the formula in linear time.
- Recall that a formula is valid iff it is a theorem.
- If we can translate any propositional logic formula into conjunctive normal form, we can check the validity of the formula!
- We know how to translate any logic formula to conjunctive normal form by its truth table.
  - This is not satisfactory. If we have to construct its truth table, we can check validity already.
- We will give an algorithm $\text{CNF}(\phi)$ to convert any propositional logic formula into conjunctive normal form without building its truth table.
Any propositional logic formula can be transformed to conjunctive normal form by the following equivalences:

\[
\begin{align*}
\phi & \iff \psi \equiv \neg \phi \lor \psi \\
\neg (\phi \land \psi) & \equiv \neg \phi \lor \neg \psi \\
\neg (\phi \lor \psi) & \equiv \neg \phi \land \neg \psi \\
\phi \land (\psi_1 \lor \psi_2) & \equiv (\phi \land \psi_1) \lor (\phi \land \psi_2) \\
\phi \lor (\psi_1 \land \psi_2) & \equiv (\phi \lor \psi_1) \land (\phi \lor \psi_2)
\end{align*}
\]

The algorithm $\text{CNF}(\phi)$ hence consists of three steps:

- Remove every implication ($\iff$) from $\phi$ (Algorithm $\text{IMPL\_FREE}(\phi)$);
- Push every negation ($\neg$) to literals (Algorithm $\text{NNF}(\phi)$);
- Apply law of distribution (Algorithm $\text{CNF}(\phi)$).
Algorithm IMPL_FREE(φ)

Input: φ : a logic formula
Output: φ′ : all implications ( ⇒ ) in φ′ are removed and φ′ ≡ φ

switch φ do
  case φ is a literal: do return φ;
  case φ is ¬φ₁: do return ¬IMPL_FREE(φ₁);
  case φ is φ₁ ∧ φ₂: do return IMPL_FREE(φ₁) ∧ IMPL_FREE(φ₂);
  case φ is φ₁ ∨ φ₂: do return IMPL_FREE(φ₁) ∨ IMPL_FREE(φ₂);
  case φ is φ₁ ⇒ φ₂: do return IMPL_FREE(¬φ₁ ∨ φ₂);
  otherwise do assert(0);

Algorithm 1: IMPL_FREE(φ)
Algorithm $\text{NNF}(\phi)$

**Input:** $\phi$: a logic formula without implication ($\implies$)

**Output:** $\phi'$: only propositional atoms in $\phi'$ are negated and $\phi' \equiv \phi$

switch $\phi$ do

- case $\phi$ is a literal: do return $\phi$;
- case $\phi$ is $\neg

\neg \phi_1$: do return $\text{NNF}(\phi_1)$;
- case $\phi$ is $\phi_1 \land \phi_2$: do return $\text{NNF}(\phi_1) \land \text{NNF}(\phi_2)$;
- case $\phi$ is $\phi_1 \lor \phi_2$: do return $\text{NNF}(\phi_1) \lor \text{NNF}(\phi_2)$;
- case $\phi$ is $\neg(\phi_1 \land \phi_2)$: do return $\text{NNF}(\neg \phi_1 \lor \neg \phi_2)$;
- case $\phi$ is $\neg(\phi_1 \lor \phi_2)$: do return $\text{NNF}(\neg \phi_1 \land \neg \phi_2)$;
- otherwise do assert(0);

Algorithm 2: $\text{NNF}(\phi)$

**Definition**

Let $\phi$ be a propositional logic formula. If only propositional atoms in $\phi$ are negated, $\phi$ is in **negation normal form**.
Algorithm $\text{CNF}(\phi)$

Input: $\phi$ : an NNF formula without implication ($\implies$)
Output: $\phi'$ : $\phi'$ is in CNF and $\phi' \equiv \phi$

switch $\phi$ do
    case $\phi$ is a literal: do return $\phi$
    case $\phi$ is $\phi_1 \land \phi_2$: do return $\text{CNF}(\phi_1) \land \text{CNF}(\phi_2)$
    case $\phi$ is $\phi_1 \lor \phi_2$: do return $\text{DISTR}($CNF($\phi_1$), CNF($\phi_2$))$

Algorithm 3: $\text{CNF}(\phi)$

Input: $\eta_1, \eta_2$ : $\eta_1, \eta_2$ are in CNF
Output: $\phi'$ : $\phi'$ is in CNF and $\phi' \equiv \eta_1 \lor \eta_2$

if $\eta_1$ is $\eta_{11} \land \eta_{12}$ then return $\text{DISTR}(\eta_{11}, \eta_2) \land \text{DISTR}(\eta_{12}, \eta_2)$
else if $\eta_2$ is $\eta_{21} \land \eta_{22}$ then return $\text{DISTR}(\eta_1, \eta_{21}) \land \text{DISTR}(\eta_1, \eta_{22})$
else return $\eta_1 \lor \eta_2$

Algorithm 4: $\text{DISTR}(\eta_1, \eta_2)$
Let $\phi$ be a propositional logic formula. Consider the following algorithm for checking its satisfiability.

1. Compute a CNF formula $\psi$ such that $\psi \equiv \neg \phi$.
2. Check the validity of $\psi$.
3. Return “$\phi$ is satisfiable” if $\psi$ is not valid; Return “$\phi$ is not satisfiable” if $\psi$ is valid.

Recall that satisfiability of propositional logic formulae is an NP-complete problem.

Is the above algorithm in polynomial time? Why?
1 Normal Forms
- Semantic equivalence, satisfiability, and validity
- Conjunctive normals forms and validity
- Horn clauses and satisfiability
Given a propositional logic formula in CNF, it is easy to check its validity; it is “hard” to check its satisfiability.

We will consider a subclass of CNF formulae whose satisfiability can be checked efficiently.

**Definition**

A Horn formula is a propositional logic formula $\phi$ of the following form:

$$
P ::= \bot | \top | p 
A ::= P | P \land A 
C ::= A \implies P 
H ::= C | C \land H.
$$

A clause of the form $C$ is called a Horn clause.

**Example:** $(p \land q \land s \implies \bot) \land (q \land r \implies p) \land (\top \implies s)$

**Nonexample:** $(p \land \neg q \land s \implies \bot) \land (q \land r \implies p \land s) \land (p \lor r \implies s)$
Satisfiability of Horn Formulae

- Consider a Horn clause $P_1 \land P_2 \land \cdots \land P_n \implies Q$.
- If $P_1, P_2, \ldots, P_n$ are assigned to T, then $Q$ must be T; otherwise, $Q$ can be an arbitrary truth value.
- We hence have the following (informal) algorithm:
  1. Mark $\top$ if it occurs in the Horn formula $\phi$;
  2. If there is a Horn clause $P_1 \land P_2 \land \cdots \land P_n \implies Q$ in $\phi$ such that all $P_j$ for $1 \leq j \leq n$ are marked, mark $Q$;
  3. If $\bot$ is marked, print “The Horn formula $\phi$ is unsatisfiable.”
  4. Print “The Horn formula $\phi$ is satisfiable.”
Algorithm $\text{Horn}(\phi)$

**Input:** $\phi$: $\phi$ is a Horn formula

**Output:** “unsatisfiable” if $\phi$ is unsatisfiable; otherwise “satisfiable.”

mark all occurrences of $\top$ in $\phi$;

while there is a Horn clause $P_1 \land P_2 \land \cdots \land P_n \implies Q$ in $\phi$ such that $P_j$ are all marked but $Q$ is not do

mark $Q$;

if $\bot$ is marked then return “unsatisfiable”;
else return “satisfiable”;

**Algorithm 5:** $\text{Horn}(\phi)$
Theorem

Let $\phi$ be a Horn formula with $n$ propositional atoms. $\text{Horn}(\phi)$ runs at most $n + 1$ iterations and decides the satisfiability of $\phi$ correctly.

Proof.

At each iteration, an unmarked atom will be marked. Since there are $n$ atoms, there are at most $n + 1$ iterations.

By induction on the number of iterations, we show that “all marked $P$ are true for all valuations where $\phi$ evaluates to $T$.” At iteration 0 (before entering the loop), only $\top$ are marked. Clearly, $\top$ must be true for any valuation. At iteration $k + 1$, consider a Horn clause $P_1 \land P_2 \land \cdots \land P_n \implies Q$ where $P_1, P_2, \ldots, P_n$ are marked but not $Q$. For a valuation $\nu$ where $\phi$ evaluates to $T$, $P_1 \land P_2 \land \cdots \land P_n \implies Q$ must evaluate to $T$. Since $P_1, P_2, \ldots, P_n$ are true in $\nu$ (by IH), $Q$ must be true in $\nu$. 
We now prove $\text{Horn}(\phi)$ answers correctly. When $\text{Horn}(\phi)$ returns “unsatisfiable,” there is a Horn clause $P_1 \land P_2 \land \cdots \land P_n \Rightarrow \bot$ where $P_1, P_2, \ldots, P_n$ are all marked. Suppose $\nu$ is a valuation where $\phi$ evaluates to T. Then $P_1, P_2, \ldots, P_n$ must be true in $\nu$. Hence $P_1 \land P_2 \land \cdots \land P_n \Rightarrow \bot$ evaluates to F. $\phi$ cannot evaluate to T under $\nu$. A contradiction.

When $\text{Horn}(\phi)$ returns “satisfiable,” define a valuation $\nu$ where all marked propositional atoms are assigned to T and all unmarked atoms are F. We claim $\phi$ evaluates to T in $\nu$. Suppose not. There is a Horn clause $P_1 \land P_2 \land \cdots P_n \Rightarrow Q$ in $\phi$ which evaluates to F under $\nu$. That is, $P_1, P_2, \ldots, P_n$ are T but $Q$ is F under $\nu$. By the definition of $\nu$, $P_1, P_2, \ldots, P_n$ are marked by the algorithm. Hence $Q$ must also be marked by the algorithm. $Q$ cannot be F in $\nu$. A contradiction.