Propositional Logic: Normal Forms

CS402, Spring 2016

Shin Yoo

Overview

- Conjunctive Normal Forms and Validity
- Horn Clauses and Satisfiability

Note that the material corresponds to Chapter 1.5.2 and 1.5.3 of *Logic in Computer Science* by M. Huth and M. Ryan, the second reference book.

Normal Forms

Advantages of Normal Forms

- A mechanical tool can handle a formula of a normal form much easier.
- There are special algorithms to solve satisfiability of a formula very efficiently if the formula is written in some normal form.

We will cover two famous normal forms: Conjunctive normal form (CNF) and Horn clauses.

Conjunctive Normal Forms and Its Validity

Conjunctive Normal Form: a conjunction of clauses, where a clause is a disjunction of literals, i.e., **an AND of ORs**.

Any formula can be transformed into an equivalent formula in CNF.

- There exists a deterministic polynomial algorithm to convert a propositional formula into CNF
- Structural induction over the formula ϕ.

Example 1

Translate the formula ϕ into CNF ϕ':

1. $\phi = (\neg p \wedge q) \rightarrow (p \wedge (r \rightarrow q))$

Conjunctive Normal Form: a conjunction of clauses, where a clause is a disjunction of literals, i.e., **an AND of ORs**.

Any formula can be transformed into an equivalent formula in CNF.

- There exists a deterministic polynomial algorithm to convert a propositional formula into CNF
- Structural induction over the formula ϕ.

Example 1

Translate the formula ϕ into CNF ϕ':

1. $\phi = (\neg p \wedge q) \rightarrow (p \wedge (r \rightarrow q))$
2. $\phi' = (p \vee \neg q \vee p) \wedge (p \vee \neg q \vee \neg r \vee q)$

Translation to CNF

The translation algorithm consists of three parts:

- Transform ϕ into the implication-free form, *phi*$_1$.
- Transform the implication-free *phi*$_1$ into Negation Normal Form (NNF), *phi*$_2$.
- Transform the implicatio-free and NNF *phi*$_2$ into CNF ψ.

Implication-Free

Eliminate all implications in ϕ by replacing implication subformulas $\phi \rightarrow \psi$ with $\neg phi \vee psi$.

$\textsc{ImplFree}(\phi)$
Input: a propositional formula, ϕ
Output: an implication-free formula, ϕ'

```
switch φ
case φ is a literal
    return phi
case φ is φ1 → φ2
    return ¬ImplFree(φ1) ∨ ImplFree(φ2)
case φ is ¬φ1
    return ¬ImplFree(φ1)
case φ is φ1 op φ2, op ≠ →
    return ImplFree(φ1) op ImplFree(φ2)
```

Negation Normal Form

Eliminate all non-literal negations in ϕ using De Morgan's law.

$\mathrm{NNF}(\phi)$

Input: an implication-free formula, ϕ

Output: an implication-free, NNF formula, ϕ'

switch ϕ
case ϕ is a literal
    return ϕ
case ϕ is $\neg\neg\phi_1$
    return $\mathrm{NNF}(\phi_1)$
case ϕ is $\phi_1 \wedge \phi_2$
    return $\mathrm{NNF}(\phi_1) \wedge \mathrm{NNF}(\phi_2)$
case ϕ is $\phi_1 \vee \phi_2$
    return $\mathrm{NNF}(\phi_1) \vee \mathrm{NNF}(\phi_2)$
case ϕ is $\neg(\phi_1 \wedge \phi_2)$
    return $\mathrm{NNF}(\neg\phi_1 \vee \neg\phi_2)$
case ϕ is $\neg(\phi_1 \vee \phi_2)$
    return $\mathrm{NNF}(\neg\phi_1 \wedge \neg\phi_2)$

Conjunctive Normal Form

$\mathrm{CNF}(\phi)$
Input: an implication-free, NNF formula, ϕ
Output: a CNF formula, ϕ'

switch ϕ
case ϕ is a literal
    return ϕ
case ϕ is $\phi_1 \wedge \phi_2$
    return $\mathrm{CNF}(\phi_1) \wedge \mathrm{CNF}(\phi_2)$
case ϕ is $\phi_1 \vee \phi_2$
    return $\textsc{Distr}(\mathrm{CNF}(\phi_1)), \textsc{Distr}(\mathrm{CNF}(\phi_2))$

DISTR Function

Essentially, recursively distribute $(p \wedge q) \vee r$ to $(p \vee r) \wedge (q \vee r)$:

$\text{Distr}(\eta_1, \eta_2)$
Input: CNF formulas, η_1, η_2
Output: a CNF formula for $\eta_1 \vee \eta_2$

(1) **switch** η_1, η_2
(2) **case** η_1 is $\eta_{11} \wedge \eta_{12}$
(3) **return** $\text{Distr}(\eta_{11}, \eta_2) \wedge \text{Distr}(\eta_{12}, \eta_2)$
(4) **case** η_2 is $\eta_{21} \wedge \eta_{22}$
(5) **return** $\text{Distr}(\eta_1, \eta_{21}) \wedge \text{Distr}(\eta_2, \eta_{22})$
(6) **default**
(7) **return** $\eta_1 \vee \eta_2$ //no conjunction

CNF Example

Transform $\phi = (\neg p \wedge q) \rightarrow (p \wedge (r \rightarrow q))$ into CNF.

Step 1. Elimindate implications. Let $I(\phi)$ denote $\textsc{ImplFree}(\phi)$:

$$
\begin{aligned}
I(\phi) &= \neg I(\neg p \wedge q) \vee I(p \wedge (r \rightarrow q)) \\
&= \neg (I(\neg p) \wedge I(q)) \vee I(p \wedge (r \rightarrow q)) \\
&= \neg (\neg p \wedge I(q)) \vee I(p \wedge (r \rightarrow q)) \\
&= \neg (\neg p \wedge q) \vee I(p \wedge (r \rightarrow q)) \\
&= \neg (\neg p \wedge q) \vee (I(p) \wedge I(r \rightarrow q)) \\
&= \neg (\neg p \wedge q) \vee (p \wedge I(r \rightarrow q)) \\
&= \neg (\neg p \wedge q) \vee (p \wedge (\neg I(r) \vee I(q))) \\
&= \neg (\neg p \wedge q) \vee (p \wedge (\neg r \vee I(q))) \\
&= \neg (\neg p \wedge q) \vee (p \wedge (\neg r \vee q))
\end{aligned}
$$

CNF Example

Transform $\phi = (\neg p \wedge q) \rightarrow (p \wedge (r \rightarrow q))$ into CNF.

Step 2. NNF. Let $N(\phi)$ denote $\mathrm{NNF}(\phi)$:

$$
\begin{aligned}
N(I(\phi)) &= N(\neg(\neg p \wedge q) \vee (p \wedge (\neg r \vee q))) \\
&= N(\neg(\neg p \wedge q)) \vee N(p \wedge (\neg r \vee q)) \\
&= N((\neg\neg p) \vee \neg q) \vee N(p \wedge (\neg r \vee q)) \\
&= (N((\neg\neg p)) \vee N(\neg q)) \vee N(p \wedge (\neg r \vee q)) \\
&= (p \vee N(\neg q)) \vee N(p \wedge (\neg r \vee q)) \\
&= (p \vee \neg q) \vee N(p \wedge (\neg r \vee q)) \\
&= (p \vee \neg q) \vee (N(p) \wedge N(\neg r \vee q)) \\
&= (p \vee \neg q) \vee (p \wedge N(\neg r \vee q)) \\
&= (p \vee \neg q) \vee (p \wedge (N(\neg r) \vee N(q)) \\
&= (p \vee \neg q) \vee (p \wedge (\neg r \vee N(q)) \\
&= (p \vee \neg q) \vee (p \wedge (\neg r \vee q))
\end{aligned}
$$

CNF Example

Transform $\phi = (\neg p \wedge q) \rightarrow (p \wedge (r \rightarrow q))$ into CNF.

Step 3. CNF. Let $C(\phi)$ denote $\mathrm{CNF}(\phi)$, $D(\phi_1, \phi_2)$ denote $\textsc{Distr}(\phi_1, \phi_2)$:

$$\begin{aligned}
C(N(I(\phi))) &= C((p \vee \neg q) \vee (p \wedge (\neg r \vee q))) \\
&= D(C(p \vee \neg q), C(p \wedge (\neg r \vee q))) \\
&= D(p \vee \neg q, C(p \wedge (\neg r \vee q))) \\
&= D(p \vee \neg q, p \wedge (\neg r \vee q)) \\
&= D(p \vee \neg q, p) \wedge D(p \vee \neg q, \neg r \vee q) \\
&= (p \vee \neg q \vee p) \wedge D(p \vee \neg q, \neg r \vee q) \\
&= (p \vee \neg q \vee p) \wedge (p \vee \neg q \vee \neg r \vee q)
\end{aligned}$$

Exercise: transform the following formula into CNF.

$$\neg(p \rightarrow (\neg(q \wedge (\neg p \rightarrow q))))$$

Why do we care about this particular normal form? CNF makes it very easy to check the validity of the given formula. Consider the following CNF formula $(\neg q \vee p \vee r) \wedge (\neg p \vee r) \wedge q$. The semantic entailment holds if and only if:

$$\models \neg q \vee p \vee r, \models \neg p \vee r, \models q$$

Lemma 1 (1.43)

*A disjunction of literals $L_1 \vee L_2 \vee \ldots \vee L_m$ is **valid** if and only if there are $1 \leq i, j \leq m, i \neq j$ such that L_i is $\neg L_j$.*

Checking validity of a CNF formula boils down to searching for $L_i = \neg L_j$ in the constituent clauses: can be done in linear time.

Satisfiability of CNF formulas

Theorem 1 (Cook-Levin Theorem)

The satisfiability of CNF formulas is NP-hard.

Satisfiability of CNF formulas

Theorem 1 (Cook-Levin Theorem)

The satisfiability of CNF formulas is NP-hard.

Satisfiability of Disjunctive Normal Form?

Satisfiability of CNF formulas

Theorem 1 (Cook-Levin Theorem)

The satisfiability of CNF formulas is NP-hard.

Satisfiability of Disjunctive Normal Form?

Trivial.

Theorem 1 (Cook-Levin Theorem)

The satisfiability of CNF formulas is NP-hard.

Satisfiability of Disjunctive Normal Form?

Trivial. But...

Satisfiability of CNF formulas

Theorem 1 (Cook-Levin Theorem)

The satisfiability of CNF formulas is NP-hard.

Satisfiability of Disjunctive Normal Form?

Trivial. But...

It takes exponential time and space to convert an arbitrary propositional formula into DNF.

Satisfiability of CNF formulas

Theorem 1 (Cook-Levin Theorem)

The satisfiability of CNF formulas is NP-hard.

Satisfiability of Disjunctive Normal Form?

Trivial. But...

It takes exponential time and space to convert an arbitrary propositional formula into DNF.

No Free Lunch theorem.

Satisfiability of CNF formulas

Theorem 1 (Cook-Levin Theorem)

The satisfiability of CNF formulas is NP-hard.

Satisfiability of Disjunctive Normal Form?

Trivial. But...

It takes exponential time and space to convert an arbitrary propositional formula into DNF.

No Free Lunch theorem.

Fortunately, there are practically important subclasses of formulas which have much more efficient ways of deciding their satisfiability.

Horn Clauses

Intuitively, a *Horn clause* is a disjunction of literals with at most one positive (i.e. unnegated) literal. In other words, its disjunctive form is $\neg p \vee \neg q \vee \ldots \neg t \vee u$, which is $p \wedge q \wedge \ldots \wedge t \rightarrow u$.

Definition 1 (1.46)

A Horn formula is a formula ϕ of propositional logic if it can be generated as an instance of H in this grammar:

1. $P ::= \mathit{false} | \mathit{true} | p$
2. $A ::= P | P \wedge A$
3. $C ::= A \rightarrow P$
4. $H ::= C | C \wedge H$

That is, a *Horn formula* is a conjunction of *Horn clauses*.

Examples of Horn formulas

- $(p \land q \land s \to p) \land (q \land r \to p) \land (p \land s \to s)$
- $(p \land q \land s \to false) \land (q \land r \to p) \land (true \to s)$
- $(p_2 \land p_3 \land p_5 \to p_{13}) \land (true \to p_5) \land (p_5 \land p_{11} \to false)$

Examples of formulas which are not Horn formulas

- $(p \land q \land s \to \neg p) \land (q \land r \to p) \land (p \land s \to s)$
- $(p \land q \land s \to false) \land (\neg q \land r \to p) \land (true \to s)$
- $(p_2 \land p_3 \land p_5 \to p_{13} \land p_{27}) \land (true \land p_5) \land (p_5 \land p_{11} \to false)$
- $(p_2 \land p_3 \land p_5 \to p_{13} \land p_{27}) \land (true \land p_5) \land (p_5 \land p_{11} \lor false)$

Horn Clauses and Satisfiability

We maintain a list of all occurrences of type P (remember: $P ::= \mathit{false}|\mathit{true}|p$, $A ::= P|P \land A$, $C ::= A \rightarrow P$) in formula ϕ, and iteratively *mark* each one of them as following:

1. Mark *true* if it occurs in the list.
2. If there is a conjunct $P_1 \land P_2 \land \ldots \land P_{k_i} \rightarrow P$ of ϕ such that all P_j with $1 \leq j \leq k_i$ are marked, mark P as well and repeat 2; otherwise, proceed to 3.
3. If *false* is marked, ϕ is unsatisfiable.
4. Else, ϕ is satisfiable.

Horn Algorithm

HORN(ϕ)
Input: A Horn formula, ϕ
Output: The satisfiability of ϕ
(1) Mark all occurrences of *true* in ϕ
(2) **while** there exists a conjunct $P_1 \wedge P_2 \wedge \ldots \wedge P_j \rightarrow P'$ of ϕ s.t. all P_js are marked but P' isn't
(3) Mark P'
(4) **if** *false* is marked **then return** UNSAT
(5) **else return** SAT

Correctness of the Horn Algorithm

Theorem 2 (1.47)

The algorithm $\textsc{Horn}$*() is correct for the satisfiability decision problem of Horn formulas and has no more than* $n+1$ *cycles in its* `while` *statement if* n *is the number of atom is in* ϕ*. In particular,* $\textsc{Horn}$*() always terminates on correct input.*

Proof.

Termination: entering the body of the loop resulting in marking an yet-unmarked P that is not a *true* literal. Since there are only a finite number of atomic Ps in ϕ, $\textsc{Horn}$() terminates. □

Corollary 1

After any number of executions of the `while` loop, all marked P are true for all valuations in which ϕ evaluates to True.

Proof.

When loop executes 0 times: we already marked all occurrences of *true*, which must be *True* in all valuations. Hence Corollary 1 holds.

Corollary 1 holds for k iterations: if we enter $k+1$-th iteration, the loop predicate is true, i.e., there exists a conjunct $P_1 \wedge \ldots \wedge P_{k_i} \rightarrow P$ such that all P_js are marked. Let ν be any interpretaion in which ϕ is true. By the induction hypothesis, $P_1 \wedge \ldots \wedge P_{k_i}$ is true, as well as $P_1 \wedge \ldots \wedge P_{k_i} \rightarrow P$ is true. Therefore, P' must be also true in ν. Therefore, Corollary 1 holds for $k+1$-th iteration. □

Proof.

UNSAT: if *false* is marked, there exists a conjunct $P_1 \wedge \ldots \wedge P_{k_i} \rightarrow \mathit{false}$ of ϕ such that all P_is are marked. If ϕ is satisfiable, by Corollary 1, this means $(\mathit{true} \rightarrow \mathit{false}) = \mathit{false}$ whenever ϕ is true. This is impossible, so ϕ is unsatisfiable. Reductio ad absurdum.

SAT: if *false* is **NOT** marked, let ν be an interpretation that assign *true* to all marked atoms, and *false* to the others.

If ϕ is not true under ν, it means that there exists a conjunct $P_1 \wedge \ldots \wedge P_{k_i} \rightarrow P'$ of ϕ that is false. By the semantics, this can only mean that $P_1 \wedge \ldots \wedge P_{k_i}$ is true but P' is false. However, by the definition of ν, all P_is are marked, which means this conjunct has been processed by our `while` loop, resulting in P' being marked. By definition of ν, the conjunct becomes true; by Corollary 1, ϕ becomes true. Reductio ad absurdum. □