Propositional Calculus
- Propositional Normal Forms

Moonzoo Kim
CS Division of EECS Dept.
KAIST

moonzoo@cs.kaist.ac.kr
http://pswlab.kaist.ac.kr/courses/cs402-07

Overview

- Logic in Computer Science 2nd ed (by M.Huth and M.Ryan)
  - 1.5.2 Conjunctive normal forms and validity
  - 1.5.3 Horn clauses and satisfiability

Normal Forms

- Advantages of normal forms
  - A mechanical tool can handle a formula of a normal form 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 validity

- Any formula can be transformed into an equivalent formula in CNF.
- Formula is valid iff any of its equivalent formula is valid.
- There exists a deterministic algorithm to convert a propositional formula into CNF
- Structural induction over the formula ϕ.

Example of CNF

- Translate formula $\phi = \neg p \wedge q \rightarrow p \wedge (r \rightarrow q)$ into CNF
- We take 3 steps
  1. Transform ϕ into implication-free formula ϕ_1
  2. Transform implication-free ϕ_1 into NNF ϕ_2
  3. Transform implication-free and NNF ϕ_2 into CNF ψ

Deterministic Algorithm

- There exists a deterministic algorithm which always computes the same output CNF for a given input ϕ.
- This algorithm, called CNF, should satisfy all of the following requirements
  1. CNF terminates for all formulas of propositional logic as input.
  2. For each such input, CNF outputs an equivalent formula.
  3. All output computed by CNF is in CNF.

Preprocessing Procedures

- IMPL_FREE
  - lies in the de Morgan rules.
  - translates away all implications in ϕ by replacing all subformulas of the form $\phi \rightarrow \psi$ by $\neg \phi \vee \psi$
- NNF (Negation Normal Form)
  - formula that contains only negations of atoms.
    - ex. $p \wedge \neg q$, but not $\neg (p \wedge q)$

IMPL_FREE function

```
function IMPL_FREE(ϕ)
    /* precondition: ϕ propositional formula */
    /* postcondition : ϕ implication free */
begin function
    case
        ϕ is a literal : return ϕ
        ϕ is ϕ1 → ϕ2 : return (¬ϕ1 ∨ ϕ2)
        ϕ is ¬ϕ1 : return (¬IMPL_FREE(ϕ1))
        ϕ is ϕ1 op ϕ2 : return (IMPL_FREE(ϕ1) op IMPL_FREE(ϕ2))
    where op is a binary logical operator except →
    end case
end function
```

NNFfunction

function NNF(ϕ)
    /* precondition: ϕ implication free */
    /* postcondition : NNF(ϕ) computes a NNF for ϕ */
begin function
    case
        ϕ is a literal : **return** ϕ
        ϕ is $\neg\neg\,\phi_1$: **return** $\mathrm{NNF}(\phi_1)$
        ϕ is $\phi_1 \wedge \phi_2$: **return** $\mathrm{NNF}(\phi_1) \wedge \mathrm{NNF}(\phi_2)$
        ϕ is $\phi_1 \vee \phi_2$: **return** $\mathrm{NNF}(\phi_1) \vee \mathrm{NNF}(\phi_2)$
        ϕ is $\neg(\phi_1 \wedge \phi_2)$: **return** $\mathrm{NNF}(\neg\phi_1 \vee \neg\phi_2)$
        ϕ is $\neg(\phi_1 \vee \phi_2)$: **return** $\mathrm{NNF}(\neg\phi_1 \wedge \neg\phi_2)$
    end case
end function

CNF function

function $\mathrm{CNF}(\phi)$

/* precondition: ϕ implication free and in NNF */

/* postcondition: $\mathrm{CNF}(\phi)$ computes an equivalent CNF for ϕ */

begin function

case

ϕ is a literal : **return** ϕ

ϕ is $\phi_1 \wedge \phi_2$: **return** $\mathrm{CNF}(\phi_1) \wedge \mathrm{CNF}(\phi_2)$

ϕ is $\phi_1 \vee \phi_2$: **return** $\mathrm{DISTR}(\mathrm{CNF}(\phi_1), \mathrm{CNF}(\phi_2))$

end case

end function

DISTR function

function $\mathrm{DISTR}(\eta_1, \eta_2)$

/* precondition : η_1 and η_2 are in CNF */

/* postcondition : $\mathrm{DISTR}(\eta_1, \eta_2)$ computes a CNF for $\eta_1 \vee \eta_2$ */

begin function

case

η_1 is $\eta_{11} \wedge \eta_{12}$: **return** $\mathrm{DISTR}(\eta_{11}, \eta_2) \wedge \mathrm{DISTR}(\eta_{12}, \eta_2)$

η_2 is $\eta_{21} \wedge \eta_{22}$: **return** $\mathrm{DISTR}(\eta_1, \eta_{21}) \wedge \mathrm{DISTR}(\eta_1, \eta_{22})$

otherwise (=no conjunctions) : **return** $\eta_1 \vee \eta_2$

end case

end function

Example of CNF

- Transform $\phi = \neg p \wedge q \rightarrow p \wedge (r \rightarrow q$)))) into implication-free formula ϕ_1

$$
\begin{aligned}
\texttt{IMPL\_FREE}\,\phi &= \neg\texttt{IMPL\_FREE}\,(\neg p \wedge q) \vee \texttt{IMPL\_FREE}\,(p \wedge (r \rightarrow q)) \\
&= \neg((\texttt{IMPL\_FREE}\,\neg p) \wedge (\texttt{IMPL\_FREE}\,q)) \vee \texttt{IMPL\_FREE}\,(p \wedge (r \rightarrow q)) \\
&= \neg((\neg p) \wedge \texttt{IMPL\_FREE}\,q) \vee \texttt{IMPL\_FREE}\,(p \wedge (r \rightarrow q)) \\
&= \neg(\neg p \wedge q) \vee \texttt{IMPL\_FREE}\,(p \wedge (r \rightarrow q)) \\
&= \neg(\neg p \wedge q) \vee ((\texttt{IMPL\_FREE}\,p) \wedge \texttt{IMPL\_FREE}\,(r \rightarrow q)) \\
&= \neg(\neg p \wedge q) \vee (p \wedge \texttt{IMPL\_FREE}\,(r \rightarrow q)) \\
&= \neg(\neg p \wedge q) \vee (p \wedge (\neg(\texttt{IMPL\_FREE}\,r) \vee (\texttt{IMPL\_FREE}\,q))) \\
&= \neg(\neg p \wedge q) \vee (p \wedge (\neg r \vee (\texttt{IMPL\_FREE}\,q))) \\
&= \neg(\neg p \wedge q) \vee (p \wedge (\neg r \vee q)).
\end{aligned}
$$

Example of CNF

- 2. Transform implication-free ϕ_1 into NNF ϕ_2

$$
\begin{aligned}
\texttt{NNF}\ (\texttt{IMPL\_FREE}\ \phi) &= \texttt{NNF}\ (\neg(\neg p \wedge q)) \vee \texttt{NNF}\ (p \wedge (\neg r \vee q)) \\
&= \texttt{NNF}\ (\neg(\neg p) \vee \neg q) \vee \texttt{NNF}\ (p \wedge (\neg r \vee q)) \\
&= (\texttt{NNF}\ (\neg\neg p)) \vee (\texttt{NNF}\ (\neg q)) \vee \texttt{NNF}\ (p \wedge (\neg r \vee q)) \\
&= (p \vee (\texttt{NNF}\ (\neg q))) \vee \texttt{NNF}\ (p \wedge (\neg r \vee q)) \\
&= (p \vee \neg q) \vee \texttt{NNF}\ (p \wedge (\neg r \vee q)) \\
&= (p \vee \neg q) \vee ((\texttt{NNF}\ p) \wedge (\texttt{NNF}\ (\neg r \vee q))) \\
&= (p \vee \neg q) \vee (p \wedge (\texttt{NNF}\ (\neg r \vee q))) \\
&= (p \vee \neg q) \vee (p \wedge ((\texttt{NNF}\ (\neg r)) \vee (\texttt{NNF}\ q))) \\
&= (p \vee \neg q) \vee (p \wedge (\neg r \vee (\texttt{NNF}\ q))) \\
&= (p \vee \neg q) \vee (p \wedge (\neg r \vee q)).
\end{aligned}
$$

Example of CNF

- Transform implication-free and NNF ϕ_2 into CNF ψ

$$
\begin{aligned}
\mathtt{CNF}\,(\mathtt{NNF}\,(\mathtt{IMPL\_FREE}\;\phi)) &= \mathtt{CNF}\,((p \vee \neg q) \vee (p \wedge (\neg r \vee q))) \\
&= \mathtt{DISTR}\,(\mathtt{CNF}\,(p \vee \neg q), \mathtt{CNF}\,(p \wedge (\neg r \vee q))) \\
&= \mathtt{DISTR}\,(p \vee \neg q, \mathtt{CNF}\,(p \wedge (\neg r \vee q))) \\
&= \mathtt{DISTR}\,(p \vee \neg q, p \wedge (\neg r \vee q)) \\
&= \mathtt{DISTR}\,(p \vee \neg q, p) \wedge \mathtt{DISTR}\,(p \vee \neg q, \neg r \vee q) \\
&= (p \vee \neg q \vee p) \wedge \mathtt{DISTR}\,(p \vee \neg q, \neg r \vee q) \\
&= (p \vee \neg q \vee p) \wedge (p \vee \neg q \vee \neg r \vee q).
\end{aligned}
$$

Horn clauses

- Definition 1.46 A Horn formula is a formula \phi of propositional logic if it can be generated as an instance of H in this grammar:
  1. $P ::= \bot \mid \top \mid p$
  2. $A ::= P \mid P \wedge A$
  3. $C ::= A \rightarrow P$
  4. $H ::= C \mid C \wedge H$.
     - Each instance of C is a Horn clause.

Examples of Horn formulas

- Examples of Horn formulas
  - $(p \wedge q \wedge s \rightarrow p) \wedge (q \wedge r \rightarrow p) \wedge (p \wedge s \rightarrow s)$
  - $(p \wedge q \wedge s \rightarrow \bot) \wedge (q \wedge r \rightarrow p) \wedge (\top \rightarrow s)$
  - $(p_2 \wedge p_3 \wedge p_5 \rightarrow p_{13}) \wedge (\top \rightarrow p_5) \wedge (p_5 \wedge p_{11} \rightarrow \bot)$.
- Examples of formulas which are not Horn formulas
  - $(p \wedge q \wedge s \rightarrow \neg p) \wedge (q \wedge r \rightarrow p) \wedge (p \wedge s \rightarrow s)$
  - $(p \wedge q \wedge s \rightarrow \bot) \wedge (\neg q \wedge r \rightarrow p) \wedge (\top \rightarrow s)$
  - $(p_2 \wedge p_3 \wedge p_5 \rightarrow p_{13} \wedge p_{27}) \wedge (\top \wedge p_5) \wedge (p_5 \wedge p_{11} \rightarrow \bot)$
  - $(p_2 \wedge p_3 \wedge p_5 \rightarrow p_{13} \wedge p_{27}) \wedge (\top \wedge p_5) \wedge (p_5 \wedge p_{11} \vee \bot)$

Horn clauses and satisfiability

- The algorithm for deciding the satisfiability of a Horn formula ϕ maintains a list of all occurrences of type P in ϕ and proceeds like this:
  1. It marks $\top$ if it occurs in that list.
  2. If there is a conjunct $P_1 \wedge P_2 \wedge \ldots \wedge P_{ki} \rightarrow P'$ of ϕ such that all P_j with $1 \leq j \leq k_i$ are marked, mark P' as well and goto 2. Otherwise (= there is no conjunct $P_1 \wedge P_2 \wedge \ldots \wedge P_{ki} \rightarrow P'$ such that all P_j are marked) goto 3.
  3. If $\perp$ is marked, print out 'The Horn formula ϕ is unsatisfiable.' and stop. Otherwise, goto 4.
  4. Print out 'The Horn formula ϕ is satisfiable.' and stop.

HORN function

function $\mathrm{HORN}(\phi)$

/* precondition: ϕ is Horn formula */

/* postcondition : $\mathrm{HORN}(\phi)$ decides the satisfiability for ϕ */

begin function

mark all occurrences of $\top$ in ϕ

while there is a conjunct $P_1 \wedge P_2 \wedge \ldots \wedge P_{ki} \rightarrow P'$ of ϕ

such that all Pj are marked but P' isn't **do**

mark P'

end while

if $\perp$ is marked **then return** 'unsatisfiable' **else return** 'satisfiable'

end function

Correctness of the HORN algorithm

- The HORN algorithm is deterministic and correct
  - The algorithm terminates on all Horn formulas ϕ, and
  - Its output is always correct.
- Theorem 1.47 the algorithm 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, HORN always terminates on correct input.