Conjunctive Normal Form & Horn Clauses

York University CSE 3401
Vida Movahedi
Overview

• Definition of literals, clauses, and CNF

• Conversion to CNF- Propositional logic

• Representation of clauses in logic programming

• Horn clauses and Programs
  – Facts
  – Rules
  – Queries (goals)

• Conversion to CNF- Predicate logic

[ref.: Clocksin- Chap. 10 and Nilsson- Chap. 2]
Conjunctive Normal Form

• A literal is either an atomic formula (called a positive literal) or a negated atomic formula (called a negated literal)
  – e.g. p, ¬q

• A clause is
  – A literal, or
  – Disjunction of two or more literals, or
  – e.g. p, p ∨ ¬q ∨ r
  – A special clause: The empty clause, shown as □, :- or {}  

• A formula α is said to be in Conjunctive Normal Form (CNF) if it is the conjunction of some number of clauses
(p ∨ q) ∧ (q ∨ ¬s ∨ r) ∧ (¬r ∨ t)
CNF- Facts

• For every formula $\alpha$ of propositional logic, there exists a formula $A$ in CNF such that $\alpha \equiv A$ is a tautology.

• A polynomial algorithm exists for converting $\alpha$ to $A$.

• For practical purposes, we use CNFs in Logic Programming.
Conversion to CNF

1. Remove implication and equivalence
   - Use
     \[(p \rightarrow q) \Rightarrow (\neg p \lor q)\]
     \[(p \equiv q) \Rightarrow (p \lor q) \land (q \lor p)\]
     \Rightarrow (\neg p \lor q) \land (\neg q \lor p)\]

2. Move negations inwards
   - Use De Morgan’s
     \[\neg(p \land q) \Rightarrow (\neg p \lor \neg q)\]
     \[\neg(p \lor q) \Rightarrow (\neg p \land \neg q)\]

3. Distribute OR over AND
   \[p \lor (q \land r) \Rightarrow (p \lor q) \land (p \lor r)\]

York University - CSE 3401
02-CNF & Horn
Conversion to CNF - example

Example:
Convert the following formula to CNF

\[ p \equiv (r \land s) \]

\[ \Rightarrow (p \rightarrow (r \land s)) \land ((r \land s) \rightarrow p) \]
\[ \Rightarrow (\neg p \lor (r \land s)) \land (\neg (r \land s) \lor p) \]
\[ \Rightarrow (\neg p \lor r) \land (\neg p \lor s) \land (\neg r \lor \neg s \lor p) \]
Representing a clause

• Consider this clause: \( \neg p \lor q \lor \neg r \lor s \)

\[ \Rightarrow \neg (p \land r) \lor q \lor s \]
\[ \Rightarrow (p \land r) \rightarrow (q \lor s) \]

• In Logic programming, it is shown as:

\( (q \lor s) \leftarrow (p \land r) \)

\[ q ; s : \neg p , r . \]

• Easy way: positive literals on the left, negative literals on the right
• A clause in the form:

\[ p_1 ; p_2 ; \ldots ; p_m : \neg q_1 , q_2 , \ldots , q_n . \]

is equivalent to:

\[ p_1 \lor p_2 \lor \ldots \lor p_m \lor \neg q_1 \lor \neg q_2 \lor \ldots \lor \neg q_n \]

or

\[ q_1 \land q_2 \land \ldots \land q_n \Rightarrow p_1 \lor p_2 \lor \ldots \lor p_m \]

if \( q_1 \land q_2 \land \ldots \land q_n \) is true, then at least one of \( p_1 , p_2 , \ldots , p_m \) is true.
Logic Programming Notation - cont.

- A formula in CNF is written as conjunction (or a set of) clauses.

- Example:

\[
(\neg p \lor r) \land (\neg p \lor s) \land (\neg r \lor \neg s \lor p)
\]

\[
\Rightarrow \begin{cases} 
  r : \neg p. \\
  s : \neg p. \\
  p : \neg r, s.
\end{cases}
\]
Example - summary

• Example:
  Write the following formula in logic programming notation

\[ p \equiv (r \land s) \]

convert to \( \text{CNF} : (\neg p \lor r) \land (\neg p \lor s) \land (\neg r \lor \neg s \lor p) \)

convert to logic programming notation:
\[
\begin{cases}
  r : \neg p. \\
  s : \neg p. \\
  p : \neg r, s.
\end{cases}
\]
Another Example

Write the following expression as Logic Programming Clauses:

\[ ((p \land (s \rightarrow r)) \lor q) \land (r \rightarrow t) \]

1- Conversion to CNF:

\[ \Rightarrow ((p \land (\neg s \lor r)) \lor q) \land (\neg r \lor t) \]

2- Symmetry of \( \land \) allows for set notation of a CNF

\[ \Rightarrow (p \lor q) \land (\neg s \lor r \lor q) \land (\neg r \lor t) \]

\[ \{ (p \lor q), (\neg s \lor r \lor q), (\neg r \lor t) \} \]

3- Symmetry of \( \lor \) allows for set notation of clauses

\[ \{ \{ p, q \}, \{ q, \neg s, r \}, \{ \neg r, t \} \} \]

4- Logic Prog. notation

\[ p ; q : \neg. \quad q ; r : \neg s. \quad t : \neg r. \]
Horn Clause

• A Horn clause is a clause with at most one positive literal:
  
  – **Rules** “head:- body.”  
    e.g. \( p_1:-q_1, q_2, \ldots, q_n. \)
  
  – **Facts** “head :-.”  
    e.g. \( p_2:-. \)
  
  – **Queries** (or goals) “:-body.”  
    e.g. \( :- r_1, r_2, \ldots, r_m. \)

• Horn clauses simplify the implementation of logic programming languages and are therefore used in **Prolog**.
A Program

• A logic programming program P is defined as a finite set of rules and facts.
  
  – For example, P={p:-q,r,, q:-., r:-a,, a:-.}

  rule1     fact1   rule2   fact2

• Rules and facts (with exactly one positive literal) are called definite clauses and therefore a program defined by them is called a definite program.
Query

• A computational query (or goal) is the conjunction of some positive literals (called subgoals), e.g. \( r_1 \land r_2 \land ... \land r_n \)

• A query is deductible from P if it can be proven on the basis of P: \( P \models \neg r_1 \land r_2 \land ... \land r_n \)

• Note this query is written as \( \neg r_1, r_2, ..., r_n \).
  which is \( \neg r_1 \lor \neg r_2 \lor ... \lor \neg r_n \) or \( \neg (r_1 \land r_2 \land ... \land r_n) \)

• Why? “Proof by contradiction” is used to answer queries:
  \[ P \models \neg r_1 \land r_2 \land ... \land r_n \] \iff \[ P \cup \{\neg (r_1 \land r_2 \land ... \land r_n)\} \] is inconsistent
Example

• P: \{ p:-q. , q:-.\}

• If we want to know about p, we will ask the query:
  :-p.

• Note that the set \{ p:-q., q:-., :-p.\} is inconsistent. (Reminder: truth table for above clauses does not have even one row where all the clauses are true)

• Therefore p is provable and your theorem proving program (e.g. Prolog) will return true.
‘Predicate Logic’ Clauses

- Same definition for literals, clauses, and CNF except now each literal is more complicated since an atomic formula is more complicated in predicate logic.

- We need to deal with quantifiers and their object variables when converting to CNF.
Conversion to CNF in Predicate Logic

1. Remove implication and equivalence
2. Move negations inwards
3. Rename variables so that variables of each quantifier are unique
4. Move all quantifiers to the front (conversion to Prenex Normal Form or PNF)
5. Skolemize (get rid of existential quantifiers)
6. Distribute OR over AND
7. Remove all universal quantifiers
Example:

Convert the following formula to CNF:

\((\forall X)((\exists Y)m(X,Y) \rightarrow n(X))\)

Step 1. Remove implication and equivalence

\((\forall X)(\neg(\exists Y)m(X,Y) \vee n(X))\)

Step 2. Move negations inwards

Note \(\neg(\exists x)p(x) \equiv (\forall x)\neg p(x)\)

\((\forall X)((\forall Y)\neg m(X,Y) \vee n(X))\)

Step 3. Rename variables so that variables of each quantifier are unique

\((\forall X)(\forall Y)(\neg m(X,Y) \vee n(X))\)

Step 4. Move all quantifiers to the front (PNF)
Example- cont.

Step 5. Skolemizing (get rid of existential quantifiers)

$$(\forall X)(\forall Y)(\neg m(X, Y) \lor n(X))$$

Step 6. Distribute OR over AND to have conjunctions of disjunctions as the body of the formula

Step 7. Remove all universal quantifiers

$$\neg m(X, Y) \lor n(X)$$

Logic Programming notation:

$$n(X) : \neg m(X, Y).$$
**Skolems**

- Skolems are used to get rid of existential quantifiers:
  - **Skolem constants:**
    When NOT in scope of another quantifier
    \[(\exists X) \text{ female } (X) \land \text{ mother } (\text{eve}, X)\]
    \[\Rightarrow \text{ female } (g1) \land \text{ mother } (\text{eve}, g1)\]
  - **Skolem functions:**
    When in scope of another quantifier
    \[(\forall X)(\exists Y) \neg \text{ human } (X) \lor \text{ mother } (Y, X)\]
    \[\Rightarrow (\forall X) \neg \text{ human } (X) \lor \text{ mother } (g2(X), X)\]
(∀X)((∃Y)m(X,Y) → (∃Y)p(Y,X))
(∀X)(((∃Y)m(X,Y) ∨ (∃Y)p(Y,X))
(∀X)((∀Y)(¬m(X,Y) ∨ (∃Y)p(Y,X))
(∀X)((∀Y)(¬m(X,Y) ∨ (∃Z)p(Z,X))
(∀X)(∀Y)(∃Z)(¬m(X,Y) ∨ p(Z,X))
(∀X)(∀Y)(¬m(X,Y) ∨ p(g(X),X))
¬m(X,Y) ∨ p(g(X),X)

p(g(X),X) : ¬m(X,Y).
**Example**

- All Martians like to eat some kind of spiced food.
  
  [from Advanced Prolog Techniques and examples- Peter Ross]

\[
\Rightarrow (\forall X)(\text{martian}(X) \rightarrow (\exists Y)(\exists Z)(\text{food}(Y) \land \text{spice}(Z) \land \text{contains}(Y,Z) \land \text{likes}(X,Y)))
\]

\[
\Rightarrow (\forall X)(\neg \text{martian}(X) \lor (\exists Y)(\exists Z)(\text{food}(Y) \land \text{spice}(Z) \land \text{contains}(Y,Z) \land \text{likes}(X,Y)))
\]

\[
\Rightarrow (\forall X)(\exists Y)(\exists Z)(\neg \text{martian}(X) \lor (\text{food}(Y) \land \text{spice}(Z) \land \text{contains}(Y,Z) \land \text{likes}(X,Y)))
\]

\[
\Rightarrow (\forall X)(\neg \text{martian}(X) \lor (\text{food}(f(X)) \land \text{spice}(s(X)) \land \text{contains}(f(X),s(X)) \land \text{likes}(X,f(X))))
\]

\[
\Rightarrow (\forall X)((\neg \text{martian}(X) \lor \text{food}(f(X))) \land (\neg \text{martian}(X) \lor \text{spice}(s(X))) \land 
\] 

\[
(\neg \text{martian}(X) \lor \text{contains}(f(X),s(X))) \land (\neg \text{martian}(X) \lor \text{likes}(X,f(X))))
\]

\[
\Rightarrow (\neg \text{martian}(X) \lor \text{food}(f(X))) \land 
\] 

\[
(\neg \text{martian}(X) \lor \text{spice}(s(X))) \land 
\] 

\[
(\neg \text{martian}(X) \lor \text{contains}(f(X),s(X))) \land 
\] 

\[
(\neg \text{martian}(X) \lor \text{likes}(X,f(X)))
\]