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

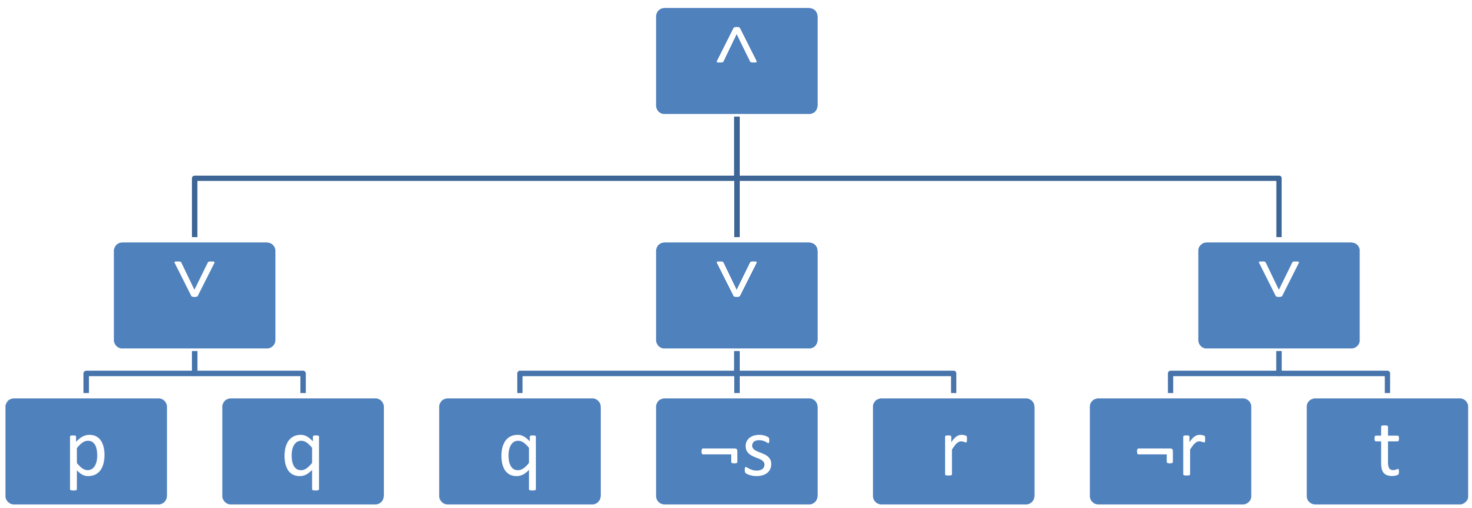
- A literal is either an atomic formula (called a positive literal) or a negated atomic formula (called a negated literal)
  - e.g. p, ¬q

- A clause is
  - A literal, or
  - Disjunction of two or more literals, or
  - The empty clause, shown as □, :- or {}
  - e.g. p, $p \vee \neg q \vee r$

- A formula α is said to be in Conjunctive Normal Form (CNF) if it is the conjunction of some number of clauses

CNF (example)

$$(p \vee q) \wedge (q \vee \neg s \vee r) \wedge (\neg r \vee t)$$



CNF- Facts

- For every formula α of propositional logic, there exists a formula A in CNF such that $\alpha \equiv A$ is a tautology
- A polynomial algorithm exists for converting α to A
- For practical purposes, we use CNFs in Logic Programming

Conversion to CNF

1. Remove implication and equivalence
   - Use $(p \to q) \Rightarrow (\neg p \vee q)$

     $(p \equiv q) \Rightarrow (p \to q) \wedge (q \to p)$

     $\Rightarrow (\neg p \vee q) \wedge (\neg q \vee p)$

2. Move negations inwards
   - Use De Morgan's

     $\neg(p \wedge q) \Rightarrow (\neg p \vee \neg q)$

     $\neg(p \vee q) \Rightarrow (\neg p \wedge \neg q)$

3. Distribute OR over AND

   $p \vee (q \wedge r) \Rightarrow (p \vee q) \wedge (p \vee r)$

Example:

$$p \equiv (r \wedge s)$$

$$\Rightarrow (\neg p \vee (r \wedge s)) \wedge$$
$$(\neg(r \wedge s) \vee p)$$

$$\Rightarrow (\neg p \vee r) \wedge$$
$$(\neg p \vee s) \wedge$$
$$(\neg r \vee \neg s \vee p)$$

Representing a clause

- Consider this clause: $\neg p \vee q \vee \neg r \vee s$
- It can be written as: $\neg(p \wedge r) \vee q \vee s \Rightarrow (p \wedge r) \rightarrow (q \vee s)$
- In Logic programming, it is shown as:

$$(q \vee s) \leftarrow (p \wedge r)$$

$$q; s :- p, r.$$

- Easy way: positive literals on the left, negative literals on the right

Logic Programming Clause

- A clause in the form:

$$p_1; p_2; \ldots; p_m :- q_1, q_2, \ldots, q_n.$$

is equivalent to:

$$p_1 \vee p_2 \vee \ldots \vee p_m \vee \neg q_1 \vee \neg q_2 \vee \ldots \vee \neg q_n$$

or $$q_1 \wedge q_2 \wedge \ldots \wedge q_n \rightarrow p_1 \vee p_2 \vee \ldots \vee p_m$$

if $q_1 \wedge q_2 \wedge \ldots \wedge q_n$ is true, then at least one of $p_1, p_2, \ldots, p_m$ is true.

Another Example

Write the following expression as Logic Programming Clauses:

$$\boxed{\left(\left(p \wedge (s \to r)\right) \vee q\right) \wedge (r \to t)}$$

1- Conversion to CNF:

$$\Rightarrow \left(\left(p \wedge (\neg s \vee r)\right) \vee q\right) \wedge (\neg r \vee t)$$

$$\Rightarrow (p \vee q) \wedge (\neg s \vee r \vee q) \wedge (\neg r \vee t)$$

2- Symmetry of ^ allows for sets notation of a CNF

$$\{(p \vee q), (\neg s \vee r \vee q), (\neg r \vee t)\}$$

3- Symmetry of ˅ allows for set notation of clauses

$$\{\ \{p, q\},\ \ \{q, \neg s, r\},\ \ \{\neg r, t\}\ \}$$

4- As Logic Prog.

$$p; q :-. \qquad q; r :- s. \qquad t :- r.$$

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, …, q_n.
  - Facts "head :-." e.g. p_2:-.
  - Queries (or goals) ":-body." e.g. :- r_1, r_2, …, 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 <u>definite</u> clauses and therefore a program defined by them is called a <u>definite</u> program.

Query

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

- A query is deductible from P if it can be proven on the basis of P: $P \mid - \; r_1 \wedge r_2 \wedge \ldots \wedge r_n$

- Note this query is written as $:- r_1, r_2, \ldots, r_n.$

  which is $\neg r_1 \vee \neg r_2 \vee \ldots \vee \neg r_n$ or $\neg(r_1 \wedge r_2 \wedge \ldots \wedge r_n)$

- Why? In logic programming theorem proving is used to answer queries:

  $P \mid - \; r_1 \wedge r_2 \wedge \ldots \wedge r_n$ iff $P \cup \{\neg(r_1 \wedge r_2 \wedge \ldots \wedge r_n)\}$ 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

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

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

4. Move all quantifiers to the front (Prenex Normal Form)

Conversion to CNF (cont.)

5. Skolemizing (get rid of existential quantifiers)
    5. <u>Skolem constants</u>

$$((\exists X)\, female(X) \wedge motherof(X, eve)) \Rightarrow female(g1) \wedge motherof(g1, eve)$$

    6. <u>Skolem functions</u>

$$(\forall X)(\exists Y)\neg human(X) \vee motherof(X, Y)$$

$$\Rightarrow (\forall X)\neg human(X) \vee motherof(X, g2(X))$$

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

7. Remove all universal quantifiers

Example

- All Martians like to eat some kind of spiced food.

  [from Advanced Prolog Techniques and examples- Peter Ross]

$$\Rightarrow (\forall X)(martian(X) \rightarrow (\exists Y)(\exists Z)(food(Y) \wedge spice(Z) \wedge contains(Y,Z) \wedge likes(X,Y)))$$

$$\Rightarrow (\forall X)(\neg martian(X) \vee (\exists Y)(\exists Z)(food(Y) \wedge spice(Z) \wedge contains(Y,Z) \wedge likes(X,Y)))$$

$$\Rightarrow (\forall X)(\exists Y)(\exists Z)(\neg martian(X) \vee (food(Y) \wedge spice(Z) \wedge contains(Y,Z) \wedge likes(X,Y)))$$

$$\Rightarrow (\forall X)(\neg martian(X) \vee (food(f(X)) \wedge spice(s(X)) \wedge contains(f(X),s(X)) \wedge likes(X,f(X))))$$

$$\Rightarrow (\forall X)((\neg martian(X) \vee food(f(X))) \wedge (\neg martian(X) \vee spice(s(X))) \wedge$$
$$(\neg martian(X) \vee contains(f(X),s(X))) \wedge (\neg martian(X) \vee likes(X,f(X))))$$

$$\Rightarrow (\neg martian(X) \vee food(f(X))) \wedge$$
$$(\neg martian(X) \vee spice(s(X))) \wedge$$
$$(\neg martian(X) \vee contains(f(X),s(X))) \wedge$$
$$(\neg martian(X) \vee likes(X,f(X)))$$