Lesson #2: Computing with Normal Forms

What's It All About?

This lesson is about logical computations with normal forms and the resolution principle. Consequently, it can be viewed as a brief overview of the Prolog inference engine.

The Normal Form Definitions

Conjuntive Normal Form

A formula F is said to be in **conjunctive normal form** if F has the form F1 ∧ F2 ∧ ... ∧ Fn where each of the Fi is a disjunction of literals.

Examples:

1. (P ∨ Q) ∧ (P ∨ ∼ Q) ∧ (∼ P ∨ ∼ Q)
2. P
3. P ∧ Q

Disjunctive Normal Form

A formula F is said to be in **disjunctive normal form** if F has the form F1 ∨ F2 ∨ ... ∨ Fn where each of the Fi is a conjunction of literals.

Examples:

1. (P ∧ Q) ∨ (P ∧ Q) ∨ (∼ P ∧ ∼ Q)
2. P
3. P ∨ Q

Multiple choice question on groups of literals

Reflecting upon the definitions of normal forms presented in the previous items, consider the following multiple choice question:

1. A literal, for example, P, can be considered a disjunction of literals.
2. A literal, for example, P, can be considered a conjunction of literals.
3. Both (1) and (2) are true.

4. Neither (1) nor (2) are true.

Normal Form Conversion Example

Any well formed formula can be converted to a normal form. This is accomplished by judiciously using logical equivalences.

Example: Converting $((P \vee \sim Q) \to R)$ to DNF

$((P \vee \sim Q) \to R)$
$\Longrightarrow \sim (P \vee \sim Q) \vee R$ (switcheroo)
$\Longrightarrow (\sim P \wedge \sim \sim Q) \vee R$ (DeMorgan)
$\Longrightarrow (\sim P \wedge Q) \vee R$ (double negation)

Pseudocode procedure for converting to normal form

The following is a rough procedure for transforming a formula to a normal form:

1. Use the following laws to eliminate the logical connectives $\to$ and $\leftrightarrow$:
   (a) $F \leftrightarrow G = (F \to G) \wedge (G \to F)$
   (b) $F \to G = \sim F \vee G$
2. Repeatedly use the double negation law and De Morgan's laws to bring the negation signs immediately before atoms:
   (a) $\sim (\sim F) = F$
   (b) $\sim (F \vee G) = (\sim F \wedge \sim G)$
   (c) $\sim (F \wedge G) = (\sim F \vee \sim G)$
3. Repeatedly use the distributive laws, and perhaps other laws, to objtain a normal form.
   (a) $F \vee (G \wedge H) = ((F \vee G) \wedge (F \vee H))$
   (b) $F \wedge (G \vee H) = ((F \wedge G) \vee (F \wedge H))$

Example: Conversion to CNF Using the Pseudocode Procedure

Transform to CNF: $(P \lor \sim Q) \to R$

$(P \lor \sim Q) \to R$
$\implies \sim (P \lor \sim Q) \lor R$ (switcheroo)
$\implies (\sim P \land \sim \sim Q) \lor R$ (DeMorgan's law)
$\implies (\sim P \land Q) \lor R$ (double negation)
$\implies R \lor (\sim P \land Q)$ (commutative law)
$\implies (R \lor \sim P) \land (R \lor Q)$ (distributive law)

Normal form conversion problems

1. Transform to DNF: $(P \land (Q \to R)) \to S$
2. Transform to DNF: $\sim (P \lor \sim Q) \land (S \to T)$
3. Transform to DNF: $(P \to Q) \to R$
4. Transform to CNF: $P \lor (\sim P \land Q \land R)$
5. Transform to CNF: $(\sim P \land Q) \lor (P \land \sim Q)$

Normal form conversion solutions

1. Transform to DNF: $(P \land (Q \to R)) \to S$

   $(P \land (Q \to R)) \to S$
   $\implies (P \land (\sim Q \lor R)) \to S$ (switcheroo)
   $\implies \sim (P \land (\sim Q \lor R)) \lor S$ (switcheroo)
   $\implies (\sim P \lor \sim (\sim Q \lor R)) \lor S$ (DeMorgan's law)
   $\implies (\sim P \lor (\sim \sim Q \land \sim R)) \lor S$ (DeMorgan's law)
   $\implies (\sim P \lor (Q \land \sim R)) \lor S$ (double negation)
   $\implies \sim P \lor (Q \land \sim R) \lor S$ (commutativity)

2. Transform to DNF: $\sim (P \lor \sim Q) \land (S \to T)$

   $\sim (P \lor \sim Q) \land (S \to T)$
   $\implies \sim (P \lor \sim Q) \land (\sim S \lor T)$ (switcheroo)
   $\implies (\sim P \land \sim \sim Q) \land (\sim S \lor T)$ (DeMorgan's law)
   $\implies (\sim P \land Q) \land (\sim S \lor T)$ (double negation)
   $\implies ((\sim P \land Q) \land \sim S) \lor ((\sim P \land Q) \land T)$ (distributive law)
   $\implies (\sim P \land Q \land \sim S) \lor ((\sim P \land Q) \land T)$ (associative law)
   $\implies (\sim P \land Q \land \sim S) \lor (\sim P \land Q \land T)$ (commutative law)

3. Transform to DNF: $(P \to Q) \to R$

   $(P \to Q) \to R$
   $\implies (\sim P \lor Q) \to R$ (switcheroo)
   $\implies \sim (\sim P \lor Q) \lor R$ (switcheroo)
   $\implies (\sim \sim P \land \sim Q) \lor R$ (DeMorgan's law)
   $\implies (P \land \sim Q) \lor R$ (double negation)

4. Transform to CNF: $P \vee (\sim P \wedge Q \wedge R)$

   $P \vee (\sim P \wedge Q \wedge R)$

   $\Longrightarrow P \vee (\sim P \wedge (Q \wedge R))$ (associative law)

   $\Longrightarrow (P \vee \sim P) \wedge (P \vee (Q \wedge R))$ (distributive law)

   $\Longrightarrow (P \vee \sim P) \wedge ((P \vee Q) \wedge (P \vee R))$ (distributive law)

   $\Longrightarrow (P \vee \sim P) \wedge (P \vee Q) \wedge (P \vee R)$ (associative law)

5. Transform to CNF: $(\sim P \wedge Q) \vee (P \wedge \sim Q)$

   $(\sim P \wedge Q) \vee (P \wedge \sim Q)$

   $\Longrightarrow ((\sim P \wedge Q) \vee P) \wedge ((\sim P \wedge Q) \vee \sim Q)$ (distributive law)

   $\Longrightarrow (P \vee (\sim P \wedge Q)) \wedge ((\sim P \wedge Q) \vee \sim Q)$ (communtative law)

   $\Longrightarrow ((P \vee \sim P) \wedge (P \vee Q)) \wedge ((\sim P \wedge Q) \vee \sim Q)$ (distributive law)

   $\Longrightarrow (P \vee \sim P) \wedge (P \vee Q) \wedge ((\sim P \wedge Q) \vee \sim Q)$ (associative law)

   $\Longrightarrow (P \vee \sim P) \wedge (P \vee Q) \wedge (\sim Q \vee (\sim P \wedge Q))$ (commutative law)

   $\Longrightarrow (P \vee \sim P) \wedge (P \vee Q) \wedge ((\sim Q \vee \sim P) \wedge (\sim Q \vee Q))$ (distributive law)

   $\Longrightarrow (P \vee \sim P) \wedge (P \vee Q) \wedge (\sim Q \vee \sim P) \wedge (\sim Q \vee Q)$ (associative law)

   $\Longrightarrow T \wedge (P \vee Q) \wedge (\sim Q \vee \sim P) \wedge (\sim Q \vee Q)$ (complementary disjunction)

   $\Longrightarrow (P \vee Q) \wedge (\sim Q \vee \sim P) \wedge (\sim Q \vee Q)$ (identity for and)

   $\Longrightarrow (P \vee Q) \wedge (\sim Q \vee \sim P) \wedge T$ (complementary disjunction)

   $\Longrightarrow (P \vee Q) \wedge (\sim Q \vee \sim P)$ (identity for and)

Definition of Logical Consequence

Formula G is a logical consequence of formulas F_1, F_2, ..., F_n if G is true for any interpretation in which F_1, F_2, ..., F_n are true.

Alternate Take 1

- **Theorem 1**: If the formula $((F_1 \wedge F_2 \wedge ... \wedge F_n) \rightarrow G)$ is valid, then G is a logical consequence of F_1, F_2, ..., F_n.

- **Proof**

  Suppose that $((F_1 \wedge F_2 \wedge ... \wedge F_n) \rightarrow G)$ is valid.

  1. Then G is true for any interpretation in which F_1, F_2, ..., F_n are true. (definitions of $\wedge$ and $\rightarrow$)
  2. So G is a logical consequence of formulas F_1, F_2, ..., F_n. (definition of logical consequence)

Alternate Take 2

- **Theorem 2**: If the formula $((F_1 \wedge F_2 \wedge ... \wedge F_n) \wedge \sim G)$ is inconsistent, then G is a logical consequence of F_1, F_2, ..., F_n.

- **Proof**

  Suppose that $((F_1 \wedge F_2 \wedge ... \wedge F_n) \wedge \sim G)$ is inconsistent.

  1. Then $\sim ((F_1 \wedge F_2 \wedge ... \wedge F_n) \wedge \sim G)$ is valid. (definitions of $\sim$ and validity)
  2. And $(\sim (F_1 \wedge F_2 \wedge ... \wedge F_n) \vee \sim \sim G)$ is valid. (DeMorgan's law)
  3. And $(\sim (F_1 \wedge F_2 \wedge ... \wedge F_n) \vee G)$ is valid. (double negagion)
  4. And $((F_1 \wedge F_2 \wedge ... \wedge F_n) \rightarrow G)$ is valid. (switcheroo)

5. So G is a logical consequence of formulas F_1, F_2, ..., F_n. (Theorem 1)

Logical Consequence Example

Show that ∼ P (G) is a logical consequence of (P → Q) and ∼ Q (F1 and F2) using:

1. the definition of logical consequence
2. the validity approach (alternate take 1)
3. the inconsistency approach (alternate take 2)

Way 1: using the definition of logical consequence ...

		F_1	F_2	G
P	Q	(P → Q)	∼ Q	∼ P
T	T	T	F	F
T	F	F	T	F
F	T	T	F	T
F	F	T	T	T

Way 2: the validity approach (alternate take 1)

		F_1	F_2	$F_1 \wedge F_2$	G	($F_1 \wedge F_2$) → G
P	Q	(P → Q)	∼ Q	(P → Q) ∧ ∼ Q	∼ P	((P → Q) ∧ ∼ Q) → ∼ P
T	T	T	F	F	F	T
T	F	F	T	F	F	T
F	T	T	F	F	T	T
F	F	T	T	T	T	T

Way 3: the inconsistency approach (alternate take 2)

		F_1	F_2	$F_1 \wedge F_2$	G	∼ G	($F_1 \wedge F_2$) ∧ ∼ G
P	Q	(P → Q)	∼ Q	(P → Q) ∧ ∼ Q	∼ P	∼ ∼ P	((P → Q) ∧ ∼ Q) ∧ ∼ ∼ P
T	T	T	F	F	F	T	F
T	F	F	T	F	F	T	F
F	T	T	F	F	T	F	F
F	F	T	T	T	T	F	F

The Resolution Principle (The Big Picture)

The truth table approach to verifying logical consequence turns out to be too cumbersome to serve as a basis for logic programming languages. A much more computationally tractable approach to verifying logical consequence is known as "resolution". Resolution inference is based on Robinsons Resolution Principle. (A Machine-Oriented Logic based on the Resolution Principle, JACM, 1965)

Robinson's Approach to Demonstrating Logical Consequence (Approach 3)

Given a set W = F1, F2, ..., Fn of WFFs and a goal WFF G, you can show that G is a logical consequence of W by the following method:

1. Convert the set W to a set S of "clauses".
2. Add the negation of G to the set S of clauses, calling the result S+.
3. Perform a "resolution deduction" of S+ to obtain the "empty clause".

This method, augmented with a means by which to "unify" variables, is the essential mechanism of computation in Prolog.

The Resolution Principle

Clauses

A **clause** is a disjunction of literals. For example, the following is a clause: `( P` $\vee$ `~ Q` $\vee$ `~ R` $\vee$ `S )`

Two literals are **complementary literals** if one is the negation of the other. For example, `P` and `~P` are complementary literals.

The Resolution Principle

For any two clauses C1 and C2, if there is a literal L1 in C1 that is complementary to a literal L2 in C2, then delete L1 and L2 from C1 and C2, repsectively, and construct the disjunction of the remaining clauses. The constructed clause is a **resovent** of C1 and C2.

Note

It turns out that the resolvent C of two clauses C1 and C2 is a logical consequence of C1 and C2.

Questions

1. What is the resolvent of: `( P` $\vee$ `R )` and `( ~ P` $\vee$ `~ Q )?`
2. What is the resolvent of: `( ~ P` $\vee$ `Q` $\vee$ `R )` and `( ~ Q` $\vee$ `S )?`

Resolution Deduction

Definition

Given a set S of clauses, a **resulution deduction** of clause C from S is a finite sequence C1, C2, ..., Ck of clauses such that each Ci is either a clause in S or a resolvent of clauses preceding Ci and Ck = C. We say that a clause C is derived from S if there is a **deduction** from S to C.

Notation

1. The symbol □ denotes a formula that is always false.
2. The symbol ■ denotes a formula that is always true.

Definition

A deduction of □ from S is called a **refulation** of S.

Observation / Important Note

To show that a clause G is a logical consequence of a set S of clauses, all you need to do is negate G and refute the set consisting of S augmented with the negation of G.

Example (logical consequence by refutation)

Show that G = P is a logical consequence of S = { (P ∨ Q), ∼ Q }.

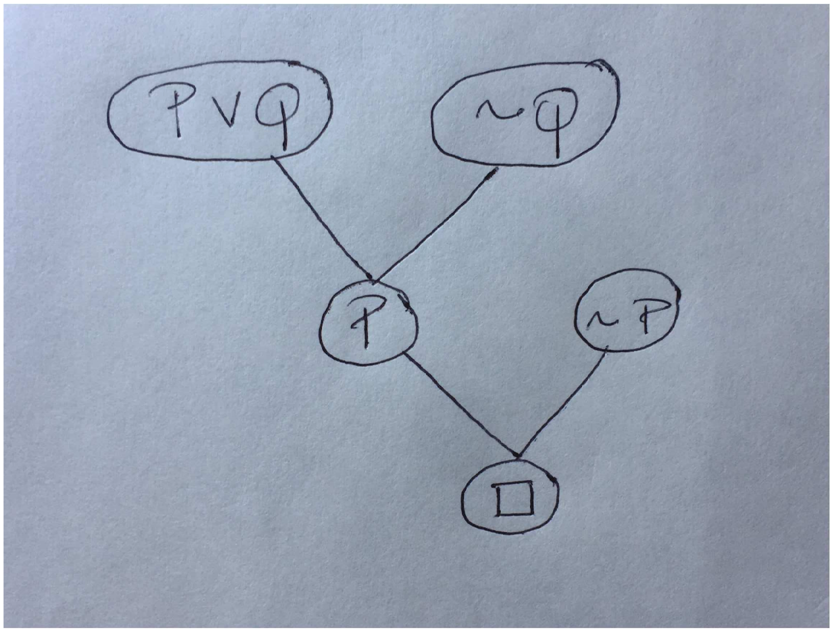
by:

1. Negating G
2. Adding the neggation of G to S, calling the result S+
3. Refuting S+

1. Determine the negation of the goal: ∼ P
2. Create S+ = { (P ∨ Q), ∼ Q, ∼ P }.
3. Do the refutation ...
   (a) (P ∨ Q) *element of S*
   (b) ∼ Q *element of S*
   (c) P *resolution principle*
   (d) ∼ P *negation of goal*
   (e) □ *resolution priniciple*

Refutation tree for the previous example

An alternative representation of the linear refutation with citations is the "refutation tree":



Example (logical consequence by refutation)

Show that G = R is a logical consequence of S = { (P $\vee$ Q), ($\sim$ Q $\vee$ R), $\sim$ P}.

1. Determine the negation of the goal: $\sim$ R
2. Create S+ = { (P $\vee$ Q), ($\sim$ Q $\vee$ R), $\sim$ P, $\sim$ R}.
3. Do the refutation ...
   (a) (P $\vee$ Q) *element of S+*
   (b) ($\sim$ Q $\vee$ R) *element of S+*
   (c) (P $\vee$ R) *resolution of (a) and (b)*
   (d) $\sim$ P *element of S+*
   (e) R *resolution (c) and (d)*
   (f) $\sim$ R *element of S+*
   (g) $\square$ *resolution of (e) and (f)*

Exercise (logical consequence by refutation)

Draw the refutation tree corresponding to the linear refutation just presented.