

Propositional Logic and First Order Logic (Review)

CS 575

August 30, 2007

1 Propositional Logic

1.1 Syntax

Formulas (or *sentences*) are constructed from *propositions* (or *statements*) and the logical connectives $\wedge, \vee, \neg, \rightarrow, \leftrightarrow$.

The BNF grammar of formulas in propositional logic is given by:

```
Formula      → Proposition | ComplexFormula
Proposition  → True | False | Symbol
Symbol       → p | q | r | ...
ComplexFormula → ¬ Formula |
                (Formula ∧ Formula) |
                (Formula ∨ Formula) |
                (Formula → Formula) |
                (Formula ↔ Formula)
```

(Sometime, the parentheses can be omitted if no confusion is possible.)

1.2 Semantics

Interpretation. An interpretation I maps each symbol p (proposition symbol) into a value $I(p) \in \{True, False\}$. (sometime, we write t and f instead of $True$ and $False$.)

p is true in an interpretation I if $I(p) = True$.

The truth value of a formula φ in an interpretation I is determined by the truth value of the proposition symbols occurring in φ and the truth table:

p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \rightarrow q$	$p \leftrightarrow q$
t	t	f	t	t	t	t
t	f	f	f	t	f	f
f	t	t	f	t	t	f
f	f	t	f	f	t	t

Satisfiable. A formula φ is *satisfiable* if there exists an interpretation I and φ is true in I . (I will be called as a *model* of φ).

Validity. A formula φ is *valid* if φ is true in every interpretation (of φ).

Knolwedge Base. A KB is a set of formulas.

Model. An interpretation I is a *model* of KB if every formula belonging to KB is true in I .

Entailment. A KB *entails* a formula φ , denoted by $KB \models \varphi$, if φ is true in every model of KB .

Equivalence. Two formulas ϕ and ψ are equivalent if and only if $\phi \leftrightarrow \psi$ is a valid formula.

1.3 Inference

Given a KB and a formula φ , to determine whether $KB \models \varphi$ holds or not, we can use one of the following methods.

- *Model theoretic approach:* We can find all the models of KB and check whether φ is true in them; if φ is false in some model of KB, then the answer is *no*; otherwise, the answer is *yes*.
- *Proof theoretic approach:* We can also use *inference rules* such as

1. The *modus ponens* rule:

$$\frac{\phi \rightarrow \psi, \phi}{\psi}$$

which says that whenever $\phi \rightarrow \psi$ and ψ are true, then so is ψ ;

2. The *and elimination* rule:

$$\frac{\phi \wedge \psi}{\phi};$$

3. The *resolution* rule:

$$\frac{p_1 \vee \dots \vee p_k, m}{p_1 \vee \dots \vee p_{i-1} \vee p_{i+1} \vee \dots \vee p_k}$$

where each p_j is literal (a proposition p or its negation $\neg p$) and m is the complementary literal of p_i (i.e., if $p_i = p$ (resp. $p_i = \neg p$) then $m = \neg p$ (resp. $m = p$)).

φ is a *consequence* of KB with respect to a set of inference rules S if we can use the inference rules in S to obtain φ from KB. We will write $KB \vdash_S \varphi$ to say that φ is derived from KB wrt. S .

Example 1 (Exercise 7.9, book of Russell and Norvig) Given the following, can you prove that the unicorn is mythical? How about magical? Horned?

If the unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.

Let

- M denotes that the unicorn is mythical.
- I denotes that the unicorn is immortal.
- X denotes that the unicorn is a mammal.
- H denotes that the unicorn is horned.
- A denotes that the unicorn is marginal.

The information given to us can be represented by the knowledge base consisting of the following formulas:

- $M \rightarrow I$ (if the unicorn is mythical, then it is immortal)
- $\neg M \rightarrow \neg I \wedge X$ (if it is not mythical, then it is a mortal mammal)
- $I \vee X \rightarrow H$ (If the unicorn is either immortal or a mammal, then it is horned)

- $H \rightarrow A$ (The unicorn is magical if it is horned)

Assume that we would like to prove that the unicorn is mythical, there are two ways:

- *Model theoretical approach:* we enumerate *all models* of the knowledge base and check for the truth value of M in them; if M is true in all of them, we conclude that M is a consequence of the theory. Otherwise, it is not. Please do this at home.
- *Proof theoretical approach:* we use the resolution rule. To do so, we first transform all the formulas with implication to disjunctions as follows.

$$\neg M \vee I \quad (1)$$

$$M \vee (\neg I \wedge X) \quad (2)$$

$$\neg(I \vee X) \vee H \quad (3)$$

$$\neg H \vee A \quad (4)$$

Distribution $\wedge$ over $\vee$ gives us

$$\neg M \vee I \quad (5)$$

$$M \vee \neg I \quad (6)$$

$$M \vee X \quad (7)$$

$$\neg I \vee H \quad (8)$$

$$\neg X \vee H \quad (9)$$

$$\neg H \vee A \quad (10)$$

Note that the second (6) and third formula (7) come from the second formula (2). Similarly, (8) and (9) are from (3).

We have the following sequence:

$$\frac{M \vee X, \neg M \vee I}{X \vee I} \Rightarrow \frac{X \vee I, \neg I \vee H}{X \vee H} \Rightarrow \frac{X \vee H, \neg X \vee H}{H}$$

This allows us to conclude H (i.e. the unicorn is horned). With this, we get A (the unicorn is magical). However, this is all we can say for sure. Strange that we cannot even conclude that the unicorn is a mammal? Please try to see why it is so at home.

1.4 Properties of inference rules

Needs to make sure that a set of inference rules S is

- *Sound:* the rule generates **only** correct answer (i.e., whenever it says that KB entails φ (or $KB \models_S \varphi$) then $KB \models \varphi$).
- *Complete:* it can generate **all** correct answers (i.e., if $KB \models \varphi$ then we can find a derivation using S such that $KB \vdash_S \varphi$).

For propositional logic, the resolution rule is complete. A *single* rule is complete inference rule!

```

function DPLL(clauses, symbols, model) return true or false
1. if every clause in clauses is true in model then return true
2. if some clauses in clauses is false in model then return false
3. compute (P, value) by the function FIND-PURE-SYMBOL(symbols, clauses, model)
4. if P is not null, then return DPLL(clauses, symbols \ {P}, Extend(P, value, model))
5. compute (P, value) by the function UNIT-CLAUSE(clauses, model)
6. if P is not null, then return DPLL(clauses, symbols \ {P}, Extend(P, value, model))
7. take a P from symbols and let rest = symbols \ {P}
8. return DPLL(clauses, rest, Extend(P, true, model))
9.           or DPLL(clauses, rest, Extend(P, false, model))

```

Figure 1: DPLL algorithm

1.5 Satisfiability Problem

Given a set of formulas *clauses*, determine whether it is satisfiable or not. Davis, Putnam, Logemann, and Loveland algorithm (Figure 1):

Notes:

- *clauses* is the set of clauses that need to be satisfied.
- *symbols* is the set of symbols that have not been assigned a value (**true** or **false**).
- *model* is the current model that contains the truth values of already assigned symbols.
- the first call to the algorithm is done with *symbols* as the set of *all* symbols and *model* is an empty set.
- a symbol is *pure* if it appears with the same sign in *all* clauses.
- a clause is a *unit clause* if the value of all but one of its symbol is known.
- *Extend(P, value, model)* creates a new model in which *P* is set to have the value *value*.

It can be shown that *DPLL* algorithm is sound and complete. Using this, we can check for entailment:

- Given a *KB* and a formula φ
- Determine whether $KB \models \varphi$

We can try to see whether $KB \cup \neg\varphi$ is satisfiable or not. If it is not, then we can conclude that $KB \models \varphi$.

2 First Order Logic

2.1 Preliminary

A *first-order theory* consists of an alphabet, a first order language, a set of axioms and a set of inference rules.

Definition 1 An alphabet consists of the following sets:

1. *variables*
2. *constants*

3. function symbols
4. predicate symbols
5. connectives: $\{\wedge, \vee, \neg, \leftrightarrow, \rightarrow\}$
6. quantifiers: $\forall, \exists$
7. punctuation symbols: $'(' , ')' , ',' , ':' , '.'$

NOTE: • The last three sets are the same for every alphabet.

- For an alphabet, only the set of constants or the set of function symbols may be empty.
- Notation convention: Variables: u, v, w, x, y , and z (possibly with indexes); constants: a, b , and c (possibly with indexes); function symbols of arities > 0 : f, g , and h (possibly with indexes); and predicate symbols of arities ≥ 0 : p, q , and r (possibly with indexes).

The precedence among the connectives: $\neg, \forall, \exists, \wedge, \vee, \rightarrow, \leftrightarrow$

Given an alphabet, a *first order language* is defined by the set of *well-formed formula* (*wff* or *sentences*) of the theory (defined below).

Definition 2 A term is either

1. a variable,
2. a constant, or
3. an expression of the form $f(t_1, \dots, t_n)$ where f is an n -ary function symbol and $t_1, \dots, t_n$ are terms.

Definition 3 A (well-formed) formula is defined inductively as follows.

1. $p(t_1, \dots, t_n)$ where p is an n -ary predicate symbol and $t_1, \dots, t_n$ are terms,
2. if P and Q are formulas then $(\neg P)$, $(P \wedge Q)$, $(P \vee Q)$, $(P \rightarrow Q)$, $(P \leftrightarrow Q)$ are formulas
3. if P is a formula and x is a variable then $(\forall x P)$ and $(\exists x P)$ are formulas.

Definition 4 A first order language given by an alphabet consists of the set of all formulas constructed from the symbols of the alphabet.

Example 2 $(\forall x(\exists y(p(x, y) \rightarrow q(x))))$ and $(\neg \exists x((p(x, a) \wedge q(f(x)))))$ are formulas. We can simplify them to $\forall x \exists y(p(x, y) \rightarrow q(x))$ and $\neg \exists x(p(x, a) \wedge q(f(x)))$.

Definition 5 The scope of $\forall x$ (resp. $\exists x$) in $\forall x F$ (resp. $\exists x F$) is F . A bound occurrence of a variable in a formula is an occurrence immediately following a quantifier or an occurrence within the scope of the quantifier, which has the same variable immediately after the quantifier. Any other occurrence of a variable is free.

Example 3 $\exists x p(x, y) \rightarrow q(x)$ – the first two occurrences of x (in $\exists x$ and $p(x, y)$) are bound but the third one (in $q(x)$) is free.

$\exists x(p(x, y) \rightarrow q(x))$ – all occurrences of x are bound (because of the parentheses!).

Definition 6 A closed formula is a formula with no free occurrences of any variable.

Example 4 $\exists x p(x, y) \rightarrow q(x)$ is not a closed formula.

$\exists x(p(x, y) \rightarrow q(x))$ is a closed formula.

Definition 7 A grounded term is a term not containing a variable. A grounded atom is an atom not containing a variable.

2.2 Interpretation

NOTE: When we say ‘a first order language L ’ we understand that the alphabet of L is given.

Definition 8 Let L be a first order language. An interpretation I of L consists of

1. a non-empty set D , called the domain of I ,
2. for each constant in L , the assignment of an element of D , (i.e., a constant c is mapped into an element $I(c) \in D$),
3. for each n -ary function symbol in L , the assignment of a mapping from D^n to D , (i.e., a function symbol f is mapped into a function f^I)
4. for each n -ary predicate symbol in L , the assignment of a mapping from D^n to $\{true, false\}$, (i.e., a predicate symbol p is mapped into a relation p^I).

Let I be an interpretation. A *variable assignment* (wrt. I) is an assignment to each variable in L an element in D .

Let I be an interpretation and V be a variable assignment (wrt. I). The term assignment (wrt. I and V) of the terms in L is defined as follows.

1. Each variable is given its assignment according to V ,
2. Each constant is given its assignment according to I ,
3. If $t'_1, \dots, t'_n$ are the term assignments of $t_1, \dots, t_n$ and f' is the assignment of the n -ary function symbol f , then $f'(t'_1, \dots, t'_n)$ is the term assignment of $f(t_1, \dots, t_n)$.

Let I be an interpretation and V be a variable assignment (wrt. I). Then, a formula L can be given a *truth value*, true or false, (wrt. I and V) as follows:

1. If $L = p(t_1, \dots, t_n)$ and $t'_1, \dots, t'_n$ are the term assignments of $t_1, \dots, t_n$ (wrt. I and V), and p' be the mapping assigned to the n -ary predicate symbol p by I , then the truth value of L is obtained by calculating the truth value of $p'(t'_1, \dots, t'_n)$,
2. If the formula has the form $(\neg P)$, $(P \wedge Q)$, $(P \vee Q)$, $(P \rightarrow Q)$, $(P \leftrightarrow Q)$ then the truth value of the formula is given by the following table

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \rightarrow Q$	$P \leftrightarrow Q$
t	t	f	t	t	t	t
t	f	f	f	t	f	f
f	t	t	f	t	t	f
f	f	t	f	f	t	t

3. If the formula has the form $\exists x F$, then the truth value of the formula is true if there exists $d \in D$ such that F has the truth value wrt. I and the variable assignment V in which x is assigned to d ; Otherwise, its truth value is false.
4. If the formula has the form $\forall x F$, then the truth value of the formula is true if for all $d \in D$, F has the truth value wrt. I and the variable assignment V in which x is assigned to d ; Otherwise, its truth value is false.

From the above, the truth value of a closed formula does not depend on the variable assignment. Thus, we can speak about the truth value of a closed formula wrt. to an interpretation (without mentioning the variable assignment).

Definition 9 A first order theory T is a set of formulas of a first order language L .

Definition 10 Let I be an interpretation of a first order language L and let F be a closed formula of L . Then, I is a model of F if F is true wrt. I .

Let S be a closed formulas of a first order language L and I be an interpretation of L . I is a model of S if every formula $F \in S$ is true wrt. I .

Definition 11 Let S be a closed formulas of a first order language L . We say

1. S is satisfiable if L has an interpretation which is a model of S ,
2. S is valid if every interpretation of L is a model of S ,
3. S is unsatisfiable if no interpretation of L is a model of S ,
4. S is nonvalid if L has an interpretation which is not a model of S .

Definition 12 Let S be a closed formulas of a first order language L and F a formula in L . S entails F , denoted by $S \models F$, if F is true wrt. to every model of S .

2.3 Unification

Definition 13 A literal is either an atom P or its negation $\neg P$.

Given a first order language L , an *expression* is either a constant, a variable, a term, a literal, a conjunction of literals, or a disjunction of literals.

Definition 14 A substitution η is a finite set of the form $\{v_1/t_1, \dots, v_n/t_n\}$ where each v_i is a variable, each t_i is a term distinct from v_i and the variables $v_1, \dots, v_n$ are distinct.

Each v_i/t_i is called a binding for v_i ;

η is called a grounded substitution if the t_i are all ground terms.

Definition 15 Let $\eta = \{v_1/t_1, \dots, v_n/t_n\}$ be a substitution and E be an expression (term, literal, conjunction of literals, or disjunction of literals). Then, $E\eta$, the instance of E by η , is the expression obtained from E by simultaneously replacing each occurrence of the variables v_i in E by the term t_i .

If $E\eta$ is ground, then it is called a ground instance of E .

Example 5 Let $E = p(x, y, f(a))$ and $\eta = \{x/b, y/x\}$ then $E\eta = p(b, x, f(a))$.

Definition 16 Let $\sigma = \{v_1/t_1, \dots, v_n/t_n\}$ and $\eta = \{u_1/s_1, \dots, u_m/s_m\}$ be two substitutions. Then the composition $\eta\sigma$ of η and σ is the substitution obtained from the set $\{u_1/s_1\sigma, \dots, u_m/s_m\sigma, v_1/t_1, \dots, v_n/t_n\}$ by deleting any binding v_j/t_j for which $v_j \in \{u_1, \dots, u_m\}$.

Example 6 Let $\sigma = \{x/a, y/b, z/y\}$ and $\eta = \{x/f(y), y/z\}$ then $\eta\sigma = \{x/f(b), z/y\}$ since $X = \{x/f(y)\sigma, y/z\sigma\} = \{x/f(b), y/y, x/a, y/b, z/y\}$ and $\eta\sigma$ is obtained from X by deleting $y/y, x/a, y/b$ because x and y are variables occurring in η .

Two expressions E and F are called *variants* if there exists a substitution η such that $E = F\eta$ and $F = E\eta$. (We also say that E is a variant of F and vice versa!)

A set of expressions $S = \{E_1, \dots, E_n\}$ is unifiable if there exists a substitution η such that $E_1\eta = E_2\eta = \dots = E_n\eta$. In that case, η is a *unifier* of S .

A unifier η of S is called a *most general unifier* (or *mgu*) of S if for each unifier σ of S there exists a substitution γ such that $\sigma = \eta\gamma$.

Example 7 The set $S = \{p(f(x), a), p(y, f(w))\}$ is not unifiable because we can not unify the constant a with $f(w)$.

The set $S = \{p(f(x, y), g(z), a), p(f(y, x), g(u), a)\}$ is unifiable since for $\eta = \{x/y, z/u\}$, $S\eta = \{p(f(y, y), g(u), a)\}$. Here, η is a mgu of S .

Definition 17 Let S be a set of simple expressions (a simple expression is a term or an atom). The disagreement set of S is defined as follows. Locate the leftmost symbol position at which not all expressions in S have the same symbol and extract from each expression expression in S the subexpression beginning at that symbol position. The set of all such subexpressions is the disagreement set.

Example 8 Let $S = \{p(f(x), h(y), a), p(f(x), z, a), p(f(x), h(y), b)\}$. Then the disagreement set is $\{h(y), z\}$.

Example 9 Let $S = \{p(f(x), h(y), a), p(f(x), z, a), p(f(x), w, b)\}$. Then the disagreement set is $\{z, w\}$.

2.4 Unification Algorithm

Let $S = \{P_1, \dots, P_m\}$ be a set of simple expressions.

S1 Put $k = 0$ and $\sigma_0 = \{\}$.

S2 If $S\sigma_k$ is a singleton ($P_i\sigma_k = P_j\sigma_k$ for every $i \neq j$), then stop; σ_k is an mgu (most general unifier) of S ; Otherwise, find the disagreement set D_k of $S\sigma_k$.

S3 If there exist v and t in D_k such that v is a variable that does not occur in t , then put $\sigma_{k+1} = \sigma_k\{v/t\}$, increment k and go to S2. Otherwise, stop; S is not unifiable.

Example 10 Let $S = \{p(f(a), g(x)), p(y, y)\}$.

S1 Put $k = 0$ and $\sigma_0 = \{\}$.

S2 $S\sigma_0 = S$ is not a singleton. So, we need to find the disagreement set D_0 of $S\sigma_0 = S$. We have: $D_0 = \{f(a), y\}$.

S3 Here, y is a variable which does not occur in $f(a)$. So, we let $\sigma_1 = \sigma_0\{y/f(a)\} = \{y/f(a)\}$ and go to S2.

S2 $S\sigma_1 = \{p(f(a), g(x)), p(f(a), f(a))\}$ is not a singleton. So, we need to find the disagreement set D_1 of $S\sigma_1 = S$. We have: $D_1 = \{g(x), f(a)\}$.

S3 Here, there is no variable in D_1 . So, we stop; S is not unifiable.

Example 11 Let $S = \{p(a, x, h(g(z))), p(z, h(y), h(y))\}$.

S1 Put $k = 0$ and $\sigma_0 = \{\}$.

S2 $S\sigma_0 = S$ is not a singleton. So, we need to find the disagreement set D_0 of $S\sigma_0 = S$. We have: $D_0 = \{a, z\}$.

S3 Here, z is a variable which does not occur in a . So, we let $\sigma_1 = \sigma_0\{z/a\} = \{z/a\}$ and go to S2.

S2 $S\sigma_1 = \{p(a, x, h(g(a))), p(a, h(y), h(y))\}$ is not a singleton. So, we need to compute the disagreement set D_1 of $S\sigma_1$. We have: $D_1 = \{x, h(y)\}$.

S3 Here, x is a variable which does not occur in $h(y)$. So, we let $\sigma_2 = \sigma_1\{x/h(y)\} = \{z/a, x/h(y)\}$ and go to S2.

S2 $S\sigma_2 = \{p(a, h(y), h(g(a))), p(a, h(y), h(y))\}$ is not a singleton. So, we need to find the disagreement set D_2 of $S\sigma_2$. We have: $D_1 = \{y, g(a)\}$.

S3 Here, y is a variable which does not occur in $g(a)$. So, we let $\sigma_3 = \sigma_2\{y/g(a)\} = \{z/a, x/h(g(a)), y/g(a)\}$ and go to S2.

S2 $S\sigma_3 = \{p(a, h(g(a)), h(g(a)))\}$ is a singleton. So we stop and one mgu of S is $\sigma_3 = \{z/a, x/h(g(a)), y/g(a)\}$.

Theorem 1 Let S be a finite of simple expressions. If S is unifiable then the algorithm terminates and gives an mgu for S . If S is not unifiable then the algorithm terminates and reports this fact.

2.5 Resolution

Definition 18 A literal is either an atom P or its negation $\neg P$.

A clause is a disjunction of literals. (sometime it is written as $P_1 \vee \dots \vee P_n$ or $\{P_1, \dots, P_n\}$)

A formula Q is said to be in conjunctive normal form (or CNF) if Q is a conjunction of clauses.

A formula Q is said to be in implicative normal form if Q is a conjunction of implication of the form $P_1 \wedge \dots \wedge P_n \rightarrow Q_1 \vee \dots \vee Q_m$ where each P_i, Q_j is an atom.

$(P \vee Q \vee \neg S) \wedge (\neg P \vee Q \vee S) \wedge (\neg P \vee \neg R \vee \neg S) \wedge (P \vee T \vee \neg S)$ is a CNF.

Theorem 2 For every formula ϕ there exists a formula ψ in CNF form such that ϕ and ψ is equivalent, i.e., $\forall(\phi \leftrightarrow \psi)$ is a valid formula.

Algorithm to convert a formula into CNF form.

1. **Eliminate implications:** Replace $p \rightarrow q$ with $\neg p \vee q$
2. **Move $\neg$ inward:** do the following
 - (a) $\neg(p \vee q)$ is replaced by $\neg p \wedge \neg q$
 - (b) $\neg(p \wedge q)$ is replaced by $\neg p \vee \neg q$
 - (c) $\neg \forall x p$ is replaced by $\exists x \neg p$
 - (d) $\neg \exists x p$ is replaced by $\forall x \neg p$
 - (e) $\neg \neg p$ is replaced by p
3. **Standardize variable:** For sentences like $(\forall x P(x)) \vee (\exists x Q(x))$ that use the same variable name twice, change the name of one of the variable.
4. **Skolemize:** Remove the existential quantifier by elimination – this includes: (1) defines a Skolem function, one for a variable occurred immediately after an existential quantification, (2) introduces a new constant, one for a variable occurred immediately after an existential quantification, (3) removes the existential quantification and substitutes x for $F^x(A^x)$ in the formula.
5. **Distribute $\wedge$ over $\vee$:** $(a \wedge b) \vee c$ becomes $(a \vee c) \wedge (b \vee c)$.
6. **Flatten nested conjunction and disjunction:** $(a \wedge b) \wedge c$ becomes $(a \wedge b \wedge c)$ and $(a \vee b) \vee c$ becomes $(a \vee b \vee c)$.

Example 12 Convert $((\neg\forall xA(x)) \vee (\forall yB(y))) \rightarrow (\neg(\forall zQ(z, f(z))))$ to CNF.

1. $\neg((\neg\forall xA(x)) \vee (\forall yB(y))) \vee (\neg(\forall zQ(z, f(z))))$ (Eliminate implication)
2. $((\neg\neg\forall xA(x)) \wedge (\neg\forall yB(y))) \vee (\neg(\forall zQ(z, f(z))))$ (Move $\neg$...)
3. $(\forall xA(x) \wedge \exists y\neg B(y)) \vee ((\exists z\neg Q(z, f(z))))$ (Move $\neg$...)
4. $\forall x((A(x) \wedge \neg B(Fy(Cy))) \vee \neg Q(Cz, f(Fz(Cz))))$ (Skolemize – Fy, Fz are two new functions and Cy, Cz are two new constants, correspond to the variable y and z respectively)
5. $(A(x) \wedge \neg B(Fy(Cy))) \vee \neg Q(Cz, f(Fz(Cz)))$ (Drop universal quantifier)
6. $(A(x) \vee \neg Q(Cz, f(Fz(Cz)))) \wedge (\neg B(Fy(Cy)) \vee \neg Q(Cz, f(Fz(Cz))))$ (Distribute $\wedge$ over $\vee$)

NOTE: 1. In the above, Fz might not be needed.

2. Implicative normal form is often used to. A formula of the form $\neg P_1 \vee \neg P_2 \vee \dots \vee \neg P_n \vee Q_1 \vee Q_2 \dots \vee Q_m$ is equivalent to $P_1 \wedge P_2 \dots \wedge P_n \rightarrow Q_1 \vee Q_2 \dots \vee Q_m$

It is easy to see that if Q is in CNF then we can convert it into implicative normal form using the above conversion.

The *resolution inference rule* If β_1 and β_2 are unifiable and η is a mgu of β_1 and β_2 , then

$$\frac{\alpha \vee \beta_1, \neg\beta_2 \vee \gamma}{\alpha\eta \vee \gamma\eta} \quad (11)$$

or

$$\frac{\neg\alpha \rightarrow \beta_1, \beta_2 \rightarrow \gamma}{\neg\alpha\eta \rightarrow \gamma\eta} \quad (12)$$

Given a set of formulas S and a formula Q , we would like to determine if $S \models Q$.

We can use (11) (or (12)) to determine whether $S \vdash Q$ holds or not.

We make the following assumptions:

1. Each formula in S is a clause (**why?**)
2. Q is a literal (**why?**)

Example 13 Let Δ be the set consisting of the following clauses:

1. $\neg P(w) \vee Q(w)$,
2. $P(x) \vee R(x)$,
3. $\neg Q(y) \vee S(y)$, and
4. $\neg R(z) \vee S(z)$.

Question: $\Delta \vdash S(A)$?

Proof.

1. $\frac{\neg P(w) \vee Q(w), P(x) \vee R(x)}{\neg P(w) \vee S(w)}$ where $\eta = \{y/w\}$
2. $\frac{\neg P(w) \vee S(w), P(x) \vee R(x)}{S(x) \vee R(x)}$ with $\{w/x\}$
3. $\frac{S(x) \vee R(x), \neg R(z) \vee S(z)}{S(A)}$ with $\{x/A, z/A\}$! **DONE!**

Refutation proof procedure. Given a set of clauses S and a literal Q . The refutation proof procedure uses resolution to determine whether $S \models Q$ holds or not.

1. **Idea:** If $S \models Q$ then $S \cup \{\neg Q\}$ is unsatisfiable, i.e., there is no model for $S \cup \{\neg Q\}$. So, we will assume that $\neg Q$ holds and try to derive a contradiction out of $S \cup \{\neg Q\}$.
2. **Algorithm:** We try to derive a proof that derives a contradiction from $S \cup \{\neg Q\}$. The algorithm can be described as follows.

A1 Let $k = 0$, $G_k = \neg Q$.

A2 If $G_k = \text{false}$ then step and answer 'yes'; Otherwise, find a clause C in S that contains a literal L which is contradictory with some L' of G_k and η is a mgu of L and L' . Go to step [A3]!

A3 Let $G_{k+1} = ((C \setminus \{L\}) \cup (G_k \setminus \{L'\}))\eta$, $k = k + 1$, and go to step [A2]!

Example 14

$$\text{Dog}(D) \quad (13)$$

$$\text{Owns}(\text{Jack}, D) \quad (14)$$

$$\text{Dog}(y) \wedge \text{Owns}(x, y) \rightarrow \text{AnimalLover}(x) \quad (15)$$

$$\text{AnimalLover}(x) \wedge \text{Animal}(y) \wedge \text{Kills}(x, y) \rightarrow \text{False} \quad (16)$$

$$\text{Kills}(\text{Jack}, \text{Tuna}) \vee \text{Kills}(\text{Curiosity}, \text{Tuna}) \quad (17)$$

$$\text{Cat}(\text{Tuna}) \quad (18)$$

$$\text{Cat}(x) \rightarrow \text{Animal}(x) \quad (19)$$

Convert to clausal form

$$\text{Dog}(D) \quad (20)$$

$$\text{Owns}(\text{Jack}, D) \quad (21)$$

$$\neg \text{Dog}(y) \vee \neg \text{Owns}(x, y) \vee \text{AnimalLover}(x) \quad (22)$$

$$\neg \text{AnimalLover}(x) \vee \neg \text{Animal}(y) \vee \neg \text{Kills}(x, y) \quad (23)$$

$$\text{Kills}(\text{Jack}, \text{Tuna}) \vee \text{Kills}(\text{Curiosity}, \text{Tuna}) \quad (24)$$

$$\text{Cat}(\text{Tuna}) \quad (25)$$

$$\neg \text{Cat}(x) \vee \text{Animal}(x) \quad (26)$$

Proving $\text{Kills}(\text{Curiosity}, \text{Tuna})$

$$G_0 = \neg \text{Kills}(\text{Curiosity}, \text{Tuna}), \eta = \{\}, \text{ Clause (24)}$$

$$G_1 = \text{Kills}(\text{Jack}, \text{Tuna}), \eta = \{x/\text{Jack}, y/\text{Tuna}\}, \text{ Clause (23)}$$

$$G_2 = \neg \text{AnimalLover}(\text{Jack}) \vee \neg \text{Animal}(\text{Tuna}), \eta = \{x/\text{Tuna}\}, \text{ Clause (26)}$$

$$G_3 = \neg \text{AnimalLover}(\text{Jack}) \vee \neg \text{Cat}(\text{Tuna}), \eta = \{x/\text{Tuna}\}, \text{ Clause (26)}$$

$$G_4 = \neg \text{AnimalLover}(\text{Jack}), \eta = \{\}, \text{ Clause (25)}$$

$$G_5 = \neg \text{Dog}(y) \vee \neg \text{Owns}(\text{Jack}, y), \eta = \{x/\text{Jack}\}, \text{ Clause (22)}$$

$$G_6 = \neg \text{Dog}(D), \eta = \{y/D\}, \eta = \{\}, \text{ Clause (21)}$$

$$G_7 = \square \text{ (or } G_7 = \text{false}), \eta = \{\}, \text{ Clause (20)! DONE}$$