Propositional Logic– Resolution (6A)

Young W. Lim
12/31/16

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using LibreOffice

Based on

Contemporary Artificial Intelligence,
R.E. Neapolitan & X. Jiang

Logic and Its Applications,
Burkey & Foxley

Definitions

A **literal** :

A proposition of the form A or ¬A,
Where A is an atomic proposition other than True or False

A **conjunctive clause** ∧

A **conjunction** of **literal**

A **disjunctive clause** ∨

A **disjunction** of **literal**

A **disjunctive normal form** proposition

The **disjunction** of **conjunctive clause**

A **conjunctive normal form** proposition

The **conjunction** of **disjunctive clause**

Definitions

A **literal** :

$A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_n, C_1, C_2, \dots, C_n$

A **conjunctive clause** $(A_1 \wedge A_2 \wedge \dots \wedge A_n)$

A **disjunctive clause** $(B_1 \vee B_2 \vee \dots \vee B_n)$

A **disjunctive normal form** proposition

$$(A_1 \wedge \dots \wedge A_n) \vee (B_1 \wedge \dots \wedge B_n) \vee (C_1 \wedge \dots \wedge C_n)$$

A **conjunctive normal form** proposition

$$(A_1 \vee \dots \vee A_n) \wedge (B_1 \vee \dots \vee B_n) \wedge (C_1 \vee \dots \vee C_n)$$

Logical Equivalences

Commutativity Law	$A \land B \equiv B \land A, \quad A \lor B \equiv B \lor A$
Distributivity Law	$A \land (B \lor C) \equiv (A \land B) \lor (A \land C), \quad A \lor (B \land C) \equiv (A \lor B) \land (A \lor C)$
De Morgan's Law	$\neg(A \land B) \equiv \neg A \lor \neg B, \quad \neg(A \lor B) \equiv \neg A \land \neg B$
Implication Elimination	$A \Rightarrow B \equiv \neg A \lor B$
If and Only If Elimination	$A \Leftrightarrow B \equiv (A \Rightarrow B) \land (B \Rightarrow A) \equiv (\neg A \lor B) \land (\neg B \lor A)$
Double Negation	$\neg\neg A \equiv A$

$$A \lor (B \land C) \equiv (A \lor B) \land (A \lor C)$$

$$(A \lor B) \land C \equiv (A \lor C) \land (B \lor C)$$

Logical Equivalences

Commutativity Law	$A \land B \equiv B \land A$, $A \lor B \equiv B \lor A$
Distributivity Law	$A \land (B \lor C) \equiv (A \land B) \lor (A \land C)$, $A \lor (B \land C) \equiv (A \lor B) \land (A \lor C)$
De Morgan's Law	$\neg(A \land B) \equiv \neg A \lor \neg B$, $\neg(A \lor B) \equiv \neg A \land \neg B$
Implication Elimination	$A \Rightarrow B \equiv \neg A \lor B$
If and Only If Elimination	$A \Leftrightarrow B \equiv (A \Rightarrow B) \land (B \Rightarrow A) \equiv (\neg A \lor B) \land (\neg B \lor A)$
Double Negation	$\neg\neg A \equiv A$

$\neg((P \Rightarrow Q) \land \neg R)$	$\equiv \neg((\neg P \lor Q) \land \neg R)$	Implication Elimination
	$\equiv \neg(\neg P \lor Q) \lor \neg\neg R$	De Morgan's Law
	$\equiv \neg(\neg P \lor Q) \lor R$	Double Negation
	$\equiv (\neg\neg P \land \neg Q) \lor R$	De Morgan's Law
	$\equiv (P \land \neg Q) \lor R$	Double Negation
	$\equiv (P \lor R) \land \neg(\neg Q \lor R)$	Distributive Law

Logical Equivalences

Commutativity Law	$A \land B \equiv B \land A, \quad A \lor B \equiv B \lor A$
Distributivity Law	$A \land (B \lor C) \equiv (A \land B) \lor (A \land C), \quad A \lor (B \land C) \equiv (A \lor B) \land (A \lor C)$
De Morgan's Law	$\neg(A \land B) \equiv \neg A \lor \neg B, \quad \neg(A \lor B) \equiv \neg A \land \neg B$
Implication Elimination	$A \Rightarrow B \equiv \neg A \lor B$
If and Only If Elimination	$A \Leftrightarrow B \equiv (A \Rightarrow B) \land (B \Rightarrow A) \equiv (\neg A \lor B) \land (\neg B \lor A)$
Double Negation	$\neg\neg A \equiv A$

$(P \land Q) \lor (R \land S) \equiv ((P \land Q) \lor R) \land ((P \land Q) \lor S)$ Distributive Law

$\equiv (P \lor R) \land (Q \lor R) \land ((P \land Q) \lor S)$ Distributive Law

$\equiv (P \lor R) \land (Q \lor R) \land (P \lor S) \land (P \lor S)$ Distributive Law

Conjunctive Normal Form Algorithm

Procedure **Conj_Normal_From** (var Proposition);

Remove all "⇔"
Remove all "⇒"

Repeat

$\neg\neg A \equiv A$

$\neg(A \wedge B) \equiv \neg A \vee \neg B$

$\neg(A \vee B) \equiv \neg A \wedge \neg B$

Until the only negations are single negations of atomic propositions

Repeat

$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$

$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C)$

Until proposition is in conjunctive normal form

Conjunctive Normal Form Algorithm – detail

Input: a proposition
Output : a logically equivalent proposition in conjunctive normal form

Procedure **Conj_Normal_From** (var Proposition);

Remove all "⇔" using the iff elimination law;
Remove all "⇒" using the implication elimination law;

Repeat

If there are any double negations ¬¬

Remove them using the double negation law;

If there are any negations of non-atomic propositions $\neg(A \wedge B)$, $\neg(A \vee B)$

Remove them using the DeMorgan's law;

Until the only negations are single negations of atomic propositions

Repeat

If there are any disjunctions in which one or more terms is a conjuction

Remove them using the following laws

$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$

$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C)$;

Until proposition is in conjunctive normal form

Refutation

An argument consisting of
the premises $A_1, A_2, \ldots, A_n$
and the conclusion B

The negation of the conclusion B

$A_1, A_2, \ldots, A_n \vDash B$	$\Leftrightarrow$	$A_1 \wedge A_2 \wedge \ldots \wedge A_n \wedge \neg B$	Contradiction
$A_1, A_2, \ldots, A_n \vDash B$	$\Leftrightarrow$	$A_1 \wedge A_2 \wedge \ldots \wedge A_n \vDash \neg B$	False

$A_1, A_2, \ldots, A_n \vDash B$	$\Leftrightarrow$	$A_1 \wedge A_2 \wedge \ldots \wedge A_n \Rrightarrow B$	Always True
$A_1, A_2, \ldots, A_n \vDash B$	$\Leftrightarrow$	$A_1 \wedge A_2 \wedge \ldots \wedge A_n \Rightarrow B$	Tautology

Resolution

(A ∨ P), (B ∨¬P) ⊨ A ∨ B   Resolution is on P   Resolvent : A ∨ B

(A ∨ P) ∧ (B ∨¬P)

When P: (A ∨ T) ∧ (B ∨F) ⊨ B
When ¬P: (A ∨ F) ∧ (B ∨T) ⊨ A

(A∨B)

(A ∨ P) ∧ (B ∨¬P)
A ∧ (B ∨¬P) ∨P ∧ (B ∨¬P)
(A ∧ B) ∨ (A ∧¬P) ∨(P ∧ B)∨(P ∧¬P)

When P: (A ∧ B) ∨ (B) ⊨ B
When ¬P: (A ∧ B) ∨ (A) ⊨ A

(A∨B)

(A ∨ ~~P~~) ∧ (B ∨~~¬P~~) ⊨ A ∨ B

Resolution

(A ∨ P), (B ∨¬P) ⊨ A ∨ B	Resolution is on P	Resolvent : A ∨ B
(Q ∨ P), (R ∨¬P) ⊨ Q ∨ R	Resolution is on P	Resolvent : Q ∨ R
(P ∨ ¬Q ∨ R), (¬S ∨ Q) ⊨ P ∨ R ∨ ¬S	Resolution is on Q	Resolvent : P∨R∨¬S

(A ∨ P) ∧ (B ∨¬P) ⊨ A ∨ B

(Q ∨ P), (R ∨¬P) ⊨ Q ∨ R

(P ∨ ¬Q ∨ R), (¬S ∨ Q) ⊨ (P ∨ R) ∨ ¬S

Resolution

$A, (A \Rightarrow B) \models B$

CNF

$A, (\neg A \lor B)$ $A \land (\neg A \lor B)$ ➡ $(A \land \neg A) \lor (A \land B)$ ➡ A, B ➡ B

1. A	Premise					
2. $(\neg A \lor B)$	Premise					
3. $\neg B$	Negation of conclusion					
4. B	Resolvent 1 & 2	$B \land \neg B$	➡	$B, \neg B$	➡	False
5. False	Resolvent of 3 & 4					

Resolution

$(A \Rightarrow B), (B \Rightarrow C) \vDash (A \Rightarrow C)$

CNF

$(\neg A \lor B), (\neg B \lor C)$

$\neg(\neg A \lor C) \equiv A \land \neg C \quad A, \neg C$

1. $(\neg A \lor B)$	Premise
2. $(\neg B \lor C)$	Premise
3. A	Added Premise from the Negation of conclusion
4. $\neg C$	Added Premise from the Negation of conclusion
5. B	Resolvent of 1 & 3
6. $\neg B$	Resolvent of 2 & 4
7. False	Resolvent of 5 & 6

$(\neg A \lor B), (\neg B \lor C), A, \neg C$

$(\neg A \lor B) \land A \Rightarrow A \land B \Rightarrow A, B$

$(\neg B \lor C) \land \neg C \Rightarrow \neg B \land \neg C \Rightarrow \neg B, \neg C$

$B, \neg B \Rightarrow$ False

Resolution

Set_of_Support: initially the negation of the conclusion
Auxiliary_Set : no two clauses in this set resolve to False (all the premises)

Perform all possible resolutions
  Where one clauses is from the Set of support
  All the Resolvents obtained in this way are added into the Set of support

each clause C in Set_of_Support
each clauses D in Auxilary_Set ∪ Set_of_Support
Resolvents = set of clauses obtained by resolving C and D;

If False ∈ Resolvents **return True**;
else New = New ∪ Resolvents **endif**

until New ⊆ Set_of_Support;

Resolution Algorithm

Set_of_Support_Resolution
Input: A set Premises containing the premises in an argument;
The Conclusion in the argument.
Output: The value **True** if Premises *entail* Conclusion; **False** otherwise

Function Premises_Entail_Conclusion (Premises, Conclusion)
Set_of_Support = clauses derived from the negation of Conclusion;
Auxiliary_Set = clauses derived from Premises;
New = { };
Repeat
    Set_of_Support = Set_of_Support ∪ New;
    for each clause C in Set_of_Support
        for each clauses D in Auxilary_Set ∪ Set_of_Support
            Resolvents = set of clauses obtained by resolving C and D;
            If False ∈ Resolvents **return True**;
            else New = New ∪ Resolvents **endif**
        endfor
    endfor
until New ⊆ Set_of_Support;
return False;

Resolution

Set_of_Support = clauses derived from the negation of Conclusion;
Set_of_Support = Set_of_Support ∪ New;

Auxiliary_Set = clauses derived from Premises;

New = { };
New = New ∪ Resolvents

each clause C in Set_of_Support
each clauses D in Auxilary_Set ∪ Set_of_Support
Resolvents = set of clauses obtained by resolving C and D;

If False ∈ Resolvents **return True**;
else New = New ∪ Resolvents **endif**

until New ⊆ Set_of_Support;

Resolution

A, (A ⇒ B) ⊨ B

	New	Set of Support	Auxiliary Set
CNF	{ }	{ }	{ }
A, (¬A ∨ B), ¬B	{ }	{¬B}	{A, (¬A ∨ B)}
¬B, (¬A ∨ B) ⊨ ¬A	{¬A}		
	{¬A}	{¬B, ¬A}	{A, (¬A ∨ B)}
¬A, A ⊨ False	{¬A, False}		

Resolution

(A ⇒ B), (B ⇒ C) ⊨ (A ⇒ C)

CNF	New	Set of Support	Auxiliary Set
(¬A ∨ B), (¬B ∨ C), A, ¬C	{ }	{ }	{ }
A, (¬A ∨ B) ⊨ B	{ }	{A, ¬C}	{(¬A ∨ B), (¬B ∨ C)}
	{B}	{A, ¬C}	{(¬A ∨ B), (¬B ∨ C)}
¬C, (¬B ∨ C) ⊨ ¬B	{B}	{A, ¬C, B}	{(¬A ∨ B), (¬B ∨ C)}
	{B,¬B}	{A, ¬C, B}	{(¬A ∨ B), (¬B ∨ C)}
¬B, B ⊨ False	{B,¬B}	{A, ¬C, B, ¬B}	{(¬A ∨ B), (¬B ∨ C)}
	{B,¬B, False}		

Logical Equivalences

¬, ∧,
∨

¬, ∧,
∨

∧∨⌐¬
⇒⇔≡⇛
⊨

∧∨⌐¬
⇒⇔≡⇛
⊨

⇒
⇔
≡

⇒
⇔
≡

References

[1] en.wikipedia.org
[2] en.wiktionary.org
[3] U. Endriss, “Lecture Notes : Introduction to Prolog Programming”
[4] http://www.learnprolognow.org/ Learn Prolog Now!
[5] http://www.csupomona.edu/~jrfisher/www/prolog_tutorial
[6] www.cse.unsw.edu.au/~billw/cs9414/notes/prolog/intro.html
[7] www.cse.unsw.edu.au/~billw/dictionaries/prolog/negation.html
[8] http://ilppp.cs.lth.se/, P. Nugues,` An Intro to Lang Processing with Perl and Prolog