Artificial Intelligence

Propositional Logic

Zahoor Tanoli (PhD)

Outline

- Motivation
- Technical Solution
  - Syntax
  - Semantics
  - Inference
- Illustration by a Larger Example
- Extensions
- Summary
- References

MOTIVATION

Logic and Deduction

- Logic is used to formalize deduction
- **Deduction** = derivation of true statements (called conclusions) from statements that are assumed to be true (called premises)
- Natural language is **not** precise, so the careless use of logic can lead to claims that false statements are true, or to claims that a statement is true, even though its truth does not necessarily follow from the premises

  => Logic provides a way to talk about truth and correctness in a rigorous way, so that we can prove things, rather than make intelligent guesses and just hope they are correct

Why Propositional Logic?

- Propositional logic is a good vehicle to introduce basic properties of logic; used to:
  - Associate natural language expressions with semantic representations
  - Evaluate the truth or falsity of semantic representations relative to a knowledge base
  - Compute inferences over semantic representations
- One of the simplest and most common logic
  - The core of (almost) all other logics

What is Propositional Logic?

- An unambiguous formal language, akin to a programming language
  - Syntax: Vocabulary for expressing concepts without ambiguity
  - Semantics: Connection to what we're reasoning about
    - *Interpretation* - what the syntax means
  - Reasoning: How to prove things
    - What steps are allowed

TECHNICAL SOLUTIONS

SYNTAX

Syntax

- Logical constants: true, false
- Propositional symbols: P, Q, S, ...
- Wrapping parentheses: (…)
- Atomic formulas: Propositional Symbols or logical constants
- Formulas are either atomic formulas, or can be formed by combining atomic formulas with the following connectives:

$\wedge$ ...and	[conjunction]
$\vee$ ...or	[disjunction]
$\rightarrow$...implies	[implication / conditional]
$\leftrightarrow$..is equivalent	[biconditional]
$\neg$ ...not	[negation]

Syntax (cont')

- A sentence (well formed formula) is defined as follows:
  - A symbol is a sentence
  - If S is a sentence, then $\neg$S is a sentence
  - If S is a sentence, then (S) is a sentence
  - If S and T are sentences, then (S $\vee$ T), (S $\wedge$ T), (S $\rightarrow$ T), and (S $\leftrightarrow$ T) are sentences
  - A sentence results from a finite number of applications of the above rules

Syntax – BNF Grammar

Sentence $\rightarrow$ *AtomicSentence* |
ComplexSentence

AtomicSentence $\rightarrow$ True | False | P | Q | R | . . .

ComplexSentence $\rightarrow$ (*Sentence*)
| *Sentence Connective Sentence*
| $\neg$ *Sentence*

Connective $\rightarrow$ $\wedge$ | $\vee$ | $\rightarrow$ | $\leftrightarrow$

Ambiguities are resolved through precedence $\neg \wedge \vee \rightarrow \leftrightarrow$
or parentheses

e.g. $\neg P \vee Q \wedge R \Rightarrow S$ is equivalent to $(\neg P) \vee (Q \wedge R)) \Rightarrow S$

Syntax – Examples

- P means “It is hot.”
- Q means “It is humid.”
- R means “It is raining.”
- $(P \wedge Q) \rightarrow R$

  “If it is hot and humid, then it is raining”
- $Q \rightarrow P$

  “If it is humid, then it is hot”

- $\neg p \wedge \neg q$
- $\neg(p \wedge q)$
- $(p \wedge q) \vee r$
- $p \wedge q \wedge r$
- $(((\neg p) \wedge q) \rightarrow r) \leftrightarrow ((\neg r) \vee p)$
- $(\neg (p \vee q) \rightarrow q) \rightarrow r$
- $((\neg p) \vee (\neg q)) \leftrightarrow (\neg r)$
- Etc.

SEMANTICS

Semantics

- Interpretations
- Equivalence
- Substitution
- Models and Satisfiability
- Validity
- Logical Consequence (Entailment)
- Theory

Semantics – Some Informal Definitions

- Given the truth values of all symbols in a sentence, it can be "evaluated" to determine its **truth value** (True or False)
- A **model** for a KB is a "possible world" (assignment of truth values to propositional symbols) in which each sentence in the KB is True
- A **valid sentence** or **tautology** is a sentence that is True under all interpretations, no matter what the world is actually like or how the semantics are defined (example: "It's raining or it's not raining")
- An **inconsistent sentence** or **contradiction** is a sentence that is False under all interpretations (the world is never like what it describes, as in "It's raining and it's not raining")
- **P entails Q**, written $P \models Q$, means that whenever P is True, so is Q; in other words, all models of P are also models of Q

Interpretations

- In propositional logic, truth values are assigned to the atoms of a formula in order to evaluate the truth value of the formula
- An assignment is a function

$$v : P \rightarrow \{T,F\}$$

v assigns a truth value to any atom in a given formula (*P* is the set of all propositional letters, i.e. atoms)
Suppose *F* denotes the set of all propositional formulas. We can extend an assignment v to a function

$$v : F \rightarrow \{T,F\}$$

which assigns the truth value *v(A)* to any formula *A* in *F.*
v is called an interpretation.

Interpretations (cont')

- Example:
  - Suppose v is an assignment for which

    $$v(p) = F, \qquad v(q) = T.$$

  - If $A = (\neg p \rightarrow q) \leftrightarrow (p \vee q)$, what is $v(A)$?

    Solution:

$$\begin{aligned} v(A) &= v((\neg p \rightarrow q) \leftrightarrow (p \vee q)) \\ &= v(\neg p \rightarrow q) \leftrightarrow v(p \vee q) \\ &= (v(\neg p) \rightarrow v(q)) \leftrightarrow (v(p) \vee v(q)) \\ &= (\neg v(p) \rightarrow v(q)) \leftrightarrow (v(p) \vee v(q)) \\ &= (\neg F \rightarrow T) \leftrightarrow (F \vee T) \\ &= (T \rightarrow T) \leftrightarrow (F \vee T) \\ &= T \leftrightarrow T \\ &= T \end{aligned}$$

Equivalence

- If A,B are formulas are such that

  $$v(A) = v(B)$$

  for **all** interpretations v, A is (logically) equivalent to B:

  $$A \equiv B$$

- Example: $\neg p \ V \ q \equiv p \rightarrow q$ since both formulas are true in all interpretations except when $v(p)$ = T, $v(q)$ = F and are false for that particular interpretation
- Caution**:** ≡ does **not** mean the same thing as ↔ :
  - $A \leftrightarrow B$ is a formula (syntax)
  - $A \equiv B$ is a relation between two formula (semantics)

  Theorem: $A \equiv B$ if and only if $A \leftrightarrow B$ is true in every interpretation; i.e. $A \leftrightarrow B$ is a tautology.

Equivalence and Substitution – Examples

- Examples of logically equivalent formulas

$$A \equiv \neg\neg A$$
$$A \vee B \equiv B \vee A \qquad A \wedge B \equiv B \wedge A$$
$$(A \vee B) \vee C \equiv A \vee (B \vee C) \qquad (A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$$
$$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$$
$$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$$
$$\neg(A \wedge B) \equiv \neg A \vee \neg B \qquad \neg(A \vee B) \equiv \neg A \wedge \neg B$$
$$A \wedge \mathit{true} \equiv A \qquad A \rightarrow \mathit{false} \equiv \neg A$$

- Example: Simplify $p \vee (\neg p \wedge q)$
  - Solution:

$$\begin{aligned} p \vee (\neg p \wedge q) &\equiv (p \vee \neg p) \wedge (p \vee q) \\ &\equiv \mathsf{T} \wedge (p \vee q) \\ &\equiv p \vee q \end{aligned}$$

Models and Satisfiability

- A propositional **formula** *A* is satisfiable iff $v(A) = T$ in **some** interpretation *v;* such an interpretation is called a **model** for *A*.
  - *A* is unsatisfiable (or, contradictory) if it is false in every interpretation
- A **set of formulas** $U = \{A_1, A_2, \ldots, A_n\}$ is satisfiable iff there exists an interpretation *v* such that $v(A_1) = v(A_2) = \ldots = v(A_n) = T$; such an interpretation is called a **model** of *U.*
  - *U* is unsatisfiable if no such interpretation exists
- Relevant properties:
  - If *U* is satisfiable, then so is $U - \{A_i\}$ for any $i = 1, 2, \ldots, n$
  - If *U* is satisfiable and *B* is valid, then $U \cup \{B\}$ is also satisfiable
  - If *U* is unsatisfiable and *B* is **any** formula, $U \cup \{B\}$ is also unsatisfiable
  - If *U* is unsatisfiable and some A_i is valid, then $U - \{A_i\}$ is also unsatisfiable

INFERENCE

Inference Methods

- Several basic methods for determining whether a given set of premises propositionally entails a given conclusion
  - Truth Table Method
  - Deductive (Proof) Systems
  - Resolution

Truth Table Method

- One way of determining whether or not a set of premises logically entails a possible conclusion is to check the truth table for the logical constants of the language
- This is called the truth table method and can be formalized as follows:
  - Step 1: Starting with a complete truth table for the propositional constants, iterate through all the premises of the problem, for each premise eliminating any row that does not satisfy the premise
  - Step 2: Do the same for the conclusion
  - Step 3: Finally, compare the two tables; If every row that remains in the premise table, i.e. is not eliminated, also remains in the conclusion table, i.e. is not eliminated, then the premises logically entail the conclusion

Example

- Simple sentences:
  - *Amy loves Pat:* $lovesAmyPat$
  - *Amy loves Quincy:* $lovesAmyQuincy$
  - *It is Monday:* $ismonday$
- Premises:
  - *If Amy loves Pat, Amy loves Quincy:*

    $lovesAmyPat \rightarrow lovesAmyQuincy$
  - *If it is Monday, Amy loves Pat or Quincy:*

    $ismonday \rightarrow lovesAmyPat \vee lovesAmyQuincy$
- Question:
  - *If it is Monday, does Amy love Quincy?*

    i.e. is $ismonday \rightarrow lovesAmyQuincy$ entailed by the premises?

Step 1: Truth table for the premises

lovesAmyPat	*lovesAmyQuincy*	*ismonday*	*lovesAmyPat* → *lovesAmyQuincy*	*ismonday* → *lovesAmyPat* ∨ *lovesAmyQuincy*
T	*T*	*T*	T	T
T	*T*	*F*	T	T
T	*F*	*T*	F	T
T	*F*	*F*	F	T
F	*T*	*T*	T	T
F	*T*	*F*	T	T
F	*F*	*T*	T	F
F	*F*	*F*	T	T

Step 1: Eliminate non-sat interpretations

lovesAmyPat	*lovesAmyQuincy*	*ismonday*	*lovesAmyPat* → *lovesAmyQuincy*	*ismonday* → *lovesAmyPat* ∨ *lovesAmyQuincy*
T	*T*	*T*	T	T
T	*T*	*F*	T	T
T	*F*	*T*	F	T
T	*F*	*F*	F	T
F	*T*	*T*	T	T
F	*T*	*F*	T	T
F	*F*	*T*	T	F
F	*F*	*F*	T	T

Step 2: Truth table for the conclusion

lovesAmyPat	*lovesAmyQuincy*	*ismonday*	*ismonday* → *lovesAmyQuincy*
T	*T*	*T*	T
T	*T*	*F*	T
T	*F*	*T*	F
T	*F*	*F*	T
F	*T*	*T*	T
F	*T*	*F*	T
F	*F*	*T*	F
F	*F*	*F*	T

Step 2: Eliminate non-sat interpretations

lovesAmyPat	*lovesAmyQuincy*	*ismonday*	*ismonday* → *lovesAmyQuincy*
T	*T*	*T*	T
T	*T*	*F*	T
T	*F*	*T*	F
T	*F*	*F*	T
F	*T*	*T*	T
F	*T*	*F*	T
F	*F*	*T*	F
F	*F*	*F*	T

Step 3: Comparing tables

- Finally, in order to make the determination of logical entailment, we compare the two rightmost tables and notice that every row remaining in the premise table also remains in the conclusion table.
  - In other words, the premises logically entail the conclusion.
- The truth table method has the merit that it is easy to understand
  - It is a direct implementation of the definition of logical entailment.
- In practice, it is awkward to manage two tables, especially since there are simpler approaches in which *only one table* needs to be manipulated
  - **Validity Checking**
  - **Unsatisfability Checking**

Validity checking

- Approach: To determine whether a set of sentences

  $$\{\varphi_1,\ldots,\varphi_n\}$$

  logically entails a sentence φ, form the sentence

  $$(\varphi_1 \wedge \ldots \wedge \varphi_n \rightarrow \varphi)$$

  and check that it is valid.
- To see how this method works, consider the previous example and write the tentative conclusion as shown below.

  $$(\text{lovesAmyPat} \rightarrow \text{lovesAmyQuincy}) \wedge (\text{ismonday} \rightarrow \text{lovesAmyPat} \vee \text{lovesAmyQuincy}) \rightarrow (\text{ismonday} \rightarrow \text{lovesAmyQuincy})$$

- Then, form a truth table for our language with an added column for this sentence and check its satisfaction under each of the possible interpretations for our logical constants

Unsatisfability Checking

- It is almost exactly the same as the validity checking method, except that it works negatively instead of positively.
- To determine whether a finite set of sentences $\{\varphi_1,\ldots,\varphi_n\}$ logically entails a sentence φ, we form the sentence

  $$(\varphi_1 \wedge \ldots \wedge \varphi_n \wedge \neg\varphi)$$

  and check that it is *unsatisfiable*.

- Both the validity checking method and the satisfiability checking method require about the same amount of work as the truth table method, but they have the merit of manipulating **only one table**

Example – A truth table

p	q	r	$p \to q$	$p \to r$	$p \to r \vee q$	$(p \to q) \wedge (p \to r) \to (p \to r \vee q)$	$p \to r \wedge q$	$(p \to q) \wedge (p \to r) \to (p \to r \wedge q)$
T	T	T	T	T	T	T	T	T
T	T	F	T	F	T	T	F	T
T	F	T	F	T	T	T	F	T
T	F	F	F	F	F	T	F	T
F	T	T	T	T	T	T	T	T
F	T	F	T	T	T	T	T	T
F	F	T	T	T	T	T	T	T
F	F	F	T	T	T	T	T	T

Deductive (proof) systems

- Semantic methods for checking logical entailment have the merit of being conceptually simple; they directly manipulate interpretations of sentences
- Unfortunately, the number of interpretations of a language grows **exponentially** with the number of logical constants.
  - When the number of logical constants in a propositional language is large, the number of interpretations may be impossible to manipulate.
- Deductive (proof) systems provide an alternative way of checking and communicating logical entailment that addresses this problem
  - In many cases, it is possible to create a “proof” of a conclusion from a set of premises that is much smaller than the truth table for the language;
  - Moreover, it is often possible to find such proofs with less work than is necessary to check the entire truth table

Rules of Inference

- The basis for proof systems is the use of correct rules of inference that can be applied directly to sentences to derive conclusions that are guaranteed to be correct under all interpretations
  - Since the interpretations are not enumerated, time and space can often be saved
- A rule of inference is a pattern of reasoning consisting of:
  - One set of sentence schemata, called *premises*, and
  - A second set of sentence schemata, called *conclusions*
- A rule of inference is sound if and only if, for every instance, the premises logically entail the conclusions

Resolution

- **Propositional resolution** is an extremely powerful rule of inference for Propositional Logic
- Using propositional resolution (without axiom schemata or other rules of inference), it is possible to build a theorem prover that is sound and complete for all of Propositional Logic
- The search space using propositional resolution is much smaller than for standard propositional logic
- Propositional resolution works only on expressions in *clausal form*
  - Before the rule can be applied, the premises and conclusions must be converted to this form

Clausal Forms

- A clause is a set of literals which is assumed (implicitly) to be a disjunction of those literals
  - Example: $\neg p \vee q \vee \neg r \quad \leftrightsquigarrow \quad \{\neg p, q, \neg r\}$
- Unit clause: clause with only one literal; e.g. {¬*q*}
- Clausal form of a formula: Implicit conjunction of clauses
- Example: $p \wedge (\neg p \vee q \vee \neg r) \wedge (\neg q \vee q \vee \neg r) \wedge (\neg q \vee p)$

  $$\Updownarrow$$

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

  Abbreviated notation: $\{p, \bar{p}q\bar{r}, \bar{q}q\bar{r}, \bar{q}p\}$
- Notation:
  - *l*-literal, l^c-complement of *l*
  - *C*-clause (a set of literals)
  - *S*-a clausal form (a set of clauses)

Converting to clausal form

Theorem: Every propositional formula can be transformed into an equivalent formula in CNF

1. Implications:

$\varphi_1 \rightarrow \varphi_2 \quad \rightarrow \quad \neg\varphi_1 \vee \varphi_2$

$\varphi_1 \leftarrow \varphi_2 \quad \rightarrow \quad \varphi_1 \vee \neg\varphi_2$

$\varphi_1 \leftrightarrow \varphi_2 \quad \rightarrow \quad (\neg\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \neg\varphi_2)$

2. Negations:

$\neg\neg\varphi \quad \rightarrow \quad \varphi$

$\neg(\varphi_1 \wedge \varphi_2) \quad \rightarrow \quad \neg\varphi_1 \vee \neg\varphi_2$

$\neg(\varphi_1 \vee \varphi_2) \quad \rightarrow \quad \neg\varphi_1 \wedge \neg\varphi_2$

3. Distribution:

$\varphi_1 \vee (\varphi_2 \wedge \varphi_3) \quad \rightarrow \quad (\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3)$

$(\varphi_1 \wedge \varphi_2) \vee \varphi_3 \quad \rightarrow \quad (\varphi_1 \vee \varphi_3) \wedge (\varphi_2 \vee \varphi_3)$

$(\varphi_1 \vee \varphi_2) \vee \varphi_3 \quad \rightarrow \quad \varphi_1 \vee (\varphi_2 \vee \varphi_3)$

$(\varphi_1 \wedge \varphi_2) \wedge \varphi_3 \quad \rightarrow \quad \varphi_1 \wedge (\varphi_2 \wedge \varphi_3)$

Example

- Transform the formula

$$(p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)$$

into an equivalent formula in CNF

Solution:

$$\begin{aligned}
&(p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p) \\
&\equiv (\neg p \vee q) \rightarrow (\neg\neg q \vee \neg p) \\
&\equiv \neg(\neg p \vee q) \vee (\neg\neg q \vee \neg p) \\
&\equiv (\neg\neg p \wedge \neg q) \vee (\neg\neg q \vee \neg p) \\
&\equiv (p \wedge \neg q) \vee (q \vee \neg p) \\
&\equiv (p \vee q \vee \neg p) \wedge (\neg q \vee q \vee \neg p)
\end{aligned}$$

Resolution (cont')

- Example: Show that *(p → q) → (¬q → ¬p)* is a valid formula
  Solution: We will show that

  $$\neg[(p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)]$$

  is not satisfiable.
  (1) Transform the formula into CNF:

$$\begin{aligned}\neg[(p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)] &\equiv (p \rightarrow q) \wedge \neg(\neg q \rightarrow \neg p) \\ &\equiv (\neg p \vee q) \wedge \neg q \wedge \neg\neg p \\ &\equiv (\neg p \vee q) \wedge \neg q \wedge p\end{aligned}$$

Resolution (cont')

Prove R

1	P v Q
2	$P \rightarrow R$
3	$Q \rightarrow R$

Step	Formula	Derivation
1	P v Q	Given
2	$\neg$ P v R	Given
3	$\neg$ Q v R	Given
4	$\neg$ R	Negated conclusion
5	Q v R	1,2

Resolution (cont')

Prove R

1	P v Q
2	$P \rightarrow R$
3	$Q \rightarrow R$

Step	Formula	Derivation
1	P v Q	Given
2	¬ P v R	Given
3	¬ Q v R	Given
4	¬ R	Negated conclusion
5	Q v R	1,2
6	¬ P	2,4
7	¬ Q	3,4
8	R	5,7
9	•	4,8

ILLUSTRATION BY LARGER EXAMPLE

Problem Example

- For each of these sets of premises, what relevant conclusion or conclusions can be drawn? Explain the rules of inference used to obtain each conclusion from the premises.

  (a) “If I eat spicy foods, then I have strange dreams.” “I have strange dreams if there is thunder while I sleep.” “I did not have strange dreams.”

  (b) “I am dreaming or hallucinating.” “I am not dreaming.” “If I am hallucinating, I see elephants running down the road.”

  (c) “If I work, it is either sunny or partly sunny.” “I worked last Monday or I worked last Friday.” “It was not sunny on Tuesday.” “It was not partly sunny on Friday.”

Solution (a)

(a) “If I eat spicy foods, then I have strange dreams.” “I have strange dreams if there is thunder while I sleep.” “I did not have strange dreams.”

- The relevant conclusions are: “I did not eat spicy food” and “There is no thunder while I sleep”.
- Let the primitive statements be:
  - *s, ‘I eat spicy foods’*
  - *d, ‘I have strange dreams’*
  - *t, ‘There is thunder while I sleep’*
- Then the premises are translated as: $s \rightarrow d$, $t \rightarrow d$, *and* $\neg d$.
- And the conclusions: $\neg s$, $\neg t$.

• Steps	Reason
1. $s \rightarrow d$	*premise*
2. $\neg d$	*premise*
3. $\neg s$	*Modus Tollens to Steps 1 and 2*
4. $t \rightarrow d$	*premise*
5. $\neg t$	*Modus Tollens to Steps 4 and 2.*

Solution (b)

(b) “I am dreaming or hallucinating.” “I am not dreaming.” “If I am hallucinating, I see elephants running down the road.”

- The relevant conclusion is: “I see elephants running down the road.”.
- Let the primitive statements be:
  - *d, ‘I am dreaming’*
  - *h, ‘I am hallucinating’*
  - *e, ‘I see elephants running down the road’*
- Then the premises are translated as: $d \vee h$, $\neg d$, *and* $h \rightarrow e$.
- And the conclusion: *e.*

• Steps	Reason
1. $d \vee h$	*premise*
2. $\neg d$	*premise*
3. h	*rule of disjunctive syllogism to Steps 1 and 2*
4. $h \rightarrow e$	*premise*
5. e	*Modus Ponens to Steps 4 and 3*

Solution (c)

(c) “If I work, it is either sunny or partly sunny.” “I worked last Monday or I worked last Friday.” “It was not sunny on Tuesday.” “It was not partly sunny on Friday.”

- There is no single relevant conclusion in this problem, its main difficulty is to to represent the premises so that one is able infer anything at all. One possible relevant conclusion is: “It was sunny or partly sunny last Monday or it was sunny last Friday.”.
- Let the primitive statements be:
  - *wm, ‘I worked last Monday’*
  - *wf , ‘I worked last Friday’*
  - *sm, ‘It was sunny last Monday’*
  - *st, ‘It was sunny last Tuesday’*
  - *sf , ‘It was sunny last Friday’*
  - *pm, ‘It was partly sunny last Monday’*
  - *pf , ‘It was partly sunny last Friday’*
- Then the premises are translated as: $wm \vee wf$, $wm \rightarrow (sm \vee pm)$, $wf \rightarrow (sf \vee pf)$, $\neg st$, and $\neg pf$.
- And the conclusion: $sf \vee sm \vee pm$.

Solution (c) – Method 1

• Steps	Reason
1. *wf → (sf ∨ pf)*	*premise*
2. *¬wf ∨ sf ∨ pf*	*expression for implication*
3. *¬pf → (¬wf ∨ sf)*	*expression for implication*
4. *¬pf*	*premise*
5. *¬wf ∨ sf*	*modus ponens to Steps 3 and 4*
6. *wf → sf*	*expression for implication*
7. *wm ∨ wf*	*premise*
8. *¬wm → wf*	*expression for implication*
9. *¬wm → sf*	*rule of syllogism to Steps 8 and 6*
10. *wm ∨ sf*	*expression for implication*
11. *¬sf → wm*	*expression for implication*
12. *wm → (sm ∨ pm)*	*premise*
13. *¬sf → (sm ∨ pm)*	*rule of syllogism to Steps 11 and 12*
14. *sf ∨ sm ∨ pm*	*expression for implication.*

Solution (c) – Method 2 (Use the rule of resolution)

• Steps	Reason
1. $wf \rightarrow (sf \vee pf)$	*premise*
2. $\neg wf \vee sf \vee pf$	*expression for implication*
3. $\neg pf$	*premise*
4. $\neg wf \vee sf$	*rule of resolution to Steps 2 and 3*
5. $wm \vee wf$	*premise*
6. $wm \vee sf$	*rule of resolution to Steps 4 and 5*
7. $wm \rightarrow (sm \vee pm)$	*premise*
8. $\neg wm \vee sm \vee pm$	*expression for implication*
9. $sf \vee sm \vee pm$	*rule of resolution to Steps 7 and 8*

EXTENSIONS

Extensions

- Propositional logic is not adequate for formalizing valid arguments that rely on the internal structure of the propositions involved

- In propositional logic the smallest atoms represent whole propositions (propositions are atomic)
  - Propositional logic does not capture the **internal structure** of the propositions
  - It is not possible to work with units smaller than a proposition

- Example:
  - “A Mercedes Benz is a Car” and “A car drives” are two individual, **unrelated** propositions
  - We cannot conclude “A Mercedes Benz drives”

Extensions

- It is possible to represent everything you want in propositional logic
  - But often this is not very efficient

- **Basic idea**: A proposition is expressed as predicate about (on or more) objects in the world
- Propositions are **predicates** and **arguments**
  - I.e. Car(Mercedes Benz).

- The most immediate way to develop a more complex logical calculus is to introduce rules that are sensitive to more fine-grained details of the sentences being used
  - When the atomic sentences of propositional logic are broken up into terms, variables, predicates, and quantifiers, they yield **first-order logic**, which keeps all the rules of propositional logic and adds some new ones