Propositional Logic
Part 1

Yingyu Liang

yliang@cs.wisc.edu

Computer Sciences Department

University of Wisconsin, Madison

[Based on slides from Louis Oliphant, Andrew Moore, Jerry Zhu]

5 is even implies 6 is odd.

Is this sentence logical?
True or false?

Logic

- If the rules of the world are presented formally, then a decision maker can use logical reasoning to make rational decisions.

- Several types of logic:
  - propositional logic (Boolean logic)
  - first order logic (first order predicate calculus)

- A logic includes:
  - syntax: what is a correctly formed sentence
  - semantics: what is the meaning of a sentence
  - Inference procedure (reasoning, entailment): what sentence logically follows given knowledge

Propositional logic syntax

Sentence	$\rightarrow$ *AtomicSentence* | *ComplexSentence*
AtomicSentence	$\rightarrow$ True | False | *Symbol*
Symbol	$\rightarrow$ P | Q | R | . . .
ComplexSentence	$\rightarrow$ $\neg$ *Sentence*
|	(*Sentence* $\wedge$ *Sentence*)
|	(*Sentence* $\vee$ *Sentence*)
|	(*Sentence* $\Rightarrow$ *Sentence*)
|	(*Sentence* $\Leftrightarrow$ *Sentence*)

BNF (Backus-Naur Form) grammar in propositional logic

$((\neg P \vee ((True \wedge R) \Leftrightarrow Q)) \Rightarrow S$ well formed

$(\neg(P \vee Q) \wedge \Rightarrow S)$ not well formed

Propositional logic syntax

() control the order of operations

Means True

$$((\neg P \vee ((\text{True} \wedge R) \Leftrightarrow Q)) \Rightarrow S)$$

Means “Not”

Means “Or” -- disjunction

Means “if-then” implication

Means “And” -- conjunction

Means “iff” -- biconditional

Propositional symbols must be specified

Propositional logic syntax

- Precedence (from highest to lowest):

$$\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$$

- If the order is clear, you can leave off parenthesis.

$\neg P \vee \text{True} \wedge R \Leftrightarrow Q \Rightarrow S$ ok

$P \Rightarrow Q \Rightarrow S$ not ok

Semantics

- An interpretation is a complete True / False assignment to propositional symbols
  - Example symbols: P means “It is hot”, Q means “It is humid”, R means “It is raining”
  - There are 8 interpretations (TTT, ..., FFF)
- The semantics (meaning) of a sentence is the set of interpretations in which the sentence evaluates to True.
- Example: the semantics of the sentence $P \vee Q$ is the set of 6 interpretations
  - P=True, Q=True, R=True or False
  - P=True, Q=False, R=True or False
  - P=False, Q=True, R=True or False
- A model of a set of sentences is an interpretation in which all the sentences are true.

Evaluating a sentence under an interpretation

- Calculated using the meaning of connectives, recursively.

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
$false$	$false$	$true$	$false$	$false$	$true$	$true$
$false$	$true$	$true$	$false$	$true$	$true$	$false$
$true$	$false$	$false$	$false$	$true$	$false$	$false$
$true$	$true$	$false$	$true$	$true$	$true$	$true$

- Pay attention to $\Rightarrow$
  - "5 is even implies 6 is odd" is True!
  - If P is False, regardless of Q, P$\Rightarrow$Q is True
  - No causality needed: "5 is odd implies the Sun is a star" is True.

Semantics example

$$\neg P \vee Q \wedge R \Rightarrow Q$$

Semantics example

$$\neg P \vee Q \wedge R \Rightarrow Q$$

P	Q	R	~P	Q^R	~PvQ^R	~PvQ^R->Q
0	0	0	1	0	1	0
0	0	1	1	0	1	0
0	1	0	1	0	1	1
0	1	1	1	1	1	1
1	0	0	0	0	0	1
1	0	1	0	0	0	1
1	1	0	0	0	0	1
1	1	1	0	1	1	1

Satisfiable: the sentence is true under some interpretations

Deciding satisfiability of a sentence is NP-complete

Semantics example

$$(P \wedge R \Rightarrow Q) \wedge P \wedge R \wedge \neg Q$$

Semantics example

$$(P \land R \Rightarrow Q) \land P \land R \land \neg Q$$

P	Q	R	~Q	R^~Q	P^R^~Q	P^R	P^R->Q	final
0	0	0	1	0	0	0	1	0
0	0	1	1	1	0	0	1	0
0	1	0	0	0	0	0	1	0
0	1	1	0	0	0	0	1	0
1	0	0	1	0	0	0	1	0
1	0	1	1	1	1	1	0	0
1	1	0	0	0	0	0	1	0
1	1	1	0	0	0	1	1	0

Unsatisfiable: the sentence is false under all interpretations.

Semantics example

$$(P \Rightarrow Q) \vee \; P \wedge \neg Q$$

Semantics example

$$(P \Rightarrow Q) \vee P \wedge \neg Q$$

P	Q	R	~Q	P->Q	P^~Q	(P->Q)vP^~Q
0	0	0	1	1	0	1
0	0	1	1	1	0	1
0	1	0	0	1	0	1
0	1	1	0	1	0	1
1	0	0	1	0	1	1
1	0	1	1	0	1	1
1	1	0	0	1	0	1
1	1	1	0	1	0	1

Valid: the sentence is true under all interpretations

Also called tautology.

Knowledge base

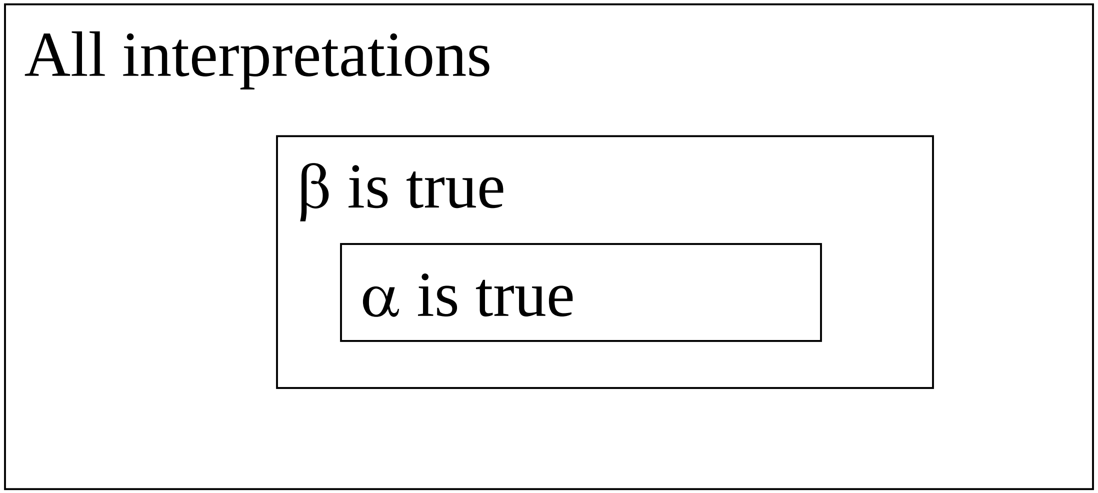
- A knowledge base KB is a set of sentences. Example KB:
  - TomGivingLecture $\Leftrightarrow$ (TodayIsTuesday $\vee$ TodayIsThursday)
  - $\neg$ TomGivingLecture
- It is equivalent to a single long sentence: the conjunction of all sentences
  - (TomGivingLecture $\Leftrightarrow$ (TodayIsTuesday $\vee$ TodayIsThursday)) $\wedge$ $\neg$ TomGivingLecture
- The model of a KB is the interpretations in which all sentences in the KB are true.

Entailment

- Entailment is the relation of a sentence β logically follows from other sentences α (i.e. the KB).

$$\alpha \models \beta$$

- $\alpha \models \beta$ if and only if, in every interpretation in which α is true, β is also true



Method 1: model checking

We can enumerate all interpretations and check this. This is called model checking or truth table enumeration. Equivalently…

- Deduction theorem: $\alpha \models \beta$ if and only if $\alpha \Rightarrow \beta$ is valid (always true)
- Proof by contradiction (refutation, *reductio ad absurdum*): $\alpha \models \beta$ if and only if $\alpha \wedge \neg\beta$ is unsatisfiable
- There are 2^n interpretations to check, if the KB has n symbols

Inference

- Let's say you write an algorithm which, according to you, proves whether a sentence β is entailed by α, without the lengthy enumeration
- The thing your algorithm does is called inference
- We don't trust your inference algorithm (yet), so we write things your algorithm finds as

$$\alpha \vdash \beta$$

- It reads "β is derived from α by your algorithm"
- What properties should your algorithm have?
  - Soundness: the inference algorithm only derives entailed sentences. If $\alpha \vdash \beta$ then $\alpha \models \beta$
  - Completeness: all entailment can be inferred. If $\alpha \models \beta$ then $\alpha \vdash \beta$

Method 2: Sound inference rules

- All the logical equivalences
- *Modus Ponens* (Latin: mode that affirms)

$$\frac{\alpha \Rightarrow \beta,\ \alpha}{\beta}$$

- And-elimination

$$\frac{\alpha \wedge \beta}{\alpha}$$

Logical equivalences

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) \quad \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) \quad \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) \quad \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) \quad \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha \quad \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) \quad \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) \quad \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) \quad \text{de Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) \quad \text{de Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \quad \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \quad \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

You can use these equivalences to modify sentences.

Proof

- Series of inference steps that leads from α (or KB) to β
- This is exactly a search problem

KB:

1. TomGivingLecture $\Leftrightarrow$ (TodayIsTuesday $\vee$ TodayIsThursday)
2. $\neg$ TomGivingLecture

β:

$\neg$ TodayIsTuesday

Proof

KB:

1. TomGivingLecture $\Leftrightarrow$ (TodayIsTuesday $\vee$ TodayIsThursday)
2. $\neg$ TomGivingLecture

3. TomGivingLecture $\Rightarrow$ (TodayIsTuesday $\vee$ TodayIsThursday) $\wedge$ (TodayIsTuesday $\vee$ TodayIsThursday) $\Rightarrow$ TomGivingLecture biconditional-elimination to 1.
4. (TodayIsTuesday $\vee$ TodayIsThursday) $\Rightarrow$ TomGivingLecture and-elimination to 3.
5. $\neg$ TomGivingLecture $\Rightarrow$ $\neg$(TodayIsTuesday $\vee$ TodayIsThursday) contraposition to 4.
6. $\neg$(TodayIsTuesday $\vee$ TodayIsThursday) Modus Ponens 2,5.
7. $\neg$TodayIsTuesday $\wedge$ $\neg$TodayIsThursday de Morgan to 6.
8. $\neg$ TodayIsTuesday and-elimination to 7.

Method 3: Resolution

- Your algorithm can use all the logical equivalences, *Modus Ponens, a*nd-elimination to derive new sentences.
- Resolution: a single inference rule
  - Sound: only derives entailed sentences
  - Complete: can derive any entailed sentence
    - Resolution is only refutation complete: if KB $|= \beta$, then KB $\wedge \neg \beta$ |- *empty*. It cannot derive *empty* |- (P $\vee \neg$P)
  - But the sentences need to be preprocessed into a special form
  - But all sentences can be converted into this form

Conjunctive Normal Form (CNF)

$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$$

- Replace all $\Leftrightarrow$ using biconditional elimination
- Replace all $\Rightarrow$ using implication elimination
- Move all negations inward using
  -double-negation elimination
  -de Morgan's rule
- Apply distributivity of $\vee$ over $\wedge$

Convert example sentence into CNF

$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$ starting sentence

$(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
biconditional elimination

$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$
implication elimination

$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$
move negations inward

$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$
distribute $\vee$ over $\wedge$

Resolution steps

- Given KB and β (query)
- Add $\neg \beta$ to KB, show this leads to empty (False. Proof by contradiction)
- Everything needs to be in CNF
- Example KB:
  - $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
  - $\neg B_{1,1}$
- Example query: $\neg P_{1,2}$

Resolution preprocessing

- Add $\neg \beta$ to KB, convert to CNF:

  a1: $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1})$

  a2: $(\neg P_{1,2} \vee B_{1,1})$

  a3: $(\neg P_{2,1} \vee B_{1,1})$

  b: $\neg B_{1,1}$

  c: $P_{1,2}$
- Want to reach goal: *empty*

Resolution

- Take any two clauses where one contains some symbol, and the other contains its complement (negative)

$$P \vee Q \vee R \qquad\qquad \neg Q \vee S \vee T$$

- Merge (resolve) them, throw away the symbol and its complement

$$P \vee R \vee S \vee T$$

- If two clauses resolve and there's no symbol left, you have reached *empty* (False). KB $\models \beta$
- If no new clauses can be added, KB does not entail β

Resolution example

a1: $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1})$

a2: $(\neg P_{1,2} \vee B_{1,1})$

a3: $(\neg P_{2,1} \vee B_{1,1})$

b: $\neg B_{1,1}$

c: $P_{1,2}$

Resolution example

a1: $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1})$

a2: $(\neg P_{1,2} \vee B_{1,1})$

a3: $(\neg P_{2,1} \vee B_{1,1})$

b: $\neg B_{1,1}$

c: $P_{1,2}$

Step 1: resolve a2, c: $B_{1,1}$

Step 2: resolve above and b: *empty*

Efficiency of the resolution algorithm

- Run time can be exponential in the worst case
  - Often much faster
- Factoring: if a new clause contains duplicates of the same symbol, delete the duplicates

$$P \vee R \vee P \vee T \;\Rightarrow\; P \vee R \vee T$$

- If a clause contains a symbol and its complement, the clause is a tautology and useless, it can be thrown away

a1: $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1})$

a2: $(\neg P_{1,2} \vee B_{1,1})$

$\Rightarrow P_{1,2} \vee P_{2,1} \vee \neg P_{1,2}$ (valid, throw away)