CS 337, Fall 2023
Logic

Scribes: Andreas Rosenmeier, Cheshta Damor, Samanthula Harsha Vardhan, Faiz Hashim
Siddharth Mahesh Patil, Harsh Poonia, Masada Jaswanthi, Vinay Mohan Vutukuri*
Edited by: Vishal Tapase

October 31, 2023

1 Logic

Logic is useful in addressing issues of bias and fairness in AI. Deep learning models while powerful and effective in numerous tasks are opaque and tough to interpret. Logic-based approaches, such as symbolic reasoning and rule-based systems, provide a structured and interpretable framework for representing knowledge and making inferences. The main role of logic in the context of AI is to represent knowledge, in the form of Knowledge Bases (a collection of sentences) and we want to infer from the KBs (Knowledge Bases).

Logic has 3 main heads:

1. **Syntax**: Defines well-formed or valid sentences in the KBs.
2. **Semantics**: Defines truth of a sentence α with respect to a model m (where a model is an assignment or configuration of the world of the full set of symbols we are interested in.)
   If a sentence α is true in model m, then we say m satisfies α or m is a model of α. This is how the sentence is interpreted w.r.t a model.

   $$M(\alpha) = \text{set of all models of } \alpha$$

   A KB is a conjunction of sentences

   $$M(\text{KB}) = \bigcap_{s \in \text{KB}} M(s)$$

   A KB is a set of constraints, and M(KB) is all the worlds that satisfy these constraints. If a new constraint is added to M then M(KB) shrinks.
3. **Inference**: Given a sentence, which other sentences will logically follow?

ENTAILMENT: A KB is said to entail some α (which is written as KB $\models \alpha$) if and only if $M(\text{KB}) \subseteq M(\alpha)$. In every model where KB is true, α should also be true and in every model where α is false, KB should also be false. Entailment is used to do Logical Inference.

*All scribes get equal credit for the final notes.

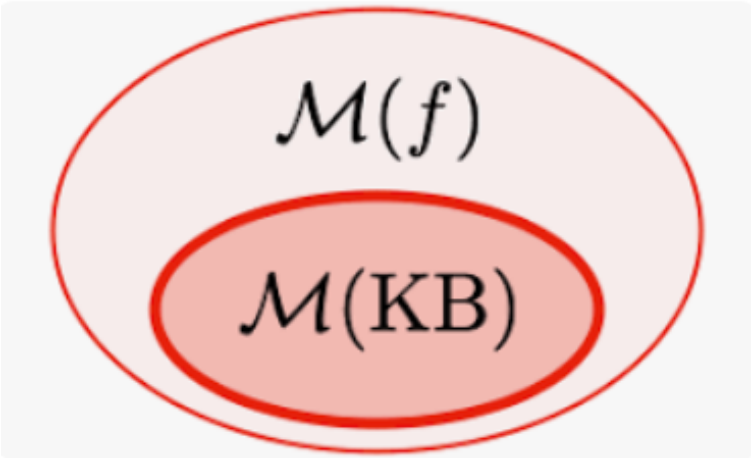

Figure 1: Image demonstrating Entailment (Here f is α)

2 Propositional Logic

2.1 Syntax

- **Atomic Sentences** contain a single propositional symbol that can take a value of either true or false. Two special atomic sentences are:
  - **TRUE**: Always true propositional symbol
  - **FALSE**: Always false propositional symbol
  - Otherwise, atomic sentences can have symbols which can take arbitrary names and they take one of two values, which are true or false.

- **Complex Sentences** are made of atomic sentences and the following five logical connectors (written in decreasing order of precedence):
  1. $\neg$ (not) Negation
  2. $\wedge$ (and) Conjunction
  3. $\vee$ (or) Disjunction
  4. $\implies$ (implies) Implication
  5. $\iff$ (if and only if) Biconditional

  eg: $\neg\alpha \vee \beta \wedge \gamma \equiv (\neg\alpha) \vee (\beta \wedge \gamma)$

2.2 Semantics

- **Truth of Atomic Sentences** can be fairly easily established:
  - **TRUE** is always True
  - **FALSE** is always False
  - Truth of other atomic sentences is with respect to the model

- **Truth of Complex Sentences** makes use of atomic sentences in conjunction with the following five rules:

1. $\neg P$ is true if and only if P is false in a model m

P	$\neg P$
T	F
F	T

2. $P \wedge Q$ is true if and only if both P and Q are true in a model m

P	Q	$P \wedge Q$
T	T	T
T	F	F
F	T	F
F	F	F

3. $P \vee Q$ is true if and only if either P or Q is true in a model m

P	Q	$P \vee Q$
T	T	T
T	F	T
F	T	T
F	F	F

4. $P \implies Q$ is true if and only if either P is false or Q is true in a model m

P	Q	$P \implies Q$
T	T	T
T	F	F
F	T	T
F	F	T

5. $P \iff Q$ is true if and only if both P and Q are either true or both P and Q are false in a model m

P	Q	$P \iff Q$
T	T	T
T	F	F
F	T	F
F	F	T

2.3 Inference

Generating the conclusions from evidence and facts is termed as Inference.

Example: Given a sentence α, does the $\boldsymbol{KB} \models \boldsymbol{\alpha}$?

Inference Rule:
An inference rule for a set of sentences $S_1, S_2, \ldots, S_m, S$ is defined as:

$$\frac{S_1, S_2, \ldots, S_m}{S} \qquad \frac{(premises)}{(conclusion)}$$

where $S_1, S_2, \ldots, S_m$ are premises, and S is the conclusion. If $S_1, S_2, \ldots, S_m$ holds, then S is implied.

The rule says that if the premises are in the KB, then you can add the conclusion to the KB.

2.3.1 Inference Methods

1) Model Checking

Based on the definition of entailment, enumerate or instantiate all models! Check if $\boldsymbol{\alpha}$ is true in all models where $\boldsymbol{KB}$ is true.

Idea:

- To test whether $\boldsymbol{KB} \models \boldsymbol{\alpha}$, enumerate all models and check truth of $\boldsymbol{KB}$ and $\boldsymbol{\alpha}$.
- $\boldsymbol{KB}$ entails $\boldsymbol{\alpha}$ if no model exists in which $\boldsymbol{KB}$ is true and $\boldsymbol{\alpha}$ is false (i.e. $(\boldsymbol{KB} \wedge \neg \boldsymbol{\alpha}$ is unsatisfiable).

2) Theorem Proving Approach

This approach makes use of Inference rules to generate local implications. Inference rules allow us to do reasoning on the formulas themselves without ever instantiating the models.

In some cases, this can be less work than model checking (i.e. generating a truth table). If you have a huge KB with lots of formulas and propositional symbols, sometimes you can draw a conclusion without instantiating the full model checking problem. This will be very important when we move to first-order logic, where the models can be infinite, and so model checking would be infeasible.

The general idea is that sentence A is what the agent knows, and sentence B is something that the agent can infer from what it knows.

2.3.2 Rules of Inference

There are various rules of inference that can be used to create proofs, i.e. chains of correct inferences.

Inference Rule	**Formula**
Modus Ponens (MP)	$\frac{\alpha \to \beta, \alpha}{\beta}$
Modus Tollens (MT)	$\frac{\alpha \to \beta, \neg\beta}{\neg\alpha}$
Hypothetical Syllogism (HS)	$\frac{\alpha \to \beta, \beta \to \gamma}{\alpha \to \gamma}$
Conjunction (CONJ)	$\frac{\alpha, \beta}{\alpha \wedge \beta}$
Simplification (SIMP)	$\frac{\alpha \wedge \beta}{\alpha}$
Addition (ADD)	$\frac{\alpha}{\alpha \vee \beta}$
Disjunctive Syllogism (DS)	$\frac{\alpha \vee \beta, \neg\alpha}{\beta}$
Resolution (RES)	$\frac{\alpha \vee \beta, \neg\beta \vee \gamma}{\alpha \vee \gamma}$
Double Negation (DN)	$\frac{\alpha}{\neg\neg\alpha}$
Converse (CONV)	$\frac{\alpha \to \beta}{\neg\beta \to \neg\alpha}$
Contrapositive (CONTRA)	$\frac{\alpha \to \beta}{\neg\beta \to \neg\alpha}$
Biconditional Introduction (BI)	$\frac{\alpha \to \beta, \beta \to \alpha}{\alpha \leftrightarrow \beta}$
Biconditional Elimination (BE)	$\frac{\alpha \leftrightarrow \beta}{\alpha \to \beta, \beta \to \alpha}$

1) Modus Ponens Rule

The Modus Ponens rule is one of the most important rules of inference, and it states that if α and $\boldsymbol{\alpha} \rightarrow \boldsymbol{\beta}$ is true, then we can infer that $\boldsymbol{\beta}$ will be true. It can be represented as:

$$\frac{\boldsymbol{\alpha} \rightarrow \boldsymbol{\beta}, \boldsymbol{\alpha}}{\boldsymbol{\beta}}$$

Example:

Statement-1: "If I am sleepy then I go to bed" :- $\boldsymbol{\alpha} \rightarrow \boldsymbol{\beta}$

Statement-2: "I am sleepy" :- $\boldsymbol{\alpha}$

Conclusion: "I go to bed." :- $\boldsymbol{\beta}$

Hence, we can say that, if $\boldsymbol{\alpha} \rightarrow \boldsymbol{\beta}$ is true and $\boldsymbol{\alpha}$ is true then $\boldsymbol{\beta}$ will be true.

NOTE Modus Ponens rule is sound but not complete.

Example:

Starting point:

KB = {Rain, Rain $\rightarrow$ Wet, Wet $\rightarrow$ Slippery}

Apply modus ponens to Rain and Rain $\rightarrow$ Wet:

KB = {Rain, Rain $\rightarrow$ Wet, Wet $\rightarrow$ Slippery, Wet}

Apply modus ponens to Wet and Wet $\rightarrow$ Slippery:

KB = {Rain, Rain $\rightarrow$ Wet, Wet $\rightarrow$ Slippery, Wet, Slippery}

Note that Wet and Slippery are derived by the KB. Hence modus ponens is sound. But there are some formulas which cannot be derived, $\neg$Wet and Rain $\rightarrow$ Slippery. Hence, modus ponens is not complete.

2) Modus Tollens Rule

The Modus Tollens rule state that if $\boldsymbol{\alpha} \rightarrow \boldsymbol{\beta}$ is true and $\neg\boldsymbol{\beta}$ is true, then $\neg\boldsymbol{\alpha}$ will also true. It can be represented as:

$$\frac{\boldsymbol{\alpha} \rightarrow \boldsymbol{\beta}, \neg\boldsymbol{\beta}}{\neg\boldsymbol{\alpha}}$$

Example:

Statement-1: "If I am sleepy then I go to bed" :- $\boldsymbol{\alpha} \rightarrow \boldsymbol{\beta}$

Statement-2: "I do not go to the bed" :- $\neg\boldsymbol{\beta}$

Conclusion: "I am not sleepy" :- $\neg\boldsymbol{\alpha}$

2.4 Logical Equivalences

Statements p and q are said to be **logically equivalent** if they have the same truth value in every model. The logical equivalence of p and q is expressed as $p \equiv q$.

The following illustrates a few general logical equivalences:

- $p \wedge q \equiv q \wedge p$ — commutativity of $\wedge$
- $p \vee q \equiv q \vee p$ — commutativity of $\vee$
- $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$ — associativity of $\wedge$
- $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$ — associativity of $\vee$
- $\neg(\neg p) \equiv p$ — double-negation elimination
- $p \implies q \equiv \neg q \implies \neg p$ — contraposition
- $p \implies q \equiv \neg p \vee q$ — implication elimination
- $p \iff q \equiv (p \implies q) \wedge (q \implies p)$ — biconditional elimination
- $\neg(p \wedge q) \equiv \neg p \vee \neg q$ — De Morgan's
- $\neg(p \vee q) \equiv \neg p \wedge \neg q$ — De Morgan's
- $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ — distributivity of $\vee$ over $\wedge$
- $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$ — distributivity of $\wedge$ over $\vee$

2.5 Properties of Inference Rules/Algorithms

Soundness

An inference algorithm is said to be **sound** if it infers entailed sentences only.
In other words, if the algorithm derives a statement from a set of premises, that statement is guaranteed to be true whenever the premises are true. A sound inference algorithm ensures that it does not derive false conclusions from true premises.

Completeness

An inference algorithm is said to be **complete** if it infers all entailed sentences.
In other words, a complete inference algorithm guarantees that if a valid conclusion can be drawn from a set of premises, the algorithm will eventually find a way to derive it.