COPYRIGHT NOTICE / THÔNG BÁO BẢN QUYỀN

VNU-HUS MAT1206E/3508: Introduction to AI

Propositional Logic
In-class Discussion

Hoàng Anh Đức

Bộ môn Tin học, Khoa Toán-Cơ-Tin học
Đại học KHTN, ĐHQG Hà Nội
hoanganhduc@hus.edu.vn

Contents

Introduction

- Logic is one of the oldest fields in AI. It was the dominant AI method from the 1950s to the 1980s (remember we talked about symbolic AI).
- Machine learning has established itself and is now the dominant AI method in every field of application. By contrast, logic no longer plays a significant role in AI.
- Nevertheless, logic remains important for understanding AI's foundations and for domains that require explicit, interpretable, and verifiable reasoning.
  - Planning systems for service robots.
  - Autonomous driving.
  - The connection between symbolic representation of knowledge in predicate logic and the implicit sub-symbolic knowledge gathered from sensors remains interesting. This is a promising application for automatic feature extraction using Deep Learning.
- We discuss propositional logic, the simplest form of logic, and its use in knowledge representation and reasoning.

Basics of Propositional Logic
A general picture

Language **S**emantics Inference **R**ules

$$(\mathcal{L}, \mathcal{S}, \mathcal{R})$$

Propositional Logic Formulas (Syntax)

- Propositional Variables Σ
- Logical Constants $\mathbf{t}$, $\mathbf{f}$
- Logical Operators Op: $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow, (,)$

Propositional Logic Formulas (Semantics)

- Truth Assignment $I : \Sigma \to \{t, f\}$
- Logical Constants $I(\mathbf{t}) = t$, $I(\mathbf{f}) = f$
- Logical Operators Op: Truth table for each operator

Propositional Logic Formulas (Premise) $\xrightarrow{\text{Derivation}}$ Propositional Logic Formula (Conclusion)

(Applying inference rules)

Exercise 1

Are you familiar with the following concepts? Review them if needed.

(1) Propositional formulas

(2) Truth assignments

(3) Semantically equivalent formulas

(4) Satisfiable, (logically) valid, unsatisfiable formulas

(5) A model of a formula

(6) KB (knowledge base – a collection of formulas) entails Q (query – a formula) (or Q follows (semantically) from KB, or $KB \models Q$)

(7) $KB \models Q$ and $\models KB \Rightarrow Q$

(8) Proof by model checking

(9) Conjunctive Normal Form (CNF) (conjunction, disjunction, literal, clause) and how to convert a formula to CNF

Premise $\vdash$ Conclusion (= "Conclusion follows from Premise syntactically")

Inference Rule

Propositional Logic Formulas (Premise) — $\dfrac{\text{Premise}}{\text{Conclusion}}$ $\rightarrow$ Propositional Logic Formula (Conclusion)

$\{A, A \Rightarrow B\} \vdash B$ or $A \land (A \Rightarrow B) \vdash B$

Modus Ponens (MP)

$A,\ A \Rightarrow B$ — $\dfrac{A,\ A \Rightarrow B}{B}$ $\rightarrow$ B

$\{A \lor B, \neg B \lor C\} \vdash A \lor C$ or $(A \lor B) \land (\neg B \lor C) \vdash (A \lor C)$

Resolution (Res)

$A \lor B,\ \neg B \lor C$ — $\dfrac{A \lor B,\ \neg B \lor C}{A \lor C}$ $\rightarrow$ $A \lor C$

Premise $\vdash$ Conclusion (= "Conclusion follows from Premise syntactically")

Inference Rules

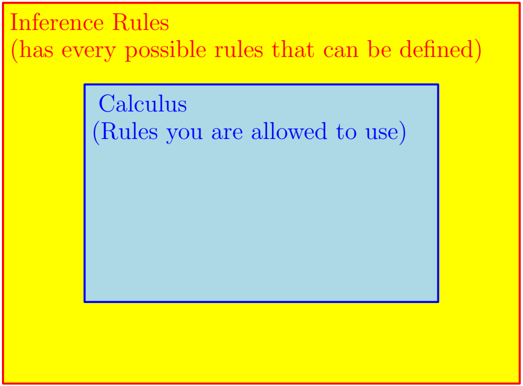
Propositional Logic Formulas (Premise) — $\boxed{}\ \boxed{}\ \boxed{} \cdots\cdots \boxed{}$ $\rightarrow$ Propositional Logic Formula (Conclusion)

Aristotle's Famous Syllogism[a]

$$\frac{\begin{array}{c}\text{All men are mortal}\\ \text{Socrates is a man}\end{array}}{\text{Socrates is mortal}}$$

[a] The term comes from the Greek word *syllogismos*, meaning "conclusion" or "inference." In a valid syllogism, if the premises are true, the conclusion must also be true. In Vietnamese, it is called "Tam đoạn luận" (three-part reasoning).

Figure: Aristotle (384–322 BC). His work (the *Organon* book) is considered as the earliest systematic study of logic. Image taken from Wikipedia.



A calculus is

- *sound* if for any two formulas KB and Q, if $KB \vdash Q$ then $KB \models Q$;
- *complete* if for any two formulas KB and Q, if $KB \models Q$ then $KB \vdash Q$.

If a calculus is *both sound and complete*, then *syntactic derivation and semantic entailment are two equivalent relations*

$$
\begin{tikzcd}[column sep=huge, row sep=huge]
KB \arrow[r, "\vdash", "\text{derivation}"'] \arrow[d, "\substack{\text{interpretation}\\ \text{(assignment)}}"'] & Q \arrow[d, "\substack{\text{interpretation}\\ \text{(assignment)}}"] \\
Mod(KB) \arrow[r, "\models", "\text{entailment}"'] & Mod(Q)
\end{tikzcd}
$$

syntactic level
(formulas)

semantic level
(interpretations/assignments)

Figure: Syntactic derivation vs. Semantic entailment

Recall: We talked about the idea of "the process of human thought can be *mechanized*".

Goal

We want to show that a knowledge base KB entails a query Q, i.e., $KB \models Q$.
Note: KB must be *consistent*, meaning that it does not contain any pair of formulas $\{P, \neg P\}$. **[What if it is not?]**

Proof System 1: Model checking

To show that $KB \models Q$, we can use a model checking algorithm. This involves checking all assignments (interpretations) of the propositional variables in KB and Q. If in every model where KB is true, Q is also true, then $KB \models Q$.

- **Pros:** Simple and straightforward.
- **Cons:** Very large computation in the worst case — 2^n assignments for n propositional variables.

Goal

We want to show that a knowledge base KB entails a query Q, i.e., $KB \models Q$.
Note: KB must be *consistent*, meaning that it does not contain any pair of formulas $\{P, \neg P\}$. **[What if it is not?]**

Proof System 1: Model checking

To show that $KB \models Q$, we can use a model checking algorithm. This involves checking all assignments (interpretations) of the propositional variables in KB and Q. If in every model where KB is true, Q is also true, then $KB \models Q$.

Proof System 2: A sound and complete calculus $\mathcal{C}$

To show that $KB \models Q$, we show $KB \vdash Q$ using the calculus $\mathcal{C}$. The soundness and completeness of $\mathcal{C}$ guarantee that $KB \vdash Q$ if and only if $KB \models Q$.

Let's discuss how we construct Proof System 2.

Note

We can assume that KB is in Conjunctive Normal Form (CNF). **[Why?]**

Let's discuss how we construct Proof System 2.

Note

We can assume that KB is in Conjunctive Normal Form (CNF). **[Why?]**

Can we use only Modus Ponens in $\mathcal{C}$?

No. Modus Ponens is sound but not complete. **[Why?]**

Let's discuss how we construct Proof System 2.

Note

We can assume that KB is in Conjunctive Normal Form (CNF). **[Why?]**

Can we use only Modus Ponens in $\mathcal{C}$?

No. Modus Ponens is sound but not complete. **[Why?]**

Can we use only Resolution in $\mathcal{C}$?

No. Resolution is sound but not complete. **[Why?]**

Let's discuss how we construct Proof System 2.

Note

We can assume that KB is in Conjunctive Normal Form (CNF). **[Why?]**

Can we use only Modus Ponens in $\mathcal{C}$?

No. Modus Ponens is sound but not complete. **[Why?]**

Can we use only Resolution in $\mathcal{C}$?

No. Resolution is sound but not complete. **[Why?]**

Can we use both Modus Ponens and Resolution in $\mathcal{C}$?

No. Resolution is a generalization of Modus Ponens **[Why?]**, and using both does not provide any additional power.

What now?

One approach is to *combine Resolution with one or more additional inference rules* — rules that are not subsumed by Resolution — to obtain a calculus that is both sound and complete.

What now?

One approach is to *combine Resolution with one or more additional inference rules* — rules that are not subsumed by Resolution — to obtain a calculus that is both sound and complete.

There are several cases that Resolution does not work

For example, $\{A, B\} \models A \vee B$, but how do you derive $A \vee B$ from the premises A and B using only Resolution? **[Try it!]**
Another example: what happens if the premise is empty?

What now?

One approach is to *combine Resolution with one or more additional inference rules* — rules that are not subsumed by Resolution — to obtain a calculus that is both sound and complete.

There are several cases that Resolution does not work

For example, $\{A, B\} \models A \vee B$, but how do you derive $A \vee B$ from the premises A and B using only Resolution? **[Try it!]**
Another example: what happens if the premise is empty?

Luckily, we can "prove by contradiction"

$KB \models Q$ means $KB \wedge \neg Q$ is unsatisfiable. **[Why?]** Instead of showing $KB \vdash Q$, we can show that $(KB \wedge \neg Q) \vdash ()$.

What now?

One approach is to *combine Resolution with one or more additional inference rules* — rules that are not subsumed by Resolution — to obtain a calculus that is both sound and complete.

There are several cases that Resolution does not work

For example, $\{A, B\} \models A \vee B$, but how do you derive $A \vee B$ from the premises A and B using only Resolution? **[Try it!]**

Another example: what happens if the premise is empty?

Luckily, we can "prove by contradiction"

$KB \models Q$ means $KB \wedge \neg Q$ is unsatisfiable. **[Why?]** Instead of showing $KB \vdash Q$, we can show that $(KB \wedge \neg Q) \vdash ()$.

New Goal

Find a calculus $\mathcal{C}$ which allows us to derive the empty clause from any unsatisfiable set of clauses.

But ...

Unfortunately, *Resolution alone is not sufficient to derive the empty clause from every unsatisfiable set of clauses*. **[Why?]** (**Hint:** Consider the Premise with two clauses $A \lor A$ and $\neg A \lor \neg A$.)

We need to add more inference rules to our calculus to make it complete.

But ...

Unfortunately, *Resolution alone is not sufficient to derive the empty clause from every unsatisfiable set of clauses*. **[Why?]** (**Hint:** Consider the Premise with two clauses $A \vee A$ and $\neg A \vee \neg A$.)

We need to add more inference rules to our calculus to make it complete.

An inference rules we can use: *Factorization*

Literals that are identical in a clause can be *factored* out. For example:

$$\frac{(A \vee A \vee B)}{(A \vee B)} \qquad \frac{(A \vee A)}{A}$$

Note: Factorization is both sound and complete.

But ...

Unfortunately, *Resolution alone is not sufficient to derive the empty clause from every unsatisfiable set of clauses*. **[Why?]** (**Hint:** Consider the Premise with two clauses $A \vee A$ and $\neg A \vee \neg A$.)
We need to add more inference rules to our calculus to make it complete.

An inference rules we can use: *Factorization*

Literals that are identical in a clause can be *factored* out. For example:

$$\frac{(A \vee A \vee B)}{(A \vee B)} \qquad \frac{(A \vee A)}{A}$$

Note: Factorization is both sound and complete.

And finally, we have $\boxed{\mathcal{C} = \{\text{Resolution, Factorization}\}}$. We call $\mathcal{C}$ the *resolution calculus*. Resolution calculus is both sound and refutation-complete.

In summary, to decide if $KB \models Q$ using the resolution calculus, we can use the following algorithm:

(1) Convert $KB \land \neg Q$ to CNF.

(2) Repeatedly apply the resolution and factorization rules until there is no resolvable pair of clauses.

(3) Every time the resolution rule is applied, add the resolvent to KB if it has not yet been included.

(4) If the empty clause is derived, then $KB \models Q$. Otherwise, $KB \not\models Q$.

Exercise 2

Confirm your understanding of the resolution calculus by seeing how the logic puzzles (“A charming English family” and “The High Jump”) in the textbook [Ertel 2025] are solved.

Basics of Propositional Logic
Horn Clauses

- In practice, many knowledge bases consist of *Horn clauses*, which are clauses with at most one positive literal.

Horn Clauses

$$\neg A_1 \vee \ldots \vee \neg A_m \vee B \equiv \underbrace{A_1 \wedge \ldots \wedge A_m}_{\text{body}} \Rightarrow \underbrace{B}_{\text{head}}$$

$$\neg A_1 \vee \ldots \vee \neg A_m \equiv A_1 \wedge \ldots \wedge A_m \Rightarrow \mathbf{f}$$

$$B \longleftarrow \text{fact}$$

- The full power of the resolution calculus is not needed to handle Horn clauses. One may use the (generalized) Modus Ponens rule instead of the general Resolution rule.

$$\frac{A_1 \wedge \cdots \wedge A_m, A_1 \wedge \cdots \wedge A_m \Rightarrow B}{B}$$

Two deriving algorithms:

- *Forward chaining* (data-driven reasoning): repeatedly apply the generalized Modus Ponens rule to derive new facts until either Q is derived or no new facts can be derived.
  - **Note:** In the case of large knowledge bases, however, Modus Ponens may derive many unnecessary formulas if one *begins with the wrong clauses*.
- *Backward chaining* (goal-driven reasoning): start from the query Q and work backwards to see if it can be derived from KB. In this case, the SLD resolution ("Selection rule driven linear resolution for definite clauses") is used.

$$\frac{A_1 \land \cdots \land A_m \Rightarrow B_1, B_1 \land B_2 \land \cdots \land B_n \Rightarrow \mathbf{f}}{A_1 \land \cdots \land A_m \land B_2 \land \cdots \land B_n \Rightarrow \mathbf{f}}$$

  - **Note:** In backward chaining, we always start applying inference rule to the negated query $\neg Q$ and further processing is always done on the currently derived clause

Exercise 3

(a) Confirm your understanding of forward and backward chaining by seeing the example in the textbook [Ertel 2025] of deriving `skiing` from the knowledge base using both forward chaining and backward chaining (SLD resolution).

(b) Summarize the proof systems (Model checking, Resolution calculus) and deriving algorithms (forward chaining, backward chaining) we have discussed so far. You can start by answering the following questions:

(1) What kind of formulas can they handle? (i.e., what are the restrictions on KB?)

(2) What is the goal of each system/algorithm?

(3) How do they work?

(4) What are the pros and cons?

(5) What are the running time complexities?

(6) Can proof in propositional logic go faster? Are there better algorithms? (**Hint:** What can we conclude from the Cook-Levin theorem?)

More Applications of Propositional Logic

- Theorem provers for propositional logic are part of the developer's everyday toolset in *digital technology*
  - *Verification of digital circuits*
  - *Generation of test patterns* for testing of microprocessors in fabrication
  - Special proof systems that work with binary decision diagrams (BDD) are also employed as a data structure for *processing propositional logic formulas*
- *Simple AI applications:* simple expert systems can work with discrete variables, few values, no cross-relations between variables
- *Probabilistic logic* uses propositional logic and probabilistic computation to model uncertainty

References

Ertel, Wolfgang (2025). *Introduction to Artificial Intelligence*. 3rd. Springer. DOI: 10.1007/978-3-658-43102-0.