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 many application areas. By contrast, logic now plays a *smaller role in mainstream 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) $\xrightarrow[\text{(Applying inference rules)}]{\text{Derivation}}$ Propositional Logic Formula (Conclusion)

Warm-up. We just saw the big picture: a *language* with its *syntax* (which strings are legal), its *semantics* (what they mean), and how a *conclusion* follows from premises. Before going into more details, here are the basic concepts we will build up, one per slide:

- *Propositional formulas*; *truth assignments* (interpretations)
- A *model* of a formula; *valid*, *unsatisfiable*, and *satisfiable but not valid* formulas
- *Semantically equivalent* formulas
- *Entailment*: when a question Q *follows from* what we know (KB) — defined carefully in a moment
- *Conjunctive Normal Form* (CNF): conjunction, disjunction, literal, clause; and how to convert to CNF

Syntax first. A *formula* (a well-formed string) is built from variables and the connectives $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$. Only *well-formed* strings count, exactly like arithmetic: $x+y$ is a legal expression, $3 + \times 4$ is not.

Quiz

Which of these is a well-formed formula?

(a) $A \wedge \vee B$ (b) $(A \wedge B) \Rightarrow \neg C$ (c) $\wedge\, A\, B$

A *truth assignment* (also called an *interpretation*) fixes every variable to t (true) or f (false).

Quiz

How many truth assignments are there for 3 variables A, B, C?

(a) 6 (b) 8 (c) 9

Now *semantics* — what a formula *says* about those assignments. A *model* of a formula is a world that makes the *whole* formula *true*; $Mod(F)$ collects all of them. For $A \vee B$, the world $A{=}t, B{=}f$ is a model, but $A{=}f, B{=}f$ is not.

Quiz

Over the variables $\{A, B\}$ there are 4 truth assignments. How many of them are models of $A \vee B$?

(a) 2 (b) 3 (c) 4

We label a formula by *how often* it is true: *valid* (a *tautology*) $=$ true in *every* assignment; *unsatisfiable* $=$ true in *no* assignment; *satisfiable but not valid* $=$ true in some assignments but not all.

Quiz

Which of these is *valid* (true in every assignment)?

(a) $A \vee \neg A$ (b) $A \wedge \neg A$ (c) $A \vee B$

Two formulas are *semantically equivalent* ($F \equiv G$) when they give the *same* truth value in *every* assignment — so they are interchangeable. For instance $A \Rightarrow B \equiv \neg A \vee B$.

Quiz

True or false: $\neg(A \wedge B) \equiv \neg A \wedge \neg B$?

(a) True (b) False

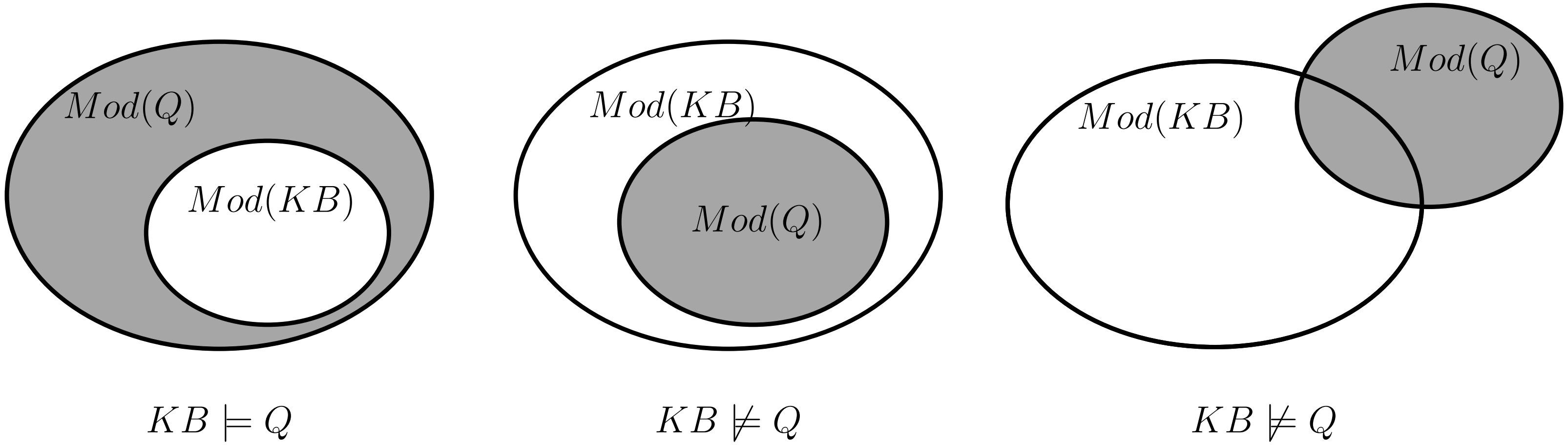
From assignments to *conclusions*. Read KB as the facts we know and Q as a question. We write $KB \models Q$ ("Q *follows from* KB") when every assignment that makes KB true also makes Q true (the *next slide* draws this). Shortcut: $KB \models Q$ holds exactly when $KB \Rightarrow Q$ is a *tautology* — the *deduction theorem*.

Quiz

Let $KB = \{A \Rightarrow B,\ A\}$ and $Q = B$. Is $\big((A \Rightarrow B) \land A\big) \Rightarrow B$ a tautology?

(a) Yes (b) No (c) only if A is true

Recall: $KB \models Q$ means *every model of KB is also a model of Q*, i.e., $Mod(KB) \subseteq Mod(Q)$ — entailment is *model containment*.



A formula is in *conjunctive normal form* (*CNF*) when it is an *and* (conjunction) of *clauses*, and each clause is an *or* (disjunction) of *literals* (a variable or its negation).

Quiz

(a) Which one is already in CNF?

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

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

(iii) $(A \Rightarrow B) \land C$

(b) Which is the CNF of $(A \Rightarrow B) \land C$?

(i) $(A \lor \neg B) \land C$

(ii) $(\neg A \lor B) \land C$

(iii) $(\neg A \land B) \land C$

Basics of Propositional Logic
Inference Rules

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

Inference Rule

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

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

Modus Ponens (MP)

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

$\{A \vee B, \neg B \vee C\} \vdash A \vee C$ or $(A \vee B) \wedge (\neg B \vee C) \vdash (A \vee C)$

Resolution (Res)

$A \vee B, \neg B \vee C$ — $\dfrac{A \vee B, \neg B \vee C}{A \vee C}$ → $A \vee C$

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

Inference Rules

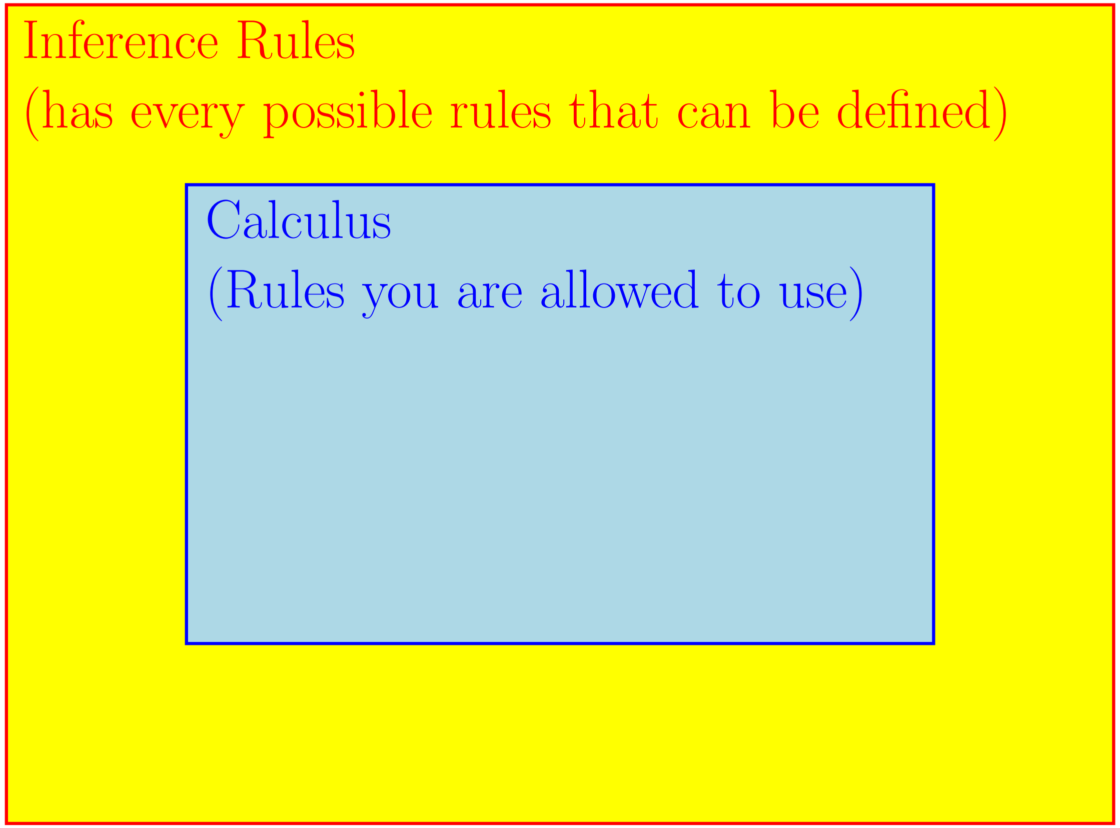
Propositional Logic Formulas (Premise) — $\boxed{}\ \boxed{}\ \boxed{} \cdots \boxed{}$ → Propositional Logic Formula (Conclusion)

Aristotle's Famous Syllogism[a]

$$\frac{\text{All men are mortal} \quad \text{Socrates is a man}}{\text{Socrates is mortal}}$$

[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 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 $KB \vdash Q$.

Basics of Propositional Logic
Calculus

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

$$\begin{tikzcd}
KB \arrow[r, "\vdash", "\text{derivation}"'] \arrow[d, "\text{interpretation}"', "\text{(assignment)}"] & Q \arrow[d, "\text{interpretation}"', "\text{(assignment)}"] & \text{syntactic level (formulas)} \\
Mod(KB) \arrow[r, "\models", "\text{entailment}"'] & Mod(Q) & \text{semantic level (interpretations/assignments)}
\end{tikzcd}$$

Figure: Syntactic derivation vs. Semantic entailment

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

Four symbols that are usually confused

- $\Rightarrow$ *inside* a formula: "if … then …", a connective like $\wedge, \vee$.
- $\models$ a statement *about* formulas: "*entails*". $KB \models Q$: every world that makes KB true makes Q true.
- $\vdash$ a statement *about* formulas: "*derives*" — reach Q by applying rules, pure symbol-pushing.
- $\equiv$ two formulas with the *same truth value in every assignment* (interchangeable).

Memory aid: $\Rightarrow$ lives *in* a formula; $\models$ and $\vdash$ talk *about* formulas ($\models$ by meaning, $\vdash$ by rules); $\equiv$ means same truth table.

Basics of Propositional Logic

Key terms in plain words (1/2)

Talking about truth

model	a world (assignment) that makes the formula *true*.
valid / *tautology*	true in *every* world.
satisfiable	true in *at least one* world.
unsatisfiable	true in *no* world.
sound	“never lies”: if it derives Q, then Q really follows.
complete	“finds everything”: if Q follows, it can derive Q.

Talking about clauses

literal	a variable or its negation: A or $\neg A$.
clause	an *or* of literals: $A \vee \neg B \vee C$.
CNF	an *and* of clauses.
empty clause $()$	nothing left $=$ a *contradiction*.
resolvent	from $(\alpha \vee \ell)$ and $(\beta \vee \neg \ell)$, Resolution cancels the pair $\ell, \neg \ell$ and builds $(\alpha \vee \beta)$.

Goal

We want to show $KB \models Q$.
Note: KB must be *consistent*, meaning that no contradiction $\varphi \wedge \neg\varphi$ can be derived from it (equivalently, KB is satisfiable). **[What if it is not?]**

Proof System 1: Model checking

To show $KB \models Q$, check *all* truth assignments of the variables in KB and Q. If Q is true in every assignment where KB is true, then $KB \models Q$.

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

Proof System 1, executed: does $KB \models Q$?

$KB = \{Study \Rightarrow Pass,\ Pass \Rightarrow Happy,\ Study\}$, $Q = Happy$. Check all $2^3 = 8$ assignments; in each row where KB is true, is Q true?

St	Pa	Ha	$St \Rightarrow Pa$	$Pa \Rightarrow Ha$	St	KB	Q
f	f	f	t	t	f	f	f
f	f	t	t	t	f	f	t
f	t	f	t	f	f	f	f
f	t	t	t	t	f	f	t
t	f	f	f	t	t	f	f
t	f	t	f	t	t	f	t
t	t	f	t	f	t	f	f
t	t	t	t	t	t	t	t

Only one row makes KB true (all t); there $Q = Happy$ is also true, so $\boxed{KB \models Q}$. We filled all $2^3 = 8$ rows.

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

To show $KB \models Q$, we instead show $KB \vdash Q$ using a 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?]**

Can we use only Modus Ponens in $\mathcal{C}$? *predict: yes / no*

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

Can we use only Resolution in $\mathcal{C}$? *predict: yes / no*

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

Can we use both Modus Ponens and Resolution in $\mathcal{C}$? *predict: yes / no*

No. Resolution *generalizes* Modus Ponens **[Why?]**

What now?

One approach is to *combine Resolution with one or more additional inference rules* — rules not subsumed by Resolution — to obtain a usable calculus.

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.

An inference rule we can use: *Factorization*

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

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

Note: Factorization is sound; it only deletes duplicate literals, so each step yields an equivalent clause.

Why Factorization is needed: $\{A \vee A,\ \neg A \vee \neg A\}$

This set is *unsatisfiable*, so a complete calculus *must* reach $()$. But binary Resolution drops *one* complementary pair and keeps the rest, so resolving two 2-literal clauses again gives a 2*-literal* clause:

$$\frac{(A \vee \underline{A}),\ (\underline{\neg A} \vee \neg A)}{(A \vee \neg A)}$$

Whatever we resolve, the only clauses ever produced are $A \vee A$, $\neg A \vee \neg A$, $A \vee \neg A$ — *never a unit*, hence never $()$. Resolution alone is *stuck*.

Factor first, then resolve

To reach $()$ we must *shrink* a clause — exactly what Factorization does:

$$\frac{(A \vee A)}{(A)} \qquad \frac{(\neg A \vee \neg A)}{(\neg A)} \qquad \text{then resolve} \qquad \frac{(A),\ (\neg A)}{()}$$

Now $()$ is reached. **Factorization was the missing piece.**

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.

The whole argument in one place

(i) Resolution *generalizes* Modus Ponens: the resolvent of $(\neg A \vee B)$ and (A) is (B) — the MP conclusion. So MP adds *no* power.

(ii) But the calculus is *not complete*: it cannot derive every entailed formula directly (e.g. $A \vee B$ from $\{A, B\}$). It is, however, *refutation*-complete — from any *unsatisfiable* set it derives the empty clause $()$.

(iii) So instead of deriving Q, we *refute* $\neg Q$: since $KB \models Q$ iff $KB \wedge \neg Q$ is unsatisfiable, we add $\neg Q$ and drive the clauses to $()$.

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

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

(2) Repeatedly apply the resolution and factorization rules.

(3) After a resolution step, add the resolvent to KB if it is new; after a factorization step, add the factorized clause if it is new.

(4) If the empty clause is derived, then $KB \models Q$. Otherwise, if no new clause can be produced, $KB \not\models Q$.

Exercise 1

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.

The algorithm, executed: prove $KB \models Q$ by refutation

Same $KB = \{Study \Rightarrow Pass,\ Pass \Rightarrow Happy,\ Study\}$, $Q = Happy$.

Step 1 — CNF of $KB \wedge \neg Q$ (each $X \Rightarrow Y$ becomes $\neg X \vee Y$; add $\neg Q$):

$$\underbrace{(\neg Study \vee Pass)}_{1} \wedge \underbrace{(\neg Pass \vee Happy)}_{2} \wedge \underbrace{(Study)}_{3} \wedge \underbrace{(\neg Happy)}_{4}$$

Step 2 — resolve, cancelling the complementary literal each time:

$$\begin{aligned}
\text{Res}(1,3) &: (\neg Study \vee Pass),\ (Study) \ \rightsquigarrow\ (Pass)_5 \\
\text{Res}(2,5) &: (\neg Pass \vee Happy),\ (Pass) \ \rightsquigarrow\ (Happy)_6 \\
\text{Res}(4,6) &: (\neg Happy),\ (Happy) \ \rightsquigarrow\ ()
\end{aligned}$$

We derived the empty clause $()$, so $KB \wedge \neg Q$ is unsatisfiable. Therefore $KB \models Q$ — matching the truth-table verdict.

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

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

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

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

- For the Horn-clause knowledge bases considered here, the full power of the resolution calculus is not needed. 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, *SLD resolution*[1] is used.

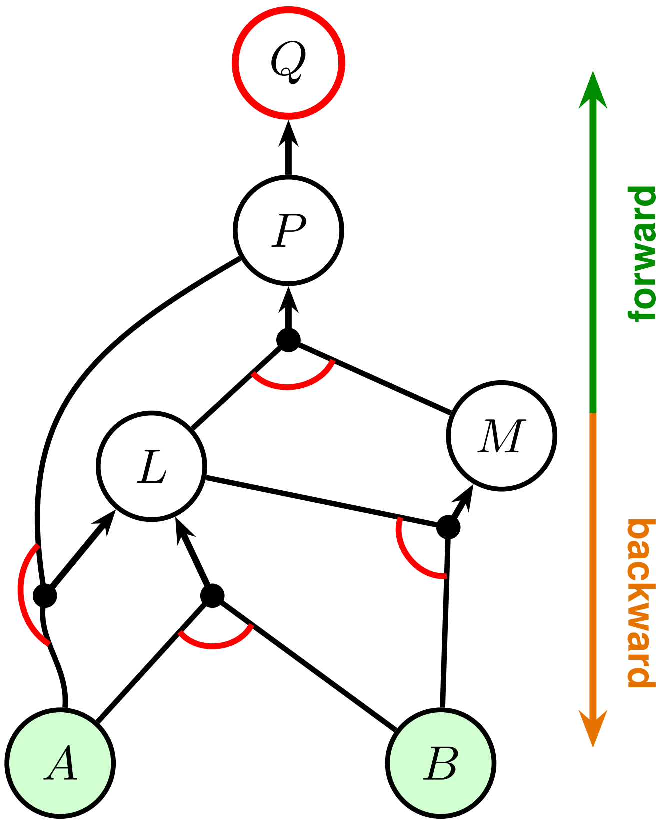
$$\frac{A_1 \wedge \cdots \wedge A_m \Rightarrow B_1, B_1 \wedge B_2 \wedge \cdots \wedge B_n \Rightarrow \mathbf{f}}{A_1 \wedge \cdots \wedge A_m \wedge B_2 \wedge \cdots \wedge B_n \Rightarrow \mathbf{f}}$$

  - **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

[1] SLD $=$ "Selection rule driven Linear resolution for Definite clauses"; a *definite clause* is a Horn clause with *exactly one* positive literal.

A worked example [Russell and Norvig 2021], Fig. 7.16:

$$A \quad B \quad \text{(facts)}$$
$$A \wedge B \Rightarrow L$$
$$A \wedge P \Rightarrow L$$
$$B \wedge L \Rightarrow M$$
$$L \wedge M \Rightarrow P$$
$$P \Rightarrow Q$$

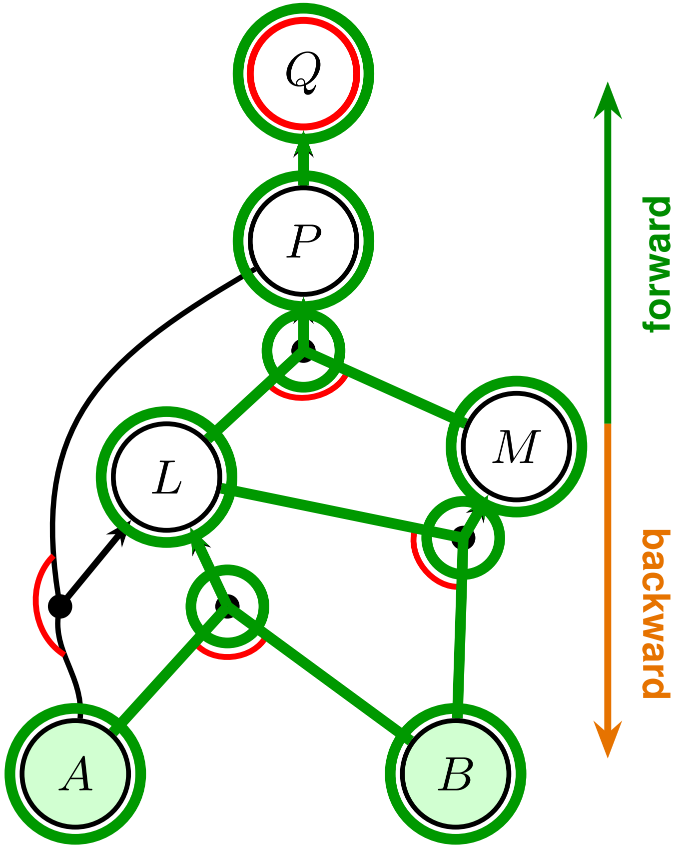


AND–OR graph [Nilsson 1980], drawn as [Russell and Norvig 2021], Fig. 7.16: a *node* per atom; each rule = a *connector* from its premises (*arc* = *and*) to its head; *several* rules into a node = an *or* (*choice*, at L); *facts* = leaves, *query* = root; $A \wedge P \Rightarrow L$ is the *back-edge* (lone *cycle*).

Forward chaining $=$ **search** (*data-driven*: set the facts, fire *up*):

1. known: facts A, B
2. $A \land B \Rightarrow L$
3. $B \land L \Rightarrow M$
4. $L \land M \Rightarrow P$
5. $P \Rightarrow Q$ ✓

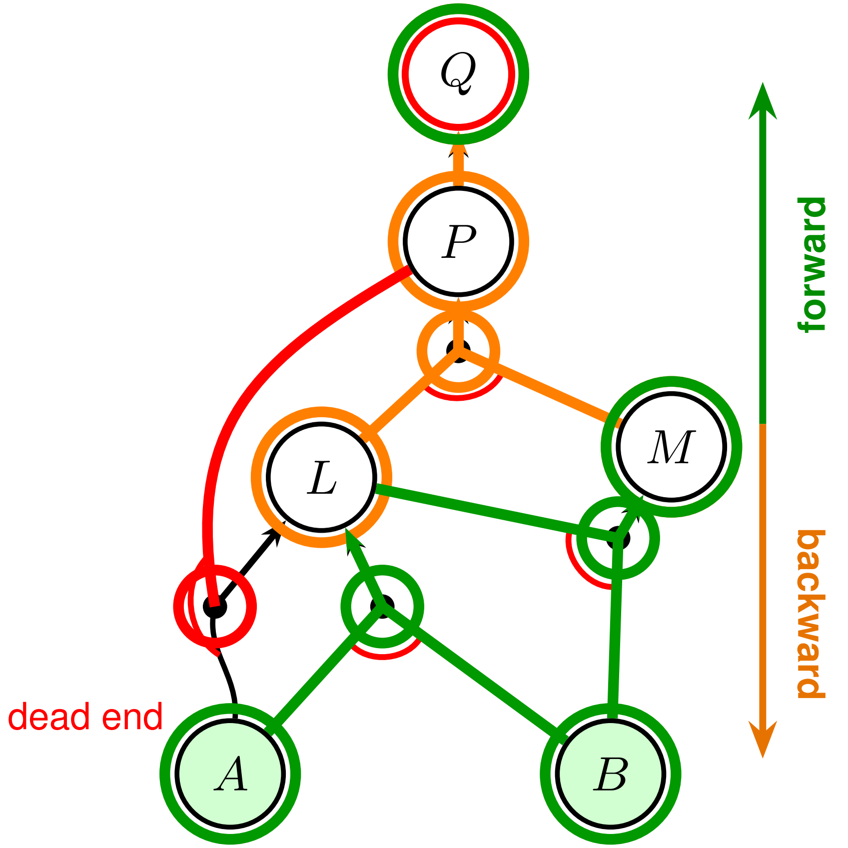
$\therefore KB \models Q$



Backward chaining = search
(goal-driven, *down* from Q on the *same* graph):

1. goal: prove Q
2. $P \Rightarrow Q$ gives subgoal P
3. $L \wedge M \Rightarrow P$ gives subgoals L, M
4. $A \wedge P \Rightarrow L$ needs open P: **dead end**
5. backtrack to $A \wedge B \Rightarrow L$: A, B facts ✓
6. $B \wedge L \Rightarrow M$: B, L known ✓

$\boxed{\therefore KB \models Q}$



Recall: $KB = \{A, B, A \wedge B \Rightarrow L, A \wedge P \Rightarrow L, B \wedge L \Rightarrow M, L \wedge M \Rightarrow P, P \Rightarrow Q\}$, Q is the query.

The same backward search as *SLD resolution* (a refutation): add the negated query $(Q \Rightarrow \mathbf{f})$ and resolve to the *empty clause* — use a rule with the subgoal as conclusion, replace it by that rule's premises, and drop facts.

$(Q \Rightarrow \mathbf{f})$	negated query
$(P \Rightarrow \mathbf{f})$	$[\,P \Rightarrow Q\,]$
$(L \wedge M \Rightarrow \mathbf{f})$	$[\,L \wedge M \Rightarrow P\,]$
$(A \wedge B \wedge M \Rightarrow \mathbf{f})$	$[\,A \wedge B \Rightarrow L\,]$ (*choice* at L: $A \wedge P$ loops)
$(M \Rightarrow \mathbf{f})$	$[\,A, B$ facts $]$
$(B \wedge L \Rightarrow \mathbf{f})$	$[\,B \wedge L \Rightarrow M\,]$
$(L \Rightarrow \mathbf{f})$	$[\,B$ fact $]$
$(A \wedge B \Rightarrow \mathbf{f})$	$[\,A \wedge B \Rightarrow L$ again $]$
$(\,)$	$[\,A, B$ facts $]$

Exercise 2

(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.
- Russell, Stuart J. and Peter Norvig (2021). *Artificial Intelligence: A Modern Approach*. 4th. Pearson.
- Nilsson, Nils J. (1980). *Principles of Artificial Intelligence*. Morgan Kaufmann.