Artificial Intelligence
(IT3160E)

Than Quang Khoat

khoattq@soict.hust.edu.vn

School of Information and Communication Technology
Hanoi University of Science and Technology

2022

Content:

- Introduction of Artificial Intelligence
- Intelligent agent
- Problem solving: Search, Constraint satisfaction
- **Logic and reasoning**
- Knowledge representation
- Machine learning

Logic

- **Logics** are formal languages for representing information such that conclusions can be drawn
- Logic = Syntax + Semantics
- **Syntax** (cú pháp) defines the sentences in the language
- **Semantics** (ngữ nghĩa) define the "meaning" of sentences
  - I.e., define the truth of a sentence in a world
- Example: The language of arithmetic
  - $(x+2 \geq y)$ is a sentence; $(x+y > \{\})$ is not a sentence
  - $(x+2 \geq y)$ is true iff the number $(x+2)$ is not less than the number y
  - $(x+2 \geq y)$ is true in a world where $x=7$, $y=1$
  - $(x+2 \geq y)$ is false in a world where $x=0$, $y=6$

Syntax

- Syntax = Language + Proof theory
- **Language**
  - Defines legal symbols, expressions, terms, formulas
  - E.g., *one plus one equal two*
- **Proof theory**
  - A set of inference rules that allow to prove (i.e., reason) expressions
  - E.g., Inference rule: *any plus zero* ⊢ *any*
- **Theorem** is a logical expression to be proven
- Proving a theorem does not need to determine the interpretation of symbols!

Semantics

- Semantics = Interpretation of symbols
- Examples:
  - $\mathcal{I}$(*one*) means **1** ($\in$ N)
  - $\mathcal{I}$(*two*) means **2** ($\in$ N)
  - $\mathcal{I}$(*plus*) means addition **+** : $N \times N \to N$
  - $\mathcal{I}$(*equal*) means equal comparison **=** : $N \times N \to$ {*true*, *false*}
  - $\mathcal{I}$(*one plus one equal two*) means *true*
- If an interpretation of an expression is true, we say that the interpretation is a **model** of the expression
- If every interpretation of an expression is true, we say that the expression is **valid**
  - Example: A OR NOT A

Entailment

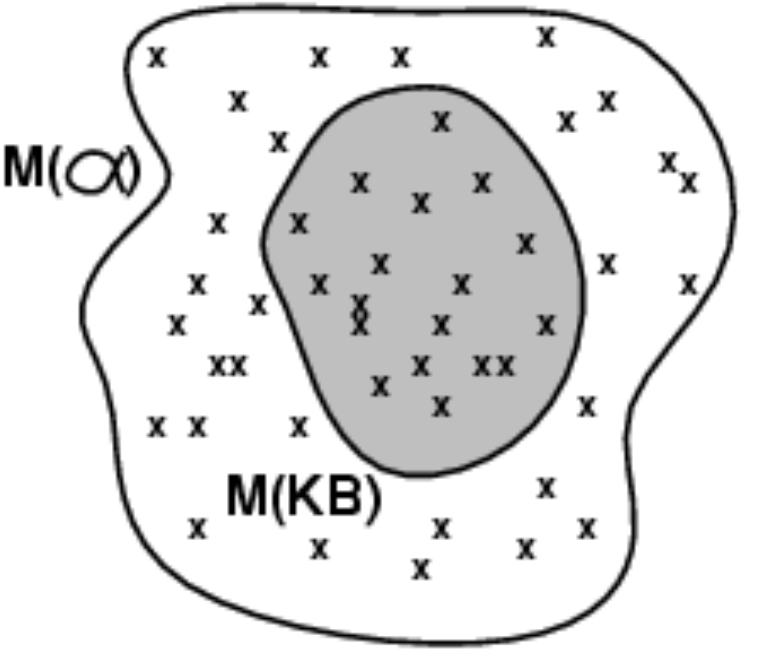
- *Entailment* means that one thing follows from another:

$$KB \models \alpha$$

- A knowledge base *KB* **entails** sentence α if and only if α is true **in all interpretations (i.e., in all worlds)** where *KB* is true
  – In other words, if *KB* is true, then α must be also true
  - Example: If a knowledge base *KB* includes the 2 sentences "Football team A won" and "Football team B won", then *KB* entails the sentence "Football team A or football team B won"
  - Example: Sentence (x+y=4) entails sentence (4=x+y)
- Entailment is a relationship between sentences that is based on semantics

Models

- Logicians typically think in terms of models
- Models are formally structured worlds with respect to which truth can be evaluated
- Definition: *m* is **a model of** a sentence α if α is true in *m*
- *M(α)* is the set of all models of α
- $KB \models \alpha$ if and only if $M(KB) \subseteq M(\alpha)$
  - Example: *KB*= {"Football team A won", "Football team B won"}, α = "Football team A won"



Logical inference (1)

- $KB \vdash_i \alpha$
  - Sentence *α* **is derived** from *KB* by (inference) procedure *i*
  - In other words, procedure *i* **infers** sentence *α* from *KB*
- **Soundness**
  - An inference procedure *i* is **sound** if procedure *i* infers **only** entailed sentences
  - Procedure *i* is sound if whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$
  - If procedure *i* infers sentence *α*, but *α* is not entailed in *KB*, then procedure *i* is unsound

Logical inference (2)

- **Completeness**
  - An inference procedure *i* is **complete** if procedure *i* can infer **all** entailed sentences
  - Procedure *i* is complete if whenever *KB* $\models \alpha$, it is also true that *KB* $\vdash_i \alpha$
- For first-order logic
  - Expressive enough to say almost anything of interest
  - There exists a *sound* and *complete* inference procedure

Logical inference (3)

- Inference can be done at the syntax level (by proofs): **Deductive reasoning**
- Inference can be done at the semantics level (by models): **Model-based reasoning**

Logical inference (4)

- Semantic inference at level of an interpretation (model):
  - Given an expression, does a model exist?: **Satisfiability**
  - Given an expression and an interpretation, check if the interpretation is a model of the expression?: **Model checking**
- Semantic inference at level of all possible interpretations: **Validity checking**

Propositional logic: Syntax (1)

- Propositional logic is the simplest logic
- **Propositional clause** (formula)
  - Any propositional symbol (S_1, S_2, …) is a propositional clause
  - The logical constant values **true** and **false** are propositional clauses
  - If S_1 is a propositional clause, then ($\neg S_1$) is also a propositional clause (**Negation**)

Propositional logic: Syntax (2)

- **Propositional clause** (*… continued*)
  - If S_1 and S_2 are propositional clauses, then $(S_1 \wedge S_2)$ is also a propositional clause (**Conjunction**)
  - If S_1 and S_2 are propositional clauses, then $(S_1 \vee S_2)$ is also a propositional clause (**Disjunction**)
  - If S_1 and S_2 are propositional clauses, then $(S_1 \Rightarrow S_2)$ is also a propositional clause (**Implication**)
  - If S_1 and S_2 are propositional clauses, then $(S_1 \Leftrightarrow S_2)$ is also a propositional clause (**Equivalence**)
  - Nothing else (apart from the above forms) is a propositional clause

Propositional logic: Examples

- p
- q
- r
- true
- false
- $\neg p$
- $(\neg p) \wedge \text{true}$
- $\neg((\neg p) \vee \text{false})$
- $(\neg p) \Rightarrow (\neg((\neg p) \vee \text{false}))$
- $(p \wedge (q \vee r)) \Leftrightarrow (p \wedge q) \vee (p \wedge r)$

Precedence of logical operators

- The precedence of the logical operators (from high to low)
  - $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$
- Use the "()" character pair to determine priority
- Examples:
  - $p \wedge q \vee r$ is equivalent to $(p \wedge q) \vee r$, but not to $p \wedge (q \vee r)$
  - $\neg p \wedge q$ is equivalent to $(\neg p) \wedge q$, but not to $\neg(p \wedge q)$
  - $p \wedge \neg q \Rightarrow r$ is equivalent to $(p \wedge (\neg q)) \Rightarrow r$, but not to $p \wedge (\neg(q \Rightarrow r))$ or $p \wedge ((\neg q) \Rightarrow r)$

Propositional logic: Semantics (1)

- An interpretation that defines the logical (i.e., true/false) value for each propositional symbol
  - Example: Given 3 propositional symbols S_1, S_2 and S_3, let's consider an interpretation m_1 is defined as follows:

$$m_1 \equiv (S_1=\text{false},\ S_2=\text{true},\ S_3=\text{false})$$

- Given 3 propositional symbols in the above example, there are 8 possible interpretations

Propositional logic: Semantics (2)

- Semantics of an interpretation *m*:
  = Rules for evaluating the truth (i.e., true/false) values of expressions in that interpretation

  $\neg S_1$ is true, if and only if S_1 is false

  $S_1 \wedge S_2$ is true, if and only if S_1 is true <u>and</u> S_2 is true

  $S_1 \vee S_2$ is true, if and only if S_1 is true <u>or</u> S_2 is true

  $S_1 \Rightarrow S_2$ is true, if and only if S_1 is false <u>or</u> both S_1 and S_2 are true
  is false, if and only if S_1 is true <u>and</u> S_2 is false

  $S_1 \Leftrightarrow S_2$ is true, if and only if $S_1 \Rightarrow S_2$ is true <u>and</u> $S_2 \Rightarrow S_1$ is true

- Example: Given an interpretation m_1 mentioned in the previous slide:

$$\neg S_1 \wedge (S_2 \vee S_3) = \mathit{true} \wedge (\mathit{true} \vee \mathit{false}) = \mathit{true} \wedge \mathit{true} = \mathit{true}$$

Semantics of propositional logic: Example (1)

- Let's consider the interpretation $m_1 \equiv (p=true,\ q=false)$:
  - $\neg p$ is *false*
  - $\neg q$ is *true*
  - $p \wedge q$ is *false*
  - $p \vee q$ is *true*
  - $p \Rightarrow q$ is *false*
  - $q \Rightarrow p$ is *true*
  - $p \Leftrightarrow q$ is *false*
  - $\neg p \Leftrightarrow q$ is *true*

Semantics of propositional logic: Example (2)

- Let's consider the interpretation $m_2 \equiv (p=false,\ q=true)$:
  - $\neg p$ is *true*
  - $\neg q$ is *false*
  - $p \wedge q$ is *false*
  - $p \vee q$ is *true*
  - $p \Rightarrow q$ is *true*
  - $q \Rightarrow p$ is *false*
  - $p \Leftrightarrow q$ is *false*
  - $\neg p \Leftrightarrow q$ is *true*

Truth tables for logical operators

S_1	S_2	$\neg S_1$	$S_1 \land S_2$	$S_1 \lor S_2$	$S_1 \Rightarrow S_2$	$S_1 \Leftrightarrow S_2$
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

Logical equivalence

- Two sentences are **logically equivalent** if and only if they are true in same models: $\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$

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

Representation by propositional logic: Example

- Assume that we have the following propositions:
  - $p \equiv$ "It's sunny this afternoon"
  - $q \equiv$ "The weather is colder than yesterday"
  - $r \equiv$ "I will go swimming"
  - $s \equiv$ "I will go playing soccer"
  - $t \equiv$ "I will be home in the evening"
- Representation of natural language statements:
  - "It is **not** sunny this afternoon **and** the weather is colder than yesterday": $\neg p \wedge q$
  - "I will go swimming **if** it's sunny this afternoon": $p \rightarrow r$
  - "**If** I will not go swimming **then** I will go playing soccer": $\neg r \rightarrow s$
  - "**If** I will go playing soccer **then** I will be home in the evening": $s \rightarrow t$

Contradiction and Tautology

- A propositional logical expression that is false in <u>every</u> interpretation is called a **contradiction**
  - Example: $(p \wedge \neg p)$
- A propositional logical expression that is true in <u>every</u> interpretation is called a **tautology**
  - Example: $(p \vee \neg p)$

$$\neg(p \wedge q) \leftrightarrow (\neg p \vee \neg q)$$

$$\neg(p \vee q) \leftrightarrow (\neg p \wedge \neg q)$$

Satisfiable and Valid

- A propositional logical expression is **satisfiable** if the expression is true in *an interpretation*
  - Example: $A \vee B$, $A \wedge B$
- A propositional logical expression is **unsatisfiable** if *there is no interpretation* for which the expression is true
  - Example: $A \wedge \neg A$
- A propositional logical expression is **valid** if the expression is true in *every interpretation*
  - Example: *true*; $A \vee \neg A$; $A \Rightarrow A$; $(A \wedge (A \Rightarrow B)) \Rightarrow B$

Logic proving problem

- Given a knowledge base (i.e., a set of premises) *KB* and an expression α to be proved (i.e., called a theorem)
- Does the knowledge base *KB* entail (semantically) α: KB $\models \alpha$?
  - In other words, can α be inferred (i.e., proven) from the knowledge base *KB*?
- **Problem:** *Is there an inference procedure that can solve the logic proving problem in a finite number of steps?*
  - For propositional logic, the answer is yes!

Solve the logic proving problem

- Goal: To answer the question KB $\models \alpha$?
- There are 3 popular proving methods:
  - Truth table
  - The inference rules
  - Translate to the problem of satisfiability (SAT) proving
    - Proving by resolution (i.e., refutation)

Proving by truth table (1)

- Proving problem: KB $\models \alpha$?
- Check **all interpretations where the *KB* is true (i.e., all models of *KB*)** to see if α is true or false
- Truth table: List the (true/false) truth values of propositions, for all possible interpretations
  - True/false value assignments for propositional symbols

		KB		α
p	q	$p \vee q$	$p \leftrightarrow q$	$(p \vee \neg q) \wedge q$
true	true	true	true	true
true	false	true	false	false
false	true	true	false	false
false	false	false	true	false

proof

Proving by truth table (2)

- KB = $(p \vee r) \wedge (q \vee \neg r)$
- $\alpha = (p \vee q)$
- KB $\models \alpha$?

p	q	r	$p \vee r$	$q \vee \neg r$	KB	α
true	true	true	true	true	true	true
true	true	false	true	true	true	true
true	false	true	true	false	false	true
true	false	false	true	true	true	true
false	true	true	true	true	true	true
false	true	false	false	true	false	true
false	false	true	true	false	false	false
false	false	false	false	true	false	false

Proving by truth table (3)

- For propositional logic, the proving method based on truth table is *sound* and *complete*
- The computational complexity
  - Exponential function for the number (*n*) of propositional symbols: 2^n
  - But there is only a (very) small subset of the possible truth value assignments in that KB and α are true

Proving by inference rules (1)

- **Modus ponens** inference rule

$$\frac{p \to q,\ \ p}{q}$$

- **And-Elimination** inference rule

$$\frac{p_1 \wedge p_2 \wedge \ldots \wedge p_n}{p_i} \qquad (i=1..n)$$

- **And-Introduction** inference rule

$$\frac{p_1,\ p_2,\ \ldots,\ p_n}{p_1 \wedge p_2 \wedge \ldots \wedge p_n}$$

- **Or-Introduction** inference rule

$$\frac{p_i}{p_1 \vee p_2 \vee \ldots \vee p_i \vee \ldots \vee p_n}$$

Proving by inference rules (2)

- **Elimination of Double Negation** inference rule

$$\frac{\neg\neg p}{p}$$

- **Resolution** inference rule

$$\frac{p \vee q, \quad \neg q \vee r}{p \vee r}$$

- **Unit Resolution** inference rule

$$\frac{p \vee q, \quad \neg q}{p}$$

- All the above inference rules are *sound*!

Proving by inference rules: Example (1)

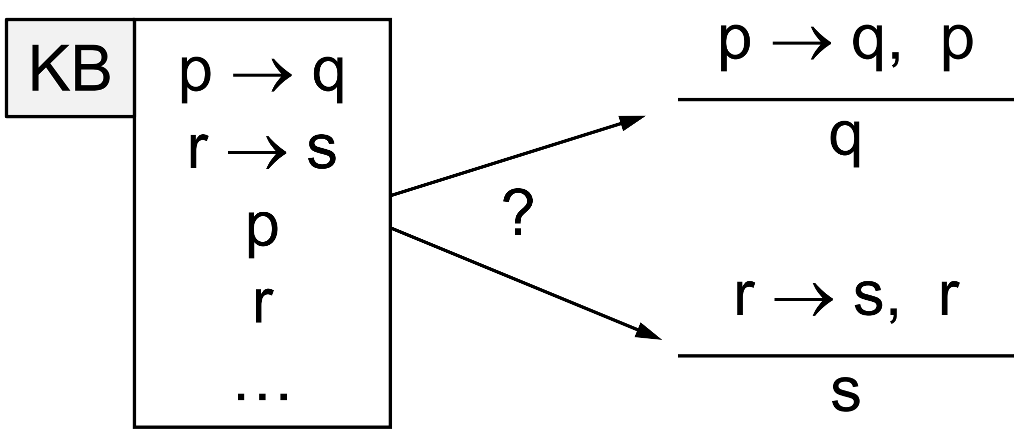
- Let's assume that we have a set of premises KB:
  1) $p \wedge q$
  2) $p \rightarrow r$
  3) $(q \wedge r) \rightarrow s$
- To prove the theorem: *s*
- From 1) and applying the And-Elimination inference rule, we have:
  4) p
- From 2), 4) and applying the Modus Ponens inference rule, we have:
  5) r

Proving by inference rules: Example (2)

- …
- From 1) and applying the And-Elimination inference rule, we have:
  6) q
- From 5), 6) and applying the And-Introduction inference rule, we have:
  7) $(q \wedge r)$
- From 7), 3) and applying the Modus-Ponens inference rule, we have:
  8) s
- So, the theorem *s* is proved true!

Logic inference and Search

- To prove that a theorem α is true given a set of premises *KB*, it is necessary to apply a proper sequence of inference rules
- Problem: At a proving step, there are several (i.e., more than 1) rules applicable
  - Which inference rule should be applied?
- This is a search problem



Logic expression conversion

- In propositional logic:
  - An expression can consist of multiple logical operators: $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$
  - An expression can consist of several other (nested) sub-expressions
- Do we need to use all of the logical operators to represent a complex expression?
  - No
  - We can rewrite (i.e., convert) a propositional logic expression into an equivalent one *containing only logical operators* $\neg$, $\wedge$, $\vee$

Normal forms

- Expressions in propositional logic can be converted to one of the normal forms
  - Helps simplify the inference (i.e., proving) process
- **Conjunctive normal form (CNF)**
  - A conjunction (i.e., AND connection) of clauses
  - Each clause is a disjunction (i.e., OR connection) of propositional symbols
  - Example: $(p \lor q) \land (\neg q \lor \neg r \lor s)$
- **Disjunctive normal form (DNF)**
  - A disjunction (i.e., OR connection) of clauses
  - Each clause is a conjunction (i.e., AND connection) of propositional symbols
  - Example: $(p \land \neg q) \lor (\neg p \land r) \lor (r \land \neg s)$

Conversion to CNF

1. Remove the logic operators $\rightarrow$ and $\leftrightarrow$, using:

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

   $(p \leftrightarrow q) \equiv ((p \rightarrow q) \wedge (q \rightarrow p)) \equiv ((\neg p \vee q) \wedge (\neg q \vee p))$

2. Move the logic operator $\neg$ to the most inner, using:

   $\neg(p \wedge q) \equiv (\neg p \vee \neg q)$

   $\neg(p \vee q) \equiv (\neg p \wedge \neg q)$

   $\neg\neg p \equiv p$

3. Convert to CNF, using the distributive rule:

   $(p \vee (q \wedge r)) \equiv (p \vee q) \wedge (p \vee r)$

Conversion to CNF: Example

Convert the following expression to CNF: $\neg(p \rightarrow q) \vee (r \rightarrow p)$

1. Remove the logic operators $\rightarrow$, $\leftrightarrow$

   $\neg(\neg p \vee q) \vee (\neg r \vee p)$

2. Move the logic operator $\neg$ to the most inner, using the DeMorgan and double negation rules

   $(p \wedge \neg q) \vee (\neg r \vee p)$

3. Use the associative and distributive rules

   $(p \vee \neg r \vee p) \wedge (\neg q \vee \neg r \vee p)$

   $= (p \vee \neg r) \wedge (\neg q \vee \neg r \vee p)$

Satisfiability (SAT) proving problem

- The goal of the satisfiability (SAT) proving problem is to determine whether an expression in conjunctive normal form (CNF) can be satisfied
  - To prove that expression is true or not
  - Example: $(p \vee q \vee \neg r) \wedge (\neg p \vee \neg r \vee s) \wedge (\neg p \vee q \vee \neg t)$

- This is a case of the constraint satisfaction problem (CSP)
  - Set of variables:
    - Propositional symbols (e.g., *p*, *q*, *r*, *s*, *t*)
    - The logic constant values (i.e., *true*, *false*)
  - Set of constraints:
    - All the clauses (connected by the AND operator) must be true
    - For each clause, at least one of the propositions must be true

Solve the SAT problem

- By the **Backtracking** method:
  - Apply the depth-first search strategy
  - For each variable (i.e., a proposition), consider possible (true/false) value assignments
  - Repeat, until all the variables are assigned values, or the value assignment of a sub-set of all variables, make **the expression false**
- **Iterative optimization methods:**
  - Start with a random assignment of true/false values to the propositional symbols
  - Change the value (i.e., true to false / false to true) for a variable
  - Heuristic: Prioritize value assignments that make more statements true
  - Use local search methods: Simulated Annealing, Walk-SAT

Logic proving problem vs. SAT problem

- **Logic proving (reasoning) problem**
  - To prove: A logic expression (theorem) α is entailed by a set of premises *KB*
  - In other words, for every interpretation in that *KB* is true, is α also true?
- **Satisfiability (SAT) problem**
  - Is there an assignment of true/false values to propositional symbols (i.e., an interpretation) such that the expression α is true?
- Connection?

$KB \models \alpha$	if and only if:
$(KB \wedge \neg\alpha)$	is **unsatisfiable**

Resolution rule (1)

- **Resolution** rule

$$\frac{p \vee q, \quad \neg q \vee r}{p \vee r}$$

- The resolution rule is applicable for logic expressions of the CNF normal form
- The resolution rule is *sound*, but *incomplete*
  - Let's consider a set of premises (i.e., knowledge base) *KB*: $(p \wedge q)$
  - To prove: $(p \vee q)$?
  - The resolution rule cannot prove it!

Resolution rule (2)

- Convert the logic proving problem to the SAT one
  - Refutation-based proving method
  - To prove a contradiction of: $(KB \wedge \neg\alpha)$
  - Equivalent to prove the entailment of: $KB \models \alpha$
- Resolution rule:
  - If the expressions in *KB* and the expression to prove α are all in the CNF normal form, then applying the resolution rule determines the unsatisfaction of $(KB \wedge \neg\alpha)$

Robinson's Resolution Algorithm

- Assume that the theorem to be proved α is false. Then, we have the expression $(\neg\alpha)$ is true
- Convert all the expressions in KB and $(\neg\alpha)$ to the CNF normal form
- Consecutively apply the resolution rule, starting by: $(KB \wedge \neg\alpha)$
  - KB is a conjunction of CNF expressions
  - Therefore, $(KB \wedge \neg\alpha)$ is also a CNF expression!
- The resolution rule application process ends when either:
  - A contradiction occurs
    - After a resolution rule application, we have an empty (i.e., contradictory) expression

$$\frac{p, \quad \neg p}{\{\}}$$

  - No new expression can be inferred (i.e., derived)

Resolution Algorithm: Example (1)

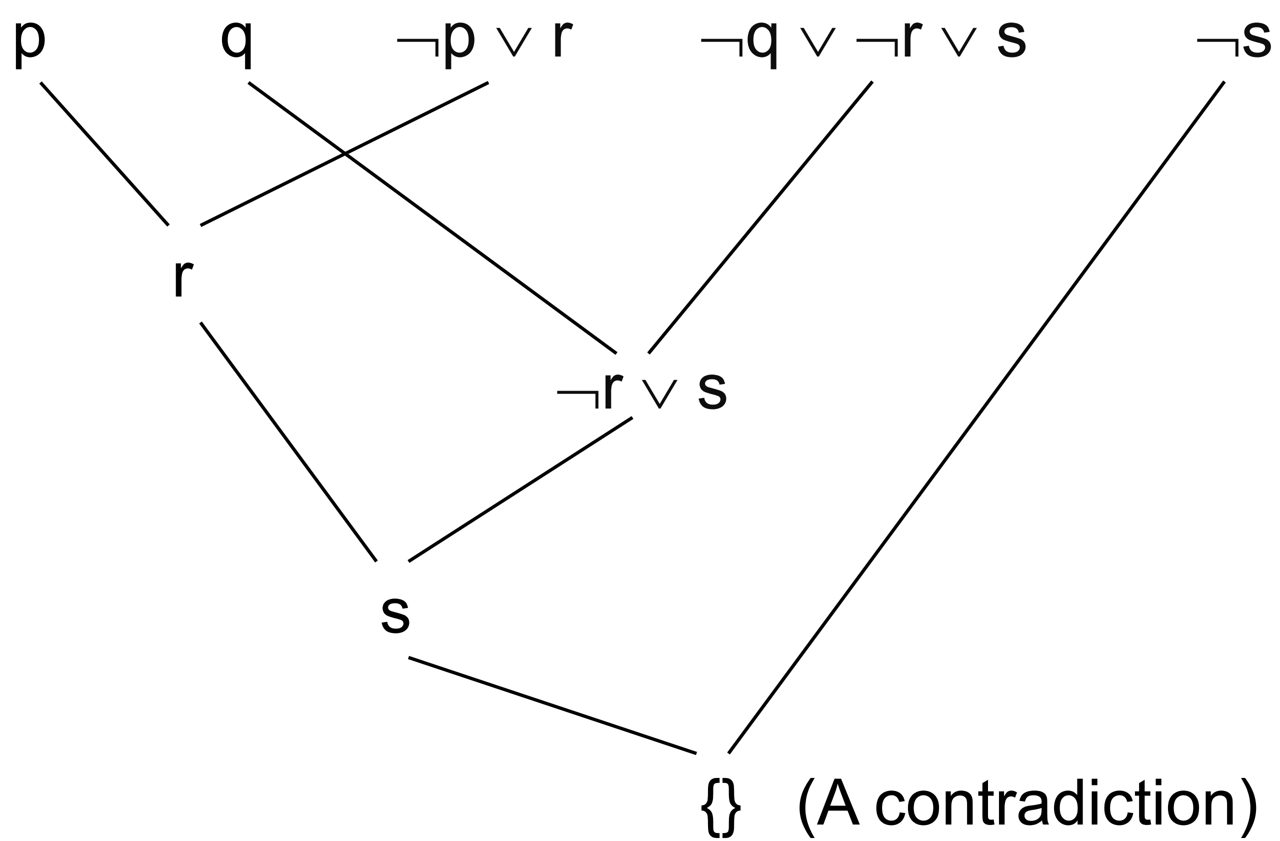
- Consider that we have a set of premises KB:
  - $p \wedge q$
  - $p \rightarrow r$
  - $(q \wedge r) \rightarrow s$
- To prove the theorem *s*
- Step 1. Assume that the theorem to be proved is false
  - $\neg s$
- Step 2. Convert all the expressions in KB to CNF
  - $(p \rightarrow r)$ is converted to $(\neg p \vee r)$
  - $((q \wedge r) \rightarrow s)$ is converted to $(\neg q \vee \neg r \vee s)$
- Step 3. Consecutively apply the resolution rule to $(KB \wedge \neg\alpha)$:

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

Resolution Algorithm: Example (2)

- At the beginning of the resolution rule application process, we have:
  1) p
  2) q
  3) $\neg p \vee r$
  4) $\neg q \vee \neg r \vee s$
  5) $\neg s$
- Resolve 1) and 3), we have
  6) r
- Resolve 2) and 4), we have
  7) $\neg r \vee s$
- Resolve 6) and 7), we have
  8) s
- Resolve 8) and 5), it results in a contradiction ({})
- It means that the theorem (*s*) is proved true

Resolution-based proving: Example (3)



Horn normal form

- An expression is in the Horn normal form if:
  - It is a conjunction (i.e., an AND combination) of clauses
  - Each clause is a disjunction (i.e., an OR combination) of literals and has a maximum of 1 (may have 0!) positive literal
  - Example: $(p \lor \neg q) \land (\neg p \lor \neg r \lor s)$
- Not all propositional expression can be converted to the Horn normal form!
- Representation of the set of premises KB in Horn normal form
  - **Rules**
    - $(\neg p_1 \lor \neg p_2 \lor \ldots \lor \neg p_n \lor q)$
    - Equivalent to the rule: $(p_1 \land p_2 \land \ldots \land p_n \rightarrow q)$
  - **Facts**
    - p, q
  - **Integrity constraints**
    - $(\neg p_1 \lor \neg p_2 \lor \ldots \lor \neg p_n)$
    - Equivalent to the rule: $(p_1 \land p_2 \land \ldots \land p_n \rightarrow \mathit{false})$

Generalized Modus Ponens rule

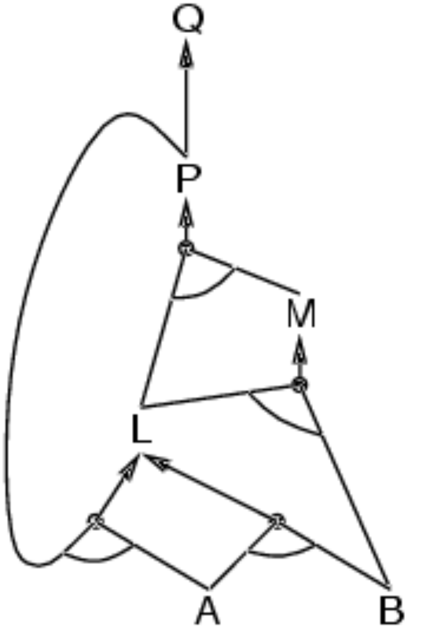
$$\frac{(p_1 \wedge p_2 \wedge \ldots \wedge p_n \rightarrow q),\ \ p_1,\ \ p_2,\ \ \ldots,\ \ p_n}{q}$$

- The Modus Ponens rule is *sound* and *complete*, given propositional symbols and the set of premises KB in Horn normal form
- The Modus Ponens rule can be used by both of the 2 reasoning approaches: Forward reasoning and Backward reasoning

Forward reasoning (chaining)

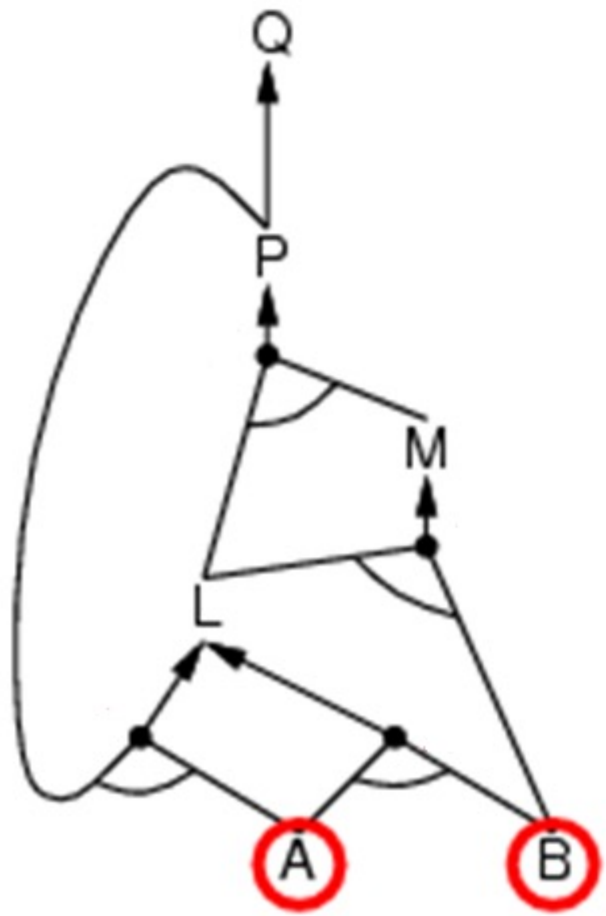
- Given a set of premises (knowledge base) *KB*, it requires to prove the expression *Q*
- Idea: Repeat the following 2 steps until inferring the expression
  - Apply a rule whose condition (IF) part is satisfied in *KB*
  - Add the applied rule's conclusion (THEN) part to *KB*

$$
\begin{aligned}
P &\Rightarrow Q \\
L \wedge M &\Rightarrow P \\
B \wedge L &\Rightarrow M \\
A \wedge P &\Rightarrow L \\
A \wedge B &\Rightarrow L \\
A & \\
B &
\end{aligned}
$$



Forward reasoning: Example (1)

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$



Forward reasoning: Example (2)

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

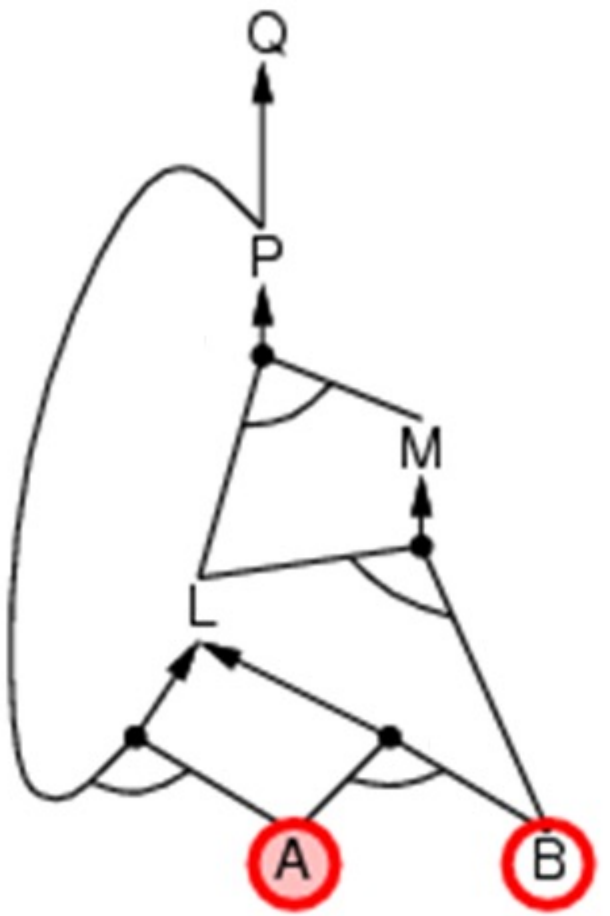
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

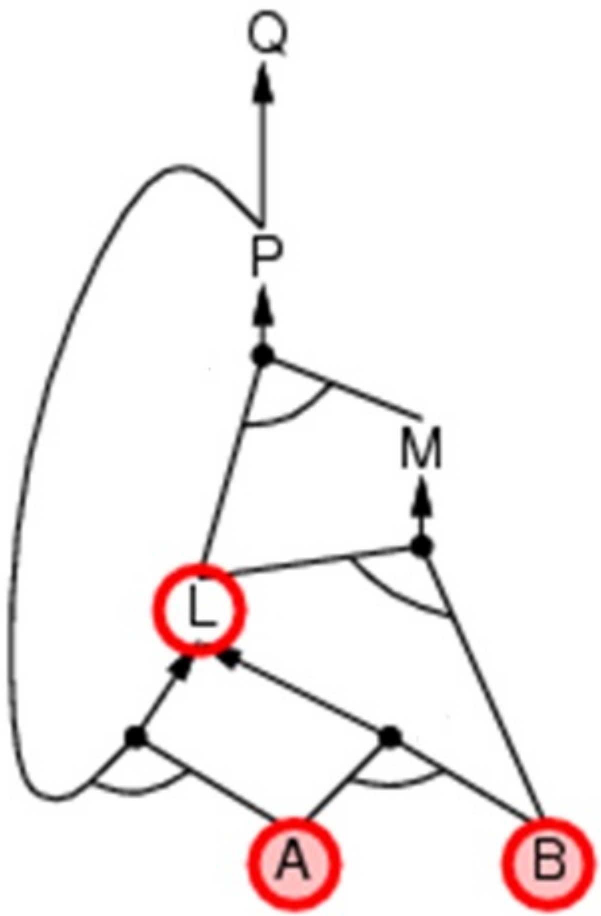
A

B



Forward reasoning: Example (3)

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$



Forward reasoning: Example (4)

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

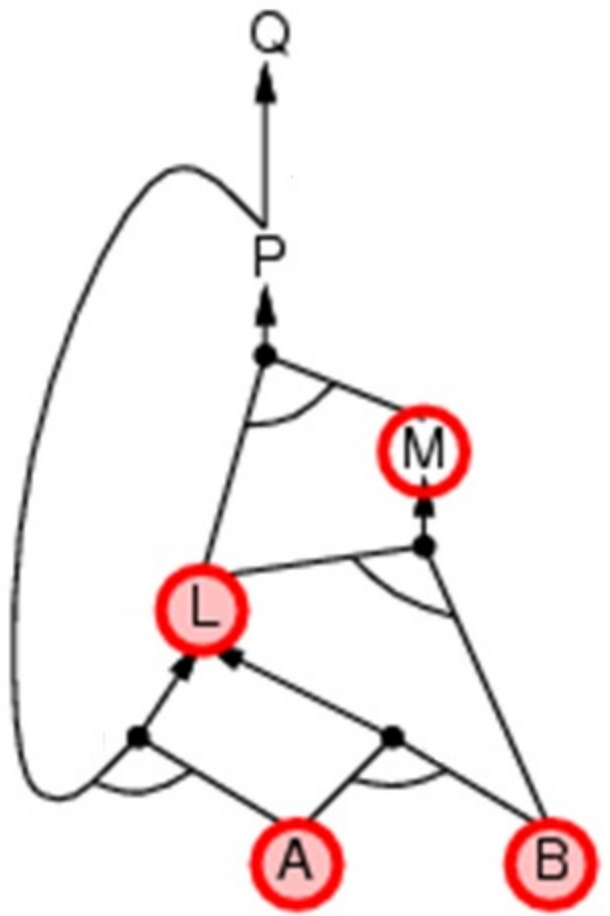
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

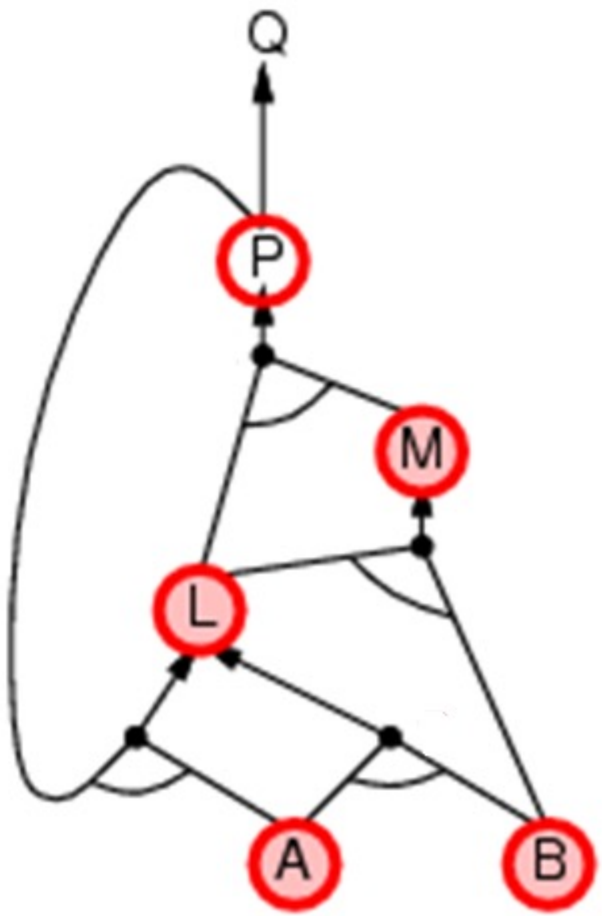
A

B



Forward reasoning: Example (5)

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$



Forward reasoning: Example (6)

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

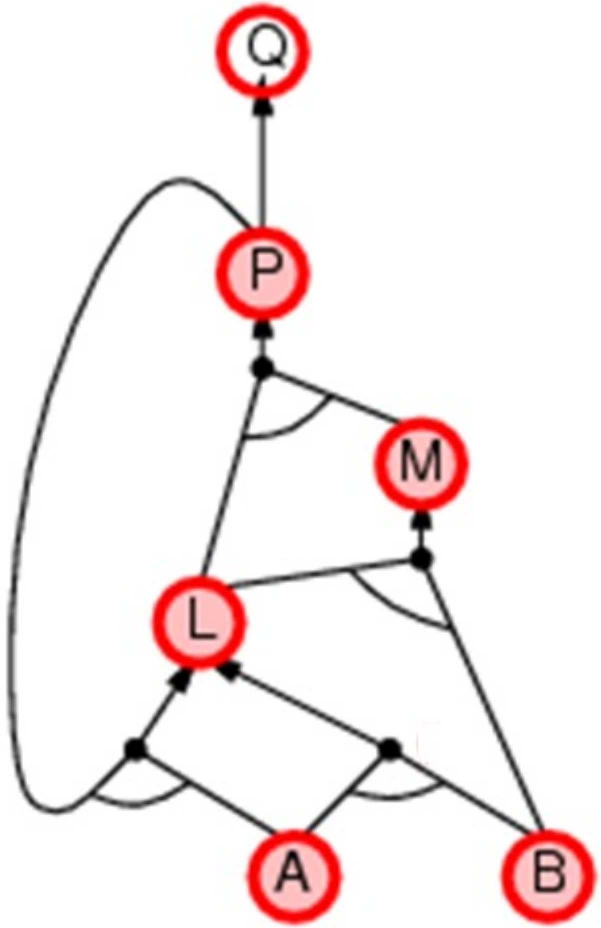
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

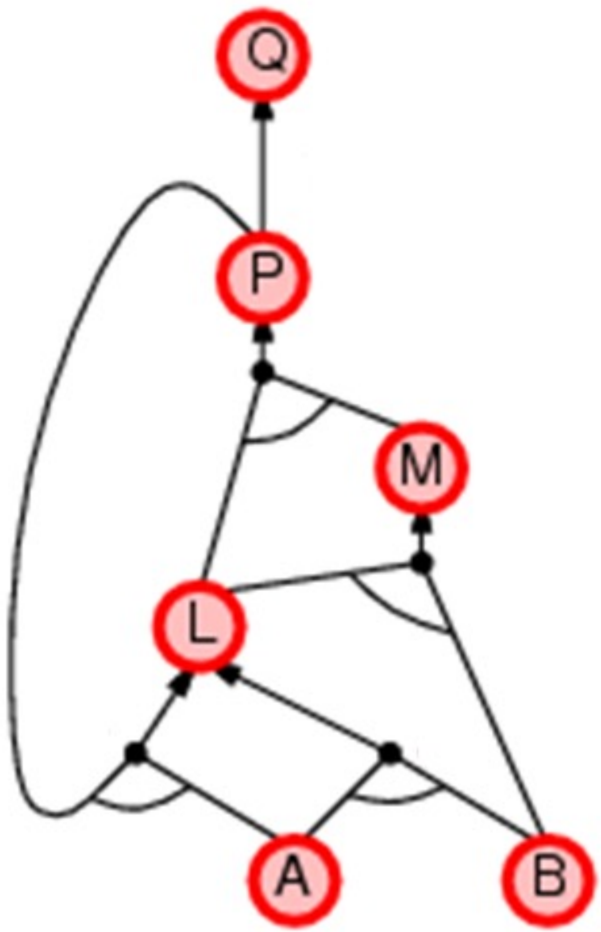
A

B



Forward reasoning: Example (7)

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

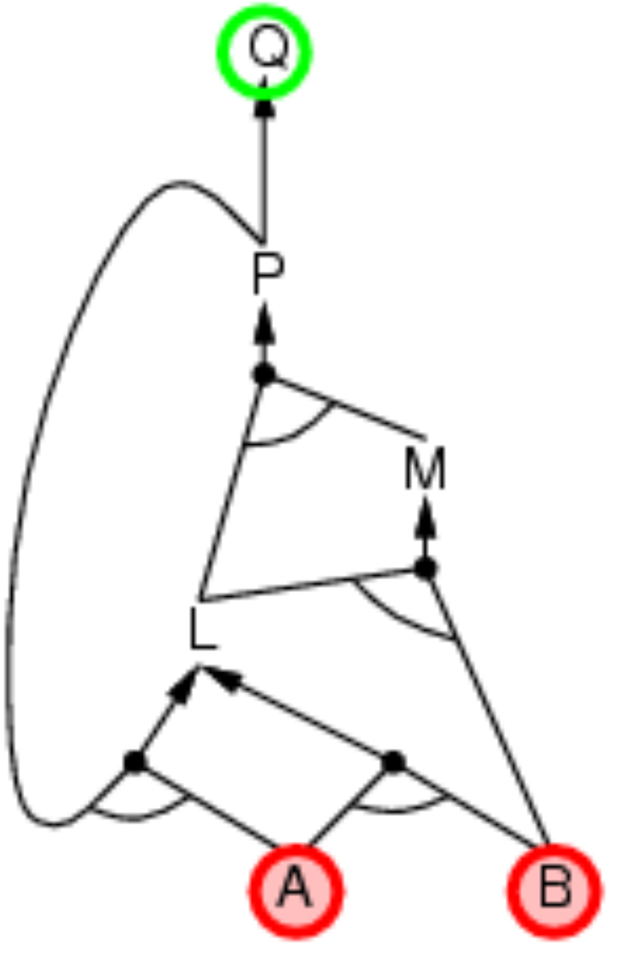


Backward reasoning (chaining)

- Idea: The reasoning process starts from the conclusion *Q*
- To prove *Q* by the set of premises (i.e., knowledge base) *KB*
  - Check if Q has been proved by *KB*,
  - If not yet, continue proving all the conditions of a rule (in *KB*) whose conclusion is *Q*
- Avoid loops
  - Check if the new expressions have been included in the list of expressions to prove? – If yes, then do not include them again!
- Avoid proving again to an expression
  - Has previously been proved true
  - Has previously been proved unsatisfiable (i.e., false) in *KB*

Backward reasoning: Example (1)

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$



Backward reasoning: Example (2)

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

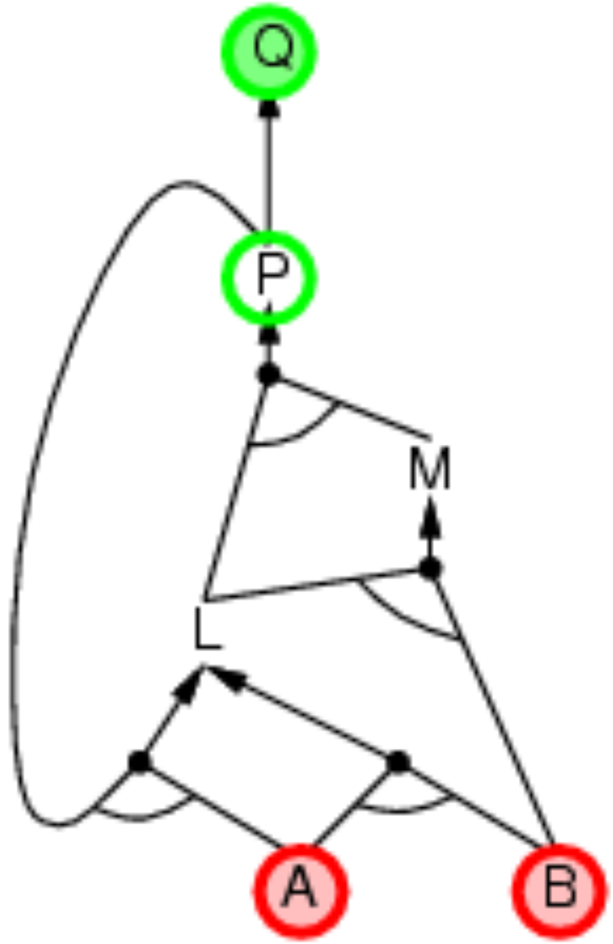
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward reasoning: Example (3)

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

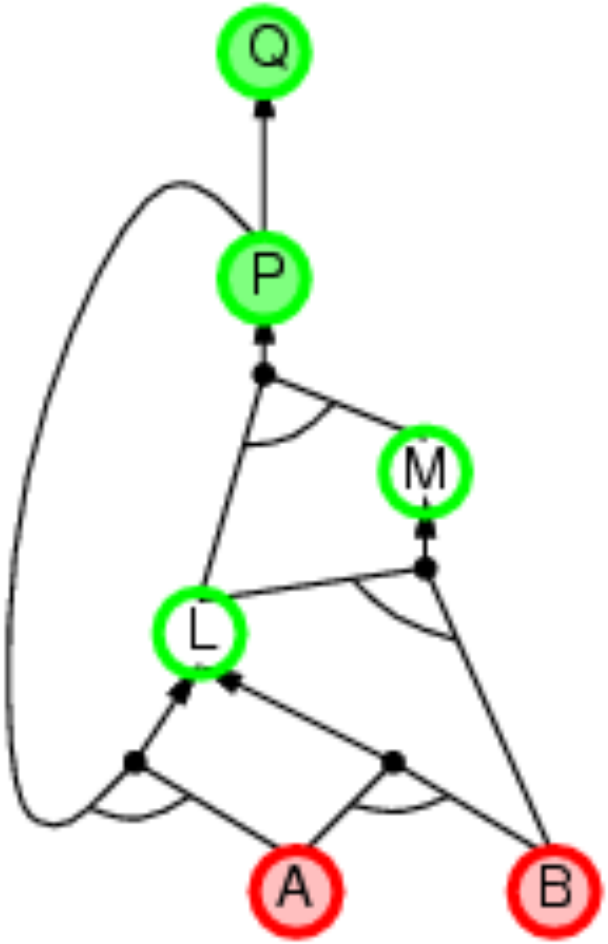
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward reasoning: Example (4)

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

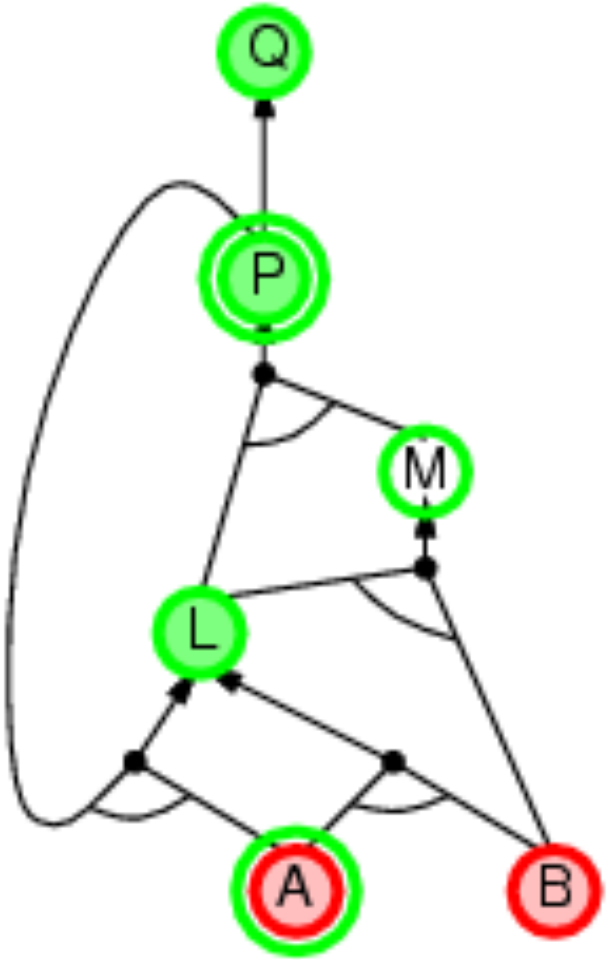
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

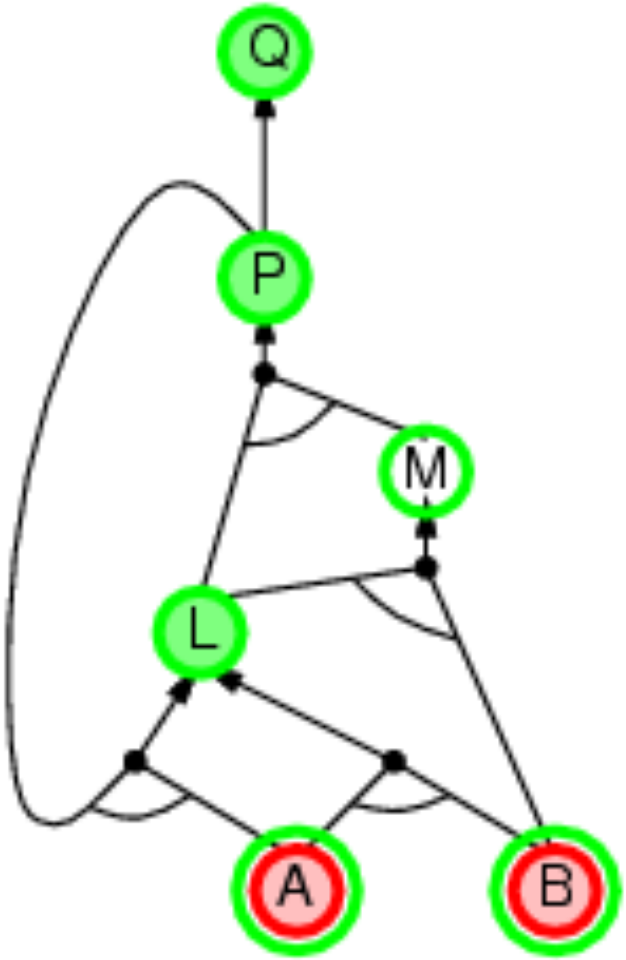
A

B



Backward reasoning: Example (5)

$P \Rightarrow Q$
$L \wedge M \Rightarrow P$
$B \wedge L \Rightarrow M$
$A \wedge P \Rightarrow L$
$A \wedge B \Rightarrow L$
A
B



Forward vs. backward reasoning?

- Forward reasoning is a data-driven process
  - Example: object recognition, decision making
- Forward reasoning may perform many redundant inference steps – irrelevant (unnecessary) to the proving goal
- Backward reasoning is a goal-driven process, suitable for problem solving

Propositional logic: Advantages and disadvantages

- (+) The propositional logic allows to easily state (represent) the knowledge base by a set of propositions
- (+) Propositional logic allows working with information in the form of negative, disjunctive
- (+) Propositional logic is structural
  - The semantics of ($S_1 \wedge S_2$) is inferred from the semantics of S_1 and the semantics of S_2
- (+) The semantics in propositional logic is context-independent
  - Unlike in natural language (meaning depends on the context of sentences)
- (-) The expressiveness ability of propositional logic is very limited
  - Propositional logic cannot express (like in natural language): “If X is a father of Y, then Y is a child of X”
  - Propositional logic must consider all possible truth value (true/false) assignment possibilities for X and Y

Limitation of Propositional logic

- Let's consider the following example:
  - Tuấn is a student of HUST
  - Every HUST student studies the algebra course
  - Since Tuấn is a HUST student, he studies the algebra course
- In propositional logic:
  - Proposition *p*: "Tuấn is a HUST student"
  - Proposition *q*: "Every HUST student studies the algebra course"
  - Proposition *r*: "Tuấn studies the algebra course"
  - BUT: (In propositional logic) *r* cannot be inferred from *p* and *q*!

First-order logic (FOL): Example

- The above example can be represented in the first-order (i.e., predicate) logic by the following expressions:
  - $HUST\_Student(Tuan)$: "Tuan is a HUST student"
  - $\forall x{:}HUST\_Student(x) \rightarrow Studies\_Algebra(x)$: "Every HUST student studies the algebra course"
  - $Studies\_Algebra(Tuan)$: "Tuan studies the algebra course"
- In the first-order logic, we can prove:

$$\{HUST\_Student(Tuan),\ \forall x{:}HUST\_Student(x) \rightarrow Studies\_Algebra(x)\} \vdash Studies\_Algebra(Tuan)$$

- For the above example, in the first-order logic:
  - The symbols *Tuan*, *x* are **terms** (*Tuan* is a constant, *x* is a variable)
  - The symbols $HUST\_Student$ and $Studies\_Algebra$ are **predicates**
  - The symbol $\forall$ is **universal quantifier**
  - Terms, predicates and quantifiers allows to represent FOL expressions

FOL: Language (1)

- 4 types of **symbols**
  - **Constants**: The names of the objects in a specific problem domain (e.g., *Tuan*)
  - **Variables**: Symbols for which values change for different objects (e.g., *x*)
  - **Function symbols**: Symbols that represent mapping (functional relations) from objects of a domain to objects of another one (e.g., *plus*)
  - **Predicates**: Relations whose logical values are true or false (e.g., *HUST_Student* and *Studies_Algebra*)

- Each function or predicate symbol has a set of arguments
  - E.g., *HUST_Student* and *Studies_Algebra* are 1-argument predicates
  - E.g., *plus* is a 2-argument function symbol

FOL: Language (2)

- **Term** is defined (recursively) as follows:
  - A constant is a term
  - A variable is a term
  - If $t_1, t_2, \ldots, t_n$ are terms and f is a *n-argument* function symbol, then $f(t_1, t_2, \ldots, t_n)$ is a term
  - Nothing else is a term
- Examples of a term
  - *Tuan*
  - *2*
  - *friend(Tuan)*
  - *friend(x)*
  - *plus(x,2)*

FOL: Language (3)

- **Atoms**
  - If $t_1, t_2, \ldots, t_n$ are terms and p is a <u>*n-argument predicate*</u>, then $p(t_1, t_2, \ldots, t_n)$ is an atom
  - E.g., *HUT_Studies(Tuan)*, *HUT_Studies(x)*, *Studies_Algebra(Tuan)*, *Studies(x)*
- **Formulas** are defined as follows:
  - An atom is a formula
  - If ϕ and ψ are formulas, then $\neg\phi$ and $\phi \wedge \psi$ are formulas
  - If ϕ is a formula and x is a variable, then $\forall x{:}\phi(x)$ is a formula
  - Nothing else is a formula
- Note that: $\exists x : \phi(x)$ is equivalent to $\neg\forall x : \neg\phi(x)$

FOL: Semantics (1)

- An **interpretation** of a formula ϕ is represented by a pair of $<\mathcal{D},\mathcal{I}>$
- **The value domain** $\mathcal{D}$ is a non-empty set
- **The interpretation function** $\mathcal{I}$ is a value assignment for each constant, function symbol and predicate:
  - For a constant `c`: $\mathcal{I}(\mathtt{c}) \in \mathcal{D}$
  - For a *n*-argument function symbol `f`: $\mathcal{I}(\mathtt{f})$: $\mathcal{D}^n \rightarrow \mathcal{D}$
  - For a *n*-argument predicate `P`: $\mathcal{I}(\mathtt{P})$: $\mathcal{D}^n \rightarrow \{\mathtt{true}, \mathtt{false}\}$

FOL: Semantics (2)

- **Interpretation of a FOL formula**. Assume that ϕ, ψ and λ are FOL formulas
  - If ϕ is $\neg\psi$, then $\mathcal{I}(\phi)$=`false` if $\mathcal{I}(\psi)$=`true`, and $\mathcal{I}(\phi)$=`true` if $\mathcal{I}(\psi)$=`false`
  - If ϕ is $(\psi \wedge \lambda)$, then $\mathcal{I}(\phi)$=`false` if $\mathcal{I}(\psi)$ or $\mathcal{I}(\lambda)$ are `false`, and $\mathcal{I}(\phi)$=`true` if both $\mathcal{I}(\psi)$ and $\mathcal{I}(\lambda)$ are `true`
  - Assume that $\forall x : \phi(x)$ is a FOL formula, then $\mathcal{I}(\forall x : \phi(x))$=`true` if $\mathcal{I}(\phi)(d)$=`true` for every value $d \in \mathcal{D}$

FOL: Semantics (3)

- A formula ϕ is **satisfiable** if and only if there exists an interpretation <$\mathcal{D}$, $\mathcal{I}$> such that $\mathcal{I}(\phi)$ – We denote: $\models_{\mathcal{I}} \phi$
- If $\models_{\mathcal{I}} \phi$, then we say that $\mathcal{I}$ is a **model** of ϕ. In other words, $\mathcal{I}$ **satisfies** ϕ
- A formula is **unsatisfiable** if and only if there exists no interpretation
- A formula ϕ is **valid** if and only if every interpretation $\mathcal{I}$ satisfies ϕ – We denote: $\models \phi$

Universal quantifier

- Syntax of **universal quantifier**:

  $$\forall\langle \mathit{Variable}_1,\ldots,\mathit{Variable}_n\rangle: \langle \mathit{Formula}\rangle$$

- E.g., All the students of the class K4 are hard-working

  $$\forall x: \mathit{In\_class}(x, K4) \Rightarrow \mathit{Hard\_working}(x)$$

- Formula ($\forall x$: *P*) is true in a model *m*, if and only if *P* is true for **every** object *x* in that model

- Formula ($\forall x$: *P*) is equivalent to a **conjunction** of all the cases of *P*

  $$\begin{aligned} & \mathit{In\_class}(Hue, K4) \Rightarrow \mathit{Hard\_working}(Hue) \\ \wedge\; & \mathit{In\_class}(Cuong, K4) \Rightarrow \mathit{Hard\_working}(Cuong) \\ \wedge\; & \mathit{In\_class}(Tuan, K4) \Rightarrow \mathit{Hard\_working}(Tuan) \\ \wedge\; & \ldots \end{aligned}$$

Existential quantifier

- Syntax of **existential quantifier**:

  $$\exists \langle \mathit{Variable}_1, \ldots, \mathit{Variable}_n \rangle : \langle \mathit{Formula} \rangle$$

- E.g., There exist a student of the class K4 who is hard working:

  $$\exists x: \mathit{In\_class}(x, \mathrm{K4}) \land \mathit{Hard\_working}(x)$$

- Formula ($\exists x: P$) is true in a model m, if and only if P is true for **an** object x in that model

- Formula ($\exists x: P$) is equivalent to a **disjunction** of all the cases of P

$$\begin{array}{ll} & \mathit{In\_class}(\mathrm{Hue}, \mathrm{K4}) \land \mathit{Hard\_working}(\mathrm{Hue}) \\ \lor & \mathit{In\_class}(\mathrm{Cuong}, \mathrm{K4}) \land \mathit{Hard\_working}(\mathrm{Cuong}) \\ \lor & \mathit{In\_class}(\mathrm{Tuan}, \mathrm{K4}) \land \mathit{Hard\_working}(\mathrm{Tuan}) \\ \lor & \ldots \end{array}$$

Characteristics of logic quantifiers

- Permutation:
  - $(\forall x\ \forall y)$ is equivalent to $(\forall y\ \forall x)$
  - $(\exists x\ \exists y)$ is equivalent to $(\exists y\ \exists x)$
- However, $(\exists x\ \forall y)$ is **not** equivalent to $(\forall y\ \exists x)$
  - $\exists x\ \forall y$: Love(x,y) - "In this world, there exists one person who loves everyone else"
  - $\forall y\ \exists x$: Love(x,y) - "Everyone in this world was loved by at least one other"
- Each logic quantifier ($\exists$ or $\forall$) can always be represented by the other
  - ($\forall x$: Love(x, Ice-Cream)) is equivalent to ($\neg\exists x$: $\neg$Love(x, Ice-Cream))
  - ($\exists x$: Love(x, Football)) is equivalent to ($\neg\forall x$: $\neg$Love(x, Football))

Use of FOL

Examples of representation of natural language statements:

- "*x* is a brother/sister of *y*" is equivalent to "*x* and *y* are sibling"

  $\forall x,y$: $Brother\_or\_sister(x,y) \Leftrightarrow Sibling(x,y)$

- "Mother of *c* is *m*" is equivalent to "*m* is female and *m* is parent of *c*"

  $\forall m,c$: $Mother(c,m) \Leftrightarrow (Female(m) \wedge Parent(m,c))$

- The relation "sibling" is symmetrical

  $\forall x,y$: $Sibling(x,y) \Leftrightarrow Sibling(y,x)$