COMP7015 Artificial Intelligence (S1, 2024-25)

Lecture 4: Knowledge Representation and Reasoning

Instructor: Dr. Kejing Yin (cskjyin@hkbu.edu.hk)

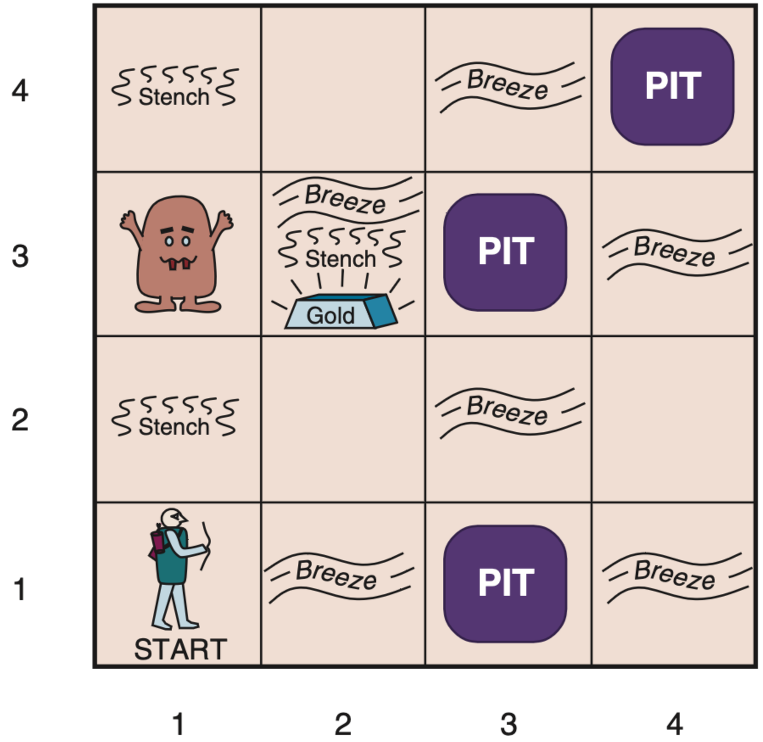
Department of Computer Science
Hong Kong Baptist University

September 27, 2024

Let's Play a Game First: Wumpus World

- **Scores:**
  - +1000 for grabbing the gold;
  - -1000 for falling into a pit or being eaten by the wumpus;
  - -10 for each action taken.

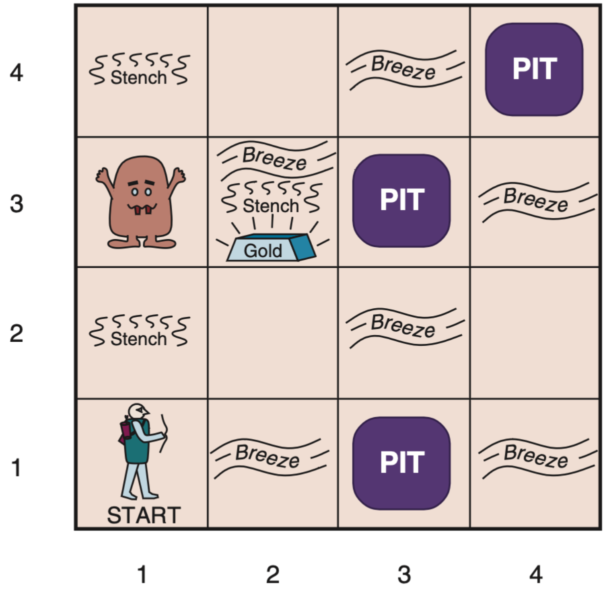
- The game **ends** when the agent either dies or climbs out of the cave.
- The agent could shoot an arrow to kill the wumpus.
- The agent can smell the stench around the wumpus.
- The agent can feel the breeze around the wumpus.

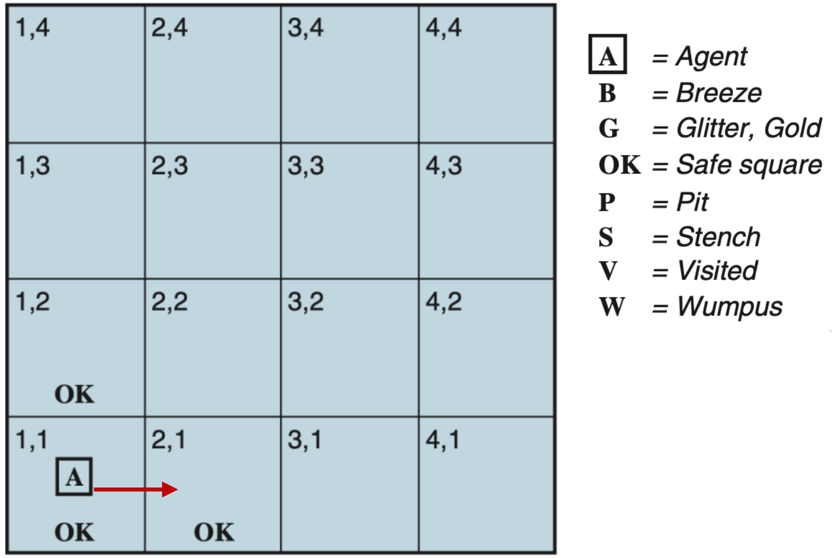


https://thiagodnf.github.io/wumpus-world-simulator/

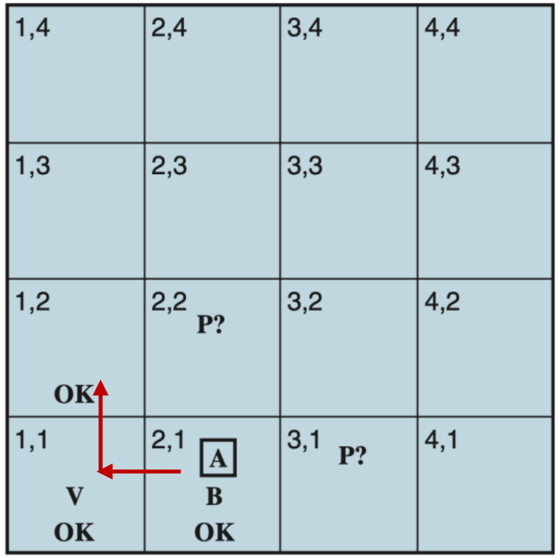
Let’s Play a Game First: Wumpus World

- **How did we make decisions? Consider a simpler 4x4 case:**





(a)



(b)

Initially at (1,1)

(1,1) is safe
→ (1,2) and (2,1) are safe

Move to (2,1)

Breeze at (2,1)
→ a pit at (2,2) and/or (3,1)

Another Motivating Example

- *Example of logic-based models: The virtual assistant*

It's raining now.
If it rains, the ground gets wet.
If the ground gets wet, it is slippery.
(knowledge)

tell

ask Is it slippery now?

answer Yes, it is slippery.

(inference)

Understand the information

Reason using the information

How Do We Represent Knowledge?

- **Knowledge bases consist of sentences.**

Knowledge base

A dime is better than a nickel.

It it is raining, it is wet.

All students like COMP7015.

It is raining now.

If the Wumpus is at (1, 3), you can smell stench at (1, 2)

Inference:

All students like COMP7015.

Tom does not like COMP7015.

→ Tom is not a student.

How Do We Represent Knowledge?

- **Is natural language a good choice?**

A dime is better than a nickel.
A nickel is better than a penny.
→ A dime is better than a penny.

A penny is better than nothing.
Nothing is better than world peace.
??? → A penny is better than world peace.

Natural language can be slippery

- **Logical language**: <u>precise</u> and <u>suitable to capture declarative knowledge</u>.
  - Propositional logic
  - First-order logic

Ingredients of Logic: Syntax, Semantics, and Inference Rules

Syntax

Syntax defines a set of valid formulas (Formulas)

What are valid expressions in the language?

Semantics

For each formula, specify a set of **models** (assignments/ configurations of the word)

What do these expressions mean?

Inference rules

Given f, what new formulas g can be added that are guaranteed to follow?

Ingredients of Logic: Syntax, Semantics, and Inference Rules

Syntax

Syntax defines a set of valid formulas (Formulas)

What are valid expressions in the language?

Examples:

- In English: "Tom ate an apple." (valid),   "Tom an apple ate." (invalid)
- In arithmetic: x + y = 4   (valid),   x4y+ = (invalid)
- In propositional logic: Rain ∧ Wet (valid),   Rain + Wet (invalid)

Ingredients of Logic: Syntax, Semantics, and Inference Rules

Semantics

Semantics defines the truth of each sentence with respect to each *possible world*.

What do these expressions mean?

Examples:

- The semantics for arithmetic specifies that <u>the sentence "x + y = 4" is true in a world where x is 2 and y is 2</u>, but <u>false in a world where x is 1 and y is 1</u>.
- In standard logics, <u>every sentence must be either true or false</u> in each possible world—there is no "in between."

Ingredients of Logic: Syntax, Semantics, and Inference Rules

Inference rules

Given f, what new formulas g can be added that are guaranteed to follow?

Examples:

All students like COMP7015.

Tom does not like COMP7015.

→ Tom is not a student.

Ingredients of Logic: Syntax, Semantics, and Inference Rules

Syntax

Syntax defines a set of valid formulas (Formulas)

What are valid expressions in the language?

Semantics

Semantics defines the truth of each sentence with respect to each *possible world*.

What do these expressions mean?

Inference rules

Given f, what new formulas g can be added that are guaranteed to follow?

Example: from Rain ∧ Wet, derive Rain

Logics

Proposition Logic

First-order Logic

Second-order Logic

...

Higher expressivity

Higher computational efficiency

Propositional Logic

- Syntax of Propositional Logic
- Semantics of Propositional Logic
- Knowledge Base
- Inference Rules of Propositional Logic

Syntax of Propositional Logic

Building blocks: propositional symbols & connectives

- Propositional symbols (atomic formulas; atoms): A, B, C, …
- Logical connectives: $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$
- Build up formulas recursively: <u>if A and B are formulas, so are the following</u>:
  - Negation (not): $\neg A$
  - Conjunction (and): $A \wedge B$ *Symbol $\wedge$ Looks like "A" for "And"*
  - Disjunction (or): $A \vee B$
  - Implication (implies): $A \Rightarrow B$
  - Biconditional (if and only if): $A \Leftrightarrow B$

Syntax of Propositional Logic

Are they valid formulas?

- ✓ A
- ✓ $\neg A$
- ✓ $\neg A \Rightarrow B$
- ✓ $\neg A \wedge (\neg B \Rightarrow C) \vee (\neg B \vee D)$
- ✓ $\neg\neg A$
- ✗ $A \neg B$
- ✗ $A + B$

Syntax of Propositional Logic

- Operator precedence: $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$
- Example: $\neg A \wedge B$ is equivalent to $(\neg A) \wedge B$ rather than $\neg(A \wedge B)$.

- When appropriate, we use parentheses and square brackets to clarify the intended sentence structure and improve readability.

- Note: They are pure symbols without any actual meaning. When we talk about syntax, we are not talking about what they mean. Semantics defines what the symbols mean.

Semantics of Propositional Logic

Fundamental Concept: **Models**

A model m in propositional logic is an assignment of truth values to propositional symbols.

In standard logic, there are only true or false, there is nothing in between.

Example:

- 3 propositional symbols: A, B, C
- $2^3 = 8$ possible models:

$$m_1 = \{A{:}\,0, B{:}\,0, C{:}\,0\}$$
$$m_2 = \{A{:}\,0, B{:}\,0, C{:}\,1\}$$
$$m_3 = \{A{:}\,0, B{:}\,1, C{:}\,0\}$$
$$m_4 = \{A{:}\,0, B{:}\,1, C{:}\,1\}$$
$$m_5 = \{A{:}\,1, B{:}\,0, C{:}\,0\}$$
$$m_6 = \{A{:}\,1, B{:}\,0, C{:}\,1\}$$
$$m_7 = \{A{:}\,1, B{:}\,1, C{:}\,0\}$$
$$m_8 = \{A{:}\,1, B{:}\,1, C{:}\,1\}$$

1: true
0: false

Semantics of Propositional Logic

Fundamental Concept: **Satisfaction**

If a sentence/formula f is true in model m, we say that m satisfies f, or we can say that m is a model of f.

We use the notation $M(f)$ to mean the set of all models of f.

Example: 3 atoms: A, B, C; 8 possible models.

$$
\begin{aligned}
m_1 &= \{A{:}\,0, B{:}\,0, C{:}\,0\} \\
m_2 &= \{A{:}\,0, B{:}\,0, C{:}\,1\} \\
m_3 &= \{A{:}\,0, B{:}\,1, C{:}\,0\} \\
m_4 &= \{A{:}\,0, B{:}\,1, C{:}\,1\} \\
m_5 &= \{A{:}\,1, B{:}\,0, C{:}\,0\} \\
m_6 &= \{A{:}\,1, B{:}\,0, C{:}\,1\} \\
m_7 &= \{A{:}\,1, B{:}\,1, C{:}\,0\} \\
m_8 &= \{A{:}\,1, B{:}\,1, C{:}\,1\}
\end{aligned}
$$

1: true
0: false

f_1="A is true"

m_5 satisfies f_1;
m_6 satisfies f_1;
m_7 satisfies f_1;
m_8 satisfies f_1;

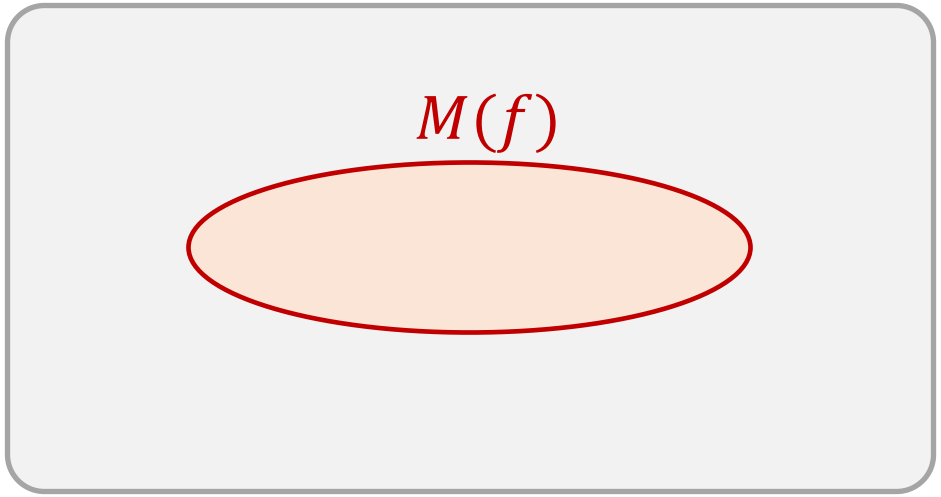
$$M(f_1) = \{m_5, m_6, m_7, m_8\}$$

Semantics of Propositional Logic

Fundamental Concept: **Satisfaction**

If a sentence/formula f is true in model m, we say that m satisfies f, or we can say that m is a model of f.

We use the notation $M(f)$ to mean the set of all models of f.



All possible models (possible worlds)

Semantics of Propositional Logic

- The semantics defines <u>the rules for determining the truth of a sentence</u> with respect to a particular model.

- In propositional logic, all sentences are constructed from atomic sentences and the five connectives. Therefore, we need to specify:
  1) how to compute the truth of <u>atomic sentences</u> and
  2) how to compute the truth of <u>sentences formed with the connectives</u>.

Semantics of Propositional Logic

- Atomic sentences are easy:
  - True (or 1) is true in every model.
  - False (or 0) is false in every model.

  - The truth value of every other proposition symbol must be specified directly in the model.
    E.g., in the model $m_5 = \{A{:}\,1, B{:}\,0, C{:}\,0\}$*, A is true, B is false, and C is false.*

Semantics of Propositional Logic

- For complex sentences, five rules hold for any subsentences P and Q, *being them atomic or complex sentences*, in any model m.

1) $\neg P$ is true iff P is false in m.
2) $P \wedge Q$ is true iff both P and Q are true in m.
3) $\mathrm{P} \vee Q$ is true iff either P or Q is true in m.
4) $\mathrm{P} \Rightarrow Q$ is true unless P is true and Q is false in m.
5) $\mathrm{P} \Leftrightarrow Q$ is true iff P and Q are both true or both false in m.

Semantics of Propositional Logic

- Truth Table

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*

Counter-intuitive: think $P \Rightarrow Q$ as saying,

"If P is true, then I am claiming that Q is true; otherwise, I am making no claim."

- "5 is even implies Sam is smart" is true, regardless of whether Sam is smart.

- Propositional logic does not require any relation of causation or relevance.
  "5 is odd implies Tokyo is the capital of Japan" is a true formula of propositional logic.

Semantics of Propositional Logic

- Truth Table

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*

Bidirectional: $P \Leftrightarrow Q$ is true whenever both $P \Rightarrow Q$ and $Q \Rightarrow P$ are true.

Semantics of Propositional Logic

- Truth Table

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*

Example:

- The formula $f_2 = \neg A \wedge (B \vee C)$, evaluated in $m_2 = \{A{:}\,0, B{:}\,0, C{:}\,1\}$, gives:

$$true \wedge (false \vee true) = true \wedge true = true$$

- Therefore, m_2 satisfies f_2.

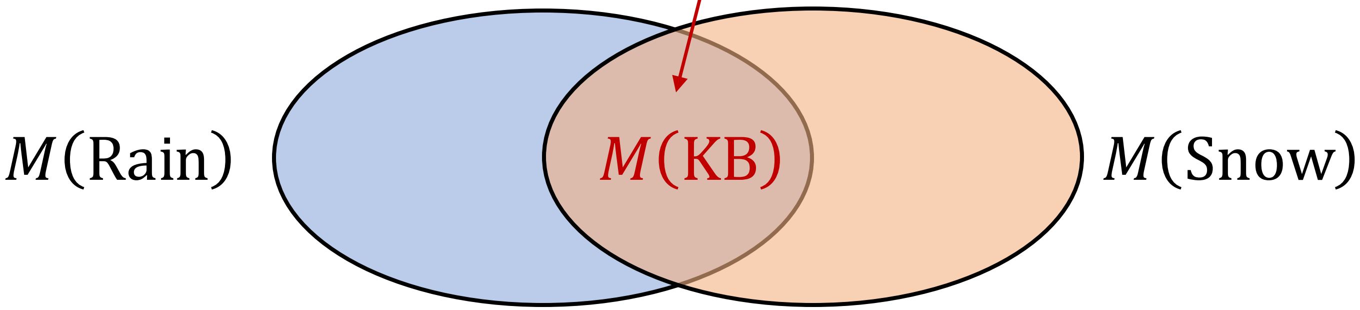
Knowledge Base

- A knowledge base KB is a set of formulas representing their intersection.

$$M(\mathrm{KB}) = \bigcap_{f \in \mathrm{KB}} M(f)$$

Example: KB = {Rain, Snow} ← *KB specifies constraints on the world.*

$M(KB)$ is the set of all worlds satisfying the constraints.



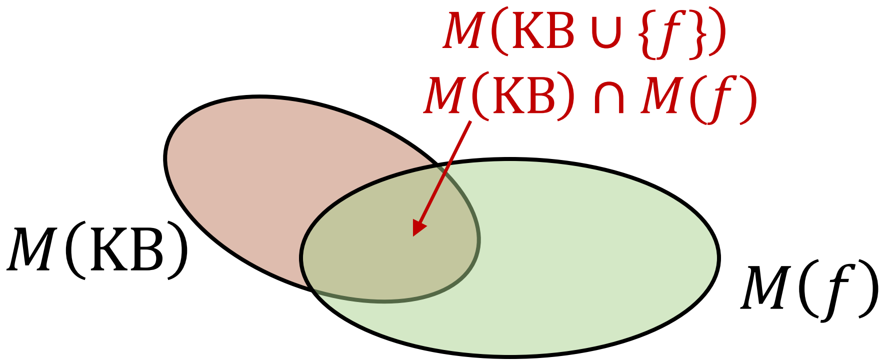
Knowledge Base: Adding knowledge

- Adding more formulas to the knowledge base:

$$\text{KB} \longrightarrow \text{KB} \cup \{f\}$$

- Shrinks the set of models:

$$M(\text{KB}) \longrightarrow M(\text{KB}) \cap M(f)$$



How much does $M(\text{KB})$ shrink?

Knowledge Base: Adding knowledge

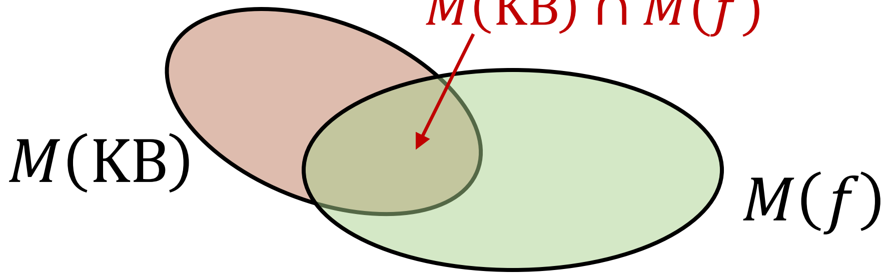
- Adding more formulas to the knowledge base shrinks the set of models:

KB → $\text{KB} \cup \{f\}$

$M(\text{KB})$ → $M(\text{KB}) \cap M(f)$

$M(\text{KB} \cup \{f\})$

$M(\text{KB}) \cap M(f)$



Another Example: 3 propositional symbols: A, B, C ($2^3 = 8$ possible models)

$m_1 = \{A: 0, B: 0, C: 0\}$
$m_2 = \{A: 0, B: 0, C: 1\}$
$m_3 = \{A: 0, B: 1, C: 0\}$
$m_4 = \{A: 0, B: 1, C: 1\}$
$m_5 = \{A: 1, B: 0, C: 0\}$
$m_6 = \{A: 1, B: 0, C: 1\}$
$m_7 = \{A: 1, B: 1, C: 0\}$
$m_8 = \{A: 1, B: 1, C: 1\}$

1. $KB = \{\} = \emptyset$ *Think of KB as "constraints": No constraints for ∅*

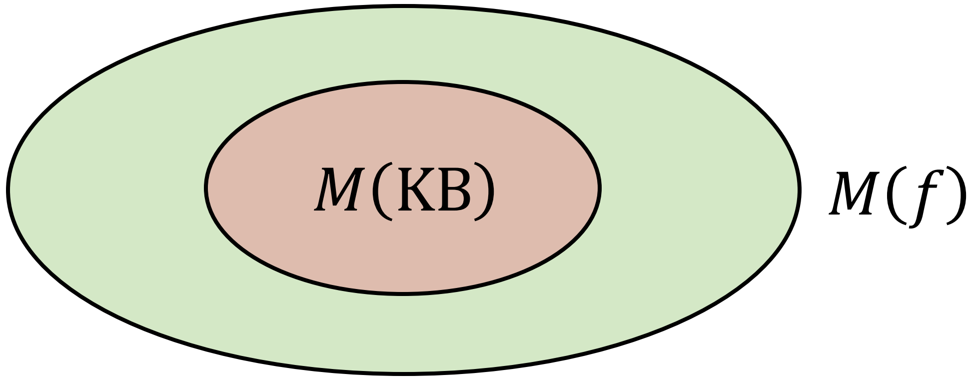
   What is $M(KB)$? $M(KB) = \{m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8\}$

2. Add a formula to KB: "A is true" *One more constraint!*

   What is KB now? $KB \leftarrow KB \cup$ { "A is true" }

   What is $M(KB)$ now? $M(KB) = \{\cancel{m_1, m_2, m_3, m_4,} m_5, m_6, m_7, m_8\}$

Knowledge Base: Adding knowledge (Entailment)

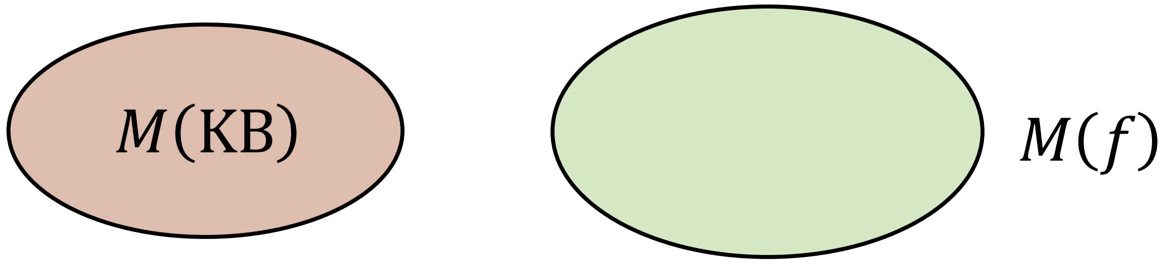


KB entails f (written KB $\vDash f$) iff $M(\mathrm{KB}) \subseteq M(f)$.

- f adds no information. It was already known.

- Example: Rain ∧ Snow ⊨ Snow
  $(x = 0) \vDash (xy = 0)$

Knowledge Base: Adding knowledge (Contradiction)

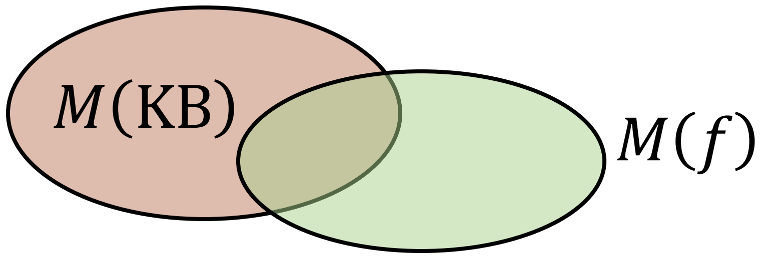


KB contradicts f iff $M(\mathrm{KB}) \cap M(f) = \emptyset$.

- f contradicts what we already know.
- Example: Rain ∧ Snow contradicts ¬Snow

Proposition: KB contradicts f iff KB entails $\neg f$.

Knowledge Base: Adding knowledge (Contingency)



$$\emptyset \subsetneq \quad M(\mathrm{KB}) \cap M(f) \quad \subsetneq M(\mathrm{KB})$$

- f adds non-trivial information to KB.
- Example: KB={Rain}, f=Snow

Knowledge Base: Tell operation

“It’s raining now.”

Tell[Rain]

?

KB

- Possible Responses:
  - Already knew that: entailment (KB $\vDash f$)
  - Don’t believe that: contradiction (KB $\vDash \neg f$)
  - Learns something new (update KB): contingent;

Knowledge Base: Ask operation

“Is it raining now?”

Ask[Rain]

?

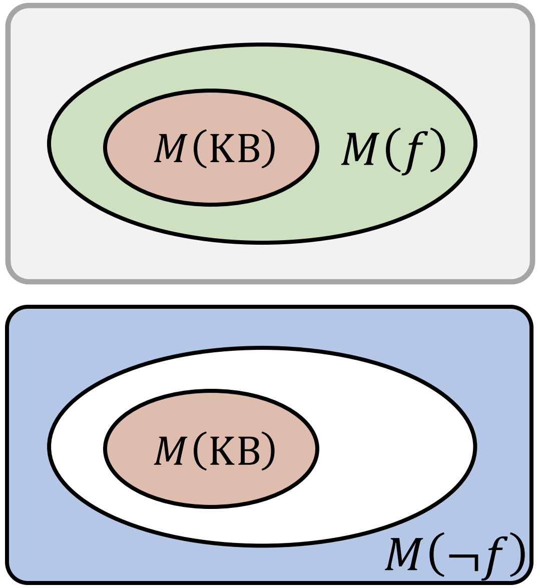
KB

- Possible Responses:
  - Yes: entailment (KB $\vDash f$)
  - No: contradiction (KB $\vDash \neg f$)
  - I don’t know: contingent;

Knowledge Base: Satisfiability

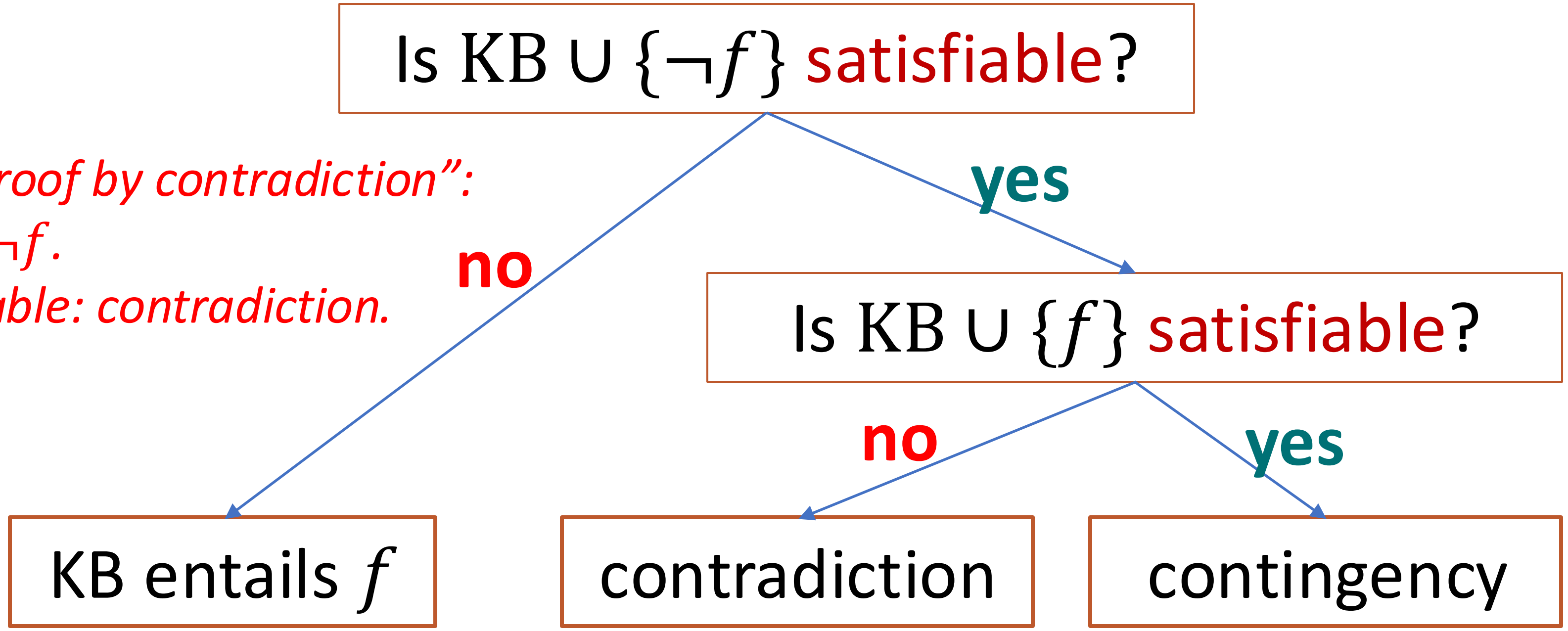
A knowledge base KB is satisfiable if $M(\text{KB}) \neq \emptyset$.

- KB is satisfiable if there is some model that satisfies all formulas in KB.
- Reduce Tell[f] and Ask[f] to satisfiability:



Think it as "proof by contradiction":

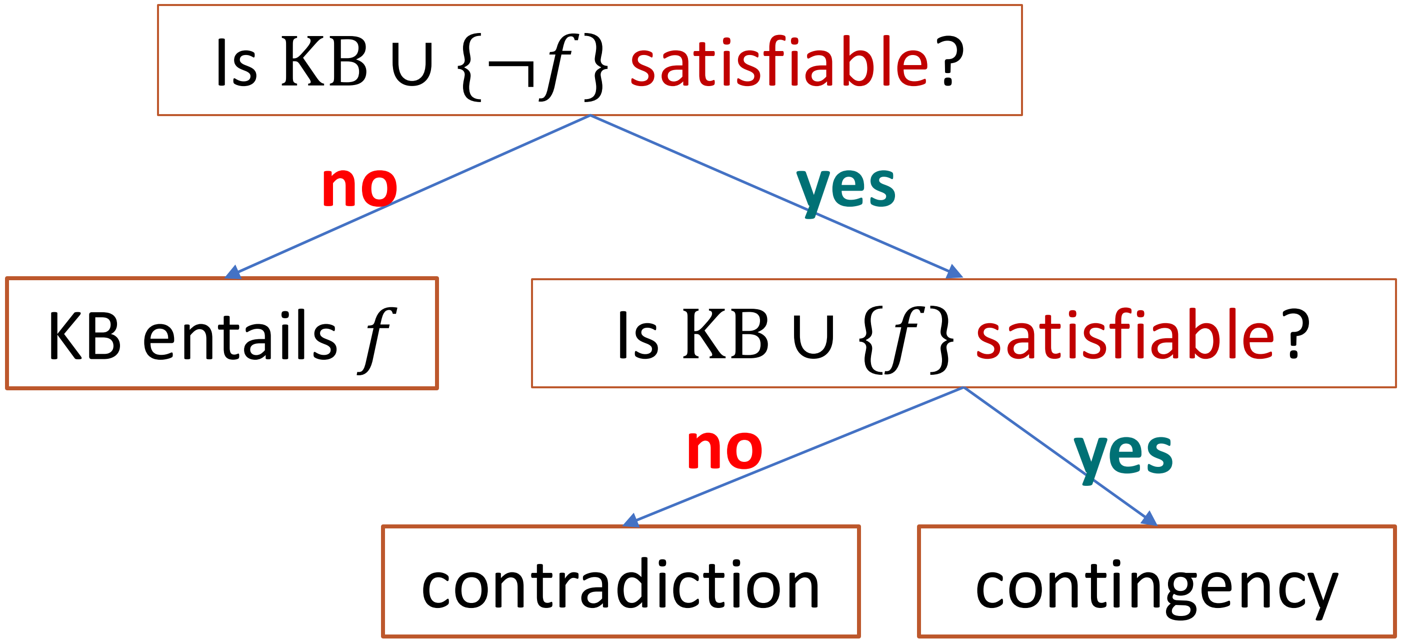
- *Assuming $\neg f$.*
- *Not satisfiable: contradiction.*



Knowledge Base: Satisfiability

A knowledge base KB is satisfiable if $M(\mathrm{KB}) \neq \emptyset$.

- KB is satisfiable if there is some model that satisfies all formulas in KB.
- Reduce Tell[f] and Ask[f] to satisfiability:



Three propositional symbols: A, B, C

$KB = \{$ "A is true" $\}$

What happens if we call Tell["A is true"]?

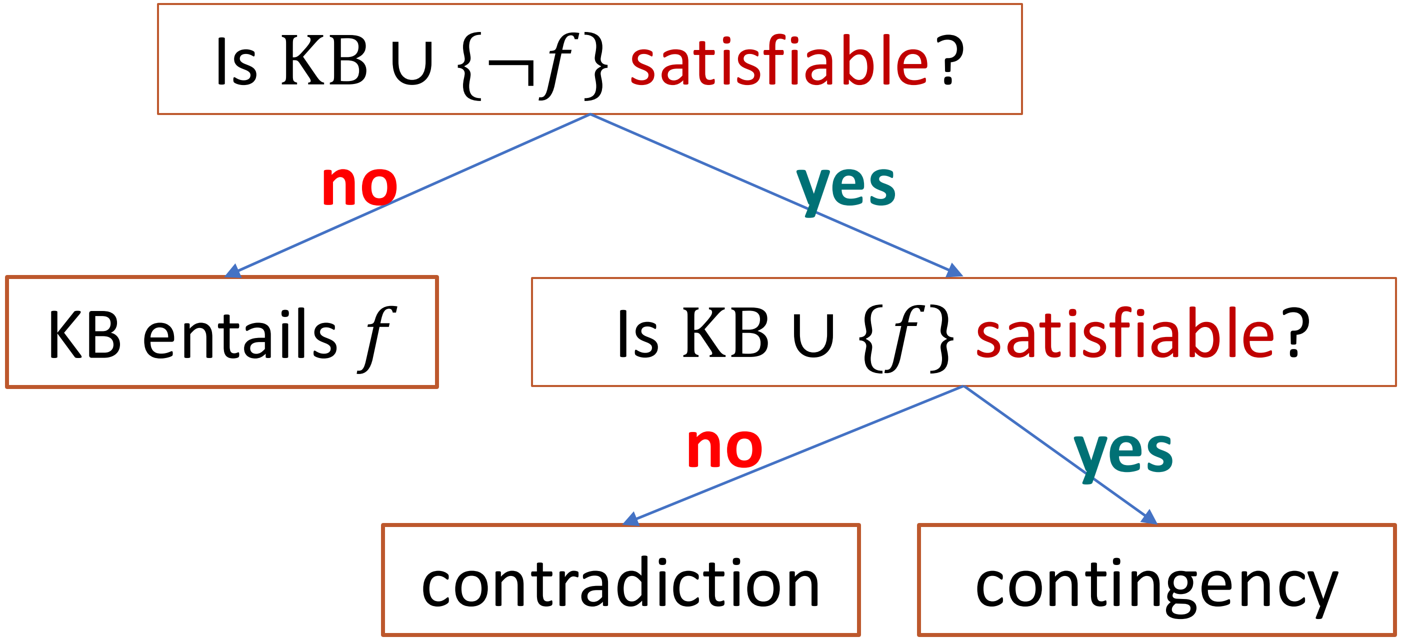
$\mathrm{KB} \cup \{\neg f\} = \{$"A is true", "A is false"$\}$

Not satisfiable → KB entails "A is true"

Knowledge Base: Satisfiability

A knowledge base KB is satisfiable if $M(\mathrm{KB}) \neq \emptyset$.

- KB is satisfiable if there is some model that satisfies all formulas in KB.
- Reduce Tell[f] and Ask[f] to satisfiability:



Three propositional symbols: A, B, C

KB = { “A is true” }

What happens if we call Tell[“A is false”]?

$\mathrm{KB} \cup \{\neg f\}$ = {“A is true”, “A is true”}

Satisfiable

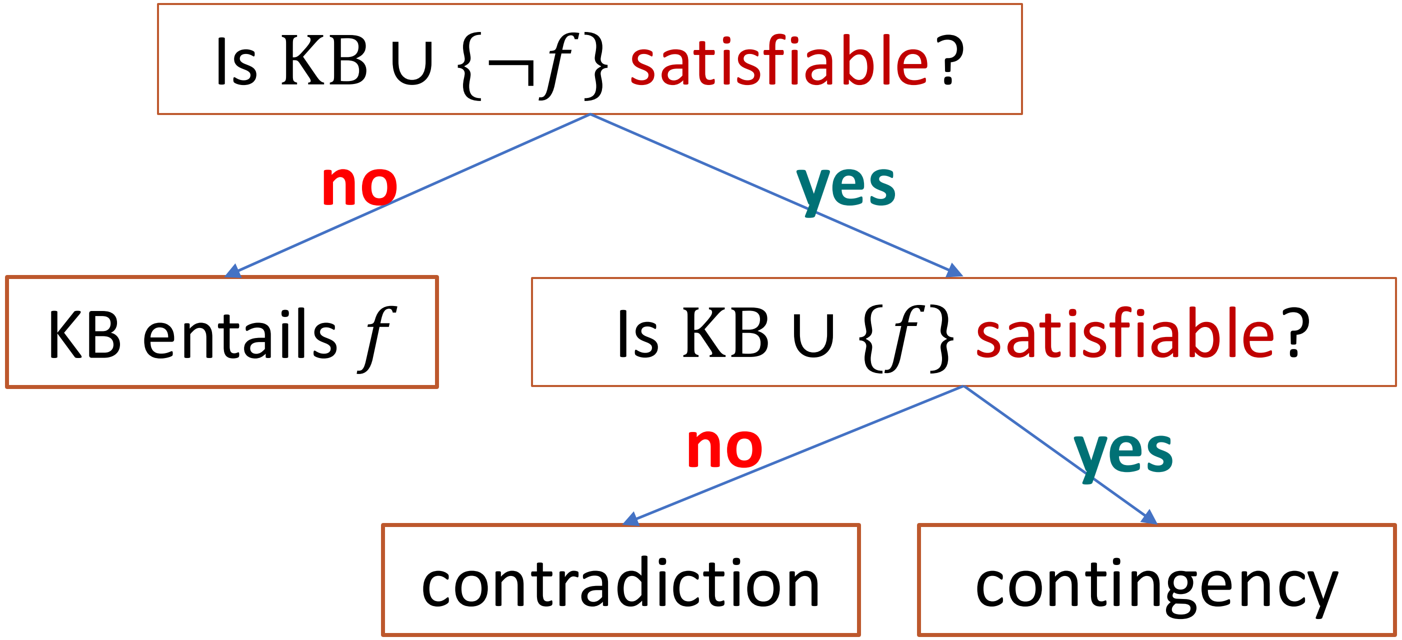
$\mathrm{KB} \cup \{f\}$ = {“A is true”, “A is false”}

Not satisfiable → KB contradicts with “A is true”

Knowledge Base: Satisfiability

A knowledge base KB is satisfiable if $M(\mathrm{KB}) \neq \emptyset$.

- KB is satisfiable if there is some model that satisfies all formulas in KB.
- Reduce Tell[f] and Ask[f] to satisfiability:



Three propositional symbols: A, B, C

KB = { "A is true" }

What happens if we call Tell["B is true"]?

$\mathrm{KB} \cup \{\neg f\}$ = {"A is true", "B is false"}

Satisfiable

$\mathrm{KB} \cup \{f\}$ = {"A is true", "B is true"}

Satisfiable → contingency

Ingredients of logic: Syntax, Semantics, and Inference Rules

Inference rules: Given f, what new formulas g can be added that are guaranteed to follow?

Examples:

All students like COMP7015.
Tom does not like COMP7015.
⇒ Tom is not a student.

Formal definition of **inference rule**:

If $f_1, \ldots, f_k, g$ are formulas, then the following is an inference rule:

$$\frac{f_1, \ldots, f_k, g}{g} \qquad \frac{\textit{(premises)}}{\textit{(conclusion)}}$$

Important: Rules operate directly on syntax, not on semantics.

Modus Ponens Inference Rule

- **Modus Ponens Inference Rule**

For any propositional symbols f and g:

$$\frac{f, \quad f \Rightarrow g}{g} \qquad \frac{\textit{(premises)}}{\textit{(conclusion)}}$$

Example:

- It is raining (Rain)
- If it is raining, then it is wet. (Rain ⇒ Wet)
- Therefore, it is wet. (Wet)

$$\frac{\text{Rain}, \quad \text{Rain} \Rightarrow \text{Wet}}{\text{Wet}}$$

And-Elimination Inference Rule

- **And-Elimination**

$$\frac{f_1 \wedge f_2 \wedge \cdots \wedge f_n}{f_i}$$

Example:

- It is raining and snowing (Rain ∧ Snow)
- Therefore, it is raining. (Rain)

$$\frac{\text{Rain} \wedge \text{Snow}}{\text{Rain}} \qquad \frac{\text{Rain} \wedge \text{Snow}}{\text{Snow}}$$

And-Introduction and Or-Introduction

Example:

- **And-Introduction** $\dfrac{f_1, f_2, \cdots, f_n}{f_1 \wedge f_2 \wedge \cdots \wedge f_n}$

$$\frac{\text{Snow, Rain}}{\text{Snow} \wedge \text{Rain}}$$

- **Or-Introduction** $\dfrac{f_i}{f_1 \vee f_2 \vee \cdots \vee f_n}$

$$\frac{\text{Snow}}{\text{Snow} \vee \text{Rain}} \qquad \frac{\text{Snow}}{\text{Snow} \vee \text{Traffic}}$$

Logically Equivalent Sentences

- Sentences (formulas) $\alpha \equiv \beta$ are logically equivalent if they are true in the same set of models.
- "$\Leftrightarrow$" is used as part of a sentence while "$\equiv$" is used between sentences.

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

Figure 7.11 Standard logical equivalences. The symbols α, β, and γ stand for arbitrary sentences of propositional logic.

All of them can be used as inference rules.

e.g.,

$$\frac{\alpha \Leftrightarrow \beta}{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}$$

$$\frac{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}{\alpha \Leftrightarrow \beta}$$

Resolution Inference Rules

- **Resolution Inference Rule**

g and $\neg g$ are complementary
(one is the negation of the other)

$$\frac{f \vee g, \neg g \vee h}{f \vee h} \quad \textit{or,} \quad \frac{f \vee g, \neg g}{f} \quad \textit{or generally,} \quad \frac{f_1 \vee \cdots \vee f_n \vee g, \quad \neg g \vee h_1 \vee \cdots \vee h_m}{f_1 \vee \cdots \vee f_n \vee h_1 \vee \cdots \vee h_m}$$

(unit resolution)

Resolves g

Example:

- It is raining, or it is snowing (Rain ∨ Snow)
- It is not snowing, or there is traffic. (¬Snow ∨ Traffic)
- Therefore, it is raining, or there is traffic. (Rain ∨ Traffic)

Resolution Inference Rules

- **Resolution Inference Rule**

g and $\neg g$ are complementary
(one is the negation of the other)

$$\frac{f \lor g, \neg g \lor h}{f \lor h} \quad \text{or,} \quad \frac{f \lor g, \neg g}{f} \quad \text{or generally,} \quad \frac{f_1 \lor \cdots \lor f_n \lor g, \quad \neg g \lor h_1 \lor \cdots \lor h_m}{f_1 \lor \cdots \lor f_n \lor h_1 \lor \cdots \lor h_m}$$

(unit resolution)

Resolves g

Important point: Only resolve one pair of complementary symbol at a time!

$$\frac{P \lor \neg Q \lor R, \neg P \lor Q}{\neg P \lor P \lor R}$$

correct

$$\frac{P \lor \neg Q \lor R, \neg P \lor Q}{\neg Q \lor Q \lor R}$$

correct

$$\frac{P \lor \neg Q \lor R, \neg P \lor Q}{R}$$

wrong (cannot resolve both P and Q at once)

Conjunctive Normal Form (CNF)

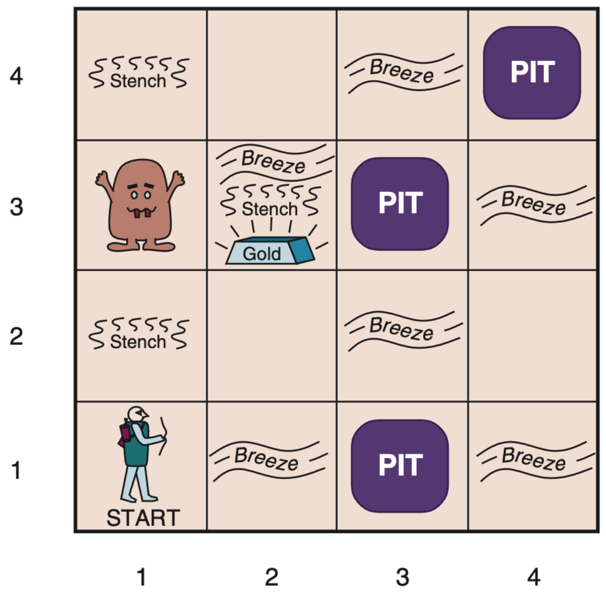
- Resolution inference rule only applies to *clauses (disjunctions of literals)*
  - *A clause is the disjunction of literals (like "$A \vee B \vee C \vee \cdots \vee K$")*
  - *A literal is either an atomic sentence (like "P") or a negated atomic sentence (like "$\neg P$").*
- Can we apply it to any propositional logic sentences?
- *Every sentence of propositional logic is logically equivalent to a conjunction of clauses.*
- *A sentence expressed as a conjunction of clauses is in conjunctive normal form (CNF).*

Conjunctive Normal Form (CNF)

- Conversion to CNF by repeatedly applying the following equivalences:
  - Eliminating "⇔": $f \Leftrightarrow g \equiv (f \Rightarrow g) \wedge (g \Rightarrow f)$
  - Eliminating "⇒": $f \Rightarrow g \equiv \neg f \vee g$
  - Move "¬" inwards: $\neg(f \wedge g) \equiv \neg f \vee \neg g$ and $\neg(f \vee g) \equiv \neg f \wedge \neg g$
  - Eliminate double negation: $\neg\neg f \equiv f$
  - Distribute "∨" over "∧": $f \vee (g \wedge h) \equiv (f \vee g) \wedge (f \vee h)$

- Example: Convert formula "*(Summer ⇒ Snow) ⇒ Bizzare*" to CNF:
  - Eliminating "⇒": ¬ (¬Summer ∨ Snow) ∨ Bizzare
  - Move "¬" inwards: (¬¬Summer ∧ ¬Snow) ∨ Bizzare
  - Eliminate double negation: (Summer ∧ ¬Snow) ∨ Bizzare
  - Distribute "∨" over "∧": (Summer ∨ Bizzare) ∧ (¬Snow ∨ Bizzare)

Propositional Logic Inference: A Simplified Wumpus World Example



Stench around the wumpus.
Breeze around the pit.
Only one Wumpus.

KB = {

$B_{11} \Leftrightarrow (P_{12} \vee P_{21})$; $B_{12} \Leftrightarrow (P_{11} \vee P_{13} \vee P_{22})$; $B_{21} \Leftrightarrow (P_{11} \vee P_{22} \vee P_{31})$; …

$S_{11} \Leftrightarrow (W_{12} \vee W_{21})$; $S_{12} \Leftrightarrow (W_{11} \vee W_{13} \vee W_{22})$; $S_{21} \Leftrightarrow (W_{11} \vee W_{22} \vee W_{31})$; …

$W_{11} \vee W_{12} \vee \cdots \vee W_{43} \vee W_{44}$ // at least one Wumpus.

$\neg W_{11} \vee \neg W_{12}$; $\neg W_{11} \vee \neg W_{13}$; … ; $\neg W_{43} \vee \neg W_{44}$; // at most one Wumpus.

$\neg W_{11}$; $\neg S_{11}$; $\neg P_{11}$; $\neg B_{11}$ // At (1,1), no Wumpus, nor stench. No pit, nor breeze.

}

Infer $\neg P_{12}$:

1. Apply $\dfrac{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}{\alpha \Leftrightarrow \beta}$ obtains $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
2. Apply And-Elimination: $((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
3. Apply $(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha)$ obtains $(\neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1}))$
4. Apply Modus Ponens: $\neg(P_{1,2} \vee P_{2,1})$
5. Apply $\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$ obtains $\neg P_{1,2} \wedge \neg P_{2,1}$

First-Order Logic

- Syntax and Semantics of First-Order Logic
- Inference Rules of First-Order Logic

Limitations of Propositional Logic

Expressivity is limited.

Higher expressivity

Proposition Logic

First-order Logic

Second-order Logic

...

Tom and Jerry both know Python

TomKnowsPython ∧ JerryKnowsPython

All students know Python

TomIsStudent ⇒ TomKnowsPython

JerryIsStudent ⇒ JerryKnowsPython

… *(100+ lines)*

Every even integer grater than 2 is the sum of two primes.

Limitations of Propositional Logic

Expressivity is limited. What are missing?

Higher expressivity

Proposition Logic

First-order Logic

Second-order Logic

...

Objects and predicates.

There are internal structures in propositions like TomKnowsPython.

Quantifiers and variables.

all is a quantifier that applies to each person.

Syntax and Semantics of First-Order Logic

- Term: a logical expression that refers to an object.
  - Constant symbols (e.g, Tom, Python, John)
  - Variable (e.g., x)
  - Function symbols (e.g., *LeftLeg(John), Sum(3, x)*)

Syntax and Semantics of First-Order Logic

- Formulas (Sentences):
  - Atomic formulas (atoms): a predicate symbol optionally followed by a parenthesized list of terms, e.g., Friend(Tom, Jerry).
  - Connectives applied to formulas, e.g., Student(x) $\Rightarrow$ Knows(x, Python).
  - Quantifiers applied to formulas, e.g., $\forall x$ Student(x) $\Rightarrow$ Knows(x, Python)

Syntax and Semantics of First-Order Logic: Quantifiers

- Universal quantification (∀; For all …)
  - All students know Python: $\forall x\ \text{Student}(x) \Rightarrow \text{Knows}(x, \text{Python})$
  - All kings are persons: $\forall x\ \text{King}(x) \Rightarrow \text{Person}(x)$
  - "$\forall x\ P$" says that "P is true for every object x".
  - "$\forall x\ P$" is true in a given model if P is true in all possible extended interpretations.

Three possible extended interpretations

$x \rightarrow$ William Shakespeare,	W. Shakespeare is a King ⇒ W. Shakespeare is a person. ✓
$x \rightarrow$ King George V,	King George V is a King ⇒ King George V is a person. ✓
$x \rightarrow$ Tom Cat	Tom Cat is a King ⇒ Tom Cat is a person. ✓

Shakespeare and Tom Cat are not King, so we say nothing about their personhood.

Syntax and Semantics of First-Order Logic: Quantifiers

- Existential quantification (∃; There exists …/ For some …)
  - Some students know Python: $\exists x\ \text{Student}(x) \land \text{Knows}(x, \text{Python})$
  - "$\exists x\ P$" says that "P is true for *at least one* object x".
  - "$\exists x\ P$" is true in a given model if P is true in *at least one* possible extended interpretations.

Three possible extended interpretations	$x \rightarrow$ Alice,	Alice is a Student ∧ Alice knows Python. ✓
	$x \rightarrow$ Harry,	Harry is a Student ∧ Harry knows Python.
	$x \rightarrow$ Tom Cat	Tom Cat is a Student ∧ Tom Cat knows Python.

Why the Universal Quantifier ∀ Always Pairs With "⇒"?

- Recall its semantics:
  "$\forall x\ P$" is true in a given model if P is true in all possible extended interpretations.

All kings are persons: $\forall x$ King(x) ⇒ Person(x) ✓

Three possible extended interpretations
- x → William Shakespeare, W. Shakespeare is a King (false) ⇒ W. Shakespeare is a person ✓
- x → King George V, King George V is a King (true) ⇒ King George V is a person (true). ✓
- x → Tom Cat Tom Cat is a King (false) ⇒ Tom Cat is a person. ✓

- How about $\forall x$ King(x) ∧ Person(x) ? ✗ *Everything is both a King and a Person*

Three possible extended interpretations
- x → William Shakespeare, W. Shakespeare is a King (false) ∧ W. Shakespeare is a person (true) ✗
- x → King George V, King George V is a King (true) ∧ King George V is a person (true). ✓
- x → Tom Cat Tom Cat is a King (false) ∧ Tom Cat is a person (false). ✗

Why the Existence Quantifier ∃ Always Pairs with "∧" ?

- Similarly, recall its semantics:
  "$\exists x\ P$" is true in a given model if P is true in at least one possible extended interpretations.

Some students know Python: $\exists x$ Student(x) ∧ Knows(x, Python)

Three possible	$x \rightarrow$ Alice,	Alice is a Student (true) ∧ Alice knows Python (true). ✓
extended	$x \rightarrow$ Harry,	Harry is a Student (false) ∧ Harry knows Python (true).
interpretations	$x \rightarrow$ Tom Cat	Tom Cat is a Student (false) ∧ Tom Cat knows Python (false).

$\exists x$ Student(x) ∧ Knows(x, Python) ✓

Why the Existence Quantifier ∃ Always Pairs with "∧" ?

Some students are from Mars: $\exists x$ Student(x) ∧ FromMars(x) ✗

possible extended interpretations

x → Alice, Alice is a Student (true) ∧ Alice is from Mars(false). ✗

x → Harry, Harry is a Student (false) ∧ Harry is from Mars(false). ✗

x → Tom Cat Tom Cat is a Student (false) ∧ Tom Cat is from Mars(false). ✗

x → … ✗

How about: $\exists x$ Student(x) ⇒ FromMars(x) ? *This formula would become true*

possible extended interpretations

x → Alice, Alice is a Student (true) ⇒ Alice is from Mars(false). ✗

x → Harry, Harry is a Student (false) ⇒ Harry is from Mars(false). ✓

x → Tom Cat Tom Cat is a Student (false) ⇒ Tom Cat is from Mars(false). ✓

$\exists x$ Student(x) ⇒ FromMars(x) is synthetically valid but cannot express our desired semantics.

Syntax and Semantics of First-Order Logic: Quantifiers

- Nested quantifiers
  - Brothers are siblings: $\forall x\ \ \forall y\ \ \text{Brothers}(x, y) \Rightarrow \text{Siblings}(x, y)$
  - Siblinghood is a symmetric relationship: $\forall x\ \ \forall y\ \text{Siblings}(x, y) \Rightarrow \text{Siblings}(y, x)$

  - Everybody loves somebody: $\forall x\ \ \exists y\ \text{Loves}(x, y)$
  - There is someone who is loved by everyone: $\exists y\ \ \forall x\ \ \text{Loves}(x, y)$

  - To avoid confusion, we always use different variable names with nested quantifiers.

Syntax and Semantics of First-Order Logic: Quantifiers

- Exercise: Write a first-order logic formula for the following English sentences.
  - There is some course that every student need to take.

    $\exists y\ \text{Course(y)} \land [\forall x\ \text{Student}(x) \Rightarrow \text{Takes}(x, y)]$
  - Every even integer greater than 2 is the sum of two primes.

    $\forall x\ \text{EvenInt}(x) \land \text{Greater}(x, 2) \Rightarrow \exists y \exists z\ \text{Equals}(x, \text{Sum}(y, z)) \land \text{Prime}(y) \land \text{Prime}(z)$
  - If a student takes a course and the course covers a concept, then the student knows that concept.

    $\forall x \forall y \forall z\ \text{Student}(x) \land \text{Takes}(x, y) \land \text{Course}(y) \land \text{Covers}(y, z) \Rightarrow \text{Knows}(x, z)$

Inference Rules of First-Order Logic: Propositionalization

- Converting the first-order knowledge base to propositional logic.

- Example:
  from the following sentence in KB

$$\forall x \; \text{King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x)$$

  we can infer any of the following:

$$\text{King(John)} \land \text{Greedy(John)} \Rightarrow \text{Evil(John)}$$

$$\text{King(Richard)} \land \text{Greedy(Richard)} \Rightarrow \text{Evil(Richard)}$$

Inference Rules of First-Order Logic: Propositionalization

- Example:

KB in first-order logic

$\text{Student}(\text{Alice}) \land \text{Student}(\text{Bob})$
$\forall x\ \text{Student}(x) \Rightarrow \text{Person}(x)$
$\exists x\ \text{Student}(x) \land \text{Creative}(x)$

Finite constant symbols

KB in propositional logic

StudentAlice ∧ StudentBob
(StudentAlice ⇒ PersonAlice) ∧ (StudentBob ⇒ PersonBob)
(StudentAlice ∧ CreativeAlice) ∨ (StudentBob ∧ CreativeBob)

Finite number of formulas

- Now, we can apply any inference algorithms for propositional logic.

Inference Rules of First-Order Logic: Generalized Modus Ponens

- Given:

$$\forall x \quad \text{Takes}(x, \text{COMP7015}) \Rightarrow \text{Knows}(x, \text{Searching})$$

and

$$\text{Takes}(\text{Alice}, \text{COMP7015})$$

- Can we infer Knows(Alice, Searching)?

No, because Takes(x, COMP7015) and Takes(Alice, COMP7015) do not match.
(*Inference rules do not know intrinsic semantics, they just do patern matching*)

- Solution: Substitution and Unification

Inference Rules of First-Order Logic: Generalized Modus Ponens

- **Substitution** *Replacing the variable in a formula with other terms.*

  A substitution θ is a mapping from variables to terms.
  $\mathrm{Subst}[\theta, f]$ returns the result of performing substitution θ on f.

- Examples:

  $\mathrm{Subst}[\{x/\mathrm{Alice}\}, P(x)] = P(\mathrm{Alice})$
  $\mathrm{Subst}[\{x/\mathrm{Alice}, y/z\}, P(x) \wedge K(x, y)] = P(\mathrm{Alice}) \wedge K(\mathrm{Alice}, z)$

Inference Rules of First-Order Logic: Generalized Modus Ponens

- **Unification**

  Unification takes two formulas f and g and returns a substitution θ which is the most general unifier:

  $$\mathrm{Unify}[f, g] \ = \ \theta \text{ such that } \mathrm{Subst}[\theta, f] \ = \ \mathrm{Subst}[\theta, g]$$

  or "fail" if no such θ exists.

- Examples:

  $\mathrm{Unify}[\mathrm{Knows}(\mathrm{Alice}, \mathrm{Python}), \mathrm{Knows}(x, \mathrm{Python})]$ = $\{x/\mathrm{Alice}\}$

  $\mathrm{Unify}[\mathrm{Knows}(\mathrm{Alice}, y), \mathrm{Knows}(x, z)]$ = $\{x/\mathrm{Alice}, y/z\}$

  $\mathrm{Unify}[\mathrm{Knows}(\mathrm{Alice}, y), \mathrm{Knows}(\mathrm{Bob}, z)]$ = fail *We can only substitute variables.*

Inference Rules of First-Order Logic: Generalized Modus Ponens

- **Generalized Modus Ponens**

$$\frac{a_1', \ldots, a_k' \quad \forall x_1 \cdots \forall x_n (a_1 \wedge \cdots \wedge a_k) \rightarrow b}{b'}$$

Get most general unifier θ on premises:

$$\theta \;=\; \mathrm{Unify}[a_1' \wedge \cdots \wedge a_k', a_1 \wedge \cdots \wedge a_k]$$

Apply θ to conclusion:

$$\mathrm{Subst}[\theta, b] = b'$$

Inference Rules of First-Order Logic: Generalized Modus Ponens

- **Example of Generalized Modus Ponens**

- Premises:
  - Takes(Alice, COMP7015)
  - Covers(COMP7015, BFS)
  - $\forall x \forall y \forall z$ Takes(x, y) ∧ Covers(y, z) ⇒ Knows(x, z)

1. Take unify: θ =Unify[Takes(Alice, COMP7015) ∧ Covers(COMP7015, searching), Takes(x, y) ∧ Covers(y, z)] $\theta = \{x/\text{Alice}, y/\text{COMP7015}, z/\text{BFS}\}$

2. Apply θ to conclusion: $\text{Subst}[\{x/\text{Alice}, y/\text{COMP7015}, z/\text{BFS}\},$ Knows(x, z)]

Derives Knows(Alice, BFS)

Summary

- Why do we need to represent knowledge and do reasoning?
- Ingredients of logic: Syntax, Semantics, and Inference Rules.
- Propositional Logic
  - Syntax: Atoms and Connectives
  - Semantics: Models, Satisfaction, Truth Table
  - Knowledge Base: Entailment, Contradiction, Contingency, Ask and Tell Operations.
  - Inference Rules: Modus Ponens, And-Elimination, Resolution
- First-Order Logic
  - Syntax and Semantics: Term, Connectives, Quantifiers (∀, ∃)
  - Inference Rules: Propositionalization, Generalized Modus Ponens

Let your voice be heard!

Thank you for your feedback! 🙌