香港浸會大學
HONG KONG BAPTIST UNIVERSITY

DEPARTMENT OF
COMPUTER SCIENCE
計算機科學系

COMP7015 Artificial Intelligence

Lecture 4: Knowledge Representation and Reasoning

Instructor: Dr. Kejing Yin

September 29, 2022

Logistics

- In-class Quiz on Oct. 20
- Sat. (Oct. 1) is public holiday.
  → Next office hour will be the following Monday (Oct. 3).

Logistics: Written Assignment

- Written Assignment 1 dues at 23:59 pm, Oct. 5.

- Scan your solutions into a single pdf file, name it using the following format:

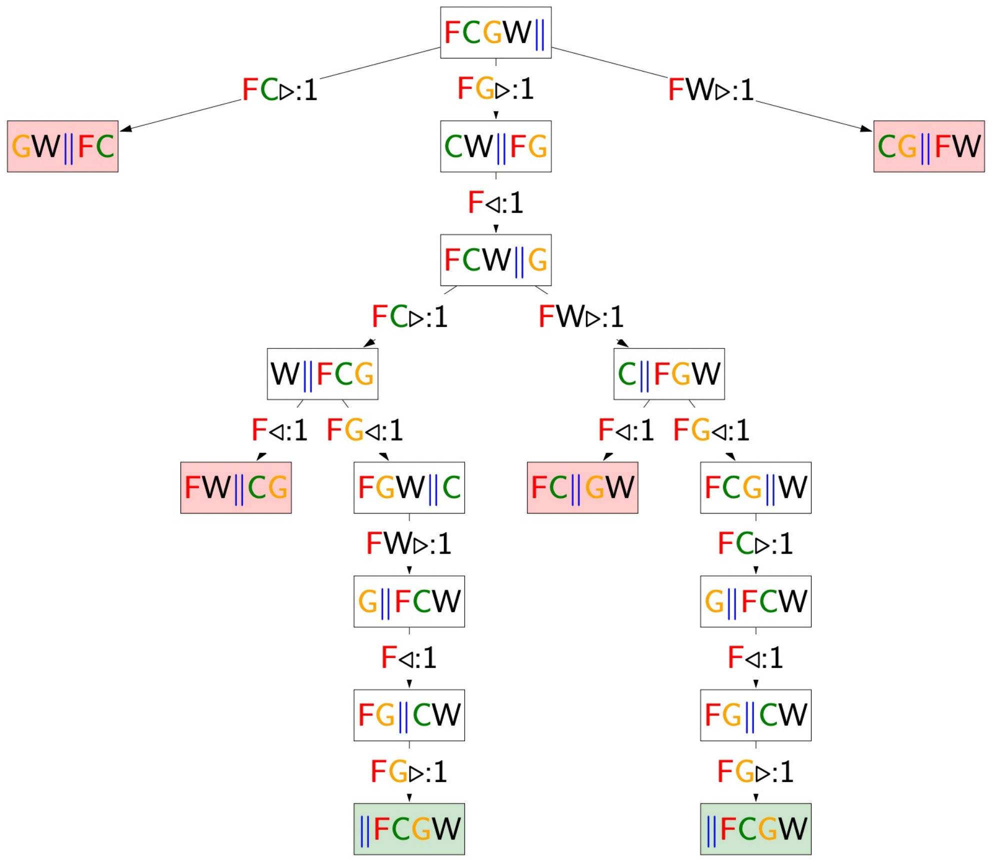
  wa1_<student id>.pdf   (*e.g.*, wa1_16483715.pdf)

- Any clarification needed?

Logistics: Written Assignment

- **Problem 1: drawing the search space**
  - Start from the initial state that you defines in Q1.
  - Draw out the resulting state of taking all possible actions. If any state violates the requirement, no need to further expand that node.

    N.B.: Nodes following the pink nodes are in the state space, but they are not "reachable", so we can ignore them.
  - For repeated states: draw them out, or simply say that they have already been drawn.

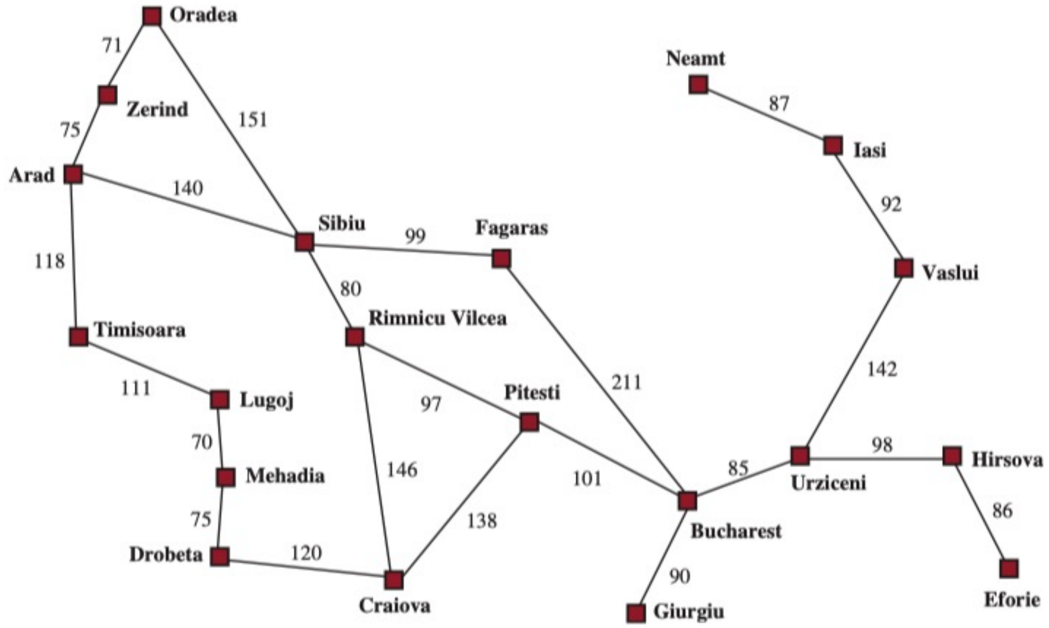


Logistics: Course Project

- Form a group of up to 4 people. You can start forming groups now.
- Topics (Choose one from the following three):

  (1) Searching; (2)Machine Learning; (3) Deep Learning
  Instructions and more *details about each topic to be released in around mid Oct.*
- Requirements:
  - Programming implementation
  - Progress report: in middle of your work
  - A final report: explain the motivation, challenges, and your solutions.
  - Presentation: record a 10 mins video to present your project

Supplement: Level of abstraction in search problem formulation



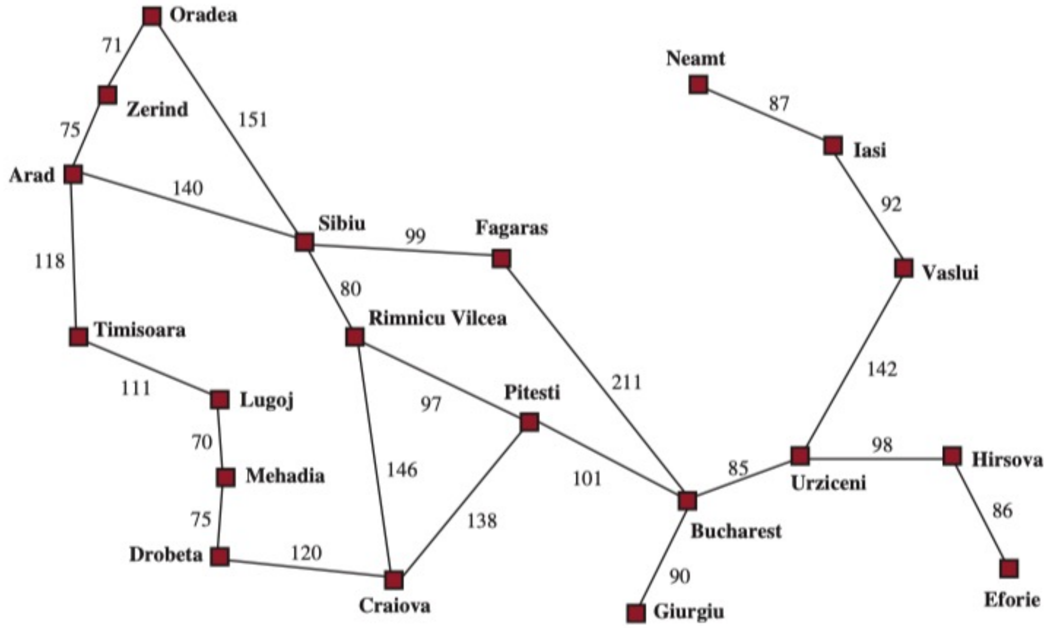
How do we formulate the path-finding problem?

e.g., states

- Option 1: *In Sibiu*, *In Fagaras*, …
- Option 2: *In Sibiu driving a red sedan*, *In Fagaras driving a white SUV with a pet,* …

Which one makes more sense to you?

Supplement: Level of abstraction in search problem formulation



How do we formulate the path-finding problem?

e.g., actions

- Option 1: *Go(Sibiu)*, *Go(Fagaras)*, …
- Option 2: *Turn on the car*, *release the brake*, *accelerate forward*, …

Which one makes more sense to you?

Supplement: Level of abstraction in search problem formulation

- **Abstraction**: the process of removing detail from a representation.

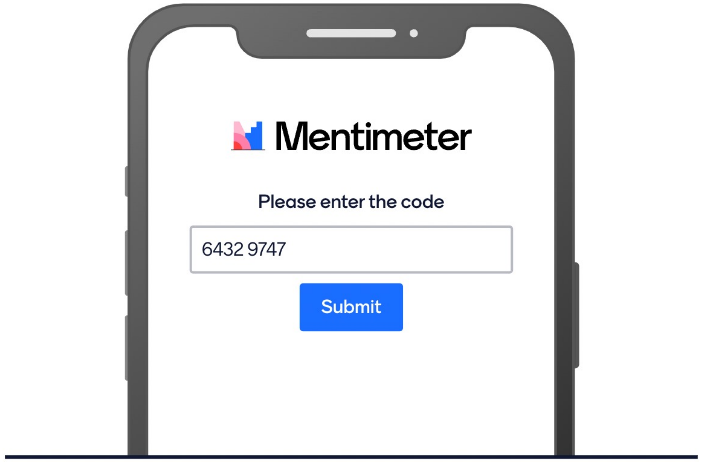
- A good problem formulation has the <u>right level of detail</u>. If we use option 2 to formulate the problem, we probably could never find the way out.
- Depends on the problem, e.g., in path-finding:
  - <u>Driving a red car</u> vs. <u>driving a black car</u>: no difference in general
  - <u>Driving a car</u> vs. <u>taking a bus</u>: there could be some difference

- *A rule of thumb: remove as much detail as possible and make only those distinctions necessary to ensure a valid solution.*

Let’s play a game first: Wumpus World

Go to

www.menti.com



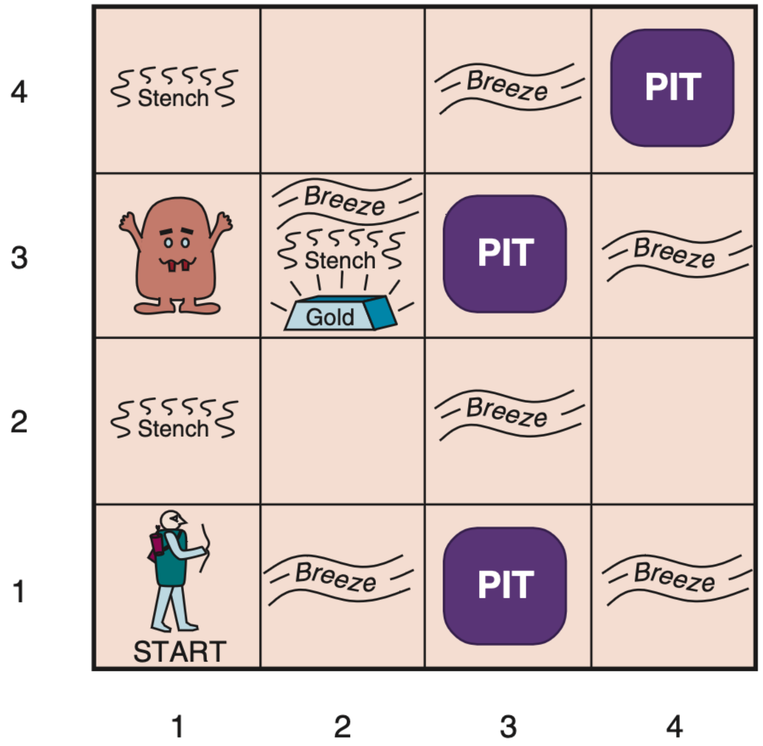
Enter the code

6432 9747

Or use QR code

Let’s play a game first: Wumpus World

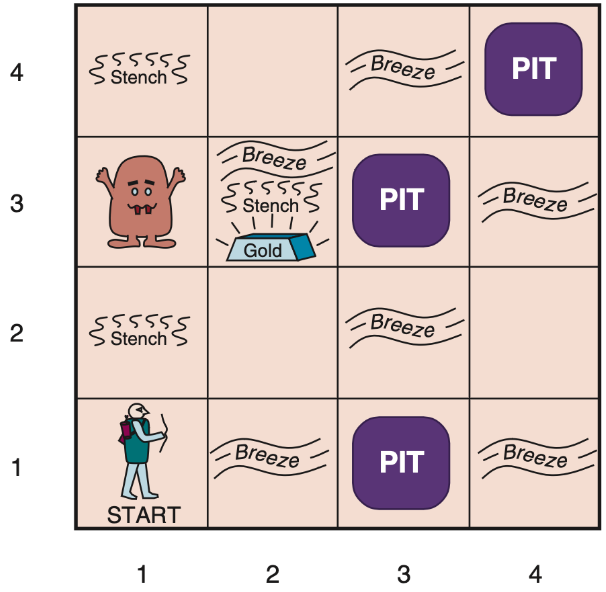
- **Scores:**
  - +1000 for grabbing the gold;
  - -1000 for falling into a pit or being eaten by the wumpus;
  - -10 for each action taken.
- The game **ends** when the agent either dies or climbs out of the cave.
- The agent could shoot an arrow to kill the wumpus.
- The agent can smell the stench around the wumpus.
- The agent can feel the breeze around the wumpus.

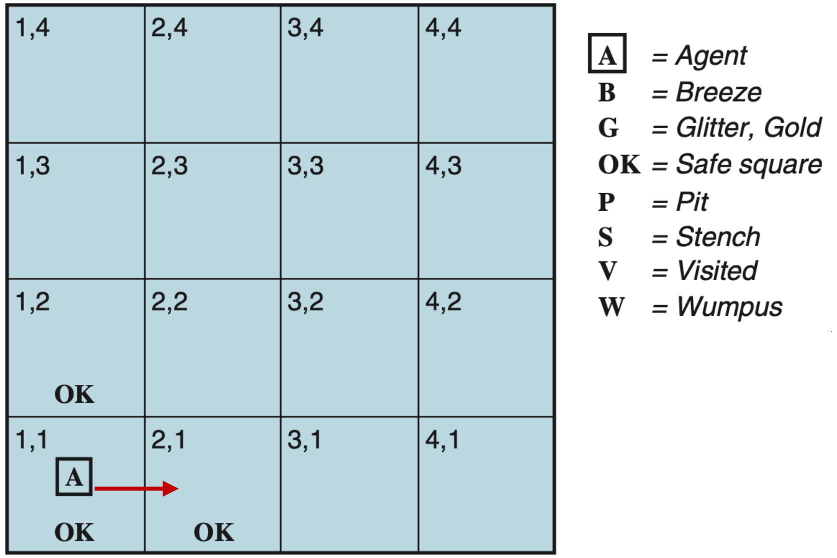


https://thiagodnf.github.io/wumpus-world-simulator/

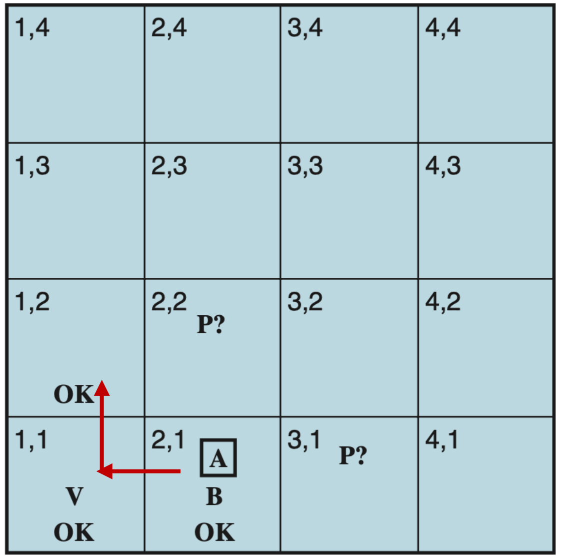
Let’s play a game first: Wumpus World

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





(a)



(b)

Initially at (1,1)

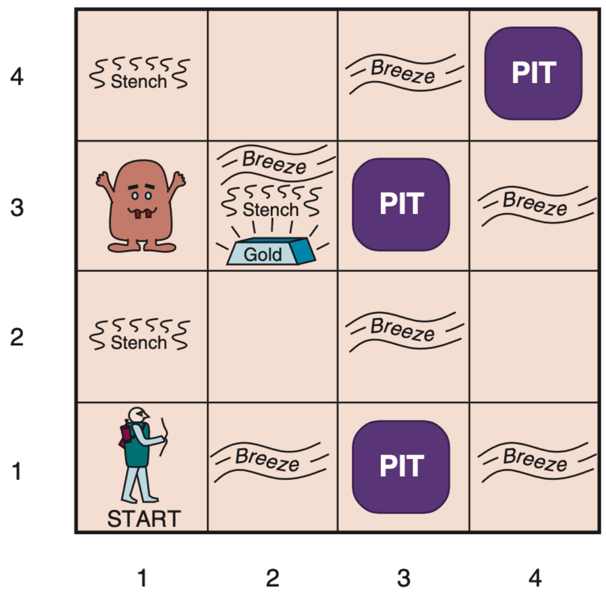
(1,1) is safe
→ (1,2) and (2,1) are safe

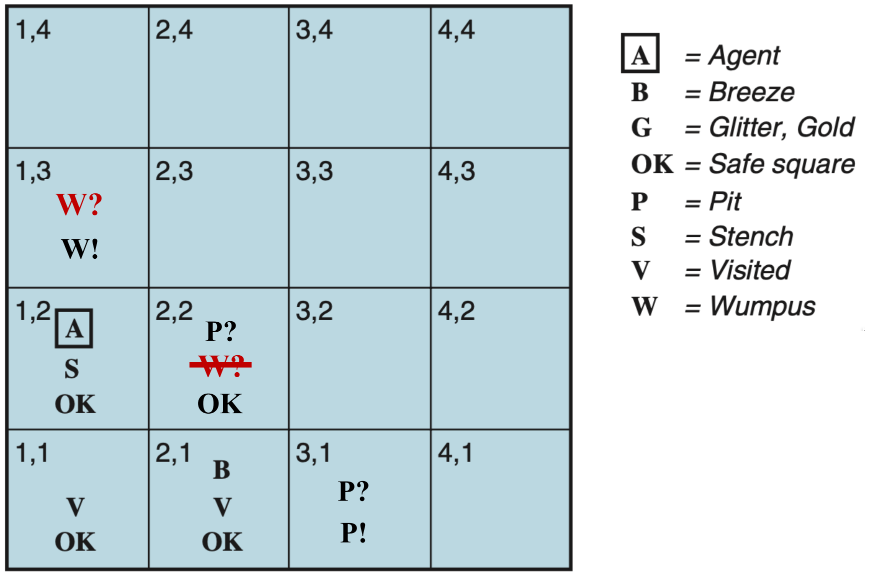
Move to (2,1)

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

Let's play a game first: Wumpus World

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





(c)

Move to (1,2)

Stench at (1,2) → Wumpus at (1,3) and/or (2,2)

No stench at (2,1) → No Wumpus at (2,2) → Wumpus at (1,3)

No breeze at (1, 2) → No pit at (2,2) → Pit at (3, 1)

Logic Reasoning

Draw conclusion from available information

The conclusion is correct if the available information is correct.

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

Logic Representation and Reasoning

- **Goal:** To enable the intelligent agent to represent and store information and derive conclusions from the available information.

Lecture Outline:

- Introduction to Logic
- Propositional Logic
- First-order Logic

Part I: Introduction to Logics

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.
$\Rightarrow$ 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: $\text{Rain} \wedge \text{Wet}$ (valid), $\text{Rain} + \text{Wet}$ (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 the sentence "$x + y = 4$" is true in a world where x is 2 and y is 2, but false in a world where x is 1 and y is 1.

- In standard logics, every sentence must be either true or false 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

Part II: Propositional Logics

Syntax of Propositional Logic

Building blocks: propositional symbols & connectives

- Propositional symbols (atomic formulas; atoms): A, B, C, …
- Logical connectives: ¬, ∧, ∨, ⇒, ⇔
- Build up formulas recursively: if A and B are formulas, so are the following:
  - Negation (not): $\neg A$
  - Conjunction (and): $A \land B$ *Symbol ∧ Looks like "A" for "And"*
  - Disjunction (or): $A \lor B$
  - 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 <u>model</u> m in propositional logic is an <u>assignment</u> 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 α_1;
m_6 satisfies α_1;
m_7 satisfies α_1;
m_8 satisfies α_1;

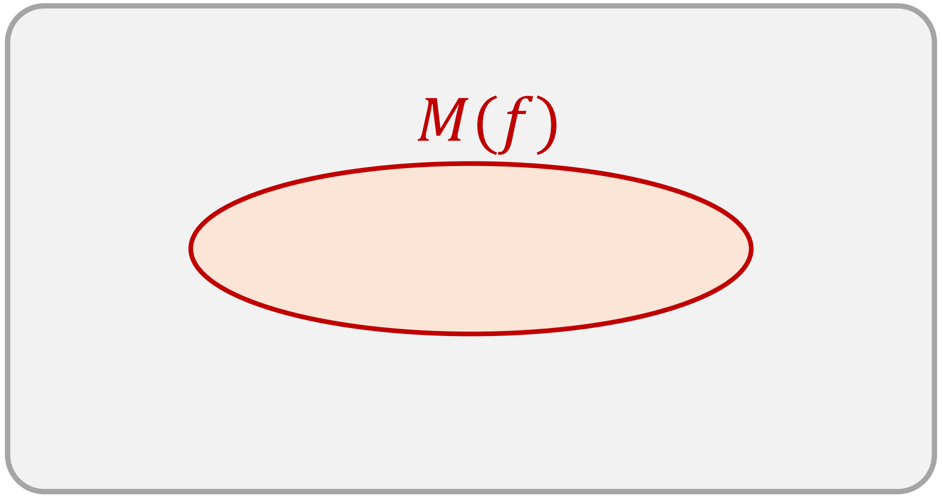
$$M(f_1) = \{m_5, m_6, m_7, m_8\}$$

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 the rules for determining the truth of a sentence 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 atomic sentences and
  2) how to compute the truth of sentences formed with the connectives.

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.

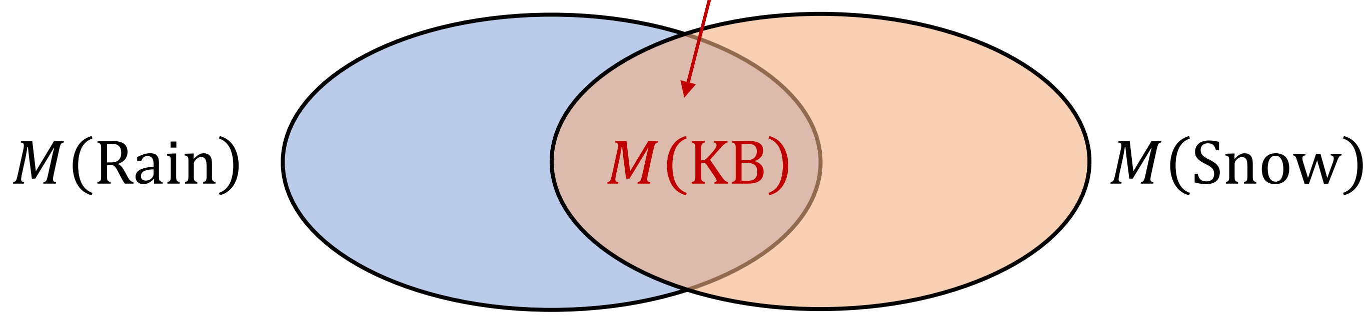
Knowledge Base

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

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

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

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



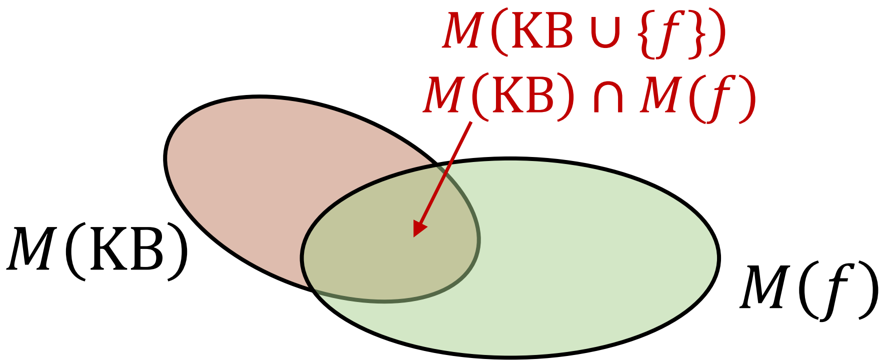
Knowledge Base: Adding knowledge

- Adding more formulas to the knowledge base:

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

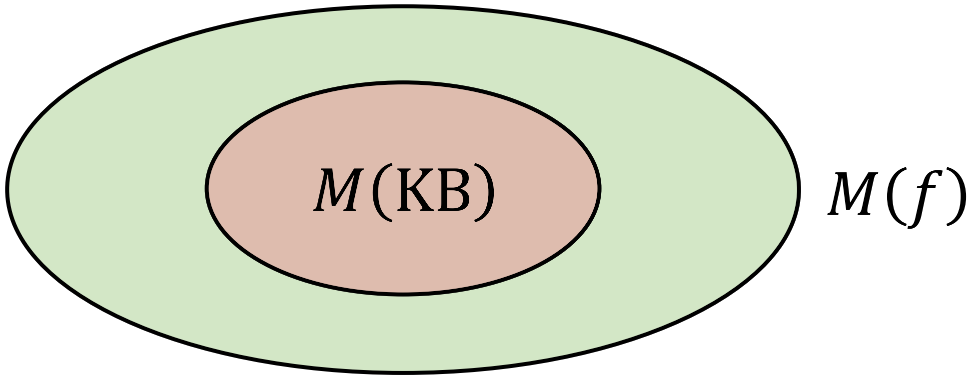
- Shrinks the set of models:

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



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

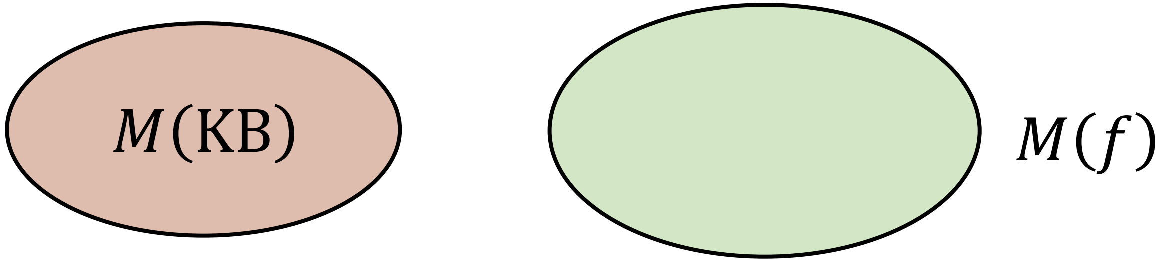
Knowledge Base: Adding knowledge (Entailment)



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

- f adds no information. It was already known.
- Example: $\text{Rain} \wedge \text{Snow} \vDash \text{Snow}$

Knowledge Base: Adding knowledge (Contradiction)

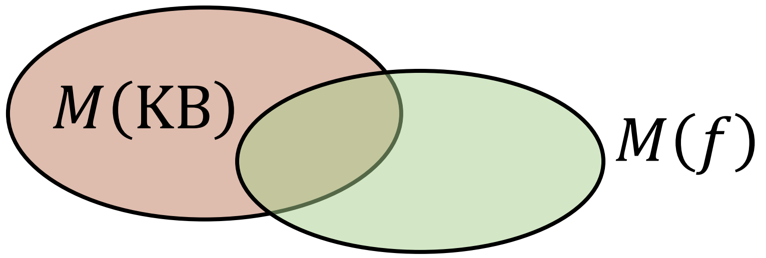


KB contradicts f iff $M(\mathrm{KB}) \cap M(f) = \emptyset$.

- 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]

?

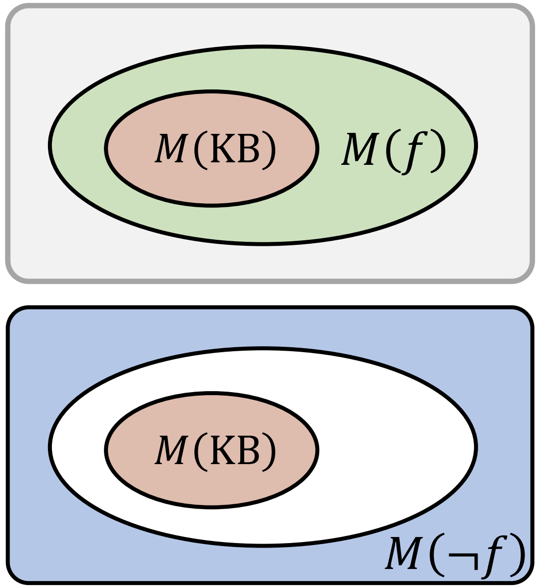
KB

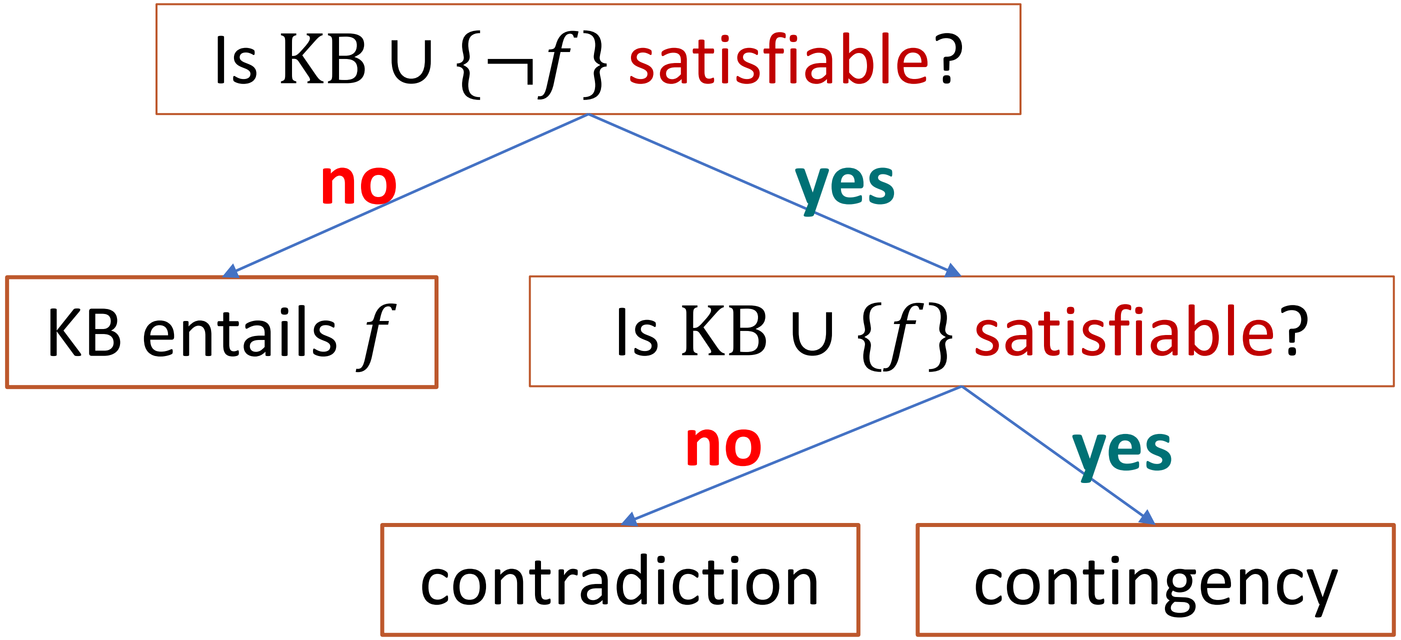
- Possible Responses:
  - Yes: entailment (KB $\models f$)
  - No: contradiction (KB $\models \neg f$)
  - I don’t know: contingent;

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:





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:

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

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

Rules operate directly on syntax, not on semantics.

Inference Rules of Propositional Logic

- **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)

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

Inference Rules of Propositional Logic

- **Resolution Inference Rule**

$$\frac{f \vee g, \neg g \vee h}{f \vee h}$$

Or more generally,

$$\frac{f_1 \vee \cdots \vee f_n \vee g, \quad \neg g \vee h_1 \vee \cdots \vee h_m}{f_1 \vee \cdots \vee f_n \vee h_1 \vee \cdots \vee h_m}$$

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)

Inference Rules of Propositional Logic

- **Modus Ponens** $\dfrac{f,\ f \Rightarrow g}{g}$
- **And-Elimination** $\dfrac{f_1 \wedge f_2 \wedge \cdots \wedge f_n}{f_i}$
- **And-Introduction** $\dfrac{f_1, f_2, \cdots, f_n}{f_1 \wedge f_2 \wedge \cdots \wedge f_n}$
- **Or-Introduction** $\dfrac{f_i}{f_1 \vee f_2 \vee \cdots \vee f_n}$
- **Resolution** $\dfrac{f \vee g, \neg g \vee h}{f \vee h}$
- **Double-Negation Elimination**
  $$\frac{\neg\neg f}{f}$$
- **Unit Resolution**
  $$\frac{f \vee g, \neg g}{f}$$

Part III: First-order Logics

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: $\forall x$ Student(x) ⇒ Knows(x, Python)
  - All kings are persons: $\forall x$ King(x) ⇒ 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: $\exists x$ Student(x) ∧ Knows(x, 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.

Syntax and Semantics of First-Order Logic: Quantifiers

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

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

$$\exists y \text{ Course(y)} \wedge [\forall x \text{ Student}(x) \Rightarrow \text{Takes}(x, y)]$$

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

$$\forall x \text{ EvenInt}(x) \wedge \text{Greater}(x, 2) \Rightarrow \exists y \exists z \text{ Equals}(x, \text{Sum}(y, z)) \wedge \text{Prime}(y) \wedge \text{Prime}(z)$$

  - If a student takes a course and the course covers a concept, then the student knows that concept.

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

Inference Rules of First-Order Logic: Propositionalization

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

- Example:
  from the following sentence in KB

$$\forall x \ \mathrm{King}(x) \wedge \mathrm{Greedy}(x) \Rightarrow \mathrm{Evil}(x)$$

we can infer any of the following:

$$\mathrm{King(John)} \wedge \mathrm{Greedy(John)} \Rightarrow \mathrm{Evil(John)}$$

$$\mathrm{King(Richard)} \wedge \mathrm{Greedy(Richard)} \Rightarrow \mathrm{Evil(Richard)}$$

Inference Rules of First-Order Logic: Propositionalization

- Example:

KB in first-order logic

Student(Alice) $\wedge$ Student(Bob)
$\forall x$ Student(x) $\Rightarrow$ Person(x)
$\exists x$ Student(x) $\wedge$ Creative(x)

Finite constant symbols

KB in propositional logic

StudentAlice $\wedge$ StudentBob
(StudentAlice $\Rightarrow$ PersonAlice) $\wedge$ (StudentBob $\Rightarrow$ PersonBob)
(StudentAlice $\wedge$ CreativeAlice) $\vee$ (StudentBob $\wedge$ 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:

$\forall x$ Takes(x, COMP7015) $\Rightarrow$ Knows(x, Searching)

and Takes(Alice, 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.
  $\text{Subst}[\theta, f]$ returns the result of performing substitution θ on f.

- Examples:

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

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:

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

  or "fail" if no such θ exists.

- Examples:

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

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

  $\text{Unify}[\text{Knows}(\text{Alice}, y), \text{Knows}(\text{Bob}, z)] = \text{fail}$ *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:

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

Apply θ to conclusion:

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

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)] $\theta = \{x/\mathrm{Alice}, y/\mathrm{COMP7015}, z/\mathrm{BFS}\}$

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

Derives Knows(Alice, BFS)

Summary of Lecture 4

- 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, Resolution
- First-Order Logic
  - Syntax and Semantics: Term, Connectives, Quantifiers ($\forall$, $\exists$)
  - Inference Rules: Propositionalization, Generalized Modus Ponens