L6

Wednesday, August 13, 2025 4:55 PM

L06 - Logical Ag...

Logical Agents

Slides adopted from
Dr. Surangika Ranathunga

Week 6: Introduction to logical agents

Week 7: Logical reasoning

Week 8: Knowledge representation

Week 9: Planning

Outline

Fundamental concepts

General principles of logic

Propositional logic

First order logic

Agents

Problem solving agent
- Solution is given, the agent executes it
- Might not face dynamic problems well

Knowledge based / Planning agents
- Given explicit goals
- Can achieve competence quickly by being told or learning
- Adapt to changes in environment

- A **Logical Agent** is a specific type of tool. It's like a brilliant, ultra-precise mathematician who is fantastic at solving specific types of puzzles as long as all the rules are perfectly clear.
- An **Intelligent Agent** is a much *broader category*. It's any system that perceives its environment and takes actions. The "mathematician" is one type of intelligent agent, but so is a self-driving car, a dog, a thermostat, and a chess-playing computer.

Think of it this way: **All Logical Agents are Intelligent Agents, but not all Intelligent Agents are Logical Agents.**

Knowledge-based Agents

Maintain a representation of the world

infer new representations of the world

Use the representation to decide what to do

representation -> reason -> take action

Summary Table

Representation	Description	Example	Power
Atomic	Whole state = 1 unit, no structure	Chess board as ID #4532	Low
Factored	Variables with values	`Temp=30, Rain=True`	Medium
Structured	Objects + Relations	`Parent(John, Mary)`	High

Representations

1. Atomic:
   State considered as a whole, No internal structure available to the agent
2. Factored:
   Assignment of values to variables
3. Structured:
   Objects and relations
   Facts: knowledge about relations

Knowledge-Based Agents

A **knowledge-based agent** is an intelligent agent that:

1. **Maintains a representation of the world** – it stores what it knows (facts, rules, models).
2. **Infers new knowledge** – it uses reasoning to derive new facts from old ones.
3. **Uses this knowledge to decide actions** – it chooses the best action based on what it knows.

The cycle is:

Representation → Reasoning → Action

Types of Representations

1. Atomic Representation
- The **state** of the world is treated as a single unit.
- No internal structure is visible to the agent.
- Example: In a chess game, you might represent the whole board position as just a unique ID, without describing piece-by-piece details.

✅ *Pros:* Simple to manage.
❌ *Cons:* No way to reason about internal details (like relationships between parts).

2. Factored Representation
- The world state is described by **variables** with assigned values.
- Example:
  - In a weather system:
    - `Temperature = 30°C`
    - `Rain = True`
    - `Humidity = 70%`
- Each state is an **assignment of values** to variables.

✅ *Pros:* More flexible than atomic, allows reasoning about variable dependencies.
❌ *Cons:* Still limited when it comes to relationships between different entities.

3. Structured Representation
- The world is described in terms of **objects** and their **relations**.
- Example (family relationships):
  - `Parent(John, Mary)` → John is a parent of Mary.
  - `Sister(Mary, Alex)` → Mary is a sister of Alex.
- Facts represent knowledge about **relationships** between objects.

✅ *Pros:* Very expressive, supports rich reasoning (logic-based, semantic networks, ontologies).
❌ *Cons:* More complex, requires logical inference systems.

Knowledge ?

"Knowing things which helps to reach goals efficiently"

Example:

How to go from San Francisco to Marin County?

How to go from University of Moratuwa to Indigaha Thotupola

👉 So, **atomic → factored → structured** is a progression of *increasing expressiveness and reasoning capability.*

Procedural

Declarative

Declarative Knowledge

- **What something is** (facts, truths, relationships).
- Explicit knowledge about the world.
- Usually expressed in **sentences, logic, rules, or facts.**
- Independent of how it is used.

✅ Example:

- "Paris is the capital of France."
- "If it rains, the ground gets wet."
- In Prolog: `capital(paris, france).`

💡 Think: **knowing that** something is true.

Procedural Knowledge

- **How to do something** (methods, procedures, algorithms).
- Tells the agent **how to act or solve a problem.**
- Often represented as **programs, instructions, or rules for execution.**

✅ Example:

- "To solve a quadratic equation, apply the quadratic formula."
- In Python:

```python
def quadratic(a, b, c):
    return (-b + (b**2 - 4*a*c)**0.5) / (2*a)
```

💡 Think: **knowing how** to do something.

Comparison

Aspect	Declarative	Procedural
Focus	Facts about the world (what is true)	Methods/steps (how to do)
Representation	Logic, statements, rules, databases	Algorithms, programs, rules of action
Flexibility	Easy to query and modify	Efficient for execution
Example (Cooking)	"Pasta is made from flour and water."	Recipe steps to cook pasta

Learning Agents?

Knowledge Base (KB)

Where the representation of the world is maintained

Consists of a set of "sentences" -> written in a knowledge representation

language Base -> a set of axioms

Axiom: ?

1. **Knowledge Base (KB):** This is the system's "brain" or database. It contains **everything the system knows** about its world at a given moment. This knowledge is stored as "sentences" in a special language the computer can understand.
2. **Axioms:** These are the **first sentences** ever put into the Knowledge Base. They are the basic, unproven truths about how the world works.
    - They define the fundamental properties and relationships of things.
    - *Example Axiom:* "All humans are mortal." or "A button is pressed if it is down."
3. **TELLing the KB:** You can add new facts to the KB. For example, you can **TELL** it: "Socrates is a human." This new fact is not an axiom; it's a specific piece of information based on the state of the world.
4. **ASKing the KB:** Once the KB has its axioms and its new facts, you can **ASK** it questions. Using logical reasoning, it can derive new knowledge.
    - *Question:* "Is Socrates mortal?"
    - *Reasoning:* The KB uses the axiom "All humans are mortal" and the fact "Socrates is a human" to logically conclude: "Yes, Socrates is mortal."

KB

Need to

Add new info to the KB (TELL)

Retrieve info from KB (ASK)

both involve inference -> derive new info from old

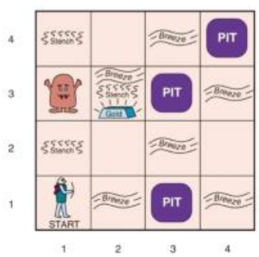
knowledge level (world representation, agents goals) vs implementation level

function KB-AGENT(*percept*) **returns** an *action*
persistent: *KB*, a knowledge base
t, a counter, initially 0, indicating time

TELL(*KB*, MAKE-PERCEPT-SENTENCE(*percept*, *t*))
action ← ASK(*KB*, MAKE-ACTION-QUERY(*t*))
TELL(*KB*, MAKE-ACTION-SENTENCE(*action*, *t*))
$t \leftarrow t + 1$
return *action*

Real-world applications of the Wumpus World :
Designing intelligent agents for autonomous vehicles, robotics, and game creation.

The Wumpus World



The Challenge:

The agent cannot see anything beyond its current room. It is **partially observable**. It must infer what is in adjacent rooms based on **sensory perceptions** (or "percepts"):

- **Stench:** You smell a terrible stench in a room adjacent to the Wumpus.
- **Breeze:** You feel a breeze in a room adjacent to a pit.
- **Glitter:** You see a glitter in the room where the gold is. (This is a direct perception).
- **Scream:** You hear a scream if the Wumpus is killed (e.g., by an arrow the agent can shoot).

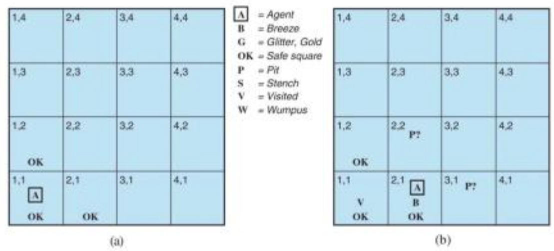
Describing the Environment

Properties of the Wumpus World

- Partially observable: The Wumpus world in AI is partially observable because the agent can only sense the immediate surroundings, such as an adjacent room.
- Deterministic: It is deterministic because the result and end of the world are already known.
- Sequential: It is sequential because the order is essential.
- Static: It is motionless because Wumpus and Pits are not moving.
- Discrete: The surroundings are distinct.
- One agent: The environment is a single agent because we only have one agent, and Wumpus is not regarded as an agent.

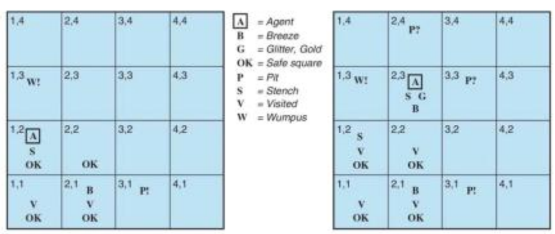
Agent's initial KB

Rules of the environment



(a) (b)

[stench, breeze, glitter, bump, scream]



(a) (b)

Logic

Fundamental concepts of logical representation and reasoning

Logical Languages

A KB has sentences

Expression of sentences should be syntactically correct

x+y =9

+2+= y

Syntax??

Syntax is the rulebook for writing sentences that the computer's reasoning engine can actually process. Without strict syntax, a Knowledge Base would be filled with incomprehensible garbage, and logical inference would be impossible. The sentence `x + y = 9` follows the rulebook. The sentence `+2 += y` does not.

Semantics

Define the truth of sentences wrt each possible world

Possible world = model = an abstraction where each sentence is either true or false

x+y = 9 ???

If a sentence ***s*** is true in a model ***m*** => ***m*** satisfies ***s*** / ***m*** is a model of ***s***

Set of *m* which are models of ***s*** **:** M(s)

The Big Idea: Syntax vs. Semantics

- **Syntax** (from the previous slide) is about **grammar and structure.** It answers: "Is this sentence written correctly?"
- **Semantics** is about **meaning and truth.** It answers: "What does this sentence actually mean, and is it true in a given situation?"

1. **Possible World = Model**
   - Think of a **model** as a single, specific, and complete configuration of the universe. It's one possible scenario where everything has a defined value.
   - *Simple Example:* Imagine a tiny world with just two numbers, `x` and `y`.
     - **Model 1 (m_1):** `x = 5`, `y = 4`
     - **Model 2 (m_2):** `x = 9`, `y = 0`
     - **Model 3 (m_3):** `x = 2`, `y = 3`
   - Each of these is a different "possible world."
2. **Semantics defines the truth of a sentence in each model.**
   - This is the core idea. For any given sentence, semantics gives us a rule to check if that sentence is **True** or **False** in a specific model.
   - Let's take the sentence `s`: `x + y = 9`
   - Now, let's check its truth in our models:
     - In m_1 (`x=5, y=4`): `5 + 4 = 9` is **True.** m_1 satisfies the sentence.
     - In m_2 (`x=9, y=0`): `9 + 0 = 9` is **True.** m_2 satisfies the sentence.
     - In m_3 (`x=2, y=3`): `2 + 3 = 5` is **not 9**. This is **False.** m_3 does *not* satisfy the sentence.
3. **"m is a model of s" / "m satisfies s"**
   - This is just a formal way of saying "In the possible world `m`, the sentence `s` is **true**."
   - From our example: **m_1 is a model of** `s`. **m_2 is a model of** `s`. **m_3 is *not* a model of** `s`.
4. **M(s) - The Set of All Models of s**
   - This is the set of *all possible worlds* where the sentence `s` is true.
   - For our sentence `s: x + y = 9`, the set `M(s)` **is infinitely large!** It contains every single pair of numbers where this is true.
   - `M(s)` = { `m₁ (5,4)`, `m₂ (9,0)`, `m₄ (6,3)`, `m₅ (10,-1)`, `m₆ (4.5,4.5)`, ... and so on forever.}

Logical Reasoning

Logical **entailment** between sentences

α ⊨ β if and only if, in every model in which α is true, β is also true

α ⊨ β if and only if M(α) ⊆ M(β)

x=0 ⊨ xy =0

What is Logical Entailment?

Entailment (symbolized by `⊨`, or `|=`) is a relationship between sentences. It means that if one sentence is true, another sentence **must necessarily also be true.**

The classic way to say it is: **"Sentence α entails sentence β"** or **"β is a logical consequence of α."**

The slide gives two definitions that mean the exact same thing:

1. **In Words:** `α ⊨ β` **if and only if, in every model (possible world) where α is true, β is also true.**
   - This means that the truth of **α forces** the truth of **β**. There is no possible scenario where α is true but β is false.
2. **In Set Theory:** `α ⊨ β` **if and only if** `M(α) ⊆ M(β)`.
   - `M(α)` is the set of all models where α is true.
   - `M(β)` is the set of all models where β is true.
   - `M(α) ⊆ M(β)` means that **every model in M(α) is also inside M(β).** This is just the set version of the first definition.

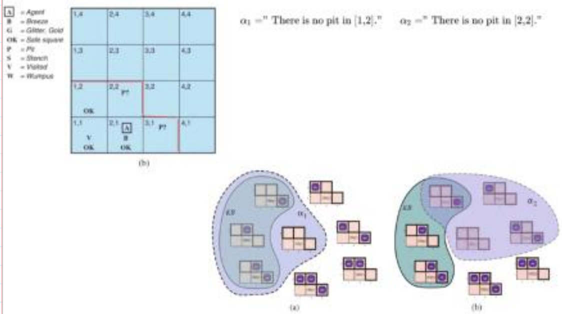
- **Sentence α (The premise):** `x = 0`
- **Sentence β (The conclusion):** `x * y = 0`

1. **Imagine all possible models (worlds) where** `x = 0` **is TRUE.**
   - In all these worlds, `x` is zero. The value of `y` can be anything—`1`, `5`, `-100`, `π`—it doesn't matter. `x` is always 0.
   - *Examples of models in M(α):* `{x=0, y=1}`, `{x=0, y=5}`, `{x=0, y=-100}`, etc.
2. **Now, check if** `x * y = 0` **is TRUE in every single one of these models.**
   - In the model `{x=0, y=1}`: `0 * 1 = 0` -> True
   - In the model `{x=0, y=5}`: `0 * 5 = 0` -> True
   - In the model `{x=0, y=-100}`: `0 * (-100) = 0` -> True
   - In **ANY** model where `x=0`, multiplying it by any `y` will always result in `0`.
3. **Conclusion:** Since `x * y = 0` is true in **every single model** where `x = 0` is true, we can say: `x = 0` **entails** `x * y = 0`. ✅

2. In Set Theory: `α ⊨ β` if and only if `M(α) ⊆ M(β)`.
- `M(α)` is the set of all models where α is true.
- `M(β)` is the set of all models where β is true.
- `M(α) ⊆ M(β)` means that **every model in M(α) is also inside M(β)**. This is just the set version of the first definition.

- In the model `{x=0, y=5}`: `0 * 5 = 0` -> True
- In the model `{x=0, y=-100}`: `0 * (-100) = 0` -> True
- In **ANY** model where `x=0`, multiplying it by any `y` will always result in `0`.

3. **Conclusion:** Since `x * y = 0` is true in **every single model** where `x = 0` is true, we can say: `x = 0` **entails** `x * y = 0`. ✅
The truth of the first sentence guarantees the truth of the second.



Why This is the Heart of Logical Reasoning

This concept of entailment is exactly what the `ASK` function in a Knowledge Base does.

- Your **KB** contains a set of sentences (facts and rules). This KB corresponds to a large set of models `M(KB)` —all the worlds that satisfy everything the KB knows.
- When you **ASK(KB, β)**, the KB checks: **"Is β true in *every single model* in `M(KB)`?"**
  - If the answer is **yes**, then `KB ⊨ β`. The KB can prove that β must be true based on what it knows. It returns `True`.
  - If there is even one model in `M(KB)` where β is false, then `KB ⊭ β`. The KB cannot be sure β is always true, so it returns `False`.

In a nutshell: Entailment is the mathematical foundation for proof and deduction.

Logical Inference

Derive conclusions using entailment

Model checking - enumerates all possible models to check that a sentence is true in all the models where the KB is true Brute force method

$KB \vdash_i \alpha$

i - inference algorithm, should be **sound**, **truth preserving** and **complete**

Model checking works if your space of models is finite

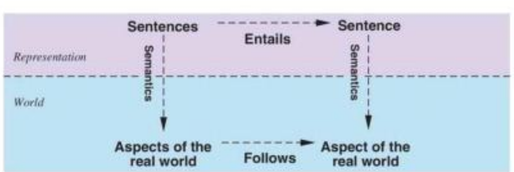
Later….. Theorem proving Smart method

What is Logical Inference?

Inference is the process of using known facts to **derive** or **prove** new facts. It's the "thinking" step. If you know "All humans are mortal" and "Socrates is human," you can *infer* that "Socrates is mortal."

The slide introduces two main ways to do this.

Grounding



if KB is true in the real world, then any sentence derived from KB by a sound inference procedure is also true in the real world

What is "Grounding"?

Grounding is the link between an AI's internal **symbols** (words, logical sentences) and the actual **real-world things** those symbols are supposed to represent.

Without grounding, an AI is just manipulating meaningless symbols. With grounding, those symbols have **meaning** because they refer to something real.

Propositional Logic - syntax

Propositional symbols - stands for a proposition that can be true or false

P,Q,R,$W_{1,2}$ and FacingEast, **True, False**

Atomic sentences - Consists of s single propositional symbol

Literal - atomic sentence or its negation (negative literal)

Complex sentences are constructed from simpler sentences, using parentheses and operators called logical connectives

¬ , ∧ , ∨ , ⇒ , ⇔

Propositional Logic is a language for talking about facts that can either be **true** or **false**, and how these facts are connected. It's like a toolkit for building complex statements out of simple, true/false pieces.

1. Propositional Symbols (The Basic Facts)

- These are the simplest parts. Each symbol represents a single fact or statement that can be True or False.
- **Examples:**
  - `P` (Could mean "It is raining")
  - `Q` (Could mean "The sprinkler is on")
  - `WumpusAlive` (A more descriptive name)
  - `True` and `False` are also special symbols representing constant truth values.

2. Atomic Sentences

- This is just a fancy name for a **single propositional symbol** by itself.
- **Examples:** `P`, `Q`, `WumpusAlive`. Each is a complete, "atomic" sentence.

3. Literal

- An atomic sentence **or its negation** (its opposite).
- **Positive Literal:** `P` ("It is raining")
- **Negative Literal:** `¬P` ("It is **not** raining")
  - The symbol `¬` means "**not**".

4. Complex Sentences (The Connectives)

This is where we build interesting sentences by connecting simpler ones with logical operators.

Connective	Symbol	Meaning	Example	Explanation
Not	`¬`	Negation	`¬P`	"It is **not** raining."
And	`∧`	Conjunction	`P ∧ Q`	"It is raining **and** the sprinkler is o
Or	`∨`	Disjunction	`P ∨ Q`	"It is raining **or** the sprinkler is on.
Implies	`⇒`	Implication	`P ⇒ Q`	"**If** it is raining, **then** the sprinkler i
If and only if	`⇔`	Biconditional	`P ⇔ Q`	"It is raining **if and only if** the spri both false)

$$
\begin{aligned}
Sentence &\rightarrow AtomicSentence \mid ComplexSentence \\
AtomicSentence &\rightarrow True \mid False \mid P \mid Q \mid R \mid \ldots \\
ComplexSentence &\rightarrow (\,Sentence\,) \\
&\mid\ \neg\, Sentence \\
&\mid\ Sentence \land Sentence \\
&\mid\ Sentence \lor Sentence \\
&\mid\ Sentence \Rightarrow Sentence \\
&\mid\ Sentence \Leftrightarrow Sentence
\end{aligned}
$$

OPERATOR PRECEDENCE : $\neg, \land, \lor, \Rightarrow, \Leftrightarrow$

- `( Sentence )`
  - **Translation:** You can put any sentence in parentheses.
  - **Example:** `(P)` is a valid complex sentence. This is useful for grouping.
- `¬ Sentence`
  - **Translation:** You can put a "not" (`¬`) in front of any sentence.
  - **Example:** `¬P` ("not P"), `¬(P ∨ Q)` ("not (P or Q)")
- `Sentence ∧ Sentence`
  - **Translation:** You can connect two sentences with an "and" (`∧`).
  - **Example:** `P ∧ Q` ("P and Q"), `(A ∨ B) ∧ ¬C` ("(A or B) and not C")
- `Sentence ∨ Sentence`
  - **Translation:** You can connect two sentences with an "or" (`∨`).
  - **Example:** `P ∨ Q` ("P or Q")
- `Sentence ⇒ Sentence`
  - **Translation:** You can connect two sentences with an "implies" (`⇒`) or "if-then".
  - **Example:** `P ⇒ Q` ("If P then Q")
- `Sentence ⇔ Sentence`
  - **Translation:** You can connect two sentences with an "if and only if" (`⇔`).
  - **Example:** `P ⇔ Q` ("P if and only if Q")

Propositional Logic - semantics

Defines the rules for determining the truth of a sentence with respect to a particular model

Models in propositional logic - simply sets the truth value—true or false— for every proposition symbol

What are Semantics?

Semantics are the rules that determine whether a sentence is **True** or **False** in a given situation.

Since we're dealing with logic, a "situation" is called a **model**.

What is a Model?

Defines the rules for determining the truth of a sentence with respect to a particular model

Models in propositional logic - simply sets the truth value—true or false— for every proposition symbol

E.g. proposition symbols P,Q, R

One possible model m_1 = {P= true, Q=true, R=false}

Semantics are the rules that determine whether a sentence is **True** or **False** in a given situation.

Since we're dealing with logic, a "situation" is called a **model**.

What is a Model?

A **model** in Propositional Logic is extremely simple. It's just a **lookup table** that assigns a truth value (**True** or **False**) to **every single propositional symbol**.

- The slide's example has three symbols: `P`, `Q`, `R`.
- One possible model, `m₁`, is: `{P = True, Q = True, R = False}`
- This is a complete description of one tiny, possible world.

Other possible models for these three symbols could be:

- `m₂ = {P = False, Q = False, R = False}`
- `m₃ = {P = True, Q = False, R = True}`
- ...and so on. For `n` symbols, there are `2ⁿ` possible models.

Computing the Truth Value of Sentences

The semantics for propositional logic must specify how to compute the truth value of any sentence, given a model

$\neg P$

$Q \wedge P$

$P \vee Q$

$P \Rightarrow Q$

$P \Leftrightarrow Q$

P	Q
false	*false*
false	*true*
true	*false*
true	*true*

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*

A Simple Knowledge Base

$P_{x,y}$ is true if there is a pit in $[x,y]$.

$W_{x,y}$ is true if there is a wumpus in $[x,y]$, dead or alive.

$B_{x,y}$ is true if there is a breeze in $[x,y]$.

$S_{x,y}$ is true if there is a stench in $[x,y]$.

$L_{x,y}$ is true if the agent is in location $[x,y]$.

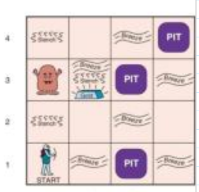
New sentences R_1 R_2 R_3
There is no pit in [1,1] R_1 =

A square is breezy if and only if there is a pit in a neighboring square(consider [1,1]) R_2 : ??

R_3 :??

$R_4 : \quad \neg B_{1,1}$

$R_5 : \quad B_{2,1}$.



Inferencing

Is a sentence α entailed by the KB??

KB ⊨ α

One way to implement an inference algorithm is model checking

Enumerate all the models

Check if α is true when KB is true

$B_{1,1}$ $B_{2,1}$ $P_{1,1}$ $P_{1,2}$ $P_{2,1}$ $P_{2,2}$ $P_{3,1}$

the agent has detected nothing in [1,1] and a breeze in [2,1]

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*false*	*true*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*false*	*true*	*true*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	<u>*true*</u>
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	<u>*true*</u>
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	<u>*true*</u>
false	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*false*	*true*	*false*

Theorem Proving

applying rules of inference directly to the sentences in our knowledge base to construct a proof of the desired sentence without consulting models

Beneficial over model checking if there are a large number of models

Logical Equivalence

Two sentences are logically equivalent if they are true in the same set of models

$\alpha \equiv \beta$ if and only if $\alpha \vDash \beta$ and $\beta \vDash \alpha$

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

Validity

sentence is valid if it is true in all models

Valid sentences are also known as tautologies

Useful in developing deduction theorems

Satisfiability

A sentence is satisfiable if it is true in, or satisfied by, some model

Satisfiability can be checked by enumerating the possible models until one is found that satisfies the sentence

Validity and satisfiability are connected:

α is valid if ¬α is unsatisfiable; contrapositively, α is satisfiable iff ¬α is not valid

Monotonicity

The set of entailed sentences can only increase as information is added to the knowledge base

$$KB \models \alpha \quad \text{then} \quad KB \wedge \beta \models \alpha$$

Inference and Proofs

inference rules that can be applied to derive a proof —a chain of conclusions that leads to the desired goal

Modes Ponens $\dfrac{\alpha \Rightarrow \beta, \;\; \alpha}{\beta}$

And elimination $\dfrac{\alpha \wedge \beta}{\alpha}$

All of the logical equivalences can be used as inference rules

$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$

This biconditional elimination yields the following inference rules

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

$R_1: \quad \neg P_{1,1}$

$R_2: \quad B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

$R_3: \quad B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

$R_4: \quad \neg B_{1,1}$

$R_5: \quad B_{2,1}$.

$R_6: \quad (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

$R_7: \quad ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

$R_8: \quad (\neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1}))$

$R_9: \quad \neg(P_{1,2} \vee P_{2,1})$

$R_{10}: \quad \neg P_{1,2} \wedge \neg P_{2,1}$

INITIAL STATE: the initial knowledge base.

ACTIONS: the set of actions consists of all the inference rules applied to all the sentences that match the top half of the inference rule.

RESULT: the result of an action is to add the sentence in the bottom half of the inference rule.

GOAL: the goal is a state that contains the sentence we are trying to prove

$$\frac{\alpha \Rightarrow \beta, \;\; \alpha}{\beta}$$

Proof by Resolution

$R_1: \neg P_{1,1}$

$R_2: B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$.

$R_3: B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

$R_4: \neg B_{1,1}$

$R_5: B_{2,1}$.

$R_6: (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

$R_7: ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

$R_8: (\neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1}))$

$R_9: \neg(P_{1,2} \vee P_{2,1})$

$R_{10}: \neg P_{1,2} \wedge \neg P_{2,1}$

$R_{11}: \neg B_{1,2}$.

$R_{12}: B_{1,2} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{1,3})$

$R_{13}: \neg P_{2,2}$

$R_{14}: \neg P_{1,3}$

$R_{15}: P_{1,1} \vee P_{2,2} \vee P_{3,1}$

$R_{16}: P_{1,1} \vee P_{3,1}$

$R_{17}: P_{3,1}$

CNF

Resolution rule applies only to disjunctions of literals (clauses)

Every sentence of propositional logic is logically equivalent to a conjunction of clauses

$$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$$

Eliminate the biconditional $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

Eliminate => $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$

CNF

CNF requires the negation to appear only in literals

$\neg(\neg\alpha) \equiv \alpha$ (double-negation elimination)

$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$ (De Morgan)

$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$ (De Morgan)

$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$$

Apply Distributive Law $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$