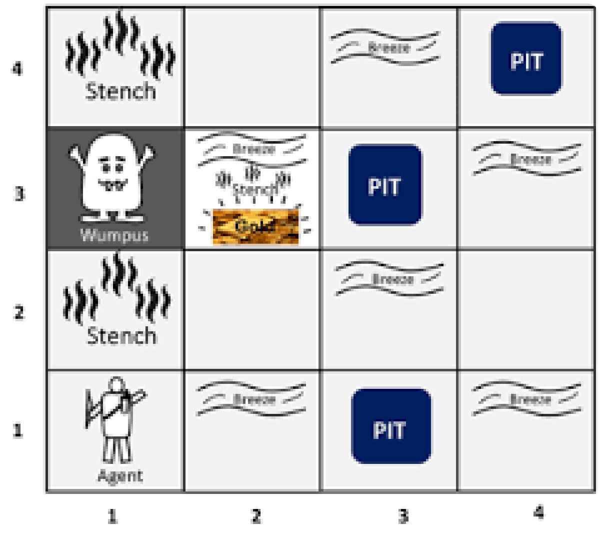
Knowledge-Based Agent

Wumpus World Problem, Logic, Propositional Logic



$\neg P_{1,1}$

$\neg W_{1,1}$

$B_{x,y} \Leftrightarrow (P_{x,y+1} \vee P_{x,y-1} \vee P_{x+1,y} \vee P_{x-1,y})$

$S_{x,y} \Leftrightarrow (W_{x,y+1} \vee W_{x,y-1} \vee W_{x+1,y} \vee W_{x-1,y})$

$W_{1,1} \vee W_{1,2} \vee \ldots \vee W_{4,4}$

$\neg W_{1,1} \vee \neg W_{1,2}$

$\neg W_{1,1} \vee \neg W_{1,3}$

…

Introduction

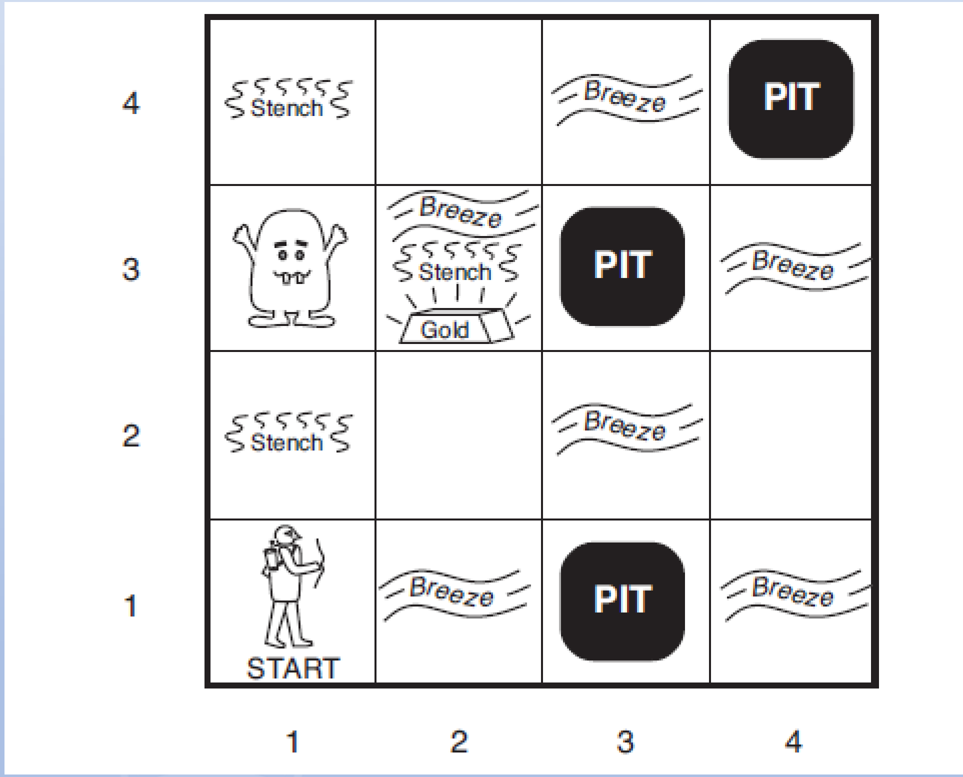
- problem-solving agents are very limiting.
- In a partially observable environment, an agent's only choice for representing what it knows about the current state is to list all possible concrete states—a hopeless prospect in large environments.
- develop logic as a general class of representations to support knowledge-based agents.
- Such agents can combine and recombine information to suit myriad purposes.

Knowledge Based Agents

- A knowledge base is a set of sentences.
- The central component of a knowledge-based agent is its knowledge base or KB.
- Each sentence is expressed in a language called a knowledge representation language and represents some assertion about the world.
- we dignify a sentence with the name axiom when the sentence is taken as given without being derived from other sentences.
- There must be a way to add new sentences to the knowledge base and a way to query what is known. The standard names for these operations are TELL and ASK.

The Wumpus World

- The Wumpus World is a cave consisting of rooms connected by passageways. Lurking somewhere in the cave is the terrible wumpus, a beast that eats anyone who enters its room. An agent can shoot the wumpus, but the agent has only one arrow. Some rooms contain bottomless pits that will trap anyone who wanders into these rooms (except for the wumpus, which is too big to fall in). The only mitigating feature of this bleak environment is the possibility of finding a heap of gold.



The PEAS Description: Wumpus World

- Performance measure: +1000 for climbing out of the cave with the gold, –1000 for falling into a pit or being eaten by the wumpus, –1 for each action taken and –10 for using up the arrow. The game ends either when the agent dies or when the agent climbs out of the cave.

The PEAS Description: Wumpus World

- Environment: A 4×4 grid of rooms. The agent always starts in the square labeled [1,1], facing to the right. The locations of the gold and the wumpus are chosen randomly, with a uniform distribution, from the squares other than the start square. In addition, each square other than the start can be a pit, with a probability of 0.2.

The PEAS Description: Wumpus World

- The action Grab can be used to pick up the gold if it is in the same square as the agent. The action Shoot can be used to fire an arrow in a straight line in the direction the agent is facing. The arrow continues until it either hits (and hence kills) the wumpus or hits a wall. The agent has only one arrow, so only the first Shoot action has any effect. Finally, the action Climb can be used to climb out of the cave, but only from square [1,1].

The PEAS Description: Wumpus World

Sensors: The agent has five sensors, each of which gives a single bit of information:

– In the square containing the wumpus and in the directly (not diagonally) adjacent squares, the agent will perceive a Stench.

– In the squares directly adjacent to a pit, the agent will perceive a Breeze.

– In the square where the gold is, the agent will perceive a Glitter.

– When an agent walks into a wall, it will perceive a Bump.

– When the wumpus is killed, it emits a woeful Scream that can be perceived anywhere in the cave.

The PEAS Description: Wumpus World

- The percepts will be given to the agent program in the form of a list of five symbols; for example, if there is a stench and a breeze, but no glitter, bump, or scream, the agent program will get [Stench, Breeze, None, None, None].

Logic

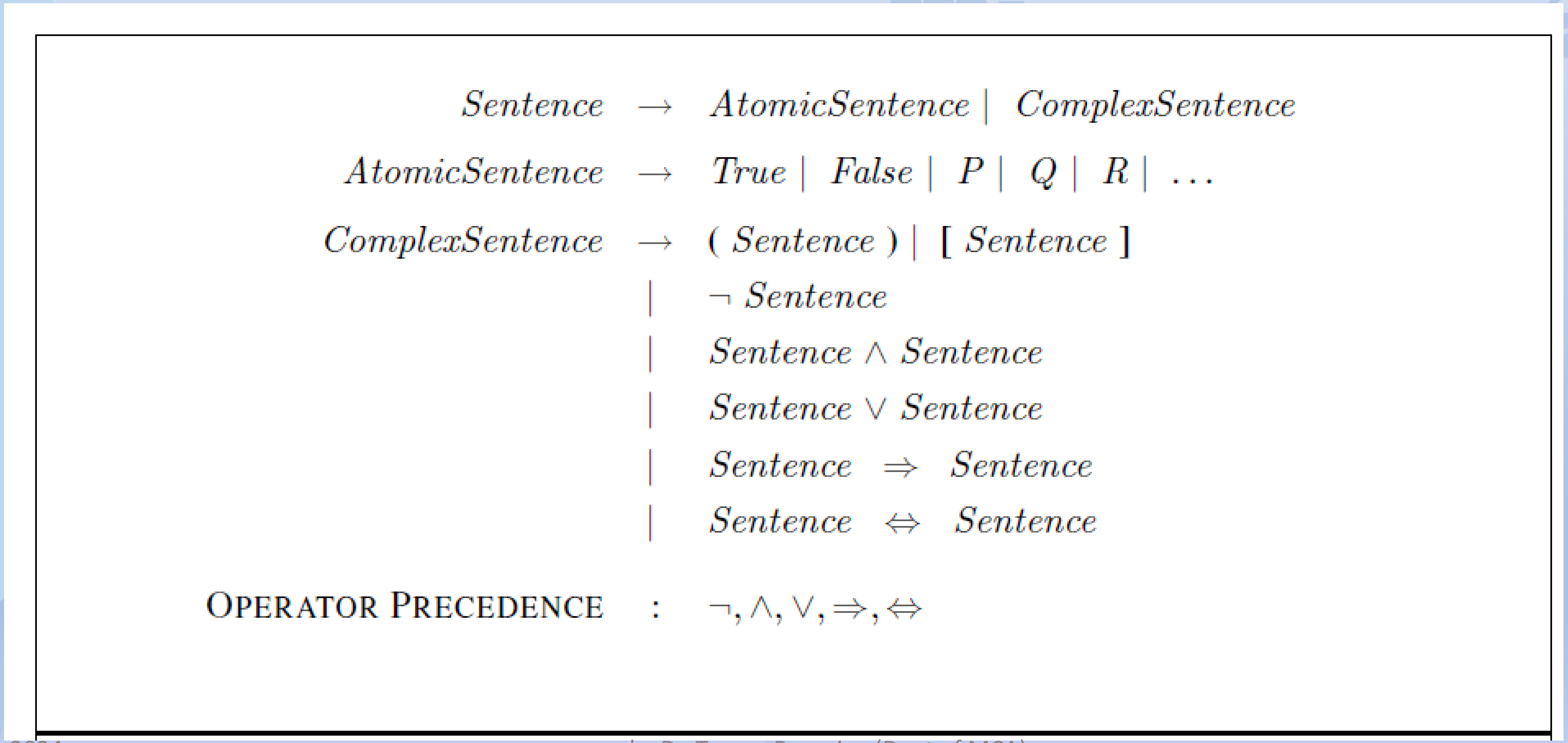
- These sentences are expressed according to SYNTAX, the syntax of the representation language, which specifies all the sentences that are well-formed.
- A logic must also define the semantics or meaning of sentences. The semantics defines the truth of each sentence with respect to each possible world.
- the term model in place of “possible world.”
- If a sentence α is true in model m, we say that m satisfies α or sometimes m is a model of α.
- This involves the relation of logical entailment between sentences—the idea that a sentence follows logically from another sentence. $\alpha \Vdash \beta$

Logic

- $\alpha \Vdash \beta$ if and only if , in every model in which α is true, β is also true.
- An inference algorithm i can derive α from KB.
- An inference algorithm that derives only entailed sentences is called sound and truth preserving.
- an inference algorithm is complete if it can derive any sentence that is entailed.

Propositional Logic

- a simple but powerful logic



$$
\begin{array}{rcl}
\mathit{Sentence} & \rightarrow & \mathit{AtomicSentence} \mid \mathit{ComplexSentence} \\
\mathit{AtomicSentence} & \rightarrow & \mathit{True} \mid \mathit{False} \mid P \mid Q \mid R \mid \ldots \\
\mathit{ComplexSentence} & \rightarrow & (\ \mathit{Sentence}\ ) \mid [\ \mathit{Sentence}\ ] \\
 & \mid & \neg\ \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \land \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \lor \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \ \Rightarrow\ \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \ \Leftrightarrow\ \mathit{Sentence} \\
 & & \\
\textsc{Operator Precedence} & : & \neg, \land, \lor, \Rightarrow, \Leftrightarrow
\end{array}
$$

Propositional Logic: Syntax

- The syntax of propositional logic defines the allowable sentences. The atomic sentences consist of a single proposition symbol. Each such symbol stands for a proposition that can be true or false.
- Complex sentences are constructed from simpler sentences, using parentheses and logical connectives. There are five connectives in common use:
- Not(¬) A sentence such as ¬W1,3 is called the negation of W1,3.

Propositional Logic: Syntax

- And(∧) A sentence whose main connective is ∧, such as $W_{1,3} \land P_{3,1}$, is called a conjunction; its parts are the conjuncts.
- Or(∨) A sentence using ∨, such as $(W_{1,3} \land P_{3,1}) \lor W_{2,2}$, is a disjunction of the disjuncts $(W_{1,3} \land P_{3,1})$ and $W_{2,2}$.
- Implies(→) A sentence such as $(W_{1,3} \land P_{3,1}) \Rightarrow \neg W_{2,2}$ is called an implication
- if and only if(↔) The sentence $W_{1,3} \Leftrightarrow \neg W_{2,2}$ is a biconditional.

Propositional Logic: Semantic

- The semantics defines the rules for determining the truth of a sentence with respect to a particular model.
- $\neg P$ is true iff P is false in m.
- $P \wedge Q$ is true iff both P and Q are true in m.
- $P \vee Q$ is true iff either P or Q is true in m.
- $P \rightarrow Q$ is true unless P is true and Q is false in m.
- $P \Rightarrow Q$ is true iff P and Q are both true or both false in m.

A Simple Inference Procedure

- $P_{x,y}$ 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 agent perceives a breeze in $[x, y]$
- $S_{x,y}$ is true if agent perceives a strench in $[x, y]$

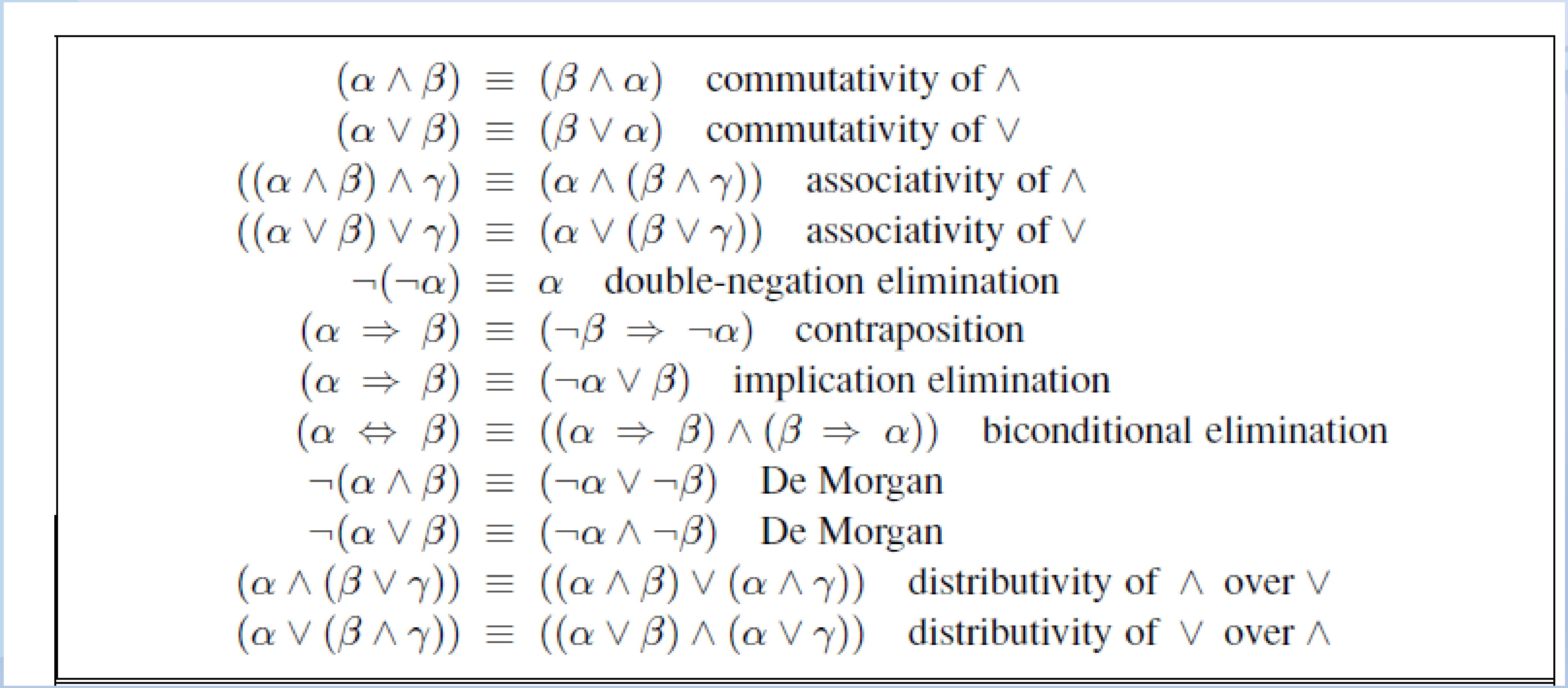
A Simple Inference Procedure

- There is no pit in [1,1] $R_1: \neg P_{1,1}$
- A square is breezy if and only if there is a pit in a neighboring square.
- $R_2: B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
- $R_2: B_{2,1} \Leftrightarrow \left(P_{1,1} \vee P_{2,2} \vee P_{3,3}\right)$
- The preceding sentences are true in all Wumpus world.
- $R_4: \neg B_{1,1}$
- $R_5: B_{2,1}$

Propositional Theorem Proving

- Two sentences α *and* β are logically equivalent if they are true in the same set of models.
- $\alpha \equiv \beta$ if and only if $\alpha \vDash \beta$ *and* $\beta \vDash \alpha$
- A sentence is valid if it is true in all models.
- A sentence is satisfiable if it is true in some model.
- A propositional formula is said to be *provable* if there is a formal proof of it in that system.
- The converse, which says that every valid formula is provable, is known as *completeness*.
- The statement that every provable formula is valid is known as *soundness*.

Propositional Theorem Proving



$(\alpha \wedge \beta)$	$\equiv$	$(\beta \wedge \alpha)$	commutativity of $\wedge$
$(\alpha \vee \beta)$	$\equiv$	$(\beta \vee \alpha)$	commutativity of $\vee$
$((\alpha \wedge \beta) \wedge \gamma)$	$\equiv$	$(\alpha \wedge (\beta \wedge \gamma))$	associativity of $\wedge$
$((\alpha \vee \beta) \vee \gamma)$	$\equiv$	$(\alpha \vee (\beta \vee \gamma))$	associativity of $\vee$
$\neg(\neg\alpha)$	$\equiv$	α	double-negation elimination
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\beta \Rightarrow \neg\alpha)$	contraposition
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\alpha \vee \beta)$	implication elimination
$(\alpha \Leftrightarrow \beta)$	$\equiv$	$((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$	biconditional 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
$(\alpha \wedge (\beta \vee \gamma))$	$\equiv$	$((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$	distributivity of $\wedge$ over $\vee$
$(\alpha \vee (\beta \wedge \gamma))$	$\equiv$	$((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$	distributivity of $\vee$ over $\wedge$

Propositional Theorem Proving: Inference and Proof

- Modus Ponens:

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

- And Elimination:

$$\frac{\alpha \wedge \beta}{\alpha}$$

Propositional Theorem Proving: Proof by Resolution

- Unit Resolution:

$$\frac{l_1 \vee l_2 \vee \cdots \vee l_k, m}{l_1 \vee l_2 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k}$$

Where l_i and m are complementary literals.

- Full Resolution:

$$\frac{l_1 \vee l_2 \vee \cdots \vee l_k, m_1 \vee m_2 \vee \cdots \vee m_n}{l_1 \vee l_2 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k \vee m_1 \vee m_2 \vee \cdots \vee m_{j-1} \vee m_{j+1} \ldots \vee m_n}$$

- Where l_i and m_j are complementary literals.

Propositional Theorem Proving: Conjunctive Normal Form

- Eliminate ⇔

$$A \Leftrightarrow B \equiv (A \Rightarrow B) \wedge (B \Rightarrow A)$$

- Eliminate ⟹

$$(A \Rightarrow B) \equiv \neg A \vee B$$

- CNF requires ¬ to appear only in literals, apply De Morgan’s law.

$$\neg(A \wedge B) \equiv \neg A \vee \neg B$$
$$\neg(A \vee B) \equiv \neg A \wedge \neg B$$
$$\neg\neg A \equiv A$$

- Distribute⋀ and ⋁ , wherever possible.

$$A \wedge (B \vee C) \equiv (A \vee B) \wedge (A \vee C)$$

A Resolution Algorithm

- First, (KB ∧ ¬α) is converted into CNF. Then, the resolution rule is applied to the resulting clauses.
- Each pair that contains complementary literals is resolved to produce a new clause, which is added to the set if it is not already present.

➢The process continues until one of two things happens:there are no new clauses that can be added, in which case KB does not entail α; or,

➢two clauses resolve to yield the empty clause, in which case KB entails α.

Horn Clause

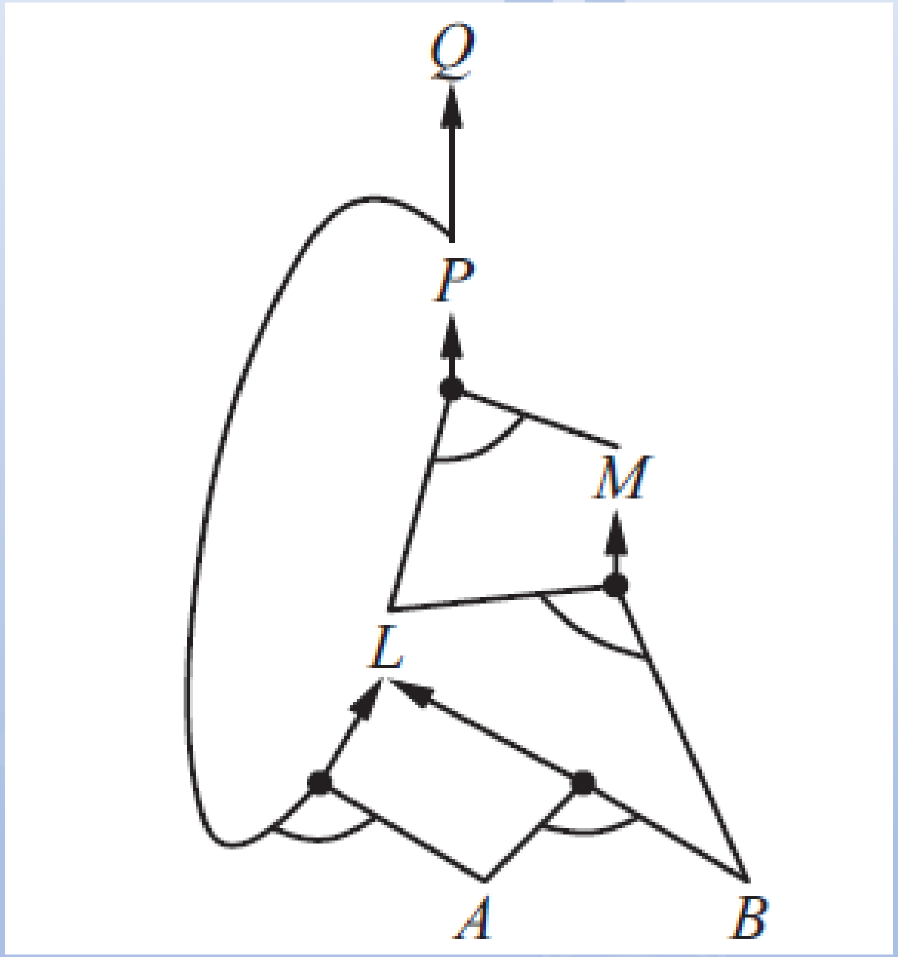
- Definite Clause: disjunction of literals of which exactly one is positive.
- Horn Clause: Horn clause, which is a disjunction of literals of which at most one is positive.

Forward Chaining

- It begins from known facts (positive literals) in the knowledge base.
- If all the premises of an implication are known, then its conclusion is added to the set of known facts.
- This process continues until the query q is added or until no further inferences can be made.

Forward Chaining

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



Backward Chaining

- The backward-chaining algorithm, as its name suggests, works backward from the query.
- If the query q is known to be true, then no work is needed. Otherwise, the algorithm finds those implications in the knowledge base whose conclusion is q.
- If all the premises of one of those implications can be proved true (by backward chaining), then q is true.
- When applied to the query Q, it works back down the graph until it reaches a set of known facts,

Thank You