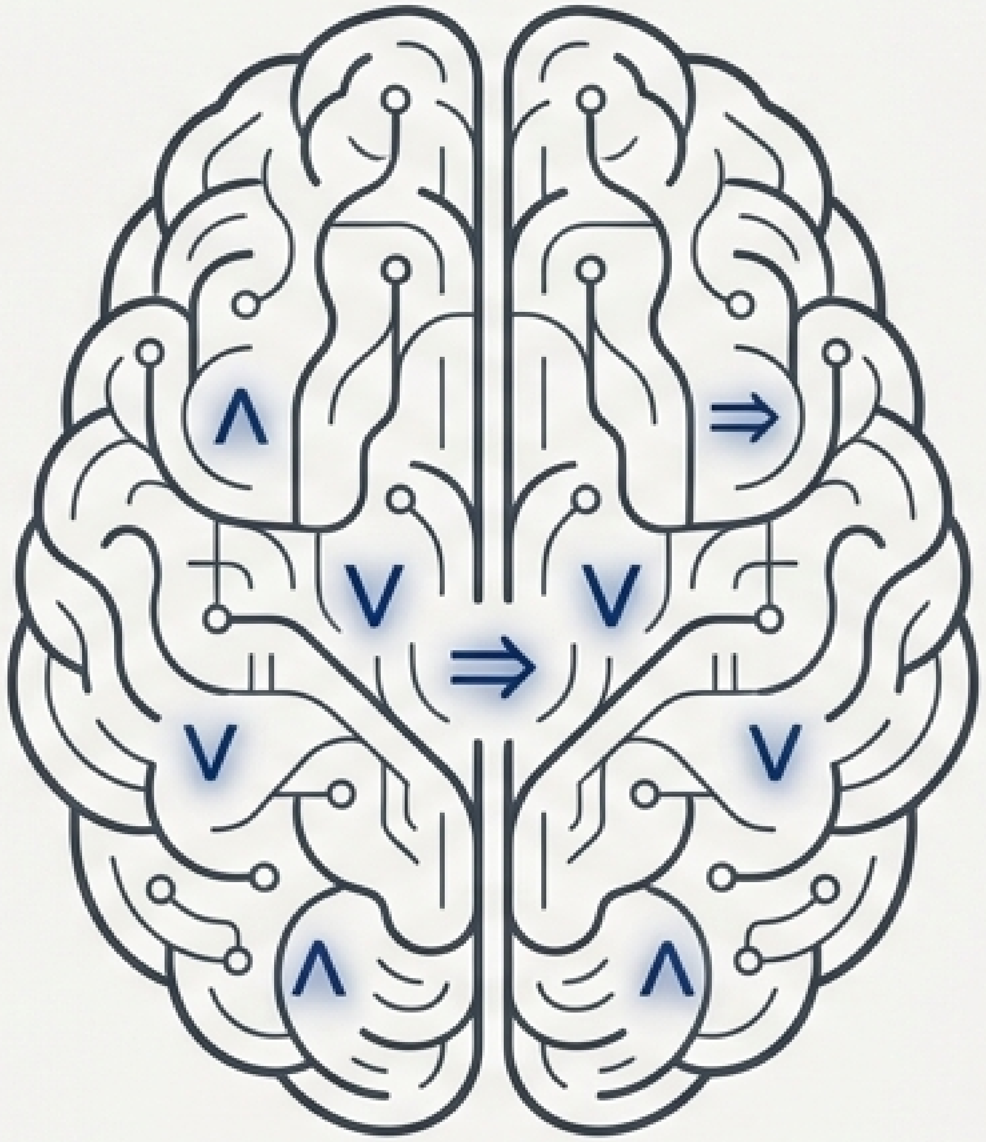
The Logic of Intelligence

A Comprehensive Summary of Logical Agents



Based on 'Fundamentals of Artificial Intelligence – Logical Agents' course materials.

An Agent's Challenge: Reasoning in the Wumpus World

Introduction

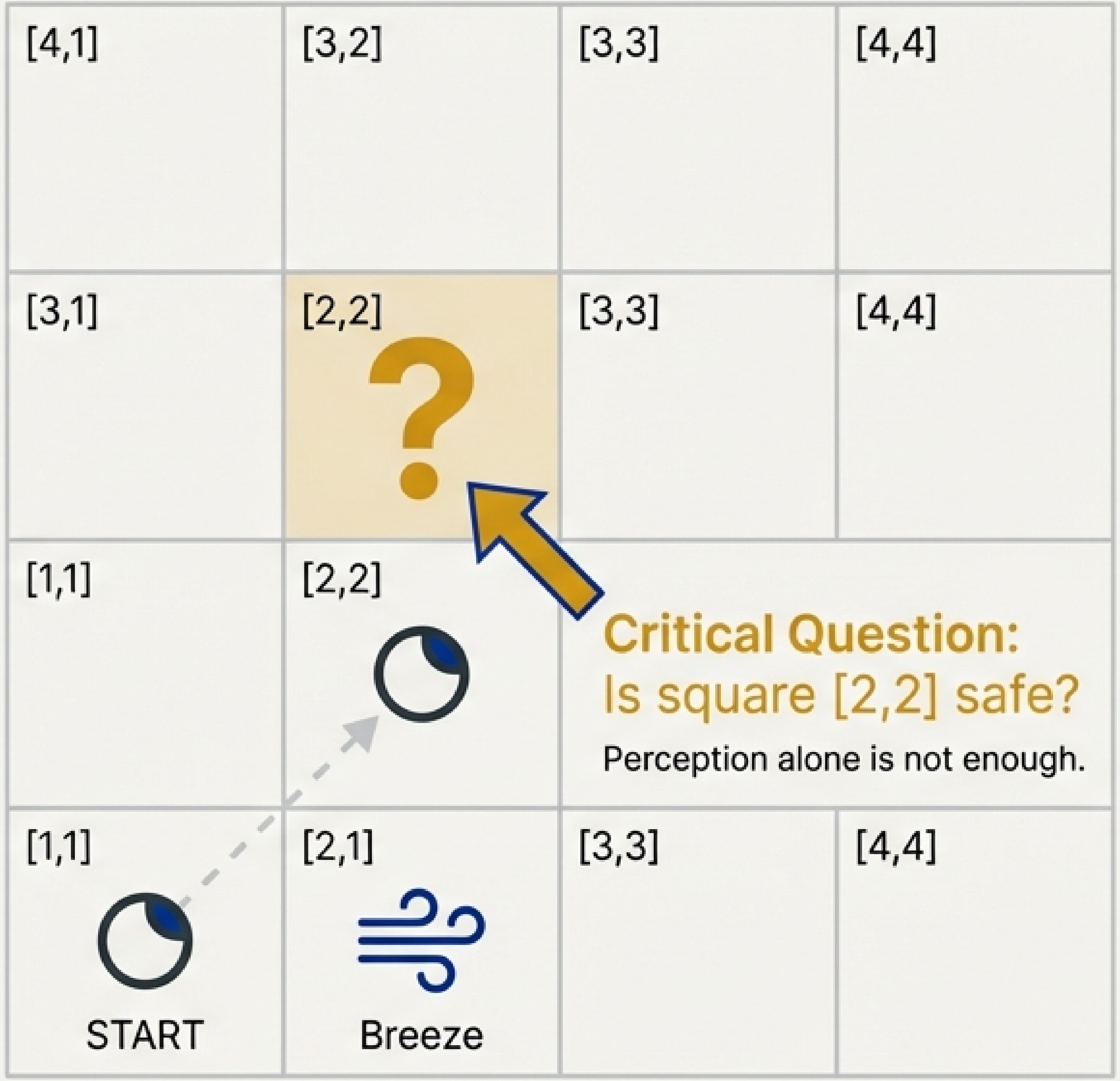
A classic AI problem environment: a cave of rooms, a Wumpus, pits, and gold. The agent's perception is local; it must infer hidden dangers.

PEAS Description

Performance: +1000 for gold, -1000 for death, -1 per step, -10 for arrow.

Environment: Partially observable (local perception), Deterministic, Sequential, Static, Discrete, Single-agent.

The Core Problem: The agent cannot see the whole map. It must use logic to reason about unseen squares.



The Language of Truth: Knowledge, Models, and Entailment

Core Components

- **Knowledge Base (KB)**: A set of sentences in a formal language representing what the agent knows.
- **Inference Engine**: Domain-independent algorithms that derive new sentences from the KB.

Foundations of Logic

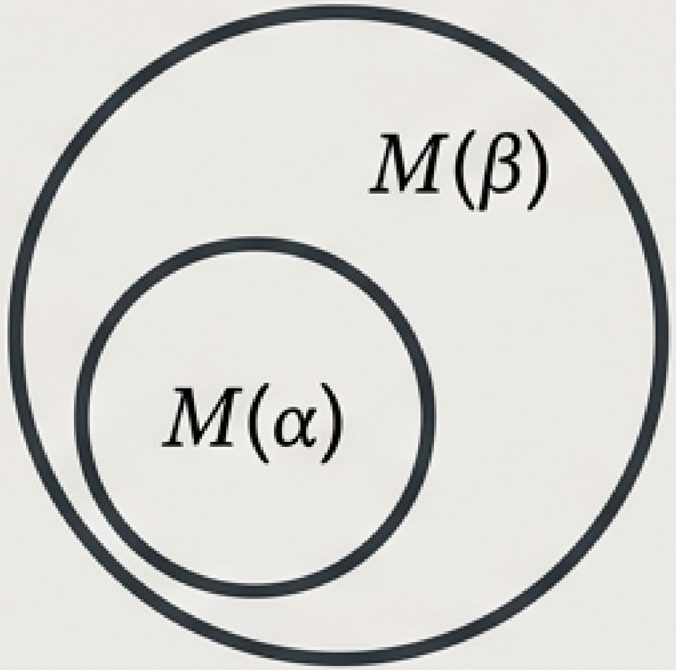
- **Syntax**: The rules for forming well-formed sentences (e.g., $A \land B$).
- **Semantics**: The meaning of sentences; the conditions under which they are true.
- **Model**: A "possible world" that assigns truth values to all proposition symbols. A model m satisfies a sentence α if α is true in m.

The Central Concept: Entailment (|=)

A sentence α entails another sentence β (written as $\alpha \models \beta$) if and only if in every model where α is true, β is also true.

$$\alpha \models \beta \text{ iff } M(\alpha) \subseteq M(\beta)$$

$M(\alpha)$ is the set of all models that satisfy α.



Our Formal Language: The Syntax and Semantics of Propositional Logic

Syntax

Atomic Propositions: Indivisible symbols representing facts (e.g., P, Q, B_1,1).

Logical Connectives:
¬ (Negation / NOT)
∧ (Conjunction / AND)
∨ (Disjunction / OR)
⇒ (Implication / IF...THEN)
⇔ (Biconditional / IF AND ONLY IF)

Semantics (Truth Table)

P	Q	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	false	false	false	true	true
false	true	false	true	true	false
true	false	false	true	false	false
true	true	true	true	true	true

Highlight Note

The implication $P \Rightarrow Q$ is true whenever its premise P is **false**. This is a crucial rule. The sentence “If 5 is even, then Tokyo is the capital of Germany” is logically true.

Classifying Sentences: Validity, Satisfiability, and Unsatisfiability

Valid (Tautology)

A sentence that is true in *all* possible models.

Example: $A \lor \neg A$

Metatheorem: A sentence α is valid if and only if $\alpha \equiv \text{True}$.
(from Problem 4.2.1)

Satisfiable

A sentence that is true in *at least one* model.

Example: $A \land B$

Note: All valid sentences are also satisfiable.

Unsatisfiable (Contradiction)

A sentence that is true in *no* models.

Example: $A \land \neg A$

Metatheorem: A sentence α is unsatisfiable if and only if $\alpha \equiv \text{False}$.
Problem 4.2.1)

Proof Method 1: Reasoning by Model Checking

To check if $KB \models \alpha$, we enumerate all models and verify that $M(KB) \subseteq M(\alpha)$. This method is also known as inference by enumeration.

Example (from Problem 4.1.5): Is the statement $(A \Leftrightarrow B) \models (\neg A \vee B)$ correct?

1. Find all models of the Knowledge Base $\alpha = (A \Leftrightarrow B)$

The sentence is true only when A and B have the same truth value.

$$M(A \Leftrightarrow B) = \{(\text{True}, \text{True}), (\text{False}, \text{False})\}$$

2. Find all models of the query $\beta = (\neg A \vee B)$

$$M(\neg A \vee B) = \{(\text{True}, \text{True}), (\text{False}, \text{True}), (\text{False}, \text{False})\}$$

3. Check if $M(\alpha) \subseteq M(\beta)$

Is {**(True, True)**, **(False, False)**} a subset of {**(True, True)**, **(False, True)**, **(False, False)**}?

Conclusion: Yes. The entailment holds.

Limitation: This method is computationally expensive, with O(2^n) complexity for n symbols.

Proof Method 2: Reasoning by Logical Equivalence

Concept

Apply a series of standard logical equivalences to transform a sentence into either True (proving it is valid) or False (proving it is unsatisfiable).

Key Equivalences

$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$ (Implication Elimination)

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

$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$ (Distributivity)

Example (from Problem 4.2.2): Is (Fire ⇒ Smoke) ∧ Fire ∧ ¬Smoke satisfiable?

(¬Fire ∨ Smoke) ∧ Fire ∧ ¬Smoke (Implication Elimination)

((¬Fire ∧ Fire) ∨ (Smoke ∧ Fire)) ∧ ¬Smoke (Distributivity)

(False ∨ (Smoke ∧ Fire)) ∧ ¬Smoke (Contradiction: $\alpha \wedge \neg\alpha \equiv$ False)

(Smoke ∧ Fire) ∧ ¬Smoke (Identity: $\alpha \vee$ False $\equiv \alpha$)

Smoke ∧ ¬Smoke ∧ Fire (Commutativity)

False ∧ Fire (Contradiction)

False (Domination: $\alpha \wedge$ False $\equiv$ False)

Conclusion: The sentence simplifies to False, therefore it is **unsatisfiable**.

Proof Method 3: Reasoning with Inference Rules

An inference procedure uses sound inference rules to generate new, true sentences from an existing Knowledge Base.

Case Study: Knights & Knaves (Problem 4.3)

On an island of Knights (always tell the truth) and Knaves (always lie). A, B, C are true if the person is a Knight.

Formalizing Statements: The rule `*Person P says Remark*' is formalized as $P \Leftrightarrow Remark$.

Knowledge Base (KB)

1. B says "A said he is a knave": $B \Leftrightarrow (A \Leftrightarrow \neg A)$
2. C says "B is lying": $C \Leftrightarrow \neg B$

Deduction

From (1): The expression $(A \Leftrightarrow \neg A)$ is a contradiction, equivalent to False. The sentence simplifies to

$$B \Leftrightarrow \text{False}$$

Using the equivalence $(X \Leftrightarrow \text{False}) \equiv \neg X$, we derive $\neg B$.

Conclusion 1: B is a Knave.

From (2): We have the rule $C \Leftrightarrow \neg B$ and we have just derived $\neg B$.

By inference, from $C \Leftrightarrow \neg B$ and $\neg B$, we conclude C.

Conclusion 2: C is a Knight.

Proof Method 4: Resolution, the Universal Inference Rule

The Big Idea

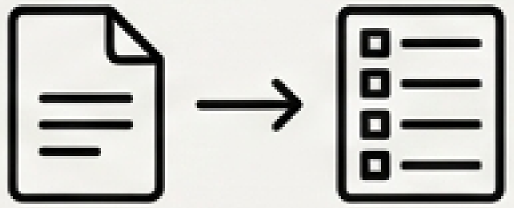
Resolution is a single, powerful inference rule that is sound and complete for propositional logic. It works via **proof by contradiction**.

The Strategy

- To prove $KB \models \alpha$, we instead prove that $KB \land \neg\alpha$ is **unsatisfiable**.
- If $KB \land \neg\alpha$ leads to a logical contradiction (**False**), then our original goal (α) must be true.

The Three-Step Process

1. Convert to CNF



Convert all sentences in $KB \land \neg\alpha$ into Conjunctive Normal Form (a conjunction of clauses).

2. Apply Resolution

$$\left.\begin{matrix}(P \lor Q) \\ (\neg P \lor R)\end{matrix}\right\} (Q \lor R)$$

Repeatedly apply the resolution rule to pairs of clauses with complementary literals to generate new clauses.

3. Find Contradiction

$$\square$$

If the process derives the **empty clause** (**False**), the proof is complete.

Resolution in Action: The Superman Problem

The Argument (from Problem 4.4)

1. If Superman were able and willing to prevent evil, he would do so.
2. If Superman were unable to prevent evil, he would be impotent.
3. If he were unwilling to prevent evil, he would be malevolent.
4. Superman does not prevent evil.
5. If Superman exists, he is neither impotent nor malevolent.

Therefore, Superman does not exist.

Formalization: Knowledge Base (KB)

A: Able, W: Willing, P: Prevents evil,
I: Impotent, M: Malevolent, E: Exists

1. $(A \wedge W) \Rightarrow P$
2. $\neg A \Rightarrow I$
3. $\neg W \Rightarrow M$
4. $\neg P$
5. $E \Rightarrow (\neg I \wedge \neg M)$

Goal: Prove $\neg E$ (Superman does not exist).

Proof Strategy: We will show that $KB \wedge \neg(\neg E)$ (which is $KB \wedge E$) is unsatisfiable.

Step 1: Converting the Knowledge Base to CNF

Objective: Transform the single sentence $KB \wedge E$ into a set of clauses (disjunctions of literals).

Conversion Process

1. Eliminate Implications (⇒): Replace $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$

- $(A \wedge W) \Rightarrow P$ becomes $\neg(A \wedge W) \vee P$
- $E \Rightarrow (\neg I \wedge \neg M)$ becomes $\neg E \vee (\neg I \wedge \neg M)$

2. Move Negation (`¬) Inwards (De Morgan's): Replace $\neg(\alpha \wedge \beta)$ with $\neg\alpha \vee \neg\beta$

- $\neg(A \wedge W) \vee P$ becomes $\neg A \vee \neg W \vee P$

3. Distribute `∨` over `∧`: Replace $\alpha \vee (\beta \wedge \gamma)$ with $(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$

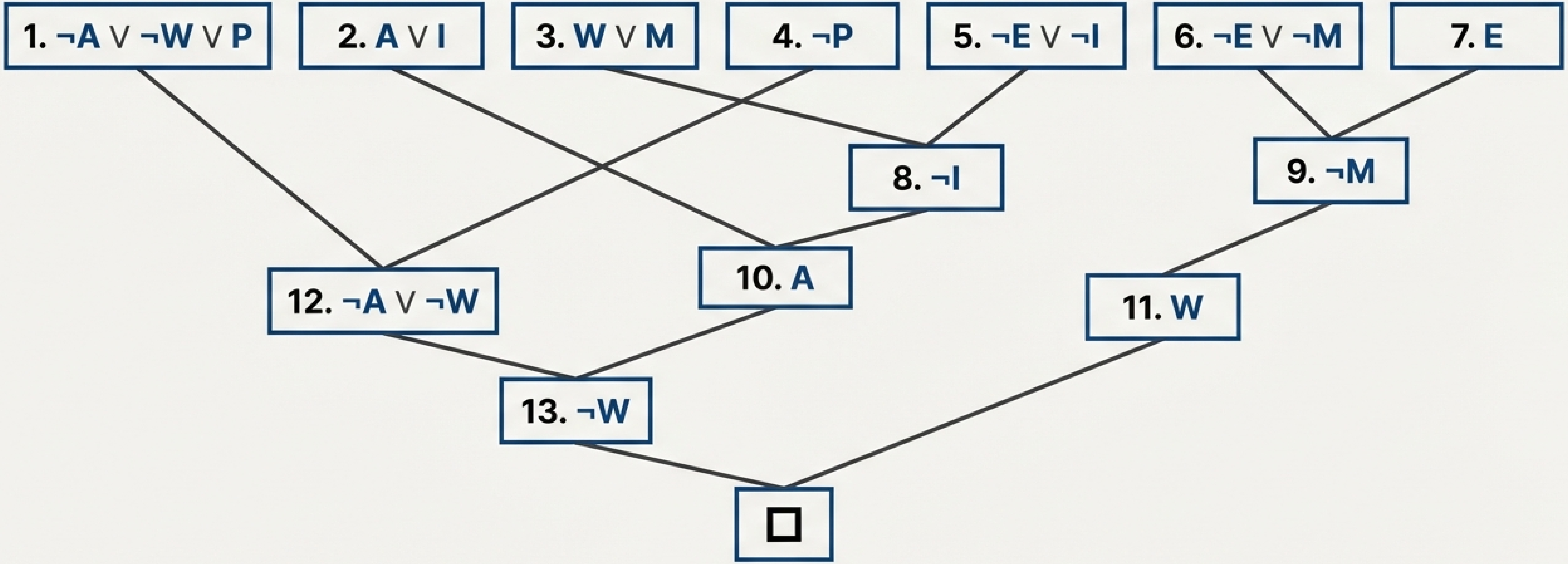
- $\neg E \vee (\neg I \wedge \neg M)$ becomes $(\neg E \vee \neg I) \wedge (\neg E \vee \neg M)$

The Final Set of Clauses

1. $\neg A \vee \neg W \vee P$
2. $A \vee I$
3. $W \vee M$
4. $\neg P$
5. $\neg E \vee \neg I$
6. $\neg E \vee \neg M$
7. E

Step 2: The Resolution Proof

Objective: Repeatedly apply the resolution rule to derive the empty clause.



Conclusion: The empty clause was derived, so **KB ∧ E** is unsatisfiable. Therefore, we have proven **KB |= ¬E**.

A Special Case for Efficient Inference: Horn Clauses

Definition

A clause with at most one positive literal. Two common forms:

1. **Definite Clause**: An implication with a conjunction of positive literals as the premise and a single positive literal as the conclusion.

$$(P_1 \land P_2 \land \ldots \land P_n) \Rightarrow Q$$

2. **Fact**: A single positive literal.

$$P \text{ (equivalent to True} \Rightarrow P\text{)}$$

Why They Matter

Inference with Horn clauses is very efficient, running in **linear time**.

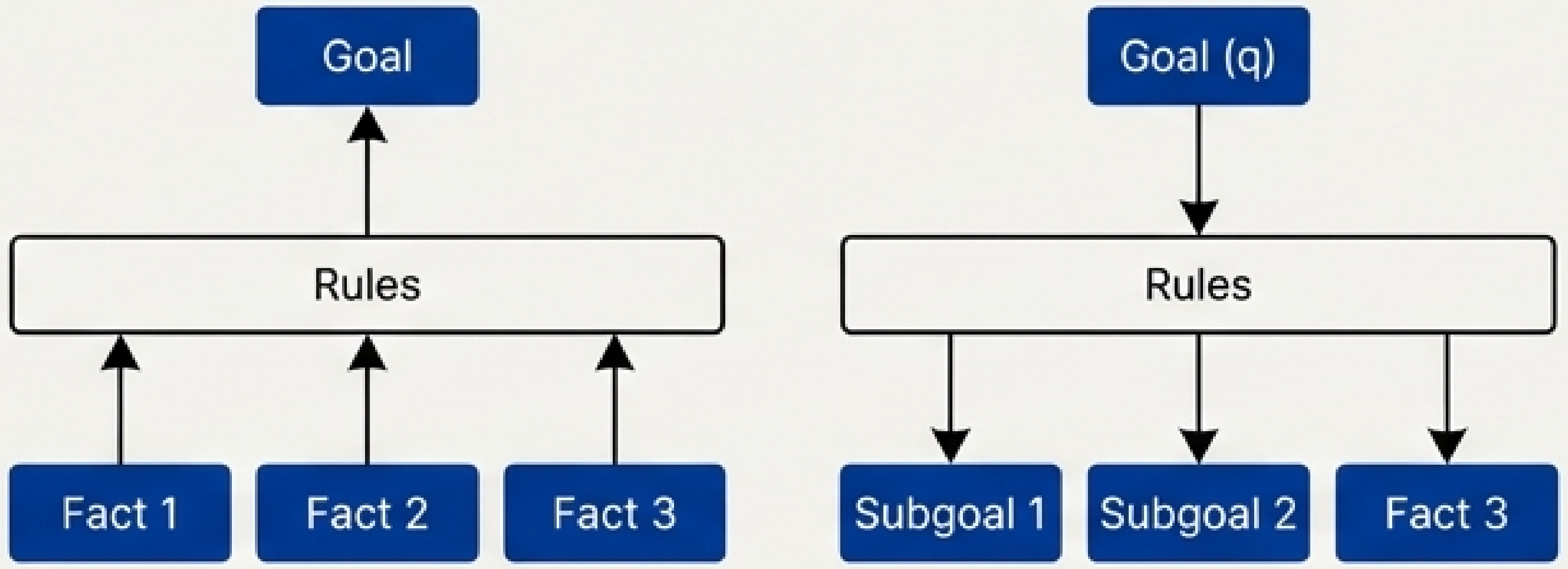
Specialized Inference Algorithms

Forward Chaining

Data-driven. Starts with known facts in the KB. Fires rules whose premises are satisfied to add new facts until the goal is derived.

Backward Chaining

Goal-driven. Starts with the query q. Finds rules that conclude q and then tries to prove their premises recursively.



How Good is Our Inference? Soundness and Completeness

Soundness: The algorithm doesn't make things up.

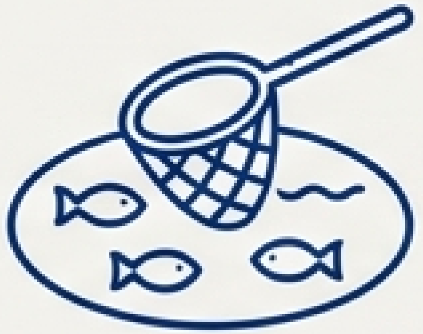
An inference algorithm is **sound** if it *only* derives sentences that are actually entailed by the knowledge base ($KB \models \alpha$).

An algorithm that *never* derives anything is perfectly **sound** (but useless), because it never makes a false claim. (from Problem 4.5.2)

Completeness: The algorithm can find any truth.

An inference algorithm is **complete** if it *can* derive *any* sentence that is entailed by the knowledge base.

An algorithm that can derive *every possible sentence* is perfectly **complete** (but unsound), because it is guaranteed to derive the entailed ones. (from Problem 4.5.1)



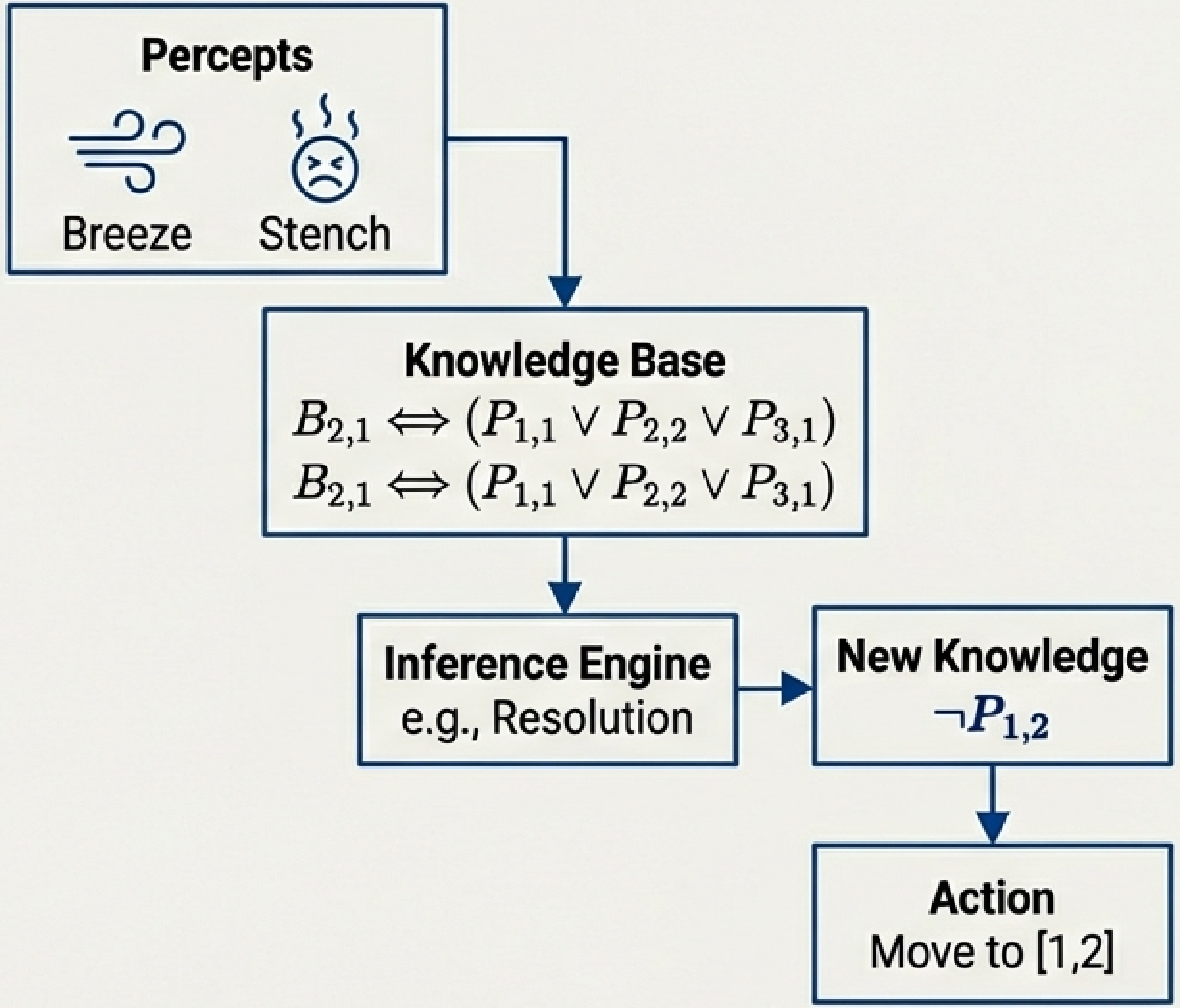
The Goal: To use algorithms like **Resolution**, which is both **sound** and **complete** for propositional logic.

The Logical Agent: From Perception to Proof

Key Takeaways

- Logical agents use a **Knowledge Base (KB)** to represent facts about the world.
- **Propositional Logic** provides the **formal syntax** and **semantics** for this knowledge.

- **Entailment ($|=$)** is the core relationship for deriving new knowledge.
- **Inference Procedures** are the algorithms that perform this derivation:
  - **Model Checking**: Direct but **computationally expensive** ($O(2^n)$).
  - **Theorem Proving (Resolution)**: Powerful, sound, and complete.
  - **Chaining (Forward/Backward)**: Highly efficient for Horn Clauses.



By combining a formal language with a sound and complete inference procedure, an agent can reason about its world, uncovering hidden facts to make intelligent decisions.