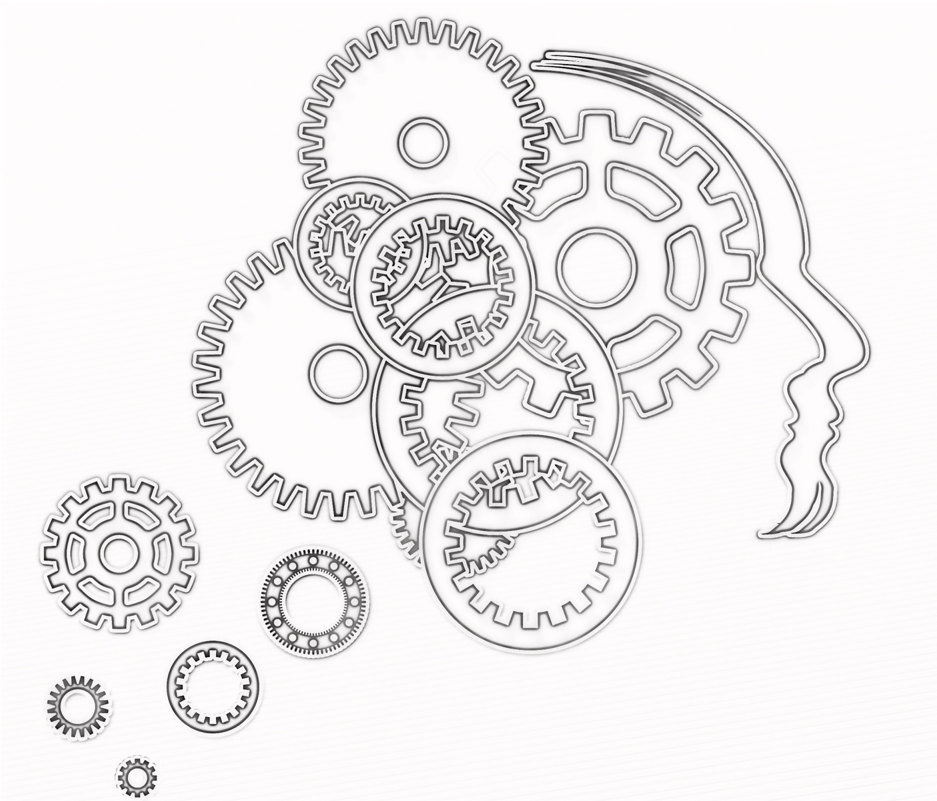
Artificial Intelligence

Logical Agents



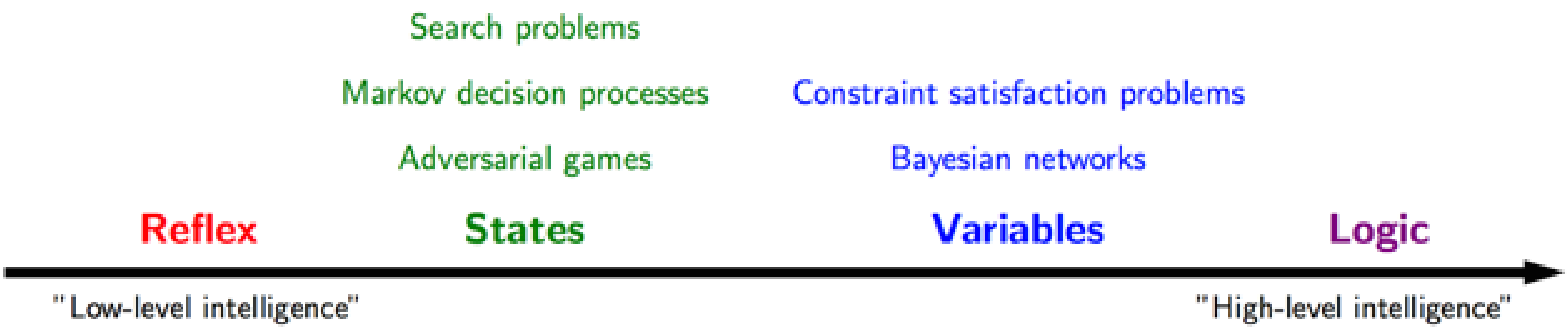
Knowledge-based agents

"**Logical AI:**
The idea is that an agent can represent knowledge of its world, its goals and the current situation by sentences in logic and decide what to do by inferring that a certain action or course of action is appropriate to achieve its goals."

John McCarthy in Concepts of logical AI, 2000.

`http://www-formal.stanford.edu/jmc/concepts-ai/concepts-ai.html`

Knowledge-based agents



Knowledge-based agents

- Intelligent agents need **knowledge** about the world to choose good actions/decisions.

- Knowledge = $\{$**sentences**$\}$ in a knowledge representation language (formal language).

- A sentence is an assertion about the world.

- A knowledge-based agent is composed of:

  1. Knowledge base: **domain-specific** content.

  2. Inference mechanism: **domain-independent** algorithms.

Knowledge-based agents

- The agent **must be able to**:
  - Represent states, actions, etc.
  - Incorporate new percepts
  - Update internal representations of the world
  - Deduce hidden properties of the world
  - Deduce appropriate actions

Knowledge-based agents

- The agent **must be able to**:
  - Represent states, actions, etc.
  - Incorporate new percepts
  - Update internal representations of the world
  - Deduce hidden properties of the world
  - Deduce appropriate actions
- **Declarative** approach to building an agent:
  - Add new sentences: **Tell** it what it needs to know
  - Query what is known: **Ask** itself what to do - answers should follow from the KB

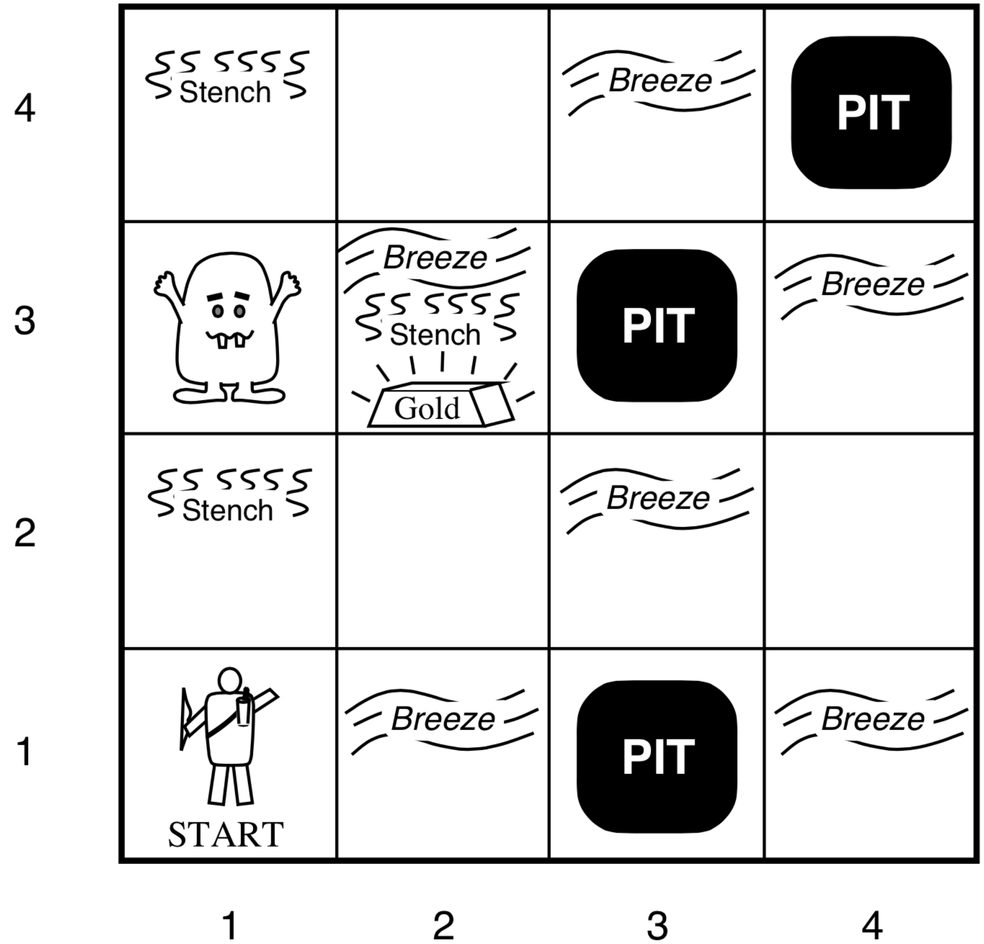
Knowledge-based agents

```
function KB-AGENT(percept) returns an action
    static: 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 ← t + 1
    return action
```

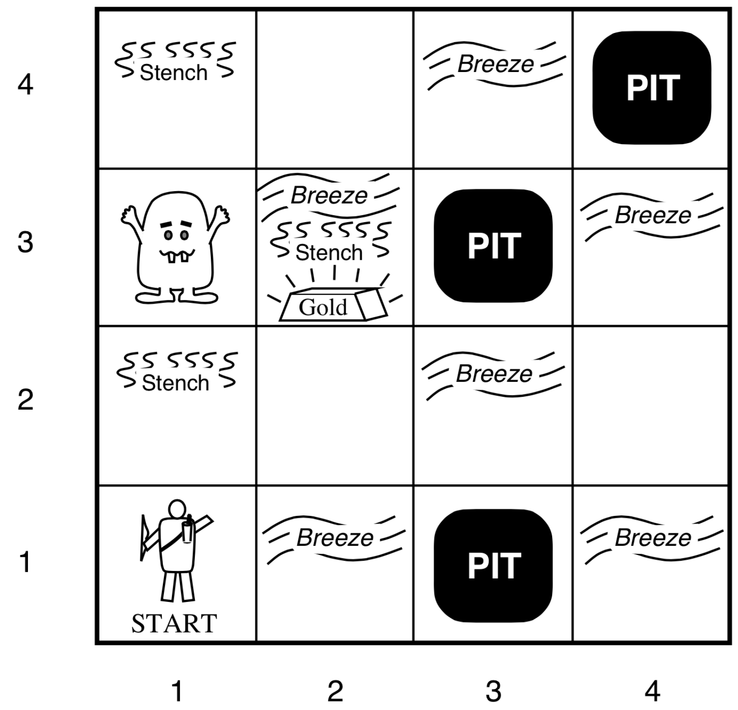
The Wumpus World

Gregory Yob (1975)



The Wumpus World

- 4 X 4 grid of rooms
- Squares adjacent to Wumpus are smelly and squares adjacent to pit are breezy
- Glitter iff gold is in the same square
- Shooting kills Wumpus if you are facing it
- Wumpus emits a horrible scream when it is killed that can be heard anywhere
- Shooting uses up the only arrow
- Grabbing picks up gold if in same square
- Releasing drops the gold in same square



Wumpus World PEAS

- **Performance measure**: gold +1000, death (eaten or falling in a pit) -1000, -1 per action taken, -10 for using the arrow. The games ends either when the agent dies or comes out of the cave.
- **Environment**
  - 4 X 4 grid of rooms
  - Agent starts in square [1,1] facing to the right
  - Locations of the gold, and Wumpus are chosen randomly with a uniform distribution from all squares except [1,1]
  - Each square other than the start can be a pit with probability of 0.2

Wumpus World PEAS

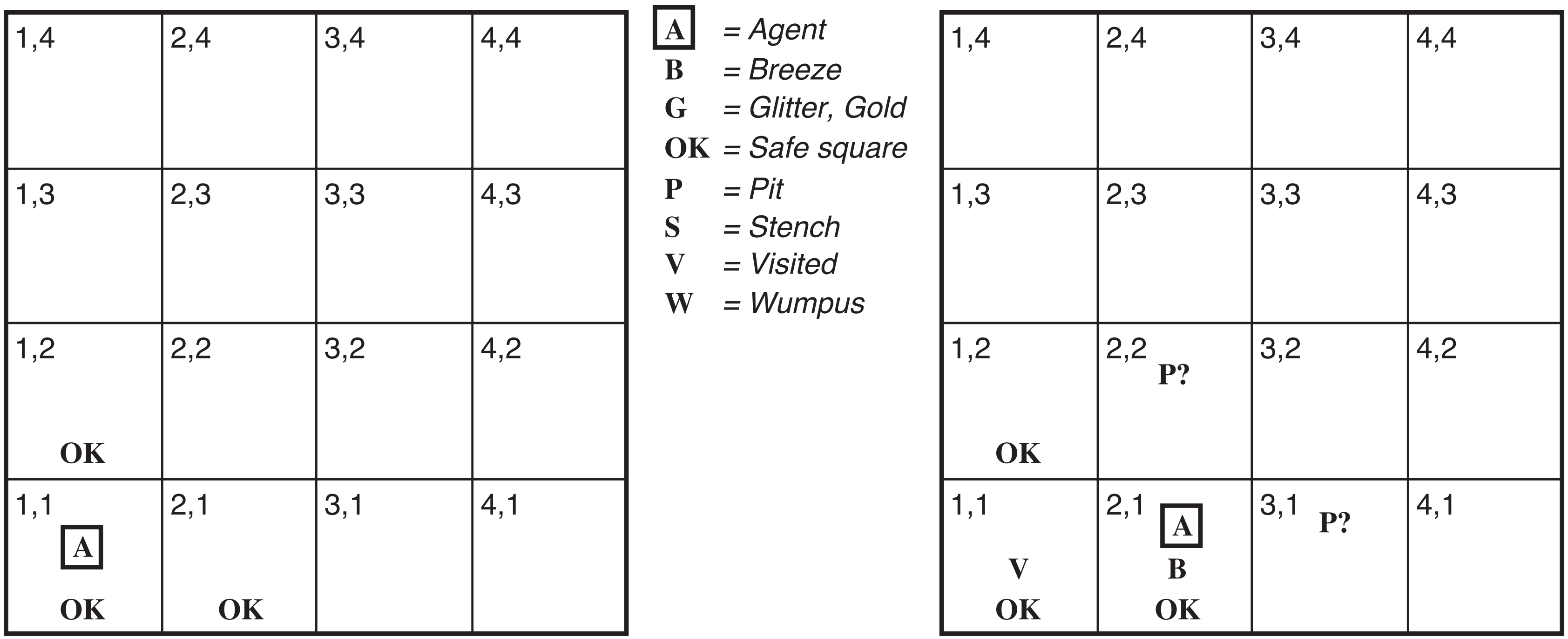
- **Actuators**:
  - Left turn, Right turn, Forward, Grab, Release, Shoot
- **Sensors**:
  - Stench, Breeze, Glitter, Bump, Scream
  - Represented as a 5-element list
  - Example: [Stench, Breeze, None, None, None]

Wumpus World properties

- Partially observable
- Static
- Discrete
- Single-agent
- Deterministic
- Sequential

Exploring Wumpus World

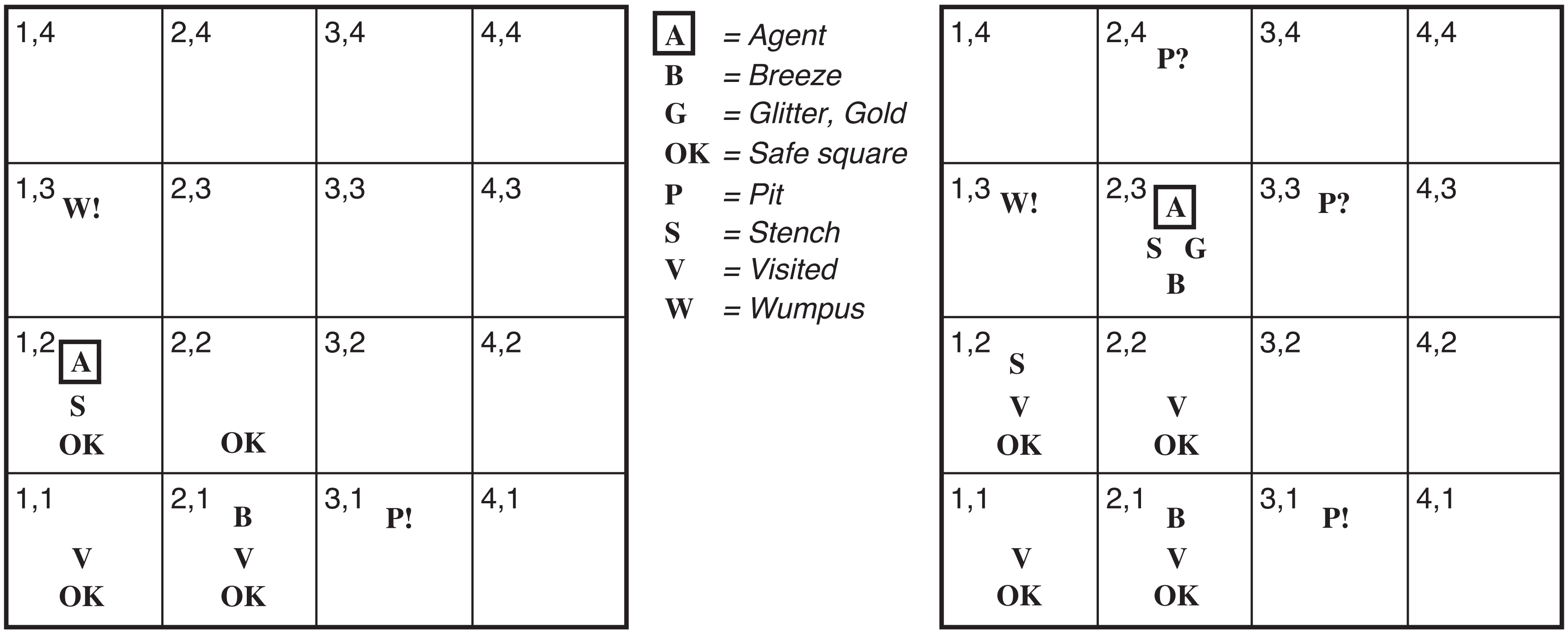
Agent's first steps:



(a) (b)

Exploring Wumpus World

Agent's later steps:



(a) (b)

Logic

- **Knowledge base**: a set of sentences in a formal representation, logic
- **Logics**: are formal languages for representing knowledge to extract conclusions
  - **Syntax**: defines well-formed sentences in the language
  - **Semantic**: defines the truth or meaning of sentences in a world
- **Inference**: a procedure to derive a new sentence from other ones.
- **Logical entailment**: is a relationship between sentences. It means that a sentence follows logically from other sentences

$$KB \models \alpha$$

Propositional logic

- Propositional logic (PL) is the simplest logic.
- **Syntax of PL**: defines the allowable sentences or propositions.
- **Definition (Proposition)**: A proposition is a declarative statement that's either True or False.
- **Atomic proposition**: single proposition symbol. Each symbol is a proposition. Notation: upper case letters and may contain subscripts.
- **Compound proposition**: constructed from atomic propositions using parentheses and logical connectives.

Atomic proposition

Examples of atomic propositions:

- 2+2=4 is a true proposition
- $W_{1,3}$ is a proposition. It is true if there is a Wumpus in [1,3]
- "If there is a stench in [1,2] then there is a Wumpus in [1,3]" is a proposition
- "How are you?" or "Hello!" are not propositions. In general, statement that are questions, commands, or opinions are not propositions.

Compound proposition

Examples of compound/complex propositions:

Let p, p_1, and p_2 be propositions

- **Negation** $\neg p$ is also a proposition. We call a **literal** either an atomic proposition or its negation. E.g., $W_{1,3}$ is a positive literal, and $\neg W_{1,3}$ is a negative literal.
- **Conjunction** $p_1 \wedge p_2$. E.g., $W_{1,3} \wedge P_{3,1}$
- **Disjunction** $p_1 \vee p_2$ E.g., $W_{1,3} \vee P_{3,1}$
- **Implication** $p_1 \rightarrow p_2$. E.g., $W_{1,3} \wedge P_{3,1} \rightarrow \neg W_{2,2}$
- **If and only if** $p_1 \leftrightarrow p_2$. E.g., $W_{1,3} \leftrightarrow \neg W_{2,2}$

Truth tables

- The semantics define the rules to determine the truth of a sentence.
- Semantics can be specified by truth tables.
- Boolean values domain: T,F
- n-tuple: $(x_1, x_2, ..., x_n)$
- Operator on n-tuples : $g(x_1 = v_1, x_2 = v_2, ..., x_n = v_n)$
- Definition: A truth table defines an operator g on n- tuples by specifying a boolean value for each tuple
- Number of rows in a truth table? $R = 2^n$

Building propositions

Negation:

p	$\neg p$
T	F
F	T

Building propositions

Conjunction:

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

Building propositions

Disjunction:

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

Building propositions

Exclusive or:

p	q	$p \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

Building propositions

Implication:

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

Building propositions

Biconditional or If and only if (IFF):

p	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

Precedence of operators

- Just like arithmetic operators, there is an operator precedence when evaluating logical operators as follows:
  1. Expressions in parentheses are processed (inside to outside)
  2. Negation
  3. AND
  4. OR
  5. Implication
  6. Biconditional
  7. Left to right
- Use parentheses whenever you have any doubt!

Building propositions

p	q	r	¬r	p ∨ q	p ∨ q → ¬r
T	T	T	F	T	F
T	T	F	T	T	T
T	F	T	F	T	F
T	F	F	T	T	T
F	T	T	F	T	F
F	T	F	T	T	T
F	F	T	F	F	T
F	F	F	T	F	T

Logical equivalence

- Two propositions p and q are logically equivalent if and only if the columns in the truth table giving their truth values agree.
- We write this as $p \Leftrightarrow q$ or $p \equiv q$.

p	q	$\neg p$	$\neg p \vee q$	$p \rightarrow q$
T	T	F	T	T
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

Properties of operators

- Commutativity: $p \wedge q = q \wedge p \qquad p \vee q = q \vee p$
- Associativity: $(p \wedge q) \wedge r = p \wedge (q \wedge r) \qquad (p \vee q) \vee r = p \vee (q \vee r)$
- Identity element: $p \wedge True = p \qquad p \vee True = True$
- $\neg(\neg p) = p$
- $p \wedge p = p \qquad p \vee p = p$
- Distributivity:
  $p \wedge (q \vee r) = (p \wedge q) \vee (p \wedge r)$
  $p \vee (q \wedge r) = (p \vee q) \wedge (p \vee r)$
- $p \wedge (\neg p) = False$ and $p \vee (\neg p) = True$
- DeMorgan's laws:
  $\neg(p \wedge q) = (\neg p) \vee (\neg q)$
  $\neg(p \vee q) = (\neg p) \wedge (\neg q)$

Tautology and contradiction

- **Tautology** is a proposition which is always true
- **Contradiction** is a proposition which is always false
- **Contingency** is a proposition which is neither a tautology or a contradiction

P	$\neg p$	$p \vee \neg p$	$p \wedge \neg p$
T	F	T	F
F	T	T	F

Contrapositive, inverse, etc.

- Given an **implication** $p \to q$
- The **converse** is: $q \to p$
- The **contrapositive** is: $\neg q \to \neg p$
- The **inverse** is: $\neg p \to \neg q$

Inference (Modus Ponens)

$$\frac{p \qquad\quad p \rightarrow q}{q}$$

Inference (Modus Ponens)

$$\frac{p \qquad p \rightarrow q}{q}$$

$$\frac{warm \qquad warm \rightarrow sunny}{sunny}$$

Inference (Modus Ponens)

Horn clauses: a proposition of the form:

$$p_1 \wedge \ldots \wedge p_n \rightarrow q$$

Modus Ponens deals with **Horn clauses**:

$$\frac{p_1, \ldots, p_n \qquad (p_1 \wedge \ldots \wedge p_n) \rightarrow q}{q}$$

Inference (Modus Tollens)

$$\frac{\neg q \qquad\qquad p \rightarrow q}{\neg p}$$

Inference (Modus Tollens)

$$\frac{\neg q \qquad p \rightarrow q}{\neg p}$$

$$\frac{\neg beach \qquad hot \rightarrow beach}{\neg hot}$$

Common Rules

- Addition: $$\frac{p}{p \vee q}$$
- Simplification: $$\frac{p \wedge q}{q}$$
- Disjunctive-syllogism: $$\frac{\begin{array}{c} p \vee q \\ \neg p \end{array}}{q}$$
- Hypothetical-syllogism: $$\frac{\begin{array}{c} p \to q \\ q \to r \end{array}}{p \to r}$$

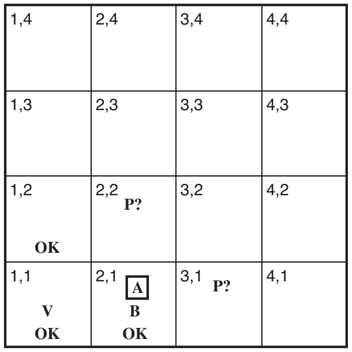
Truth Tables for connectives

Summary:

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*

Wumpus world KB

Let's build the KB for the reduced Wumpus world.



- Let $P_{i,j}$ be true if there is a pit in $[i, j]$
- Let $B_{i,j}$ be true if there is a breeze in $[i, j]$

  $\neg P_{1,1}$

- "A square is breezy if and only if there is an adjacent pit"

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

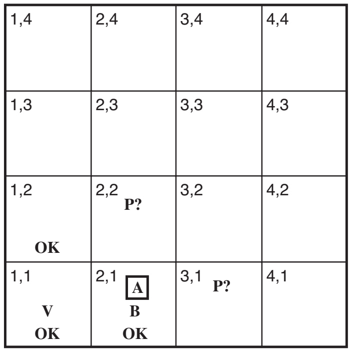
  $B_{2,1} \Leftrightarrow P_{1,1} \vee P_{2,2} \vee P_{3,1}$

  $\neg B_{1,1}$

  $B_{2,1}$

Wumpus world KB

Let's build the KB for the reduced Wumpus world.



- Let $P_{i,j}$ be true if there is a pit in $[i, j]$
- Let $B_{i,j}$ be true if there is a breeze in $[i, j]$

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

- "A square is breezy if and only if there is an adjacent pit"

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

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

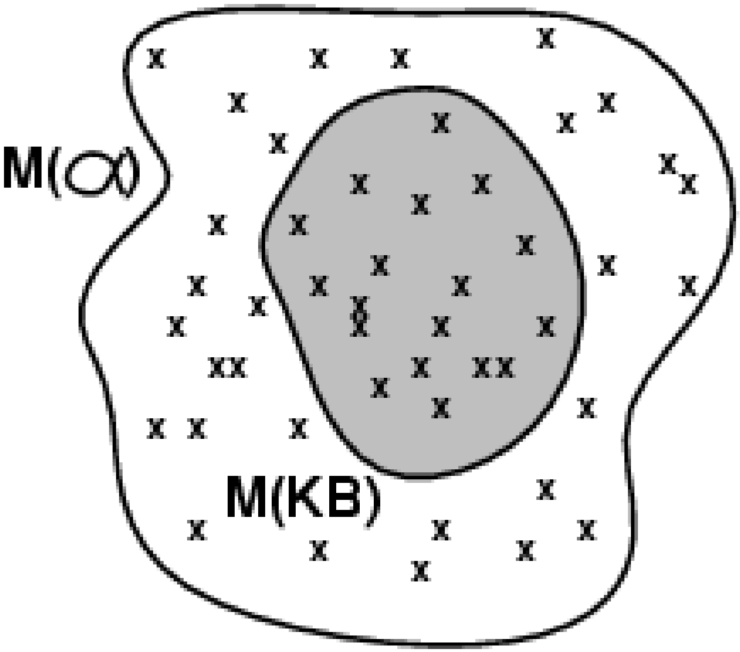
R_4: $\neg B_{1,1}$

R_5: $B_{2,1}$

Questions: Based on this KB, is $KB \models P_{1,2}$? Is $KB \models P_{2,2}$?

Models

- Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated
- We say *m* is a model of a sentence α if α is true in *m*
  - Alternative phrasing: m satisfies a
- *M(α)* is the set of all models of α
  - All models where α is true
- Then KB $\models$ α iff *M(KB)* $\subseteq$ *M*(α)
  - E.g. *KB* = Giants won and Reds won α = Giants won

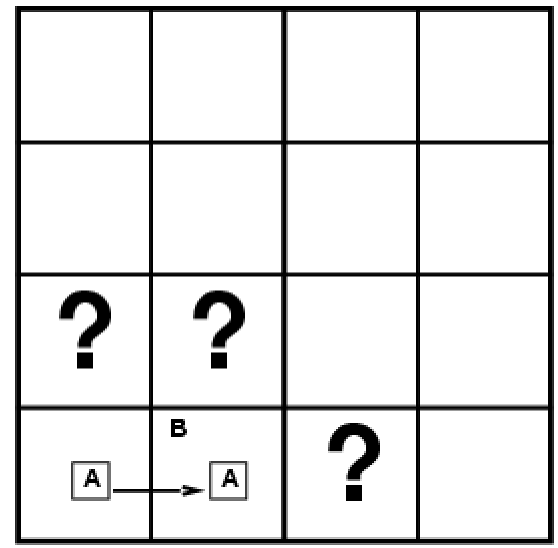


Entailment in the Wumpus World

Situation after detecting nothing in [1,1], moving right, breeze in [2,1]

Consider possible models for *KB* assuming only pits

? Boolean choices

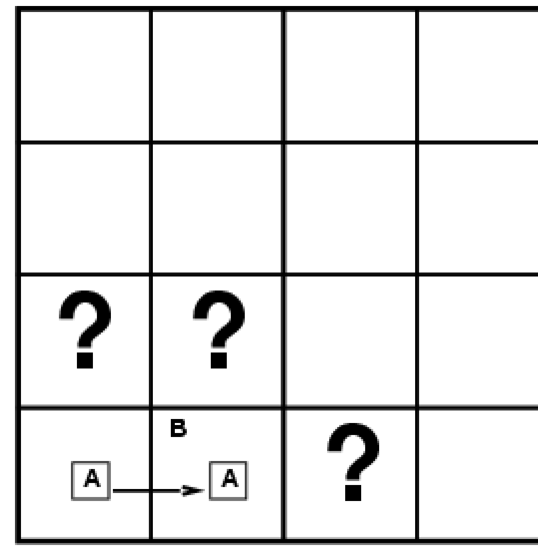


Entailment in the Wumpus World

Situation after detecting nothing in [1,1], moving right, breeze in [2,1]

Consider possible models for *KB* assuming only pits

3 Boolean choices $\Rightarrow$? possible models

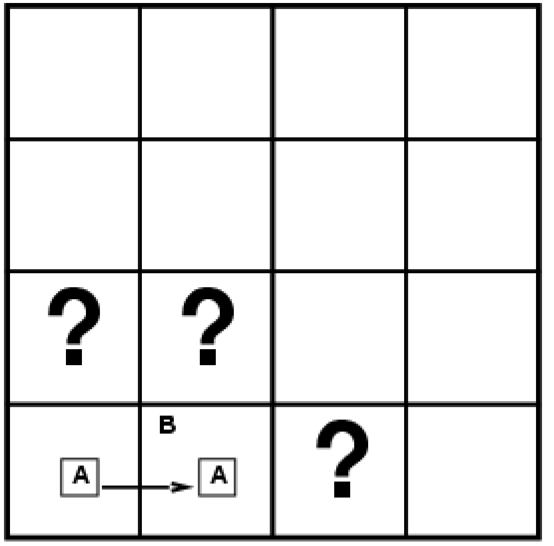


Entailment in the Wumpus World

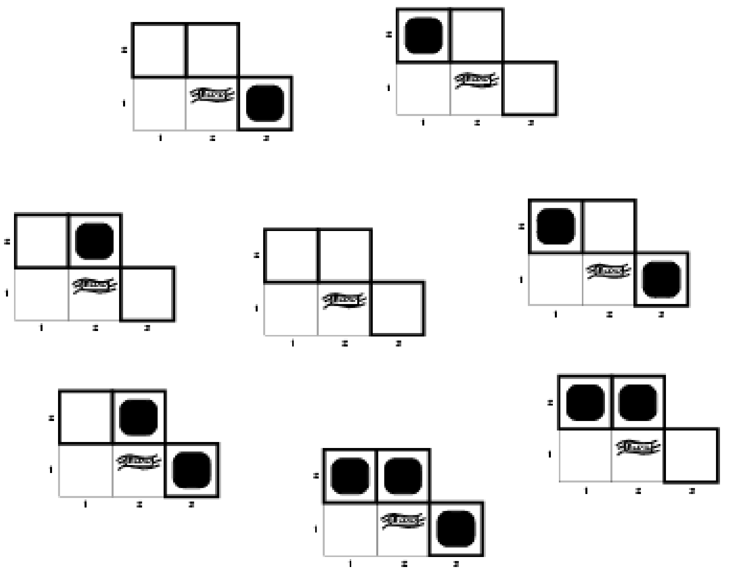
Situation after detecting nothing in [1,1], moving right, breeze in [2,1]

Consider possible models for *KB* assuming only pits

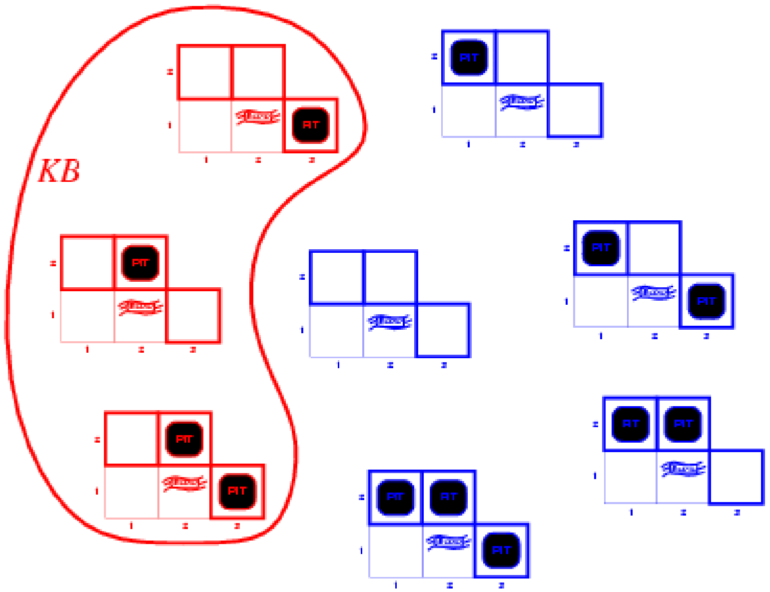
3 Boolean choices $\Rightarrow$ 8 possible models



Wumpus Models

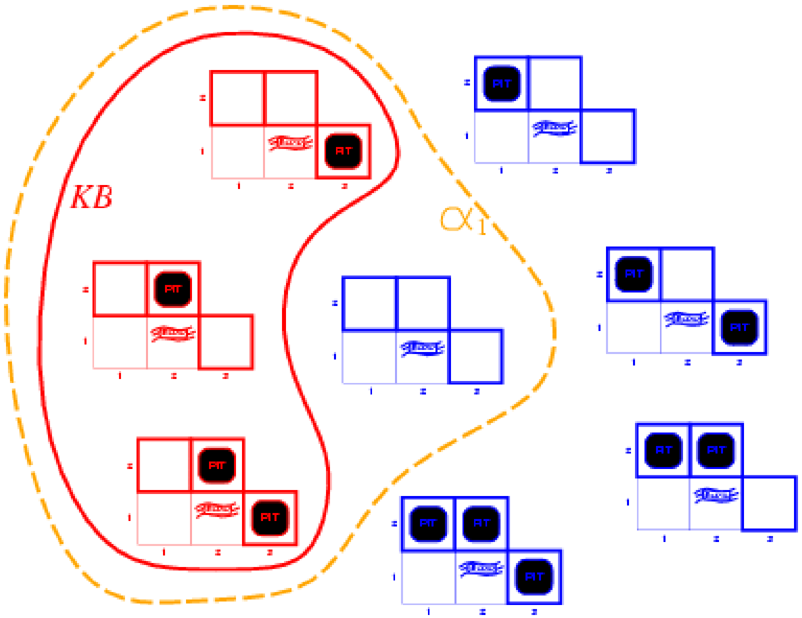


Wumpus Models



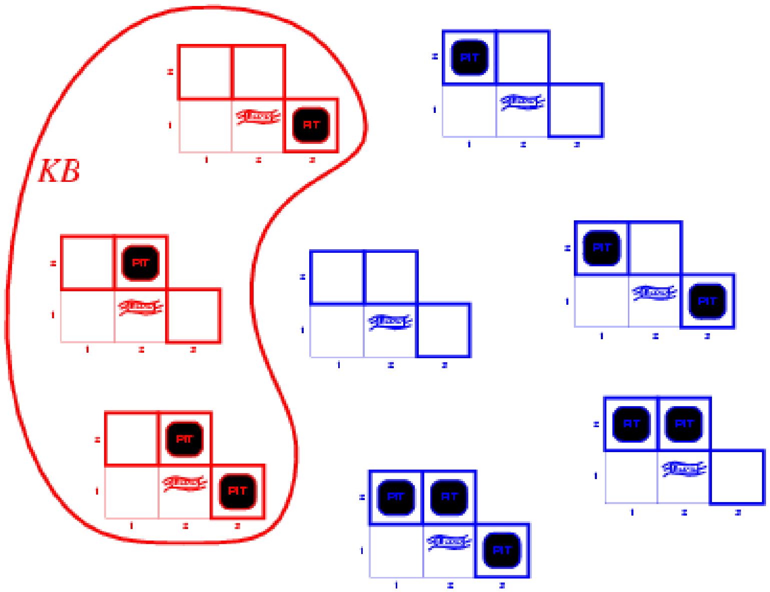
- *KB* = wumpus-world rules + observations

Wumpus Models



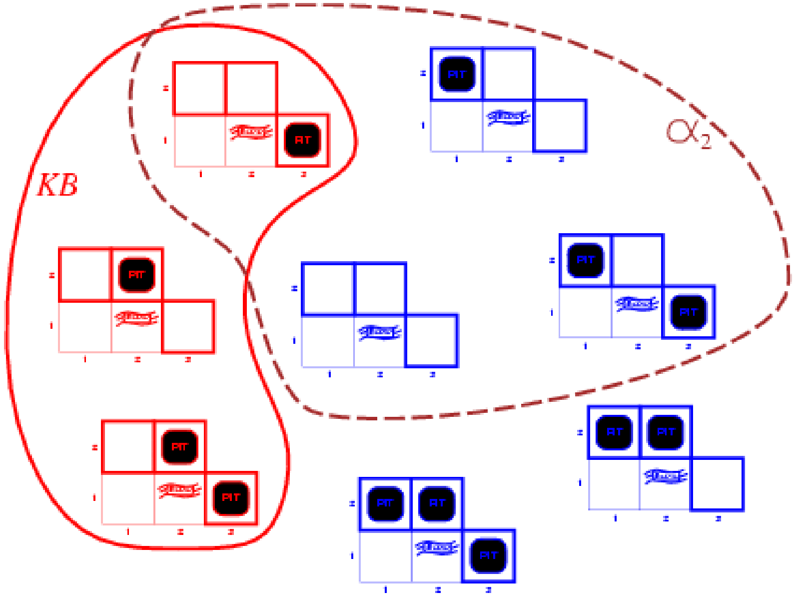
- *KB* = wumpus-world rules + observations
- α_1 = "[1,2] is safe", *KB* $\models \alpha_1$, proved by model checking

Wumpus Models



- *KB* = wumpus-world rules + observations

Wumpus Models



- *KB* = wumpus-world rules + observations
- α_2 = "[2,2] is safe", *KB* $\not\models \alpha_2$

Entailment and Inference

- **Semantics**: Determine entailment by **Model Checking**, that is enumerate all models and show that the sentence α must hold in all models.

$$KB \models \alpha$$

- **Syntax**: Determine entailment by **Theorem Proving**, that is apply rules of inference to KB to build a proof of α *without enumerating and checking all models*.

$$KB \vdash \alpha$$

- But how are entailment and inference related?

Soundness & Completeness

- We want an inference algorithm that is:

  1. **Sound**: does not infer false formulas, that is, derives only entailed sentences.

  2. **Complete**: derives ALL entailed sentences.

Validity & satisfiability

- A sentence is valid (aka tautology) if it is true in all models,
  e.g., $True$, $p \vee \neg p$, $p \Rightarrow p$, $(p \wedge (p \Rightarrow q)) \Rightarrow q$

- Validity is connected to inference via the Deduction Theorem:
  $KB \models \alpha \ IFF \ (KB \Rightarrow \alpha)$ is valid

- A sentence is satisfiable if it is true in some model
  e.g., $p \vee q, r$

- A sentence is unsatisfiable if it is true in no models
  e.g., $p \wedge \neg p$

- Satisfiability is connected to inference via the following:
  $KB \models \alpha \ IFF \ (KB \wedge \neg\alpha)$ is unsatisfiable
  i.e., prove α by contradiction

Determining entailment

- Given a Knowledge Base (KB) (set of sentences in PL), given a query α, output whether KB entails α, noted: $KB \models \alpha$
- We will see two ways of doing proofs in PL:
  - **Model checking** enumerate the models (truth table enumeration, exponential).
  - **Application of inference rules** (proof checking/theorem proving): Syntactic derivations with rules like Modus Ponens (Backward chaining and forward chaining). A proof is a sequence of inference rule applications.

Model Checking

- Truth Table for inference
- Model: assignment T/F to every propositional symbol.
- Check that α is true in every model in which KB is true.

Model Checking

- Truth Table for inference
- Model: assignment T/F to every propositional symbol.
- Check that α is true in every model in which KB is true.

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	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*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
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*

Inference Wampus World

In order to prove if $\neg P_{1,2}$

R_2 : $B_{1,1} \leftrightarrow (P_{1,2} \vee P_{2,1}) \equiv B_{1,1} \rightarrow (P_{1,2} \vee P_{2,1}) \wedge ((P_{1,2} \vee P_{2,1}) \rightarrow B_{1,1})$

By using AND Elimination we take:

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

Logical equivalence with contrapositive:

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

$\neg B_{1,1}$

$\neg (P_{1,2} \vee P_{2,1})$ By Modus Ponens

Take negation inside and use DeMorgan's Rule:

$\neg P_{1,2} \wedge \neg P_{2,1}$

No pit at [1,2] and [2,1]

Inference as a search problem

- **Initial state:** The initial KB
- **Actions:** all inference rules applied to all sentences that match the top of the inference rule
- **Results:** add the sentence in the bottom half of the inference rule
- **Goal:** a state containing the sentence we are trying to prove.

Theorem proving

- Search for proofs is a more efficient way than enumerating models (We can ignore irrelevant information)
- Truth tables have an exponential number of models.
- The idea of inference is to repeat applying *inference rules* to the KB.
- Inference can be applied whenever suitable premises are found in the KB.
- Inference is sound. How about completeness?

Theorem proving

- Two ways to ensure completeness:
  - **Proof by resolution**: use powerful inference rules (resolution rule)
  - **Forward or Backward chaining**: use of modus ponens on a restricted form of propositions (Horn clauses)
- Resolution: ONE single inference rule
- Invented by Robinson, 1965
- Resolution + Search = complete inference algorithm.

Proof by Resolution

- Resolution & Wumpus world:

1,4	2,4	3,4	4,4
1,3 **W!**	2,3	3,3	4,3
1,2 $\boxed{\textbf{A}}$ **S** **OK**	2,2 **OK**	3,2	4,2
1,1 **V** **OK**	2,1 **B** **V** **OK**	3,1 **P!**	4,1

Proof by Resolution

- **Unit resolution**:

$$\frac{\ell_1 \vee \cdots \vee \ell_k \qquad m}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k}$$

  where ℓ_i and m are complementary literals.

- Example:

$$\frac{P_{1,3} \vee P_{2,2} \qquad \neg P_{2,2}}{P_{1,3}}$$

- We call a **clause** a disjunction of literals.

- Unit resolution: Clause + Literal = New clause.

Proof by Resolution

- **Resolution inference rule (for CNF)**:

$$\frac{\ell_1 \vee \cdots \vee \ell_k \qquad m_1 \vee \cdots \vee m_n}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

  where ℓ_i and m_j are complementary literals.

- Resolution **applies only to clauses**

- **Fact:** Every sentence in PL is logically equivalent to a conjunction of clauses.

- Conjunctive Normal Form (CNF): Conjunction of disjunction of literals:

- Example: $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

- Resolution inference rule (for CNF): sound and complete for propositional logic

Conversion to CNF

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

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.
   $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

Conversion to CNF

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

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)$.
   $(B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$

2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \lor \beta$.
   $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg(P_{1,2} \lor P_{2,1}) \lor B_{1,1})$

Conversion to CNF

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

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)$.
   $(B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$

2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \lor \beta$.
   $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg(P_{1,2} \lor P_{2,1}) \lor B_{1,1})$

3. Move $\neg$ inwards using de Morgans rules and double-negation:
   $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land ((\neg P_{1,2} \land \neg P_{2,1}) \lor B_{1,1})$

Conversion to CNF

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

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)$.
   $(B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$

2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \lor \beta$.
   $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg(P_{1,2} \lor P_{2,1}) \lor B_{1,1})$

3. Move $\neg$ inwards using de Morgans rules and double-negation:
   $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land ((\neg P_{1,2} \land \neg P_{2,1}) \lor B_{1,1})$

4. Apply distributivity law ($\lor$ over $\land$) and flatten:
   $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1})$

Resolution algorithm

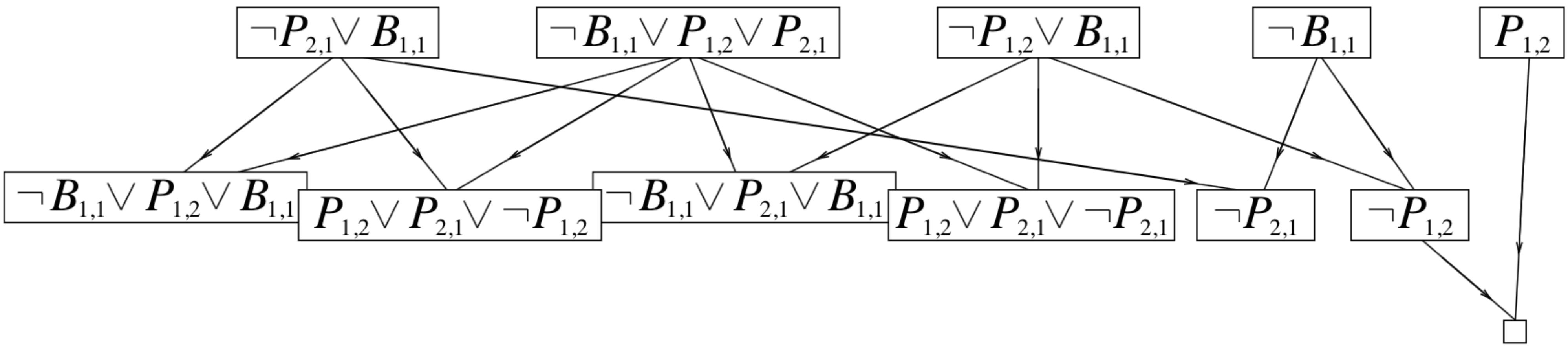
function PL-RESOLUTION(KB, α) **returns** *true* or *false*
  inputs: KB, the knowledge base, a sentence in propositional logic
    α, the query, a sentence in propositional logic

  clauses ← the set of clauses in the CNF representation of $KB \wedge \neg\alpha$
  new ← { }
  loop do
    for each C_i, C_j **in** *clauses* **do**
      resolvents ← PL-RESOLVE(C_i, C_j)
      if *resolvents* contains the empty clause **then return** *true*
      new ← *new* ∪ *resolvents*
    if *new* ⊆ *clauses* **then return** *false*
    clauses ← *clauses* ∪ *new*

Resolution example

$$KB = R_4 \wedge R_2 = (B_{1,1} \Leftrightarrow P_{1,2} \vee P_{2,1}) \wedge \neg B_{1,1}$$

$$\alpha = \neg P_{1,2}$$



Resolution Example

KB:

$Cold \wedge Precipitation \rightarrow Snow$

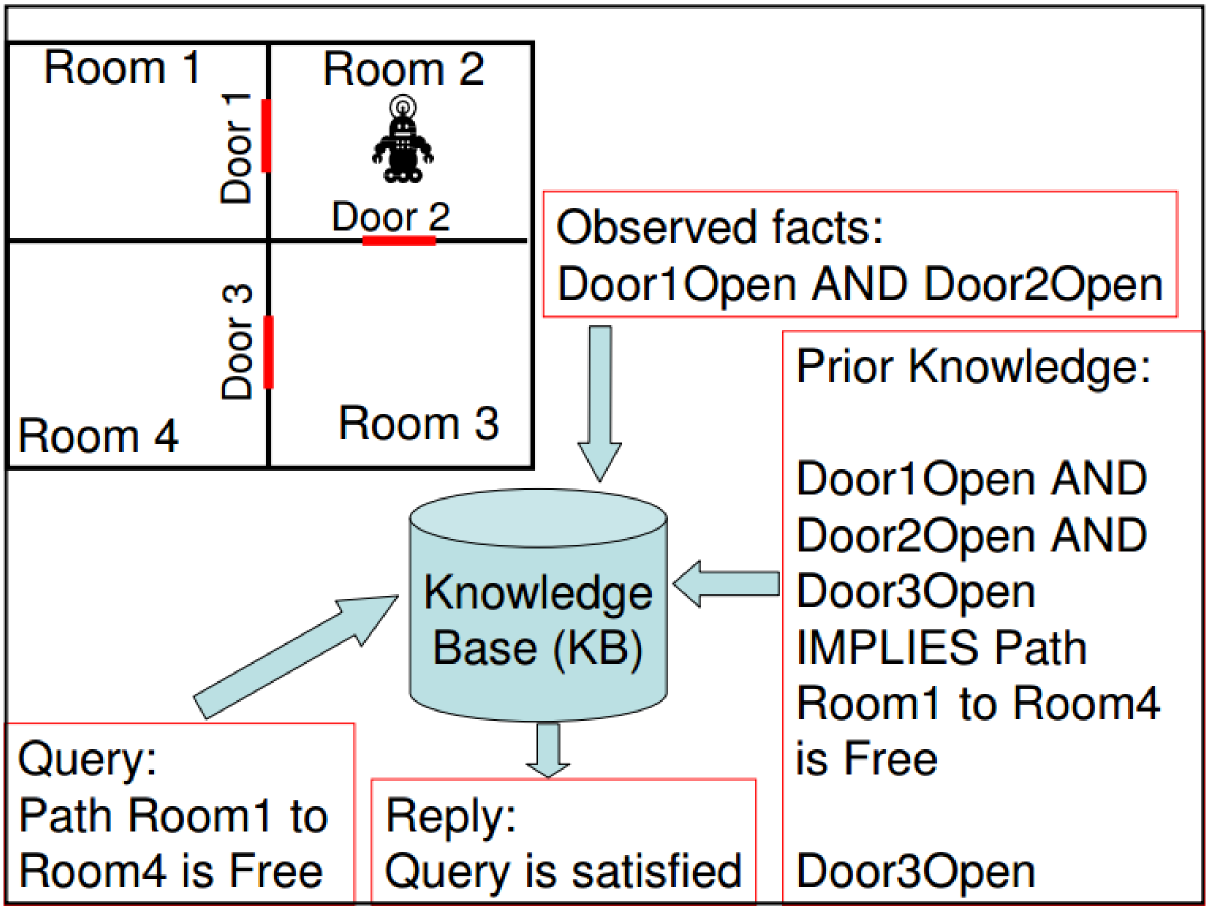
$January \rightarrow Cold$

$Clouds \rightarrow Precipitation$

$January$

$Clouds$

Q: is snow true


Room 1
Room 2
Door 1
Door 2
Door 3
Room 4
Room 3
Observed facts:
Door1Open AND Door2Open
Prior Knowledge:
Door1Open AND Door2Open AND Door3Open IMPLIES Path Room1 to Room4 is Free
Door3Open
Knowledge Base (KB)
Query:
Path Room1 to Room4 is Free
Reply:
Query is satisfied

Forward/backward chaining

- KB = conjunction of Horn clauses
- Horn clauses: logic proposition of the form: $p_1 \wedge \ldots \wedge p_n \rightarrow q$
- Modus Ponens (for Horn Form): complete for Horn KBs

$$\frac{p_1, \ldots, p_n \qquad p_1 \wedge \ldots \wedge p_n \rightarrow q}{q}$$

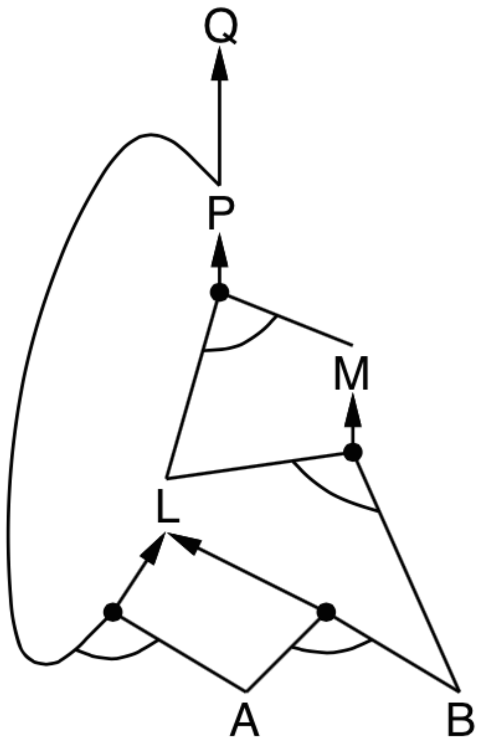
- Can be used with **forward chaining** or **backward chaining**.
- These algorithms are very natural and run in linear time

Forward chaining

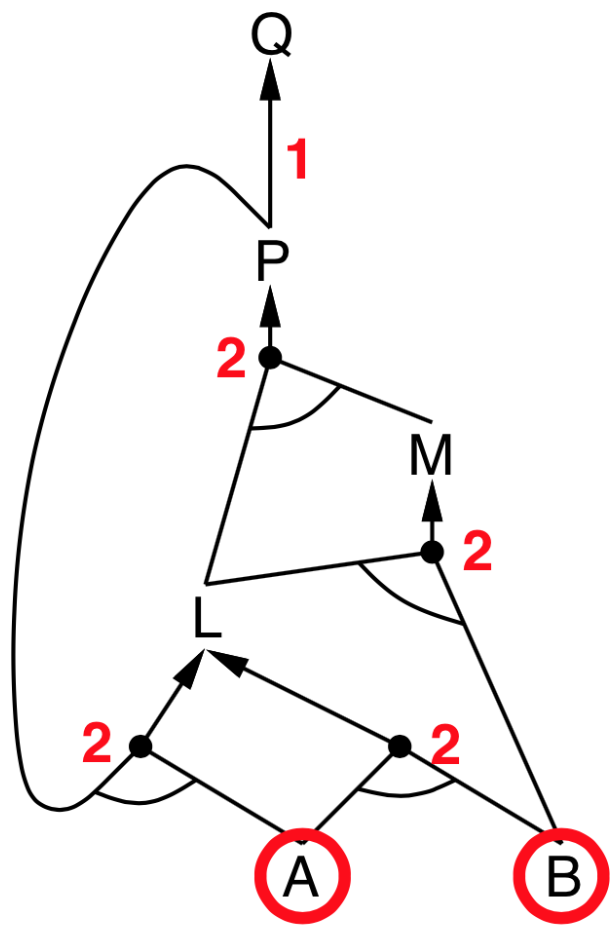
Idea:

Fire any rule whose premises are satisfied in the KB, add its conclusion to the KB, until query is found

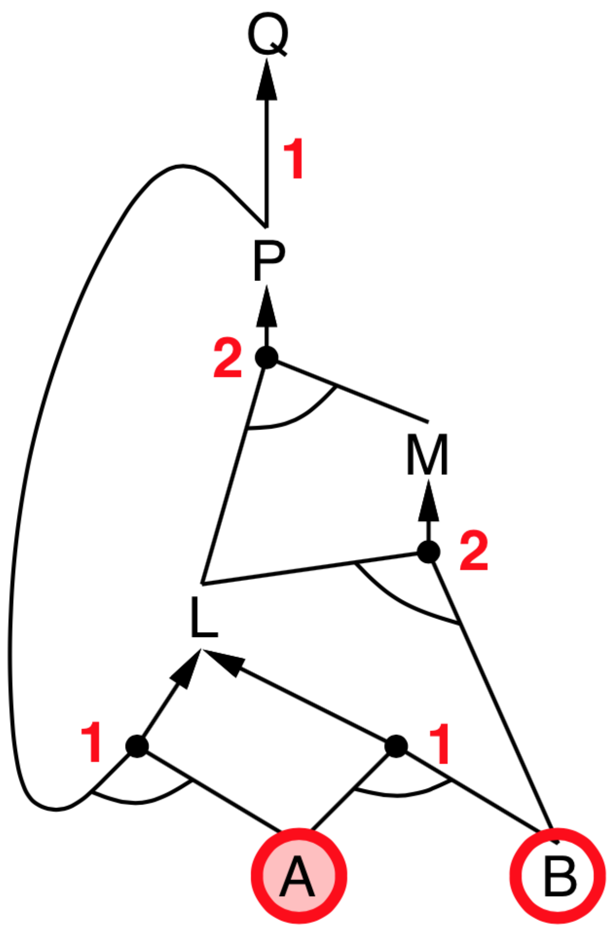
$$
\begin{aligned}
&P \Rightarrow Q\\
&L \wedge M \Rightarrow P\\
&B \wedge L \Rightarrow M\\
&A \wedge P \Rightarrow L\\
&A \wedge B \Rightarrow L\\
&A\\
&B
\end{aligned}
$$



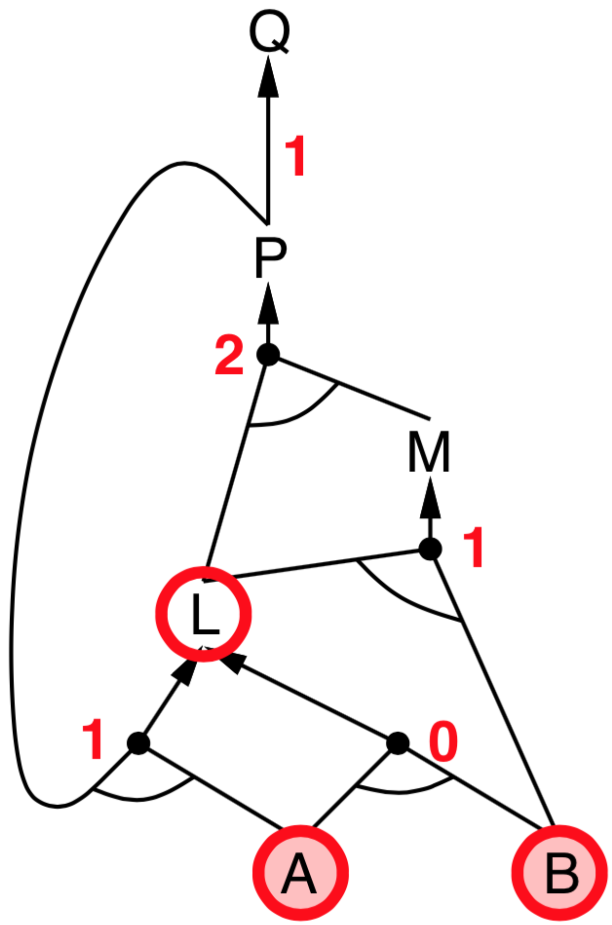
Forward chaining example



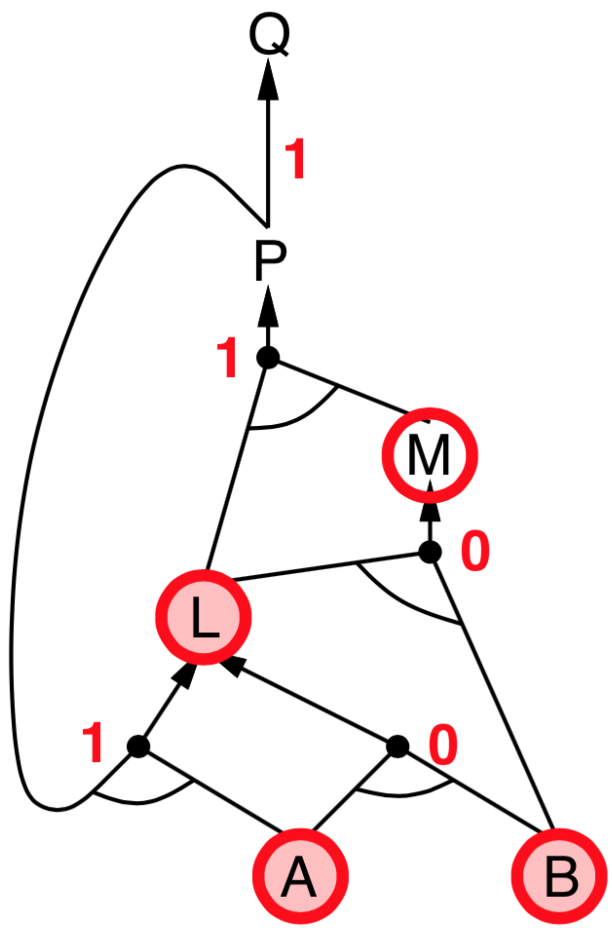
Forward chaining example



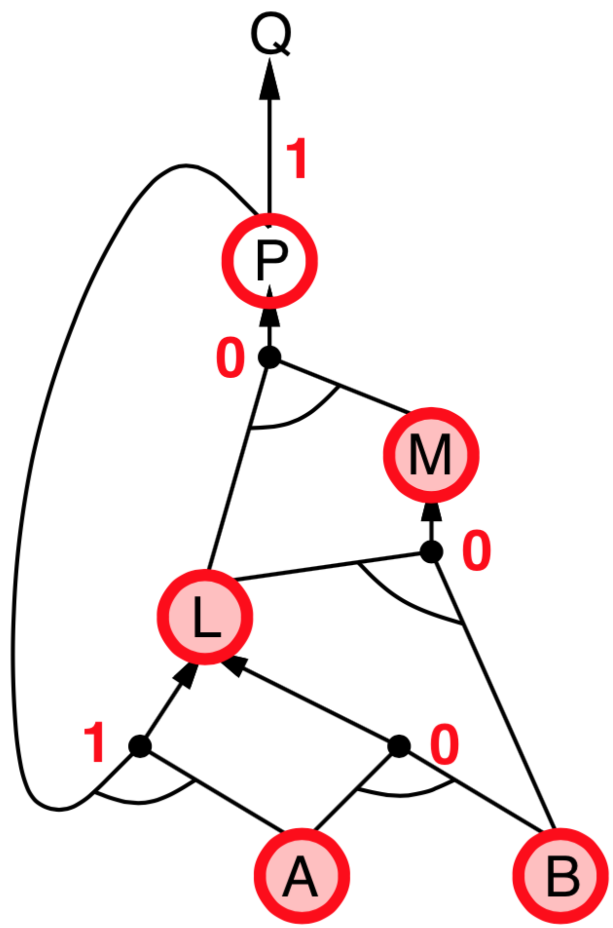
Forward chaining example



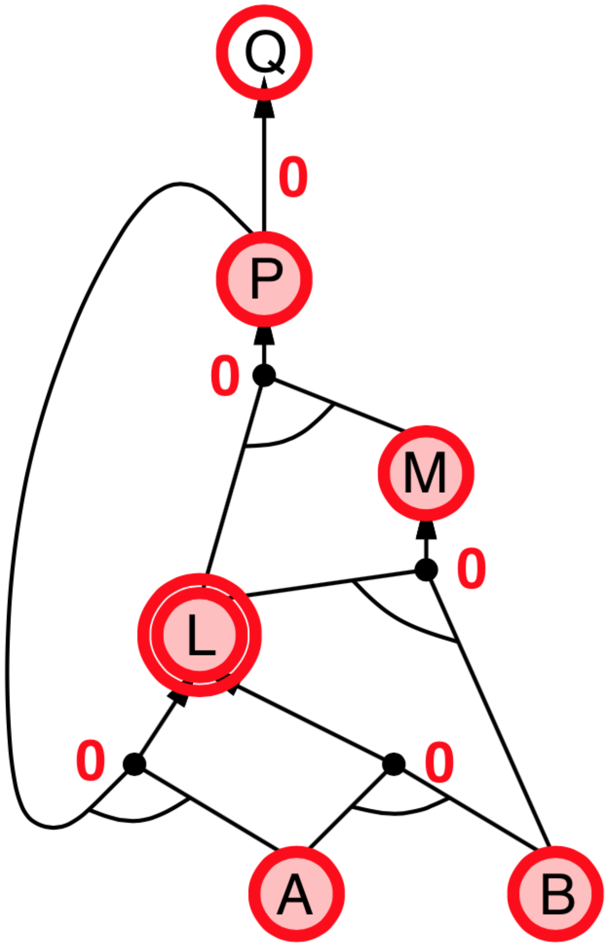
Forward chaining example



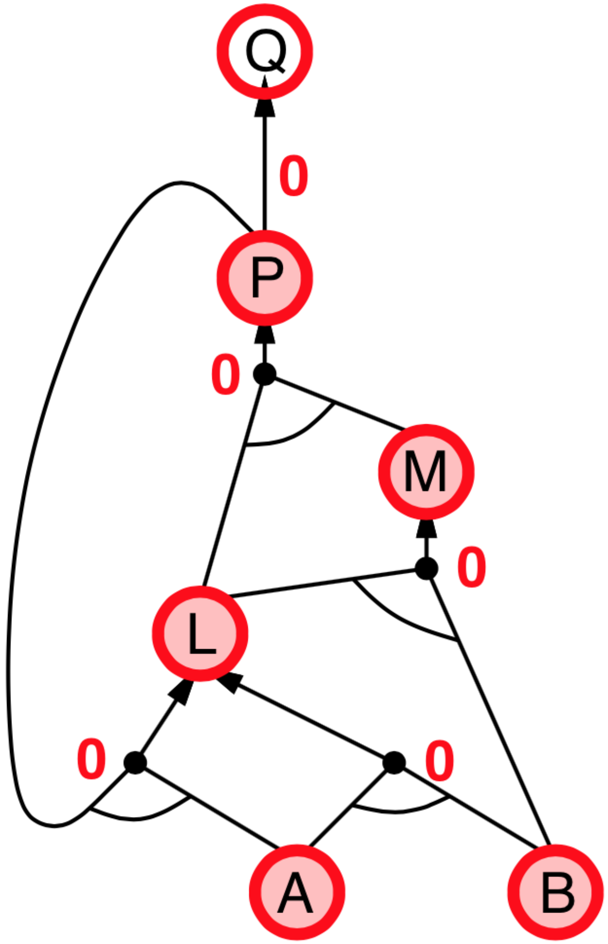
Forward chaining example



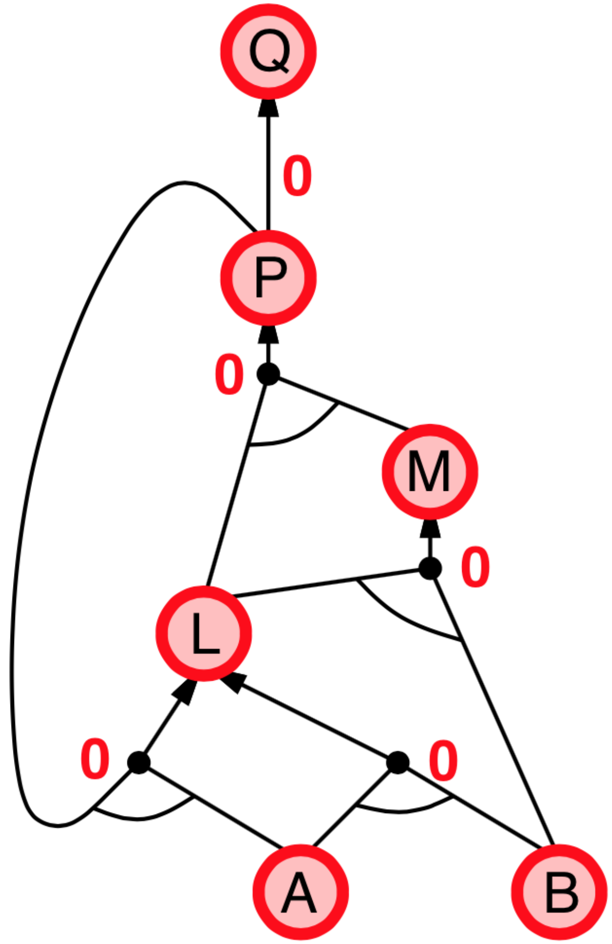
Forward chaining example



Forward chaining example



Forward chaining example

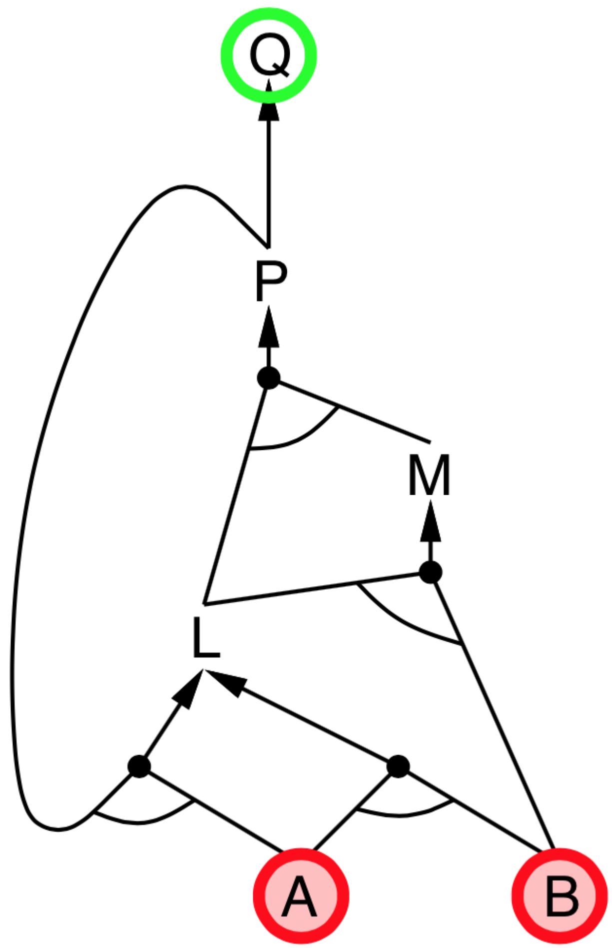


Backward chaining

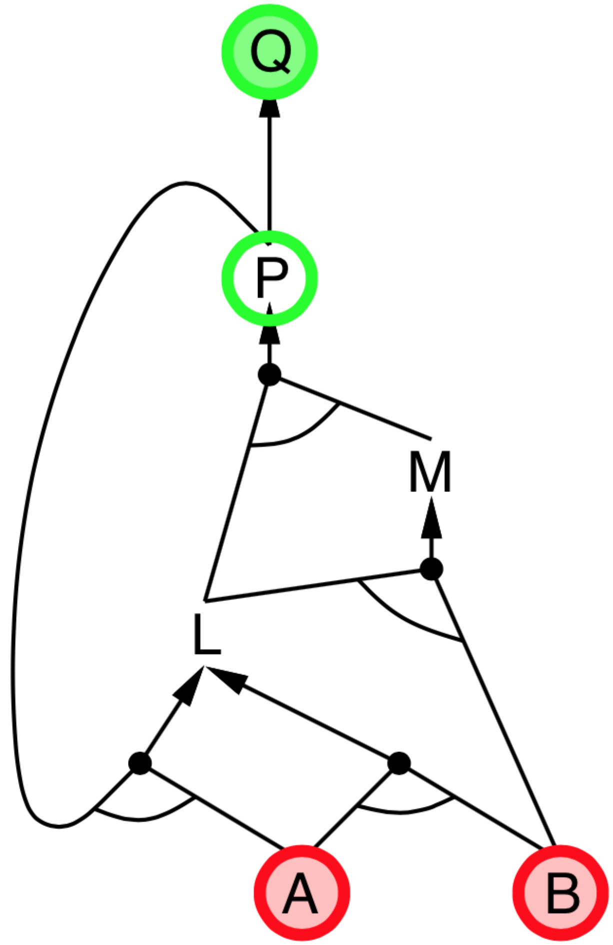
Idea: Works backwards from the query q

- to prove q by Backward Chaining:
  - Check if q is known already, or
  - Prove by Backward Chaining all premises of some rule concluding q
- Avoid loops: check if new subgoal is already on the goal stack
- Avoid repeated work: check if new subgoal
  - has already been proved true, or
  - has already failed

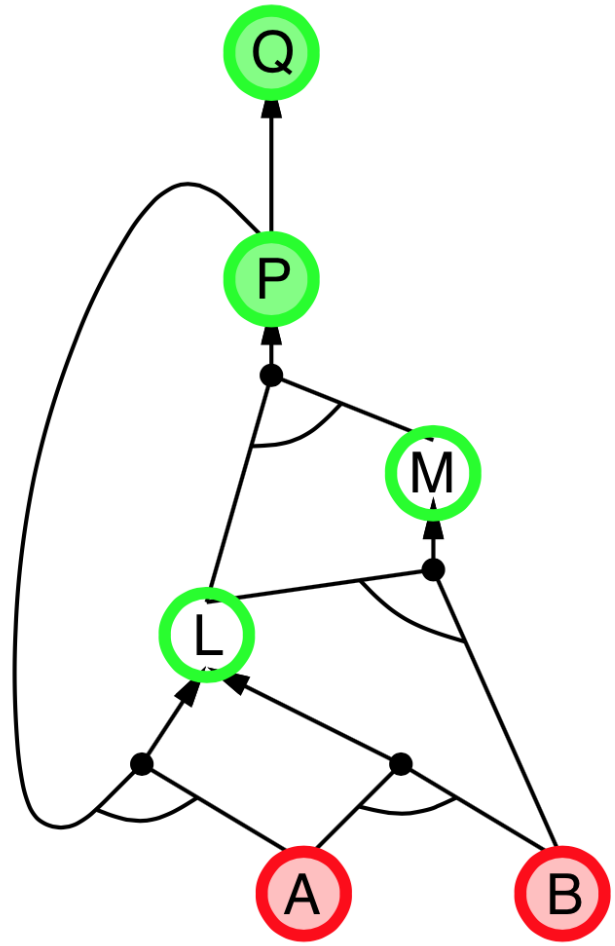
Backward chaining example



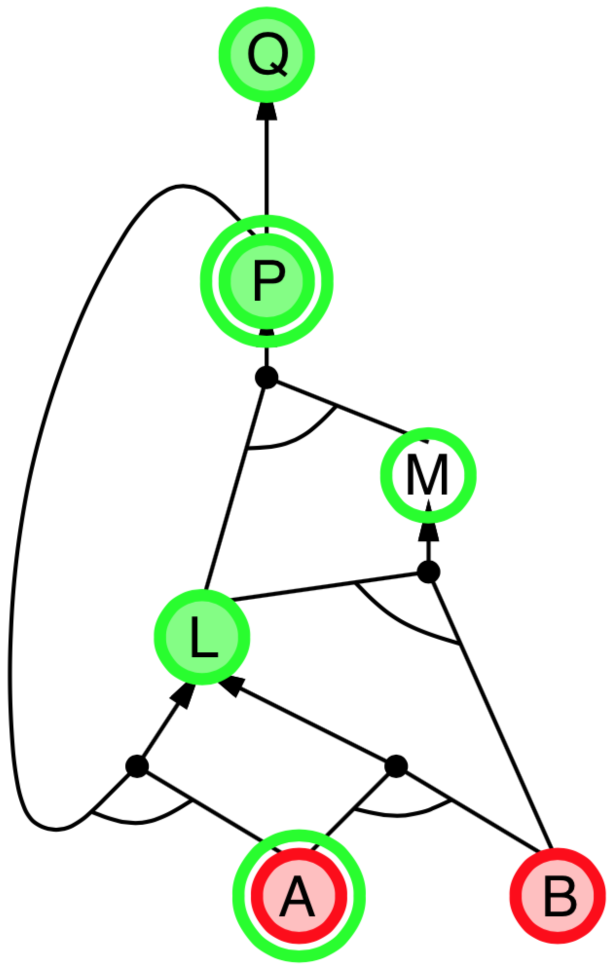
Backward chaining example



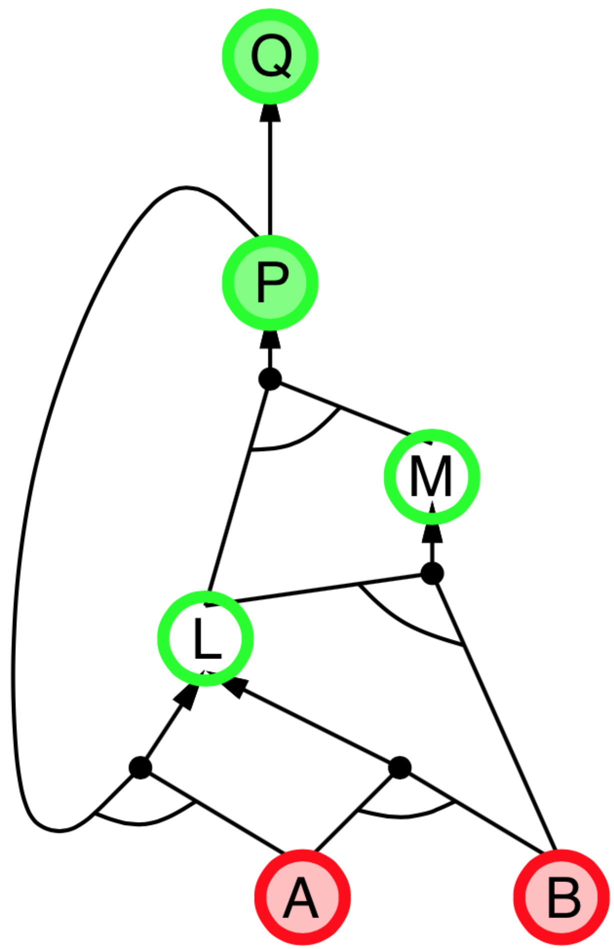
Backward chaining example



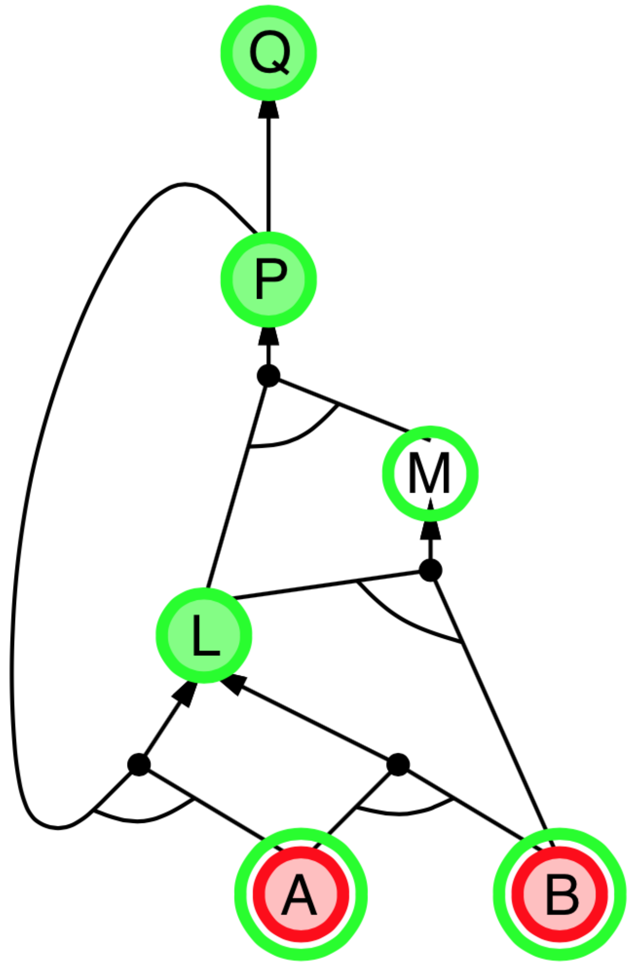
Backward chaining example



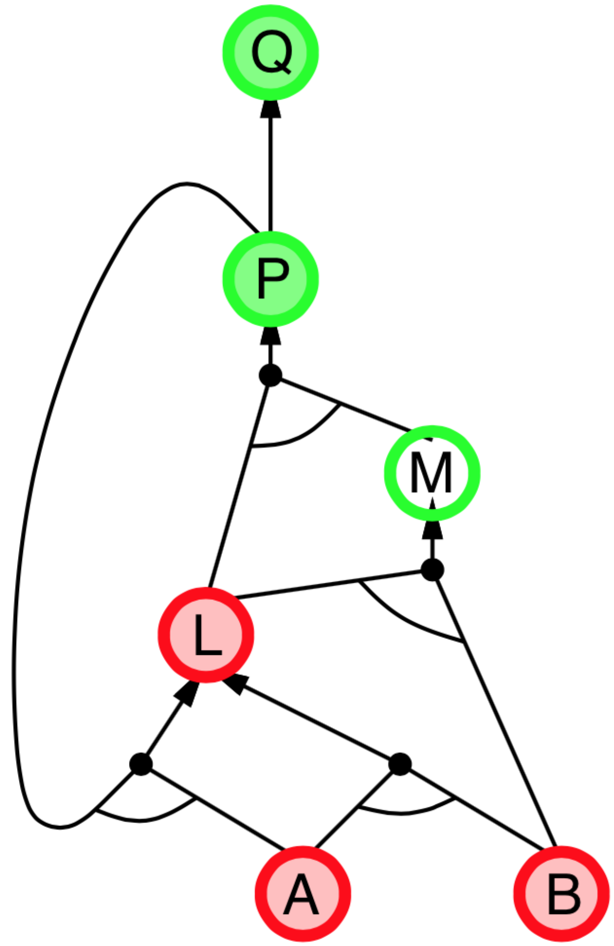
Backward chaining example



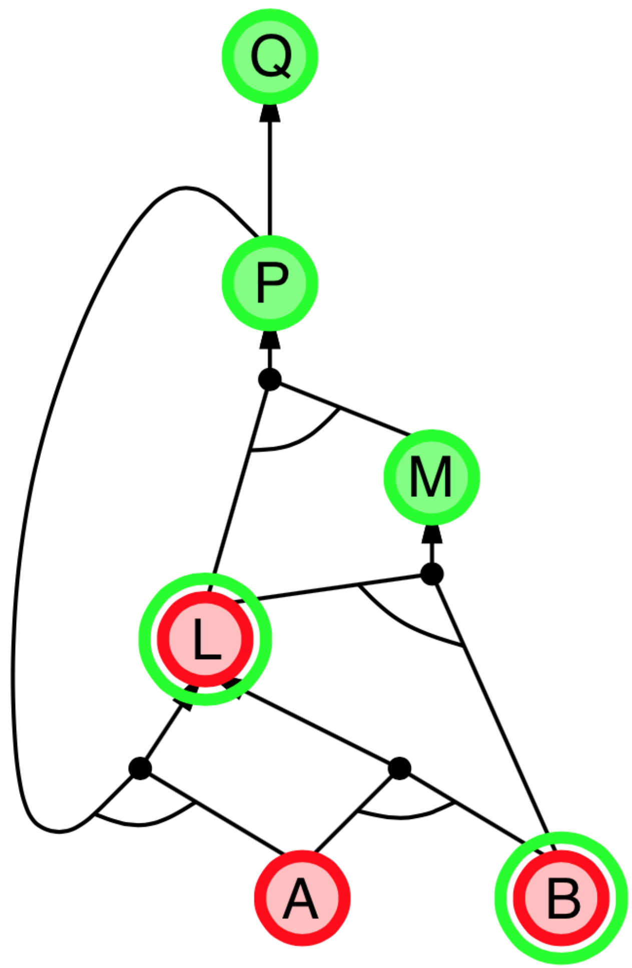
Backward chaining example



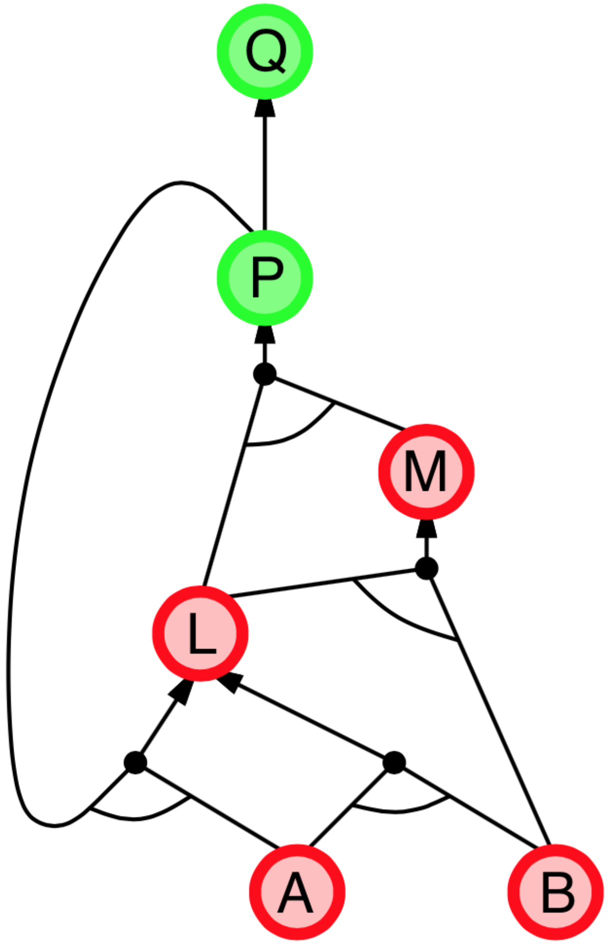
Backward chaining example



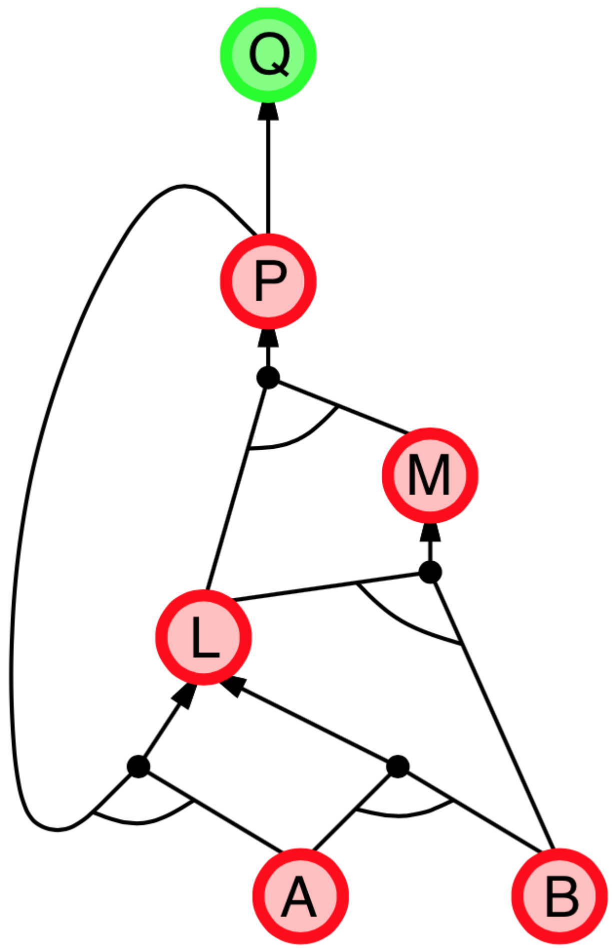
Backward chaining example



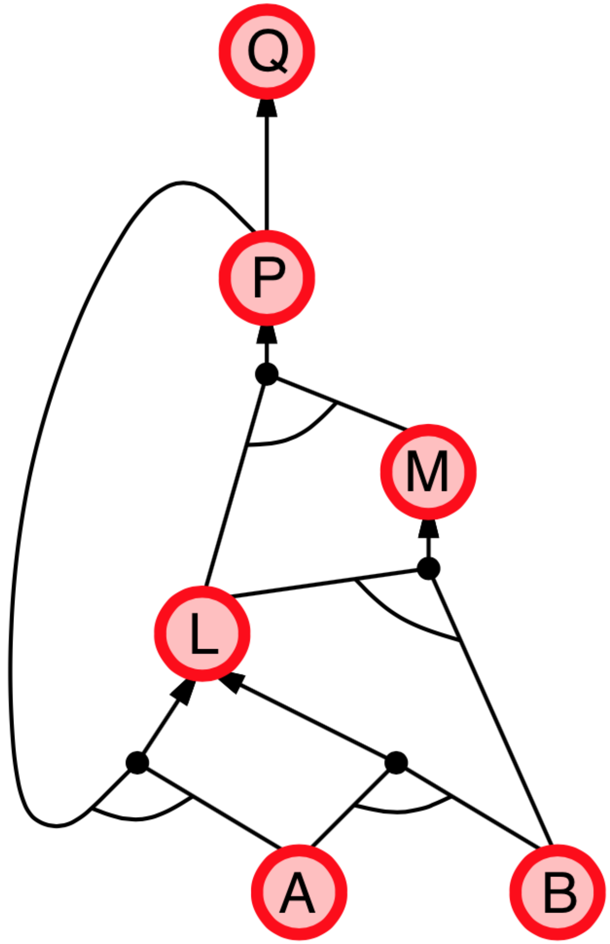
Backward chaining example



Backward chaining example



Backward chaining example



Forward vs. Backward

- Forward chaining:
  - Data-driven, automatic, unconscious processing,
  - May do lots of work that is irrelevant to the goal
- Backward chaining:
  - Goal-driven, appropriate for problem-solving,
- Complexity of BC can be much less than linear in size of KB

Propositional Logic

- Propositional Logic (PL) is a formal language to describe the world around us.
- Logic can be used by an agent to model the world.
- Sentences in PL have a fixed **syntax**.
- With symbols and connectives we can form logical sentences:
  - Symbols or terms that can be either True or False or unknown.
  - Logical connectives

  Example: $hot \wedge sunny \Rightarrow beach \vee pool$
- Syntax and **Semantic** represent two important and distinct aspects in PL.
- Semantic: configurations/instantiations of the world.

Propositional Logic

- Modus Ponens inference rule:

$$\frac{p_1, \ldots, p_n, \qquad (p_1 \wedge \ldots \wedge p_n) \rightarrow q}{q}$$

- Example:

$$\frac{Warm \qquad Warm \rightarrow Sunny}{Sunny}$$

- Modus Ponens deals with Horn clauses

- Horn clauses: logic proposition of the form: $p_1 \wedge \ldots \wedge p_n \rightarrow q$

- **Inference:** we want an inference algorithm that is:
  1. sound (does not infer false formulas), and
  2. ideally, complete too (derives all true formulas).

- **Inference in PL with horn clauses is sound and complete.**

Propositional Logic

- Limits of PL?
  1. PL is not expressive enough to describe all the world around us. It can't express information about different object and the relation between objects.
  2. PL is not compact. It can't express a fact for a set of objects without enumerating all of them which is sometimes impossible.
- Example: We have a vacuum cleaner (Roomba) to clean a 10×10 squares in the classroom. Use PL to express information about the squares.

Propositional Logic

- The proposition $square_1\_is\_clean$ expresses information about square1 being clean. How can one write this in a less heavy way?

- How can we express that all squares in the room are clean?

$$square_1\_is\_clean \wedge square_2\_is\_clean \wedge \ldots \wedge square_{100}\_is\_clean$$

- How can we express that some squares in the room are clean?

$$square_1\_is\_clean \vee square_2\_is\_clean \vee \ldots \vee square_{100}\_is\_clean$$

- How can we express that some squares have chairs on them?

$$square_1\_has\_chair \vee square_2\_has\_chair \vee \ldots \vee square_{100}\_has\_chair$$

Summary

- Logical agents apply inference to a knowledge base to derive new information and make decisions
- Basic concepts of logic:
  - **Syntax**: formal structure of sentences
  - **Semantics**: truth of sentences wrt models
  - **Entailment**: necessary truth of one sentence given another
  - **Inference**: deriving sentences from other sentences
  - **Soundness**: derivations produce only entailed sentences
  - **Completeness**: derivations can produce all entailed sentences
- Wumpus world requires the ability to represent partial and negated information, reason by cases, etc.
- Forward, backward chaining are linear in time, complete for Horn clauses Resolution is complete for propositional logic.

Summary

- Building logical agents was a main research trend in AI before the mid-nineties
- Logic is used in AI to represent the environment of the agent and reason about that environment
- Pros and cons of logical agents:
  - \- Do not handle uncertainty, probability does
  - \- Rule-based and do not use data, ML does
  - \- It is hard to model every aspect of the world
  - \+ Intelligibility of models: models are encoded explicitly