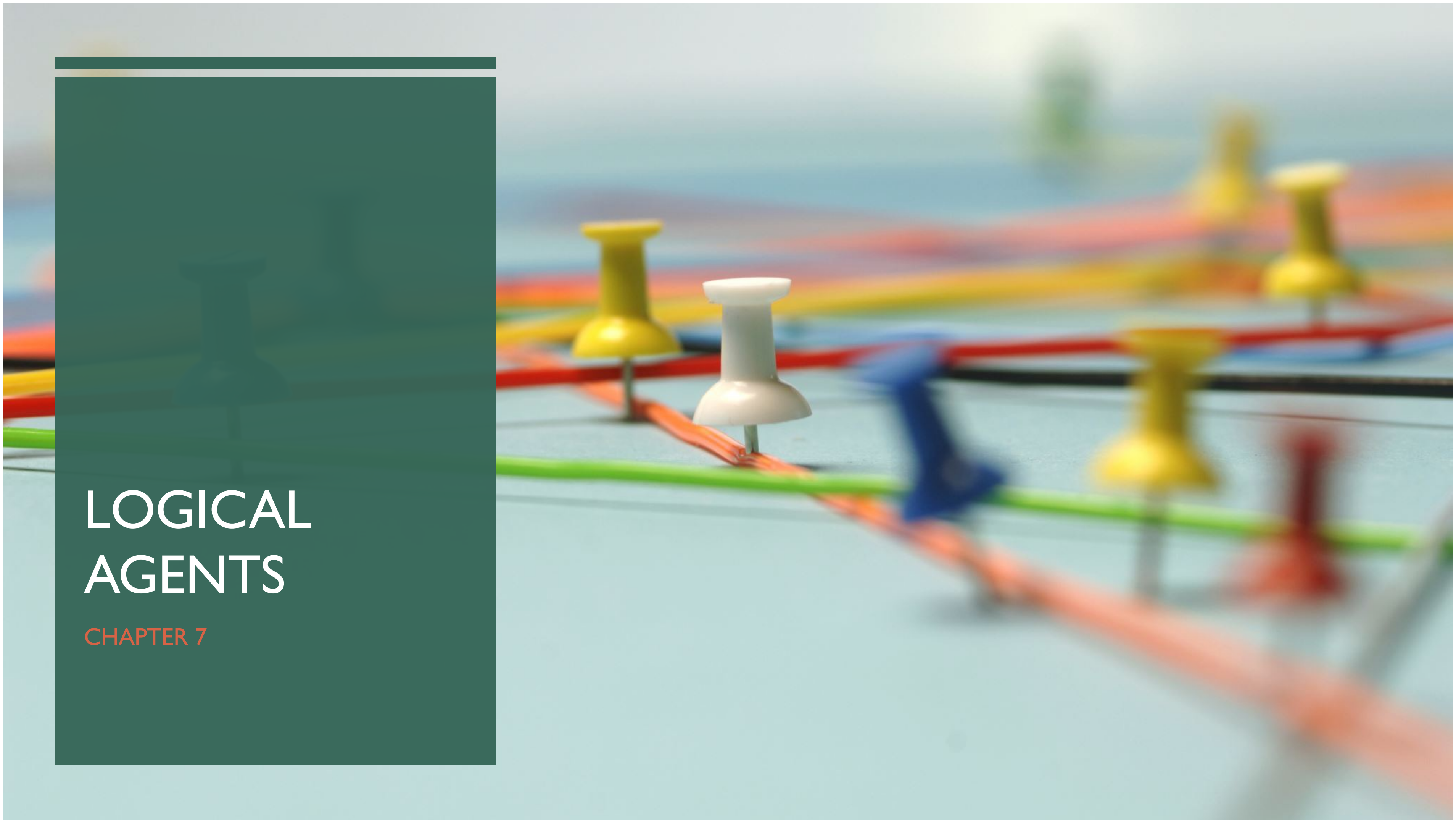


LOGICAL AGENTS

CHAPTER 7

OUTLINE

KNOWLEDGE-BASED AGENT

Knowledge Base:

- A Knowledge base (KB) is a special kind of database for knowledge management.
- An information repository that provides a means for information to be collected, organized, shared, searched and utilized
- A collection of facts and rules for problem solving

KNOWLEDGE-BASED AGENT

A knowledge-based agent must be able to:

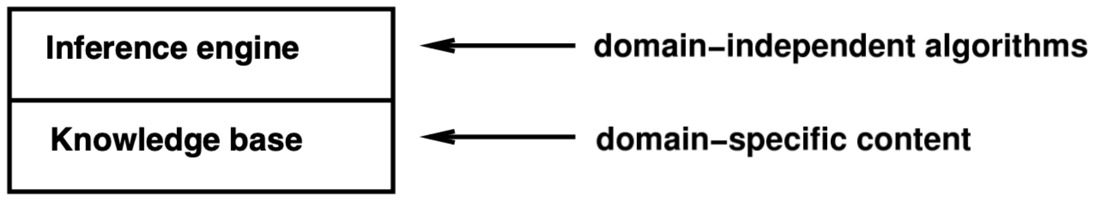
- Represent states, actions, etc.
- Incorporate new percepts
- Update internal representations of the world
- Deduce hidden properties of the world
- Deduce appropriate actions

Developing an intelligent agent:

Knowledge representation

- How to represent knowledge
- How to reason using that knowledge

KNOWLEDGE BASES (KB)



Knowledge base = set of **sentences** in a **formal** language

Building the KB:

Learning: agent discovers what it knows

Telling: agent is given what it knows (declarative)

Main actions of intelligent agent:

- **Tell** information to KB in the form of percept (adding new information)
- **Ask** KB what to do in the form of action (query)

***Answer should follow from KB.*

*** An inference engine is composed of domain-independent algorithms that are used to determine what follows from the KB*

KNOWLEDGE BASES

Agents can be viewed:

- at the **knowledge level:**

  i.e., what they know, regardless of how implemented

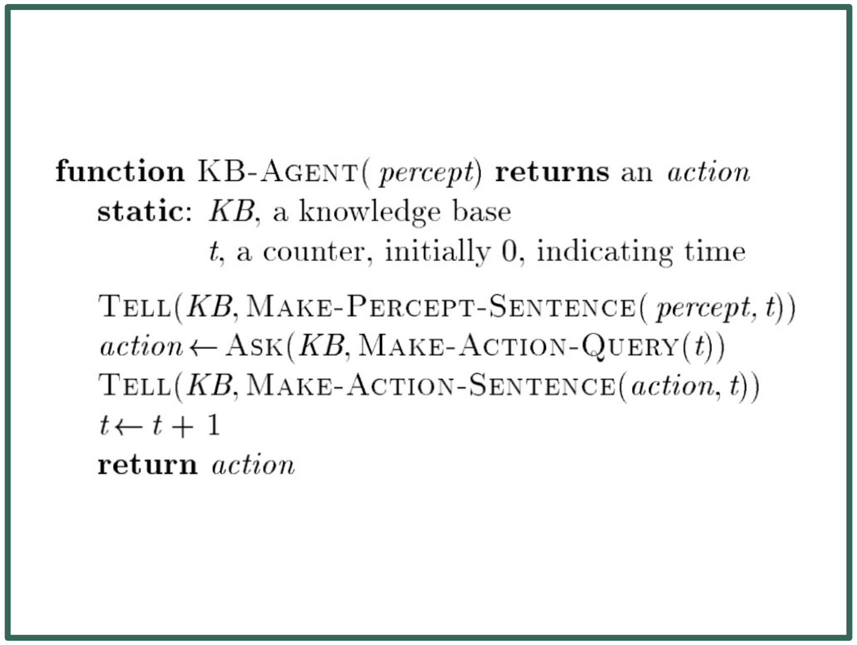
- at the **logic level:**

  level of sentence encoding

- Or at the **implementation level:**

  i.e., data structures in KB and algorithms that manipulate them

ALGORITHM: A SIMPLE KNOWLEDGE-BASED AGENT



```
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

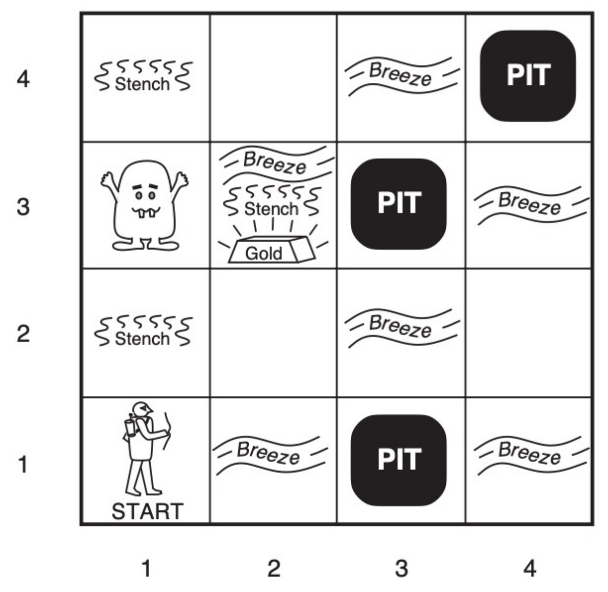


Figure 7.2 A typical wumpus world. The agent is in the bottom left corner.

PEAS Description:

- **Performance Measure**
- **Environment:** A 4 × 4 grid of rooms. The agent always starts in the square labeled [1,1], facing to the right.
- **Actuators:** Left turn, Right turn, Forward, Grab, Shoot
- **Sensors:** The agent has five sensors, each of which gives a single bit of information:

Percepts Stench, Breeze, Glitter, Bump, or Scream

Goals Get gold back to start without entering pit or wumpus square

THE WUMPUS WORLD

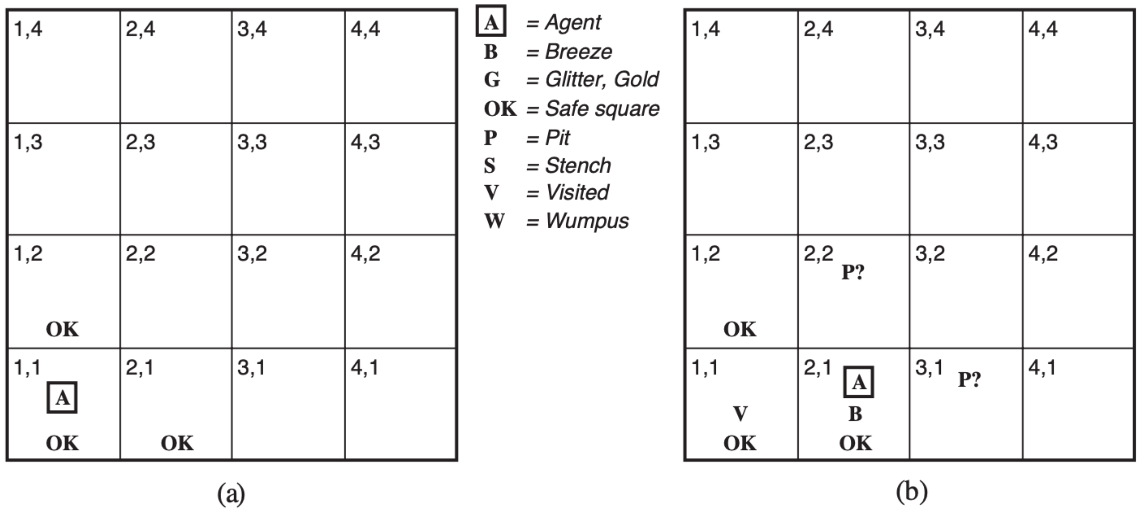


Figure 7.3 The first step taken by the agent in the wumpus world. (a) The initial situation, after percept [*None*, *None*, *None*, *None*, *None*]. (b) After one move, with percept [*None*, *Breeze*, *None*, *None*, *None*].

THE WUMPUS WORLD

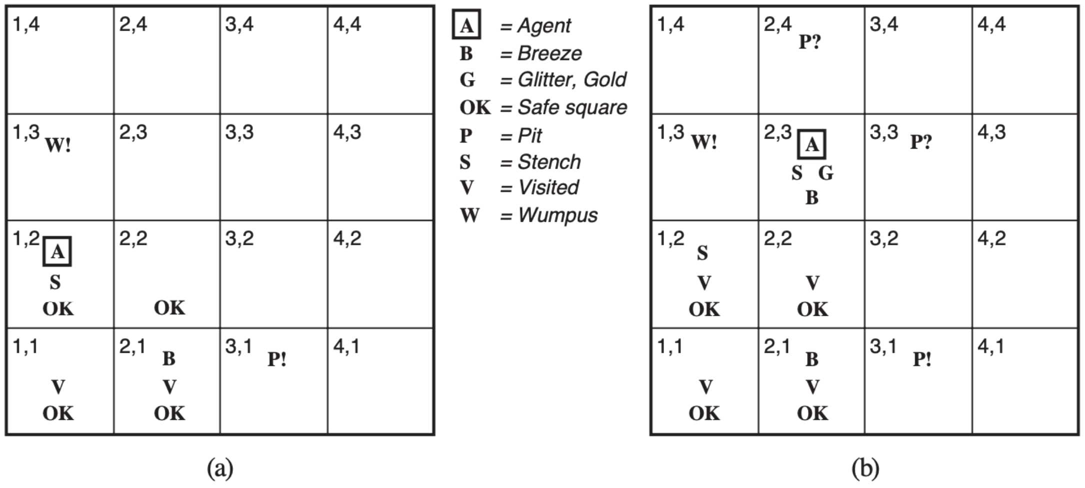


Figure 7.4 Two later stages in the progress of the agent. (a) After the third move, with percept [*Stench*, *None*, *None*, *None*, *None*]. (b) After the fifth move, with percept [*Stench*, *Breeze*, *Glitter*, *None*, *None*].

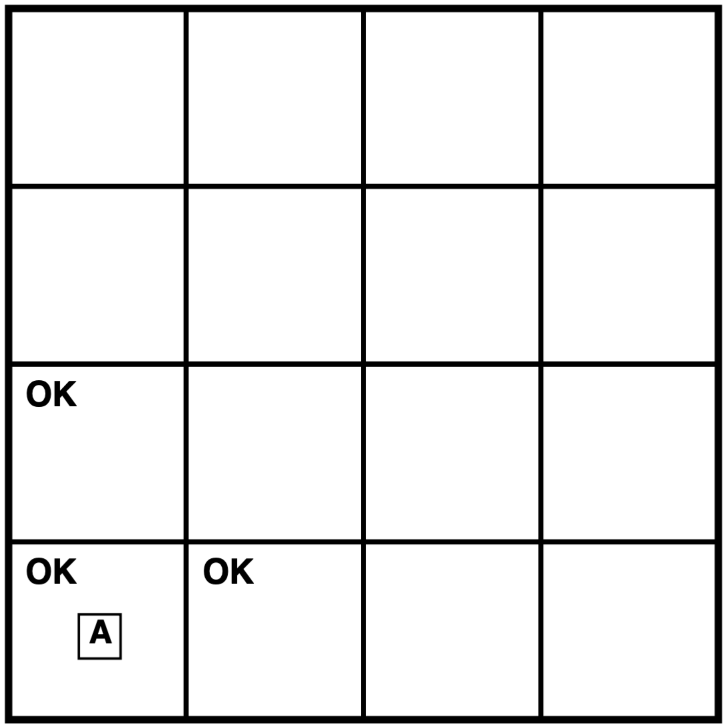
WUMPUS WORLD CHARACTERIZATION

- Is the world deterministic??
- Is the world fully accessible??
- Is the world static??
- Is the world discrete??

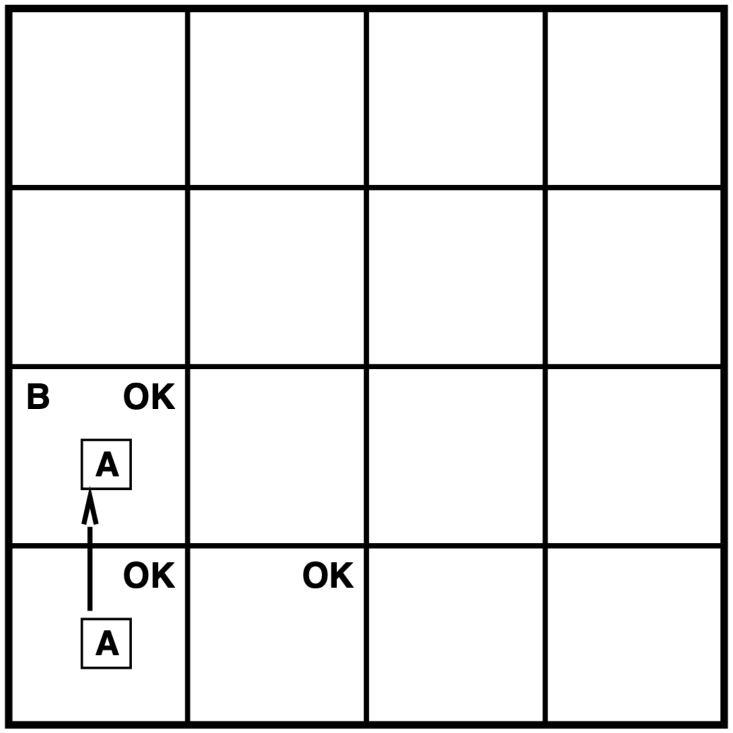
WUMPUS WORLD CHARACTERIZATION

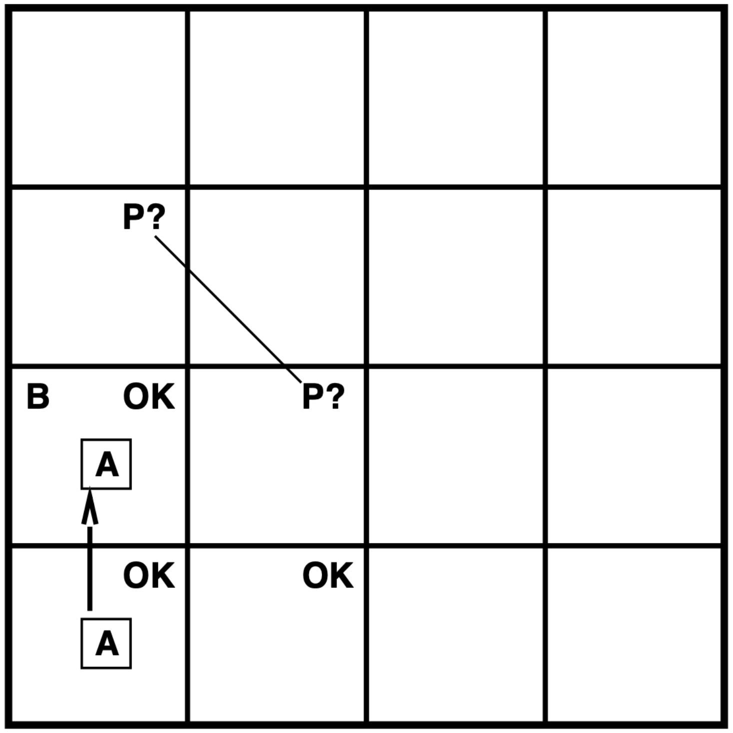
Is the world deterministic??	*Yes-outcomes exactly specified*
Is the world fully accessible??	*No-only local perception*
Is the world static??	*Yes-Wumpus and Pits do not move*
Is the world discrete??	*Yes*

EXPLORING WUMPUS WORLD (Another state)



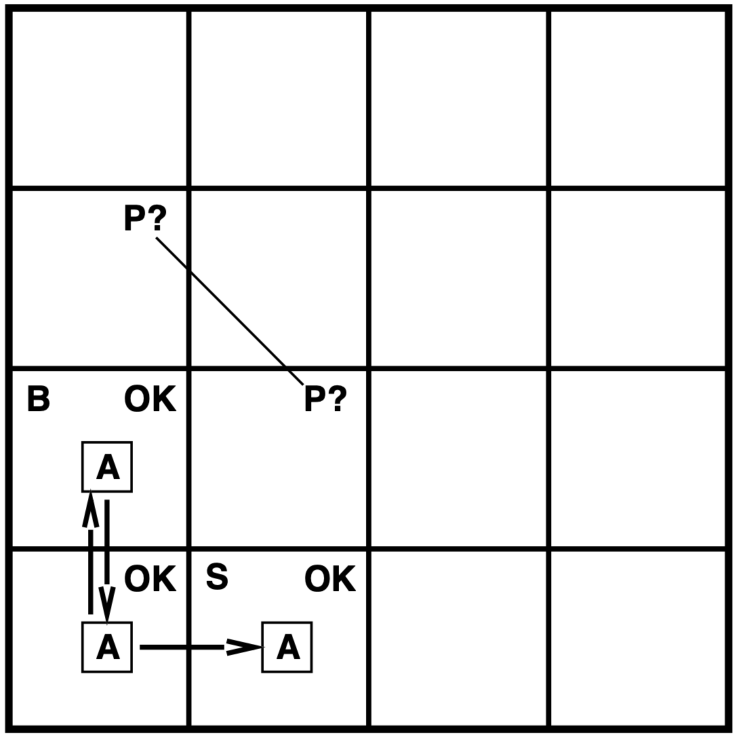
EXPLORING WUMPUS WORLD

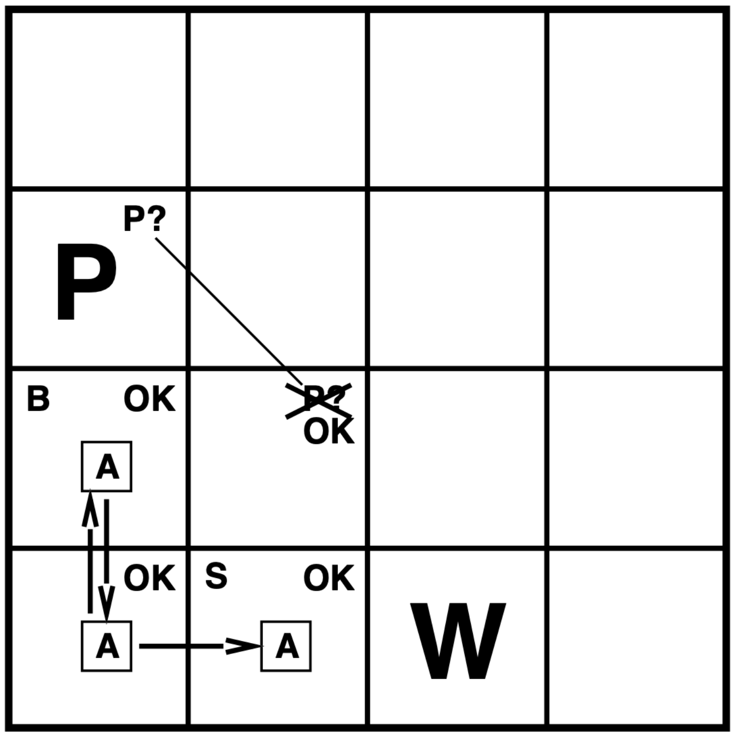




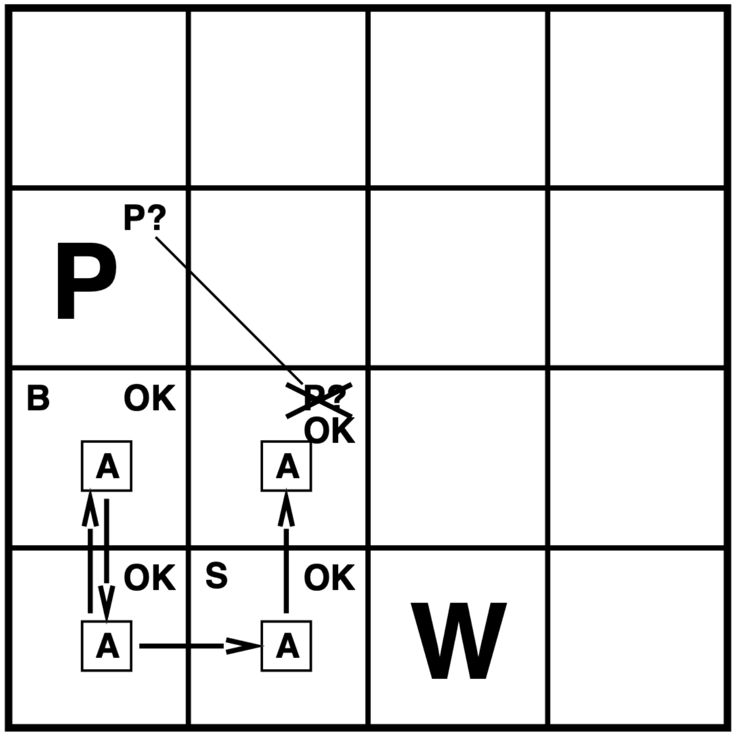
EXPLORING WUMPUS WORLD

EXPLORING WUMPUS WORLD

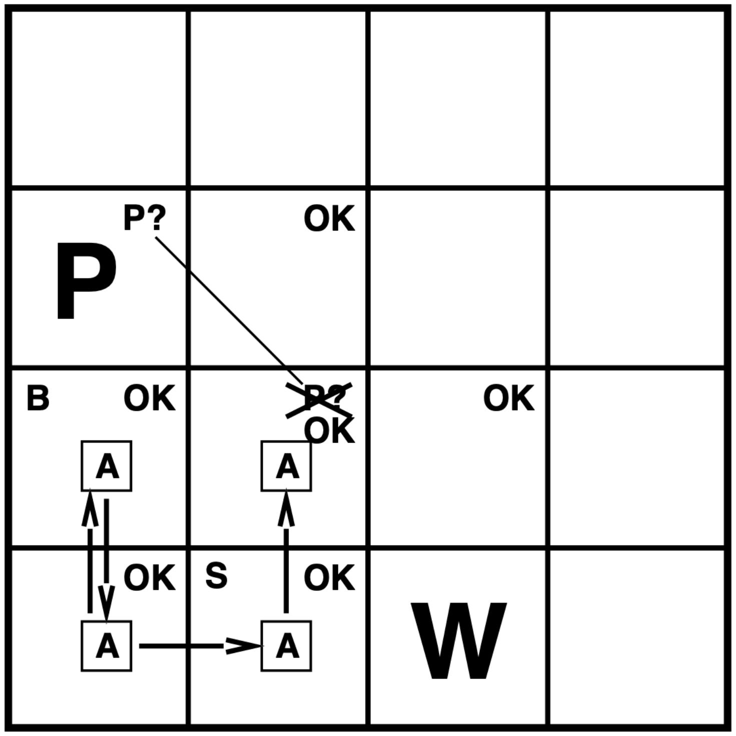




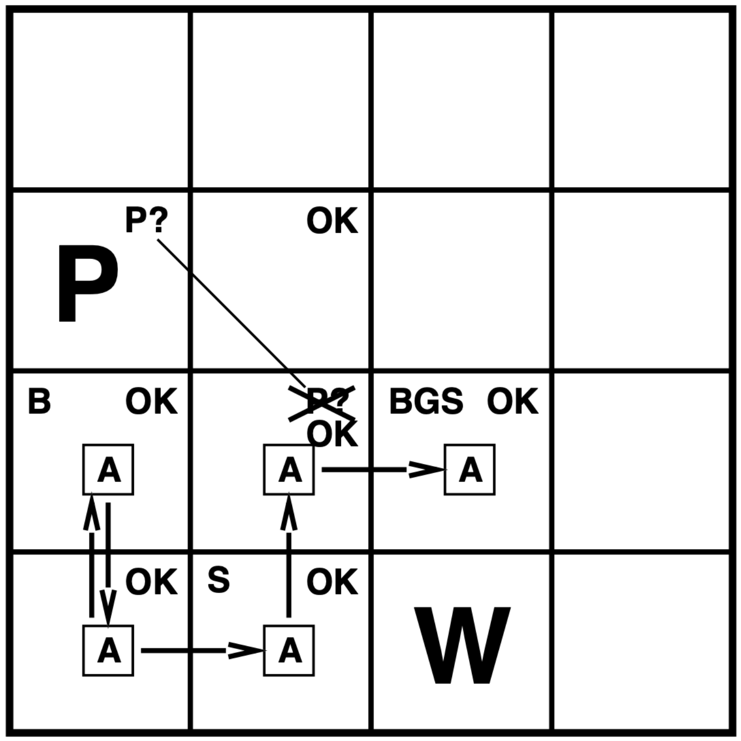
EXPLORING WUMPUS WORLD



EXPLORING WUMPUS WORLD



EXPLORING WUMPUS WORLD



EXPLORING WUMPUS WORLD

GENERAL LOGIC

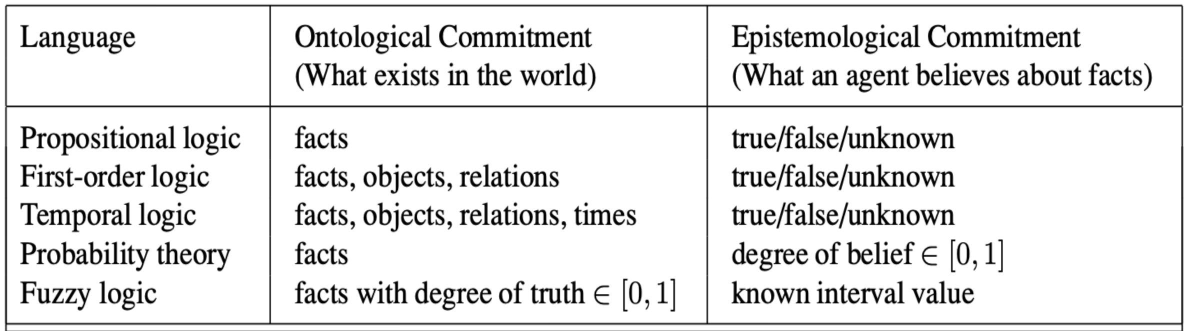
- **Logics** are formal languages for representing information such that conclusions can be drawn
- **Syntax** defines the format of legal sentences in the language
- **Semantics** define the “meaning” of sentences
  – i.e., define truth of a sentence with respect to a particular world (state)

Example: The language of arithmetic

- Syntax: $x + 2 \geq y$ is a legal sentence; $x2 + y >$ is not
- Semantics:
  – $x + 2 \geq y$ is true iff the number $x + 2$ is no less than the number y
  – $x + 2 \geq y$ is true in a world where $x = 7$, $y = 1$
  – $x + 2 \geq y$ is false in a world where $x = 0$, $y = 6$

GENERAL LOGIC

- **Types of Logic**

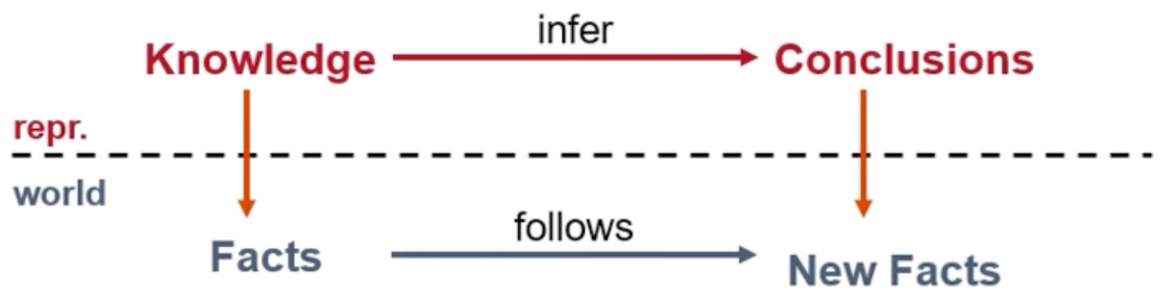


Language	Ontological Commitment (What exists in the world)	Epistemological Commitment (What an agent believes about facts)
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief $\in [0, 1]$
Fuzzy logic	facts with degree of truth $\in [0, 1]$	known interval value

Figure 8.1 Formal languages and their ontological and epistemological commitments.

GENERAL LOGIC

- Facts are claims about the world that are true/false
- Sentences represent facts in the world



**New facts follow from the original facts

GENERAL LOGIC

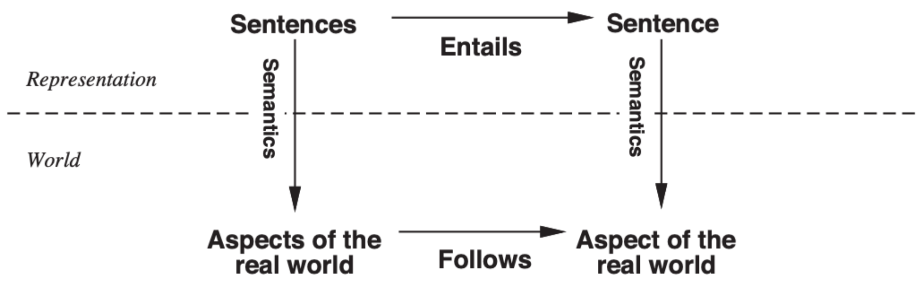


Figure 7.6 Sentences are physical configurations of the agent, and reasoning is a process of constructing new physical configurations from old ones. Logical reasoning should ensure that the new configurations represent aspects of the world that actually follow from the aspects that the old configurations represent.

ENTAILMENT

- Entailment means that one thing follows from another:
  - KB |= α
  - Knowledge base KB entails sentence α
    if and only if
    α is true in all worlds where KB is true
- Example:

  KB: “sky is blue” = true, “sun is shining” = true

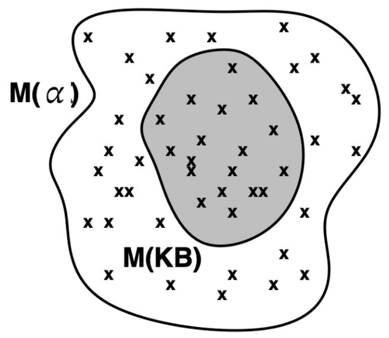
  entails α: “sky is blue and sun is shining” = true

  - α represents a true fact as long as facts represented in KB are true

MODELS

- Model = a possible “world”.
  – Formally structured expression of world state with respect to which truth can be evaluated
  – Basically, a collection of logical sentences describing a world or state
  – We say m is a model of a sentence α if α is true in m
  – Notation: M (α) is the set of all models of α

  - Then KB |= α if and only if M (KB) ⊆ M (α)
    • KB entails α iff, in every model where KB is true, α is also true.
    • Note that KB is the stronger statement here



WUMPUS MODELS

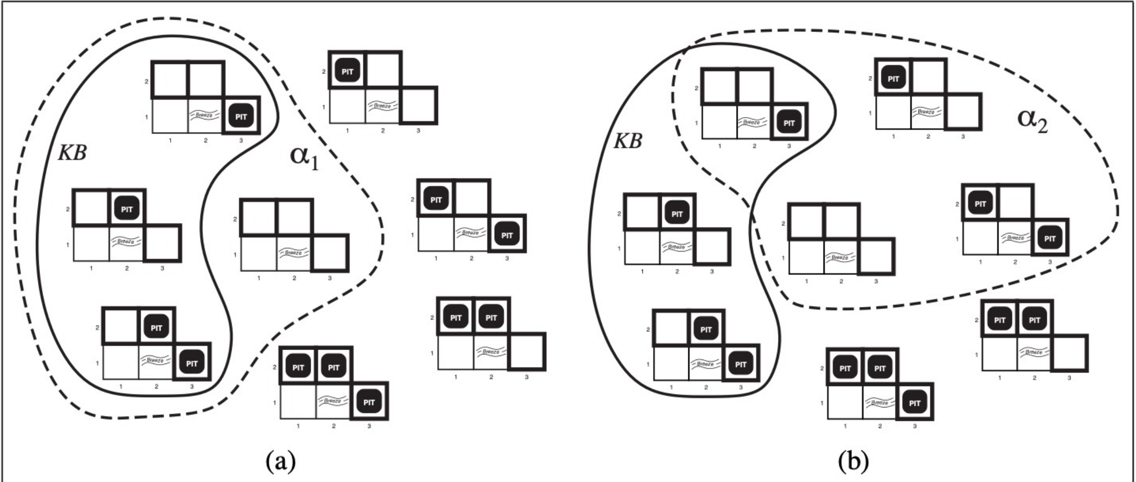


Figure 7.5 Possible models for the presence of pits in squares [1,2], [2,2], and [3,1]. The KB corresponding to the observations of nothing in [1,1] and a breeze in [2,1] is shown by the solid line. (a) Dotted line shows models of α_1 (no pit in [1,2]). (b) Dotted line shows models of α_2 (no pit in [2,2]).

α_1 = "There is no pit in [1,2]."
α_2 = "There is no pit in [2,2]."

INFERENCE

- Model checking is one possible algorithm for logical inference
- $KB \vdash_i \alpha =$ sentence α can be derived from KB by procedure i
- **Soundness:** i is sound if

  whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$
- **Completeness:** i is complete if

  whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$

PROPOSITIONAL LOGIC

Syntax

- **Logical constants**: true, false
- **Propositional symbols**: P, Q, S, ... (**atomic sentences**)
- Wrapping **parentheses**: (…)
- Sentences are combined by **connectives**:

∧ ...and	[conjunction]
∨ ...or	[disjunction]
⇒...implies	[implication / conditional] (**if – then** statements)
⇔..is equivalent	[biconditional] (**if and only if**)
¬ ...not	[negation]

- **Literal**: atomic sentence (positive literal) or negated atomic sentence (a negative literal)

Syntax

$$
\begin{array}{rcl}
\textit{Sentence} & \rightarrow & \textit{AtomicSentence} \mid \textit{ComplexSentence} \\
\textit{AtomicSentence} & \rightarrow & \textit{True} \mid \textit{False} \mid P \mid Q \mid R \mid \ldots \\
\textit{ComplexSentence} & \rightarrow & (\ \textit{Sentence}\ ) \mid [\ \textit{Sentence}\ ] \\
 & \mid & \neg\ \textit{Sentence} \\
 & \mid & \textit{Sentence} \land \textit{Sentence} \\
 & \mid & \textit{Sentence} \lor \textit{Sentence} \\
 & \mid & \textit{Sentence} \ \Rightarrow\ \textit{Sentence} \\
 & \mid & \textit{Sentence} \ \Leftrightarrow\ \textit{Sentence} \\
 & & \\
\text{OPERATOR PRECEDENCE} & : & \neg, \land, \lor, \Rightarrow, \Leftrightarrow
\end{array}
$$

Figure 7.7 A BNF (Backus–Naur Form) grammar of sentences in propositional logic, along with operator precedences, from highest to lowest.

SEMANTICS

- Each model specifies **true/false** for each proposition symbol
- Example,

$P_{1,2}$	$P_{2,2}$	$P_{3,1}$
False	False	True

← Pit in [3,1]. No pit in [2,2] and [1,2]

Three symbols: 8 possible models

- **Semantics:** Rules for evaluating truth with respect to some model m
- For complex sentence,
  - $\neg P$ 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.
  - $P \Leftrightarrow Q$ is true iff P and Q are both true or both false in m.

  $P \Leftrightarrow Q$, is true whenever both $P \Rightarrow Q$ and $Q \Rightarrow P$ are true.

TRUTH TABLE FOR LOGICAL CONNECTIVES

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:

$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1})$

Gives, true ∧ (false ∨ true) = true ∧ true = true.

a square is breezy *if* a neighboring square has a pit, and a square is breezy *only if* a neighboring square has a pit.

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

where $B_{1,1}$ means that there is a breeze in [1,1].

PROPOSITIONAL INFERENCE: ENUMERATION METHOD

Let $\alpha = A \lor B$ and $KB = (A \lor C) \land (B \lor \neg C)$

Is it the case that $KB \models \alpha$?

Check all possible models—α must be true wherever KB is true

A	B	C	$A \lor C$	$B \lor \neg C$	KB	α
$False$	$False$	$False$				
$False$	$False$	$True$				
$False$	$True$	$False$				
$False$	$True$	$True$				
$True$	$False$	$False$				
$True$	$False$	$True$				
$True$	$True$	$False$				
$True$	$True$	$True$				

PROPOSITIONAL INFERENCE: SOLUTION

A	B	C	$A \vee C$	$B \vee \neg C$	KB	α
$False$	$False$	$False$	$False$	$True$	$False$	$False$
$False$	$False$	$True$	$True$	$False$	$False$	$False$
$False$	$True$	$False$	$False$	$True$	$False$	$True$
$False$	$True$	$True$	$True$	$True$	$True$	$True$
$True$	$False$	$False$	$True$	$True$	$True$	$True$
$True$	$False$	$True$	$True$	$False$	$False$	$True$
$True$	$True$	$False$	$True$	$True$	$True$	$True$
$True$	$True$	$True$	$True$	$True$	$True$	$True$

KNOWLEDGE BASE FOR THE WUMPUS WORLD

- Symbols:

$P_{x,y}$ 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 the agent perceives a breeze in $[x, y]$.
$S_{x,y}$ is true if the agent perceives a stench in $[x, y]$.

- The sentences we write will suffice to derive $\neg P_{1,2}$ (there is no pit in [1,2])
- We label each sentence R_i so that we can refer to them:

- There is no pit in [1,1]:

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

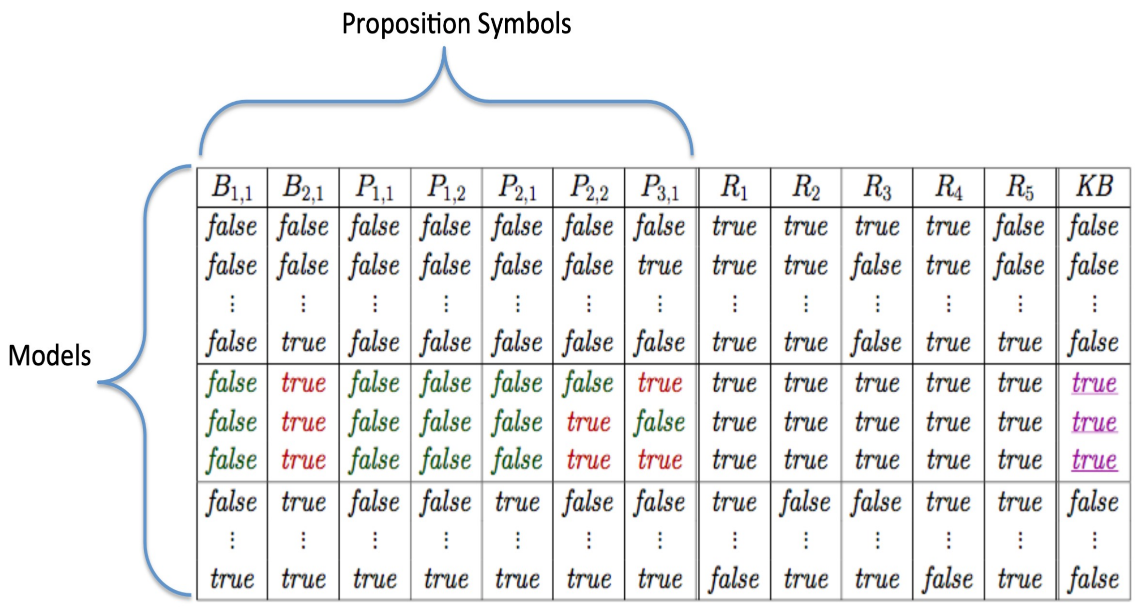
- A square is breezy if and only if there is a pit in a neighboring square. This has to be stated for each square; for now, we include just the relevant squares:

$$R_2 : \quad B_{1,1} \quad \Leftrightarrow \quad (P_{1,2} \lor P_{2,1}) .$$
$$R_3 : \quad B_{2,1} \quad \Leftrightarrow \quad (P_{1,1} \lor P_{2,2} \lor P_{3,1}) .$$

- The preceding sentences are true in all wumpus worlds. Now we include the breeze percepts for the first two squares visited in the specific world the agent is in, leading up to the situation in Figure 7.3(b).

$$R_4 : \quad \neg B_{1,1} .$$
$$R_5 : \quad B_{2,1} .$$

TRUTH TABLE FOR KNOWLEDGE BASE


Proposition Symbols
Models

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

KB is true if all rules (Rs) are true
– True in just three models

In all 3 rows,
- $P_{1,2}$ is false
  (there is no pit in [1,2])
- On the other hand, there might (or might not) be a pit in [2,2].

PROPOSITIONAL THEOREM PROVING

- Additional concepts related to entailment:

○ Logical equivalence

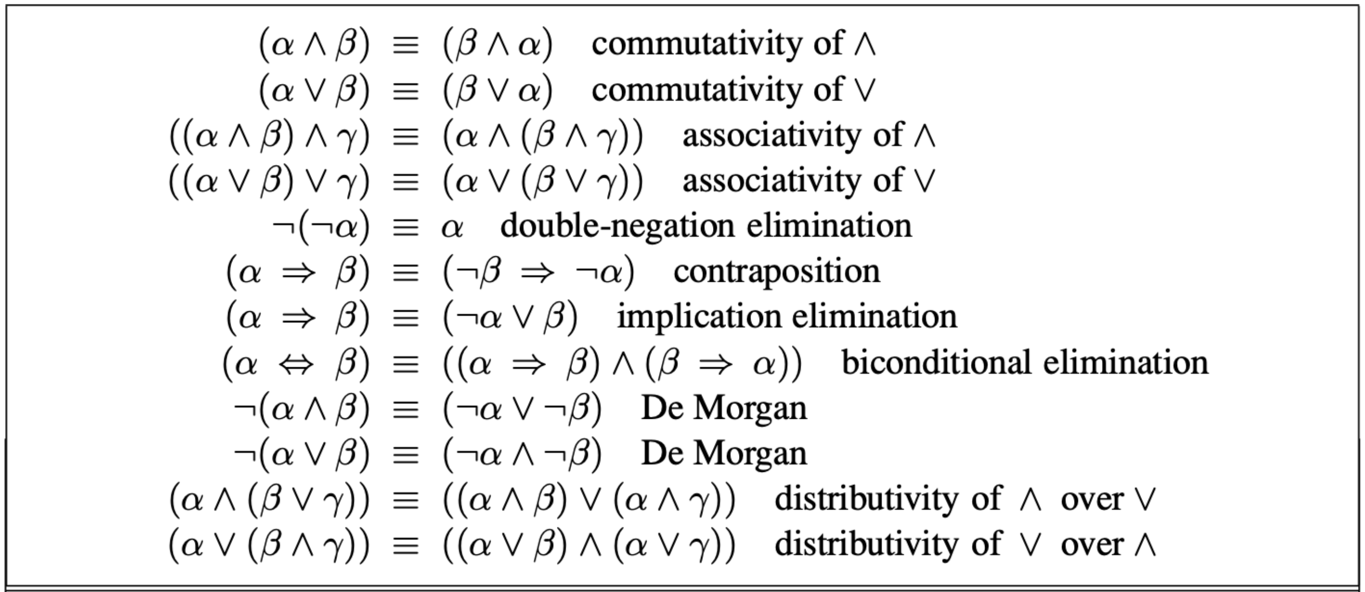
- two sentences α and β are logically equivalent if they are true in the same set of models.

- $\alpha \equiv \beta$

- alternatively, any two sentences α and β are equivalent only if each of them entails the other:

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

PROPOSITIONAL THEOREM PROVING



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

Figure 7.11 Standard logical equivalences. The symbols α, β, and γ stand for arbitrary sentences of propositional logic.

PROPOSITIONAL THEOREM PROVING

- Additional concepts related to entailment:

- Validity

  - A sentence is valid if it is true in *all* models (**tautologies**)

  - For example, the sentence P ∨ ¬P is valid

- - Gives us deduction theorem: α |= β if and only if α ⇒ β is valid
  - can decide if α |= β by checking that α ⇒ β is true in all models

- Satisfiability

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

  - For example, the knowledge base given earlier, (R1 ∧ R2 ∧ R3 ∧ R4 ∧ R5), is satisfiable because there are three models in which it is true.

PROPOSITIONAL THEOREM PROVING

- Inference Rule:

RULE	PREMISE	CONCLUSION
Modus Ponens	$A, A \Rightarrow B$	B
And Introduction	A, B	$A \wedge B$
And Elimination	$A \wedge B$	A
Double Negation	$\neg\neg A$	A
Unit Resolution	$A \vee B, \neg B$	A
Resolution	$A \vee B, \neg B \vee C$	$A \vee C$

$$\frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta} \longleftarrow$$ Modus Ponens

- all of the logical equivalences can be used as inference rules

PROPOSITIONAL THEOREM PROVING

- Inference Rule:
- All of the logical equivalences can be used as inference rules

  - For example, the equivalence for biconditional elimination yields the two inference rules:

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

PROPOSITIONAL THEOREM PROVING

- Let’s see how these inference rules and equivalences can be used in the wumpus world?
- How to prove $\neg P_{1,2}$ (there is no pit in [1,2])?
- Recall the Knowledge base R_1 through R_5 -

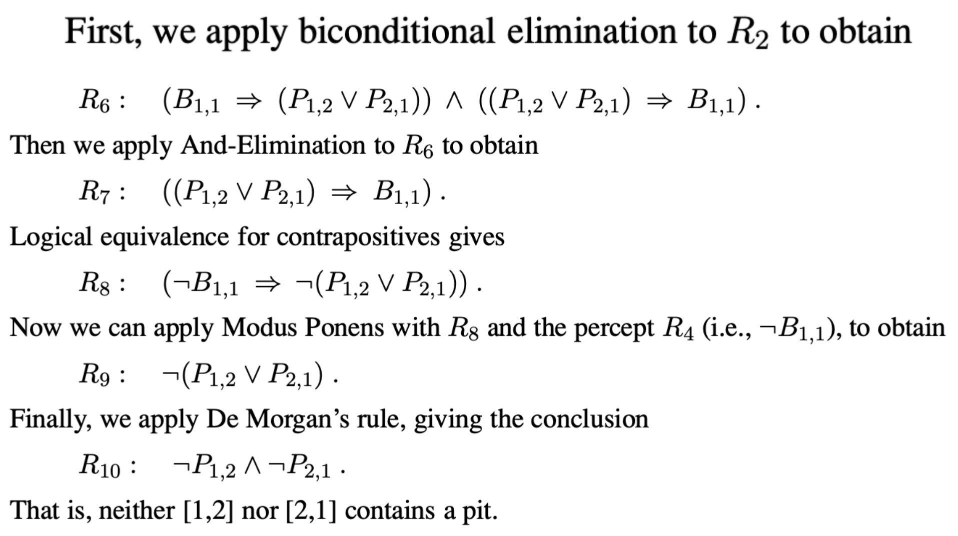
R1: $\neg P_{1,1}$

R2: $B_{1,1} \Leftrightarrow (P_{2,1} \vee P_{1,2})$

R3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

R4: $\neg B_{1,1}$

R5: $B_{2,1}$



First, we apply biconditional elimination to R_2 to obtain

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

Then we apply And-Elimination to R_6 to obtain

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

Logical equivalence for contrapositives gives

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

Now we can apply Modus Ponens with R_8 and the percept R_4 (i.e., $\neg B_{1,1}$), to obtain

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

Finally, we apply De Morgan’s rule, giving the conclusion

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

That is, neither [1,2] nor [2,1] contains a pit.

PROPOSITIONAL THEOREM PROVING

- Could apply any search algorithm:

  – Initial state: initial KB

  – 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.

PROPOSITIONAL THEOREM PROVING

- Proof by Resolution:

  - a single inference rule, **resolution**, that yields a complete inference algorithm when coupled with any complete search algorithm.

- Knowledge Base (Wumpus world):

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

True for all WW $\rightarrow$ $R_{12}: \quad B_{1,2} \Leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{1,3})$.

$$\left.\begin{array}{ll} R_{13}: & \neg P_{2,2} \, . \\ R_{14}: & \neg P_{1,3} \, . \end{array}\right\} \longleftarrow$$

Derived rules. (Prove it)

Apply Biconditional elimination to R_3, then And-elimination, then Modus Ponens with R_5 :

$$R_{15}: \quad P_{1,1} \lor P_{2,2} \lor P_{3,1} \, .$$

Apply the resolution rule: R_{13} resolves with R_{15}:

$$R_{16}: \quad P_{1,1} \lor P_{3,1} \, .$$

Apply the resolution rule: R_1 resolves with R_{16}:

$$R_{17}: \quad P_{3,1} \, .$$

$\neg P_{1,1}$

PROPOSITIONAL THEOREM PROVING

- The resolution rule applies only to clauses (a disjunction of literals).
- the unit resolution rule takes a **clause**, and a literal and produces a new clause.
- complementary literals (one is the negation of the other)

❖ **Conjunctive normal form (CNF):**

- Every sentence of propositional logic is logically equivalent to a conjunction of clauses.
- A sentence expressed as a conjunction of clauses is said to be in **conjunctive normal form** or **CNF**
- See Figure 7.14 (A grammar for conjunctive normal form)

PROPOSITIONAL THEOREM PROVING

- **Conversion to CNF:**
- converting the sentence

  $B1,1 \Leftrightarrow (P1,2 \lor P2,1)$ into CNF.

The original sentence is now in CNF, as a conjunction of three clauses. →

Example:

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

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 Morgan's 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:

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

PROPOSITIONAL THEOREM PROVING

- **A resolution algorithm:**
- Idea: Proof by contradiction

    - to show that KB |= α → we show that (KB ∧ ¬α) is unsatisfiable

- Plan:

    - first, convert (KB ∧ ¬α) 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;

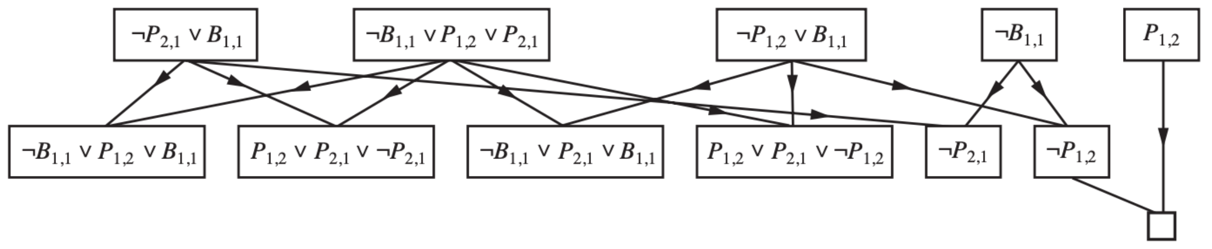
        -> or, two clauses resolve to yield the *empty* clause, in which case KB entails α.

PROPOSITIONAL THEOREM PROVING

❖ **Example:**

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

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



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

Figure 7.13 Partial application of PL-RESOLUTION to a simple inference in the wumpus world. $\neg P_{1,2}$ is shown to follow from the first four clauses in the top row.

PROPOSITIONAL THEOREM PROVING

- **Horn clauses and definite clauses**

  Definite clause: a disjunction of literals of which *exactly one is positive*.

  - Ex: the clause $(\neg L_{1,1} \lor \neg Breeze \lor B_{1,1})$ is a definite clause, whereas $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1})$ is not.

  Horn clause: a disjunction of literals of which at most one element is positive.

  - Ex: the clause $(\neg L_{1,1} \lor \neg Breeze \lor B_{1,1})$; $\neg B_{2,2}$

  - Can be written as implications (whose premise is a conjunction of positive literals and whose conclusion is a single positive literal.): $(L_{1,1} \land Breeze) \Rightarrow B_{1,1}$

  (conjunction of symbols) $\Rightarrow$ symbol

- All definite clauses are Horn clauses.
- Can be use in forward/backward chaining proof algorithm – These algorithms are very natural and run in linear time !

PROPOSITIONAL THEOREM PROVING

- **Read Section 7.5.4**
- **Forward chaining:**
- **Idea:** Work forward from the known facts to try to reach the target goal

  - It begins from known facts (positive literals) (true) in the knowledge base.

  - If all the premises of an implication are known, then its conclusion is added to the set of known facts.

  - For example, if $L_{1,1}$ and Breeze are known and $(L_{1,1} \land \text{Breeze}) \Rightarrow B_{1,1}$ is in the knowledge base, then $B_{1,1}$ can be added.

  - This process continues until the query q is added or until no further inferences can be made.

- Horn clause and Modus Ponens can be used with forward chaining

PROPOSITIONAL THEOREM PROVING

- **Forward chaining**

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

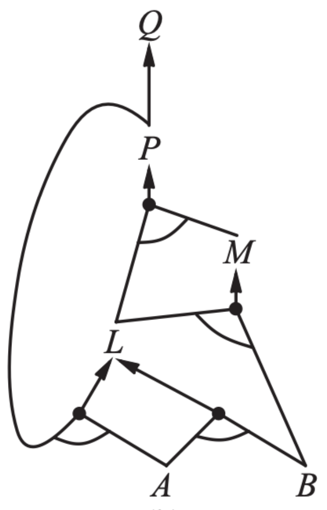
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

Known facts $\longrightarrow$ $\left\{ \begin{array}{l} A \\ B \end{array} \right.$

(a)



(b)

multiple links joined by an arc indicate a conjunction

Figure 7.16 (a) A set of Horn clauses. (b) The corresponding AND–OR graph.

PROPOSITIONAL THEOREM PROVING

- **Read Section 7.5.4**
- **Backward chaining**

HOMEWORK!

- https://www.swi-prolog.org/download/stable
- Read Chapter 7 (Section 7.1 – 7.5)