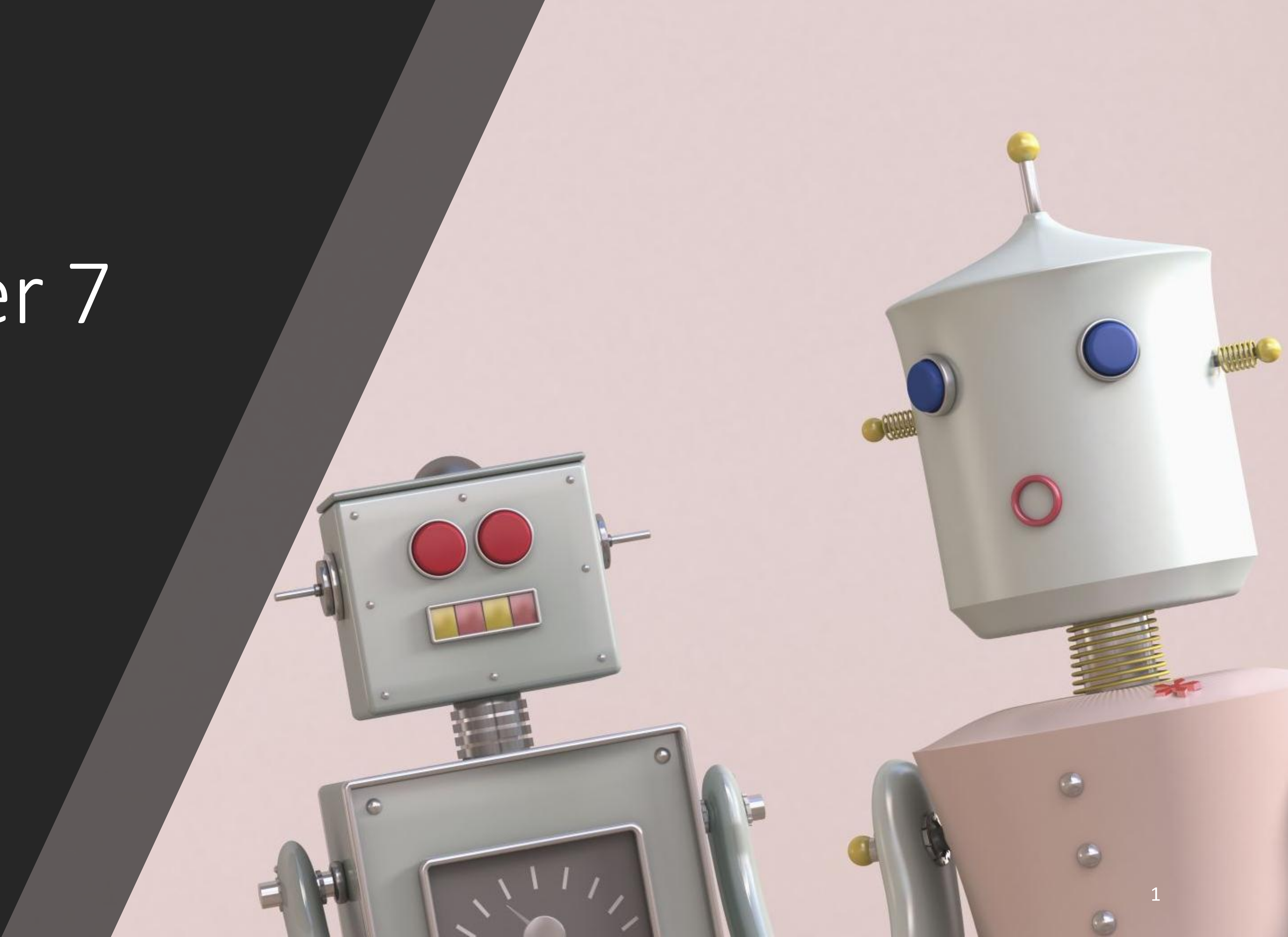
Chapter 7

Logical Agents



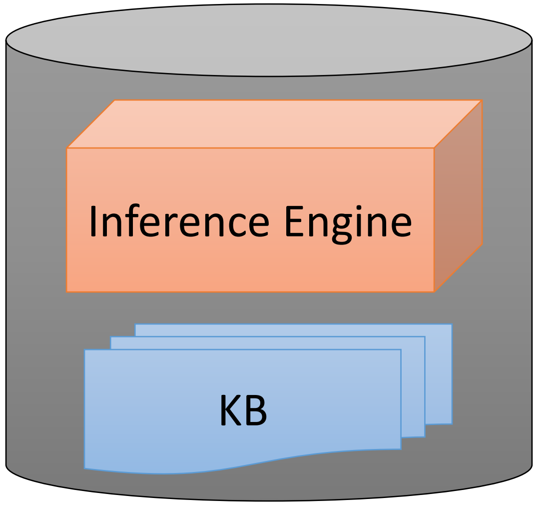
Introduction

- Previously, we saw problem-solving agents: know things, but only in a very limited, inflexible sense
- **Knowledge-based agents**: use processes of reasoning that operate on internal representations of knowledge
  - Develop **logic** as a general class of representations to support knowledge-based agents

Knowledge-based agents

- Basic component of a knowledge-based agent is its knowledge base, or KB
- Knowledge base: is a set of sentences expressed in a language called a knowledge representation language and represents some assertion about the world
    - Example sentence: α = It is raining
- Add new sentences to the KB: TELL
- Query what is known: ASK

Involve inference



Example

Automated taxi

- goal: take a passenger from San Francisco to Marin County
- KB: contains knowledge that the Golden Gate Bridge is the only link between the two locations
  - Then: we can expect it to cross the Golden Gate Bridge *because it knows that that will achieve its goal*

KB-Agent

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 ← *t* + 1
  return *action*

agent perceived the given percept at the given time

asks what action should be done at the current time

constructs a sentence asserting that the chosen action was executed

KB agents construction

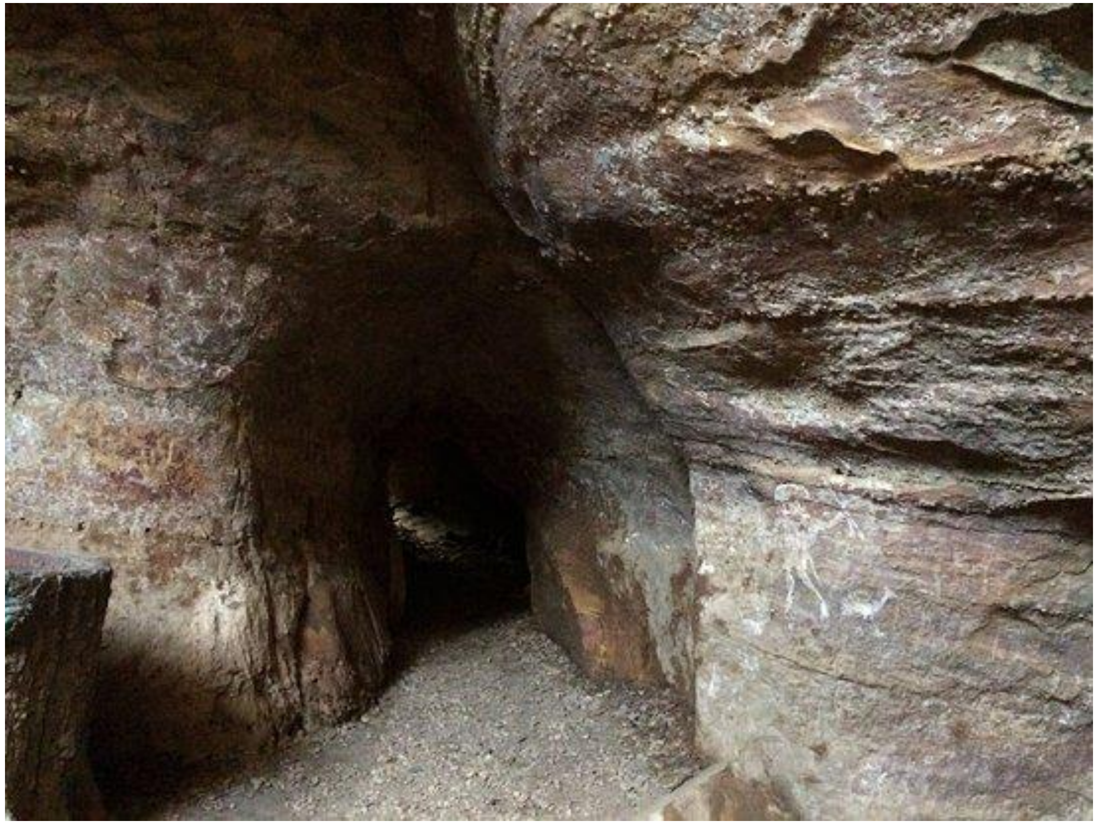
1. **Declarative** approach: Starting with an empty KB, the agent designer can TELL sentences one by one until the agent knows how to operate in its environment
2. **Procedural** approach: encodes desired behaviors directly as program code

- Usually combine the two

Wumpus World

Wumpus world: a cave consisting of rooms connected by passageways.

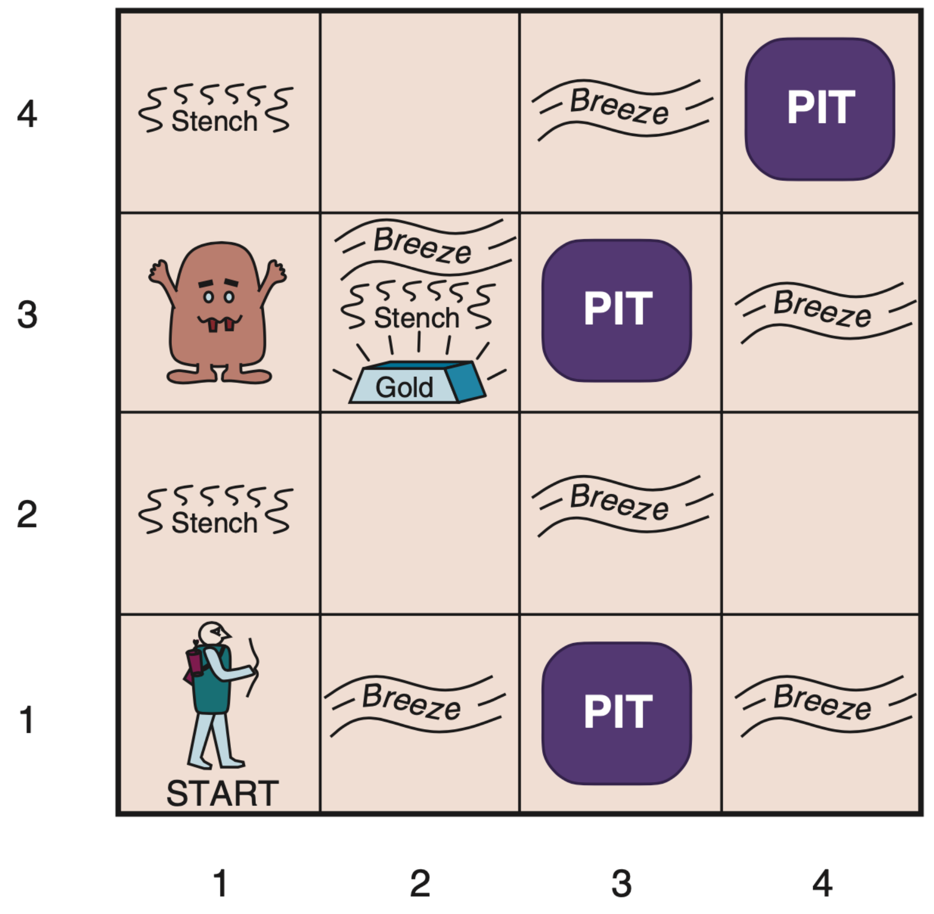
- The terrible Wumpus eats anyone who enters its room
- Wumpus can be shot by an agent, but the agent has only one arrow
- Some rooms contain pits that will trap anyone who wanders into these
- One room has a heap of gold



Wumpus World PEAS description

Environment:

- Squares adjacent to the Wumpus are smelly
- Squares adjacent to the pit are breezy
- **Sensors**: Stench, Breeze, Glitter, Bump, Scream
- **Actions**: turn Left, turn Right, Forward, Grab, Release, Shoot



Exploring the Wumpus world

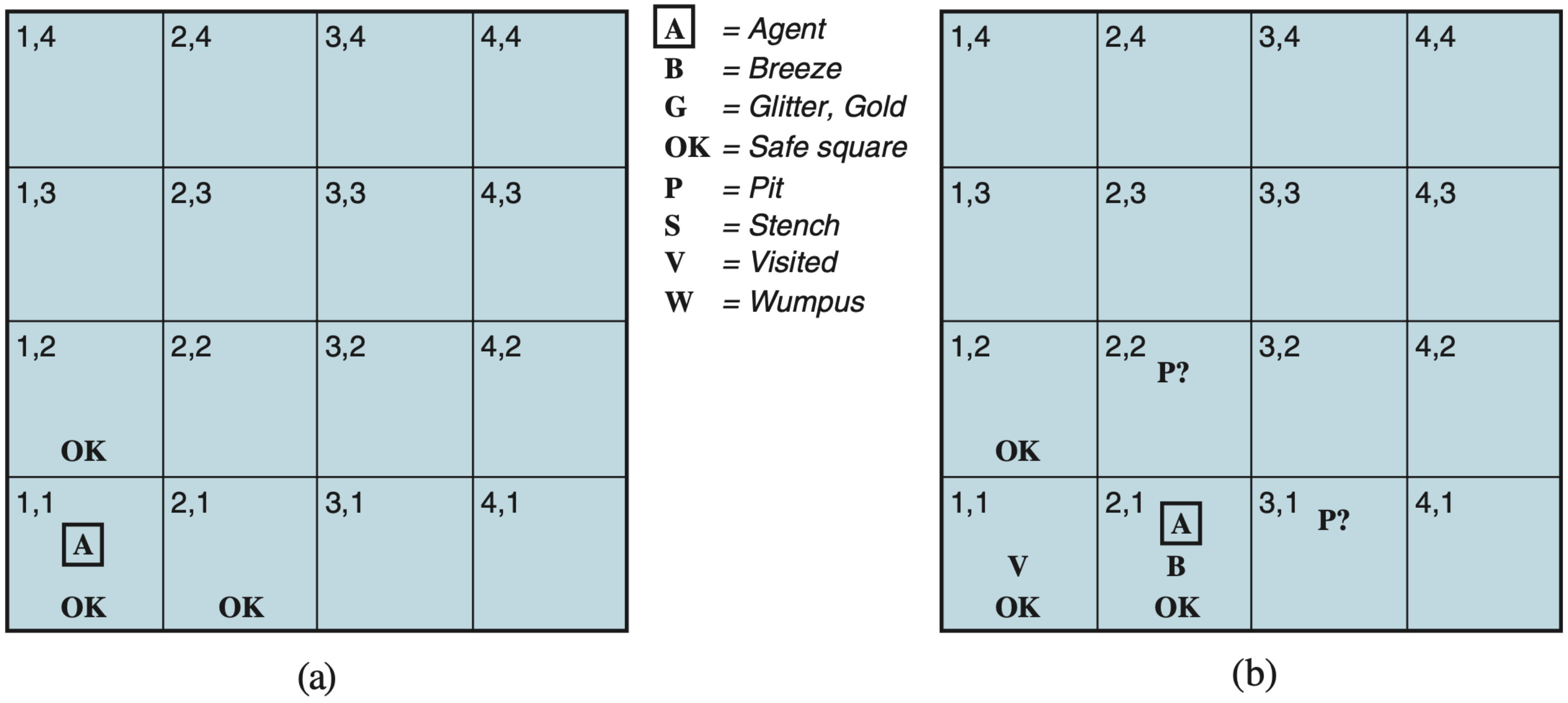


Figure 7.3 The first step taken by the agent in the wumpus world. (a) The initial situation, after percept $[None, None, None, None, None]$. (b) After moving to [2,1] and perceiving $[None, Breeze, None, None, None]$.

Sensors: Stench, Breeze, Glitter, Bump, Scream

Exploring the Wumpus world

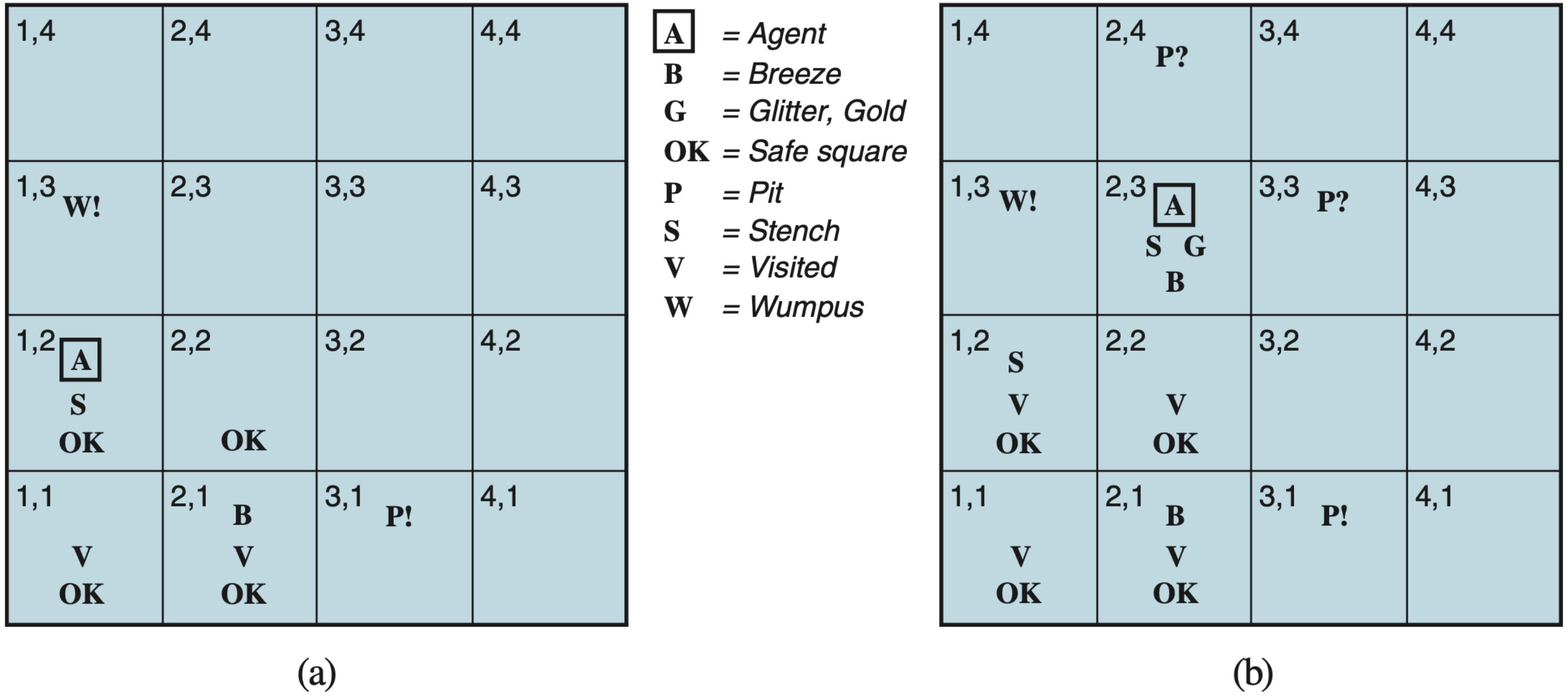


Figure 7.4 Two later stages in the progress of the agent. (a) After moving to [1,1] and then [1,2], and perceiving [*Stench*, *None*, *None*, *None*, *None*]. (b) After moving to [2,2] and then [2,3], and perceiving [*Stench*, *Breeze*, *Glitter*, *None*, *None*].

Logic in general

Logics are formal languages for representing information such that conclusions can be drawn

- **Syntax** defines how the sentences in the language are constructed
  - Called well-formed sentences
- **Semantics** define the "meaning" of sentences;
  - define **truth** of a sentence in a **possible world** or **model**
- Example: the language of arithmetic
  - Syntax: $x + 2 \geq y$ is a sentence; $x2 + y > \{\}$ is not a sentence
  - Semantics: $x + 2 \geq y$ is true iff the number $x + 2$ is no less than the number y
  - Semantics: $x + 2 \geq y$ is true in a world where $x = 7, y = 1$
  - Semantics: $x + 2 \geq y$ is false in a world where $x = 0, y = 6$

Models of sentence α: $x + y = 4$

Real World

4 women(x) + 0 men(y) = 4

Models

m_1	m_2	m_3
$x = 4, y = 0$	$x = 3, y = 1$	$x = 2, y = 2$
m_4	m_5	m_6
$x = 1, y = 3$	$x = 0, y = 4$	$x = 1, y = 4$

- Model m **satisfies** α, or m **is a model of** α
- Model m **does not satisfy** α, or m **is not a model of** α
- $M(\alpha)$ is the set of all **models** of α

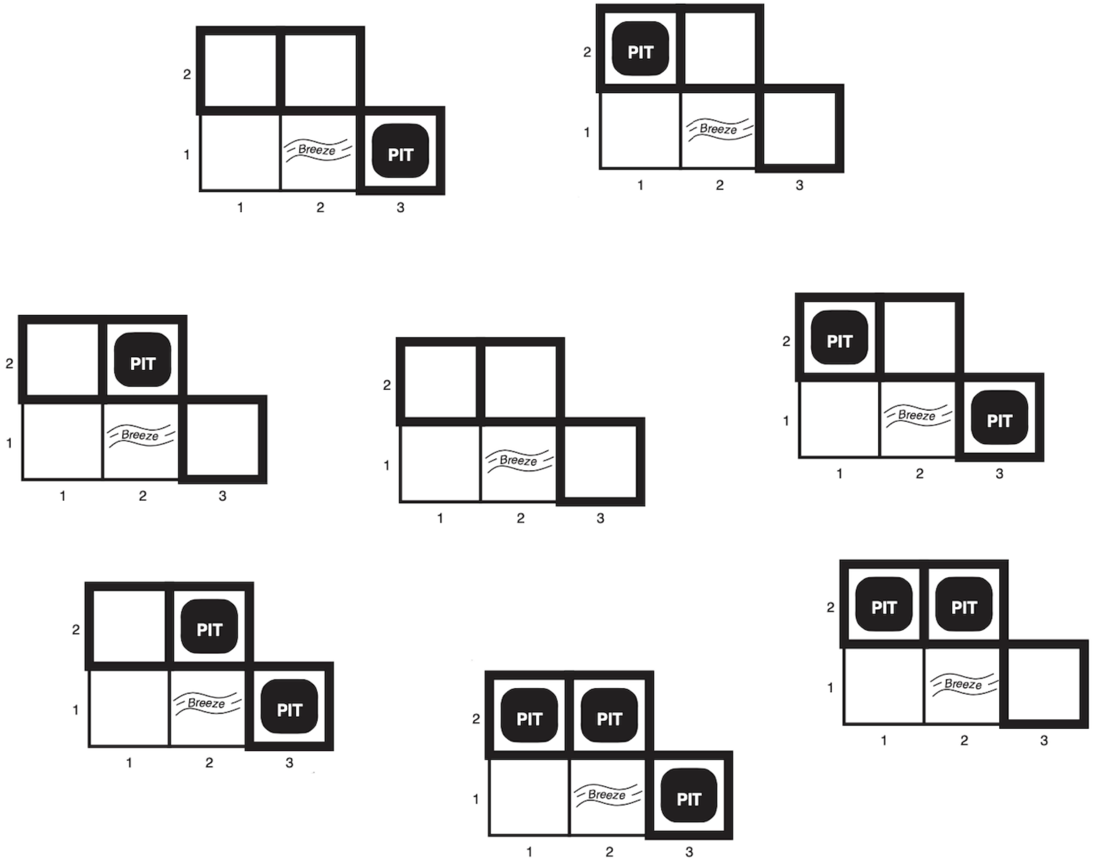
Logical reasoning: entailment

- Logical **entailment** between sentences: a sentence *follows logically* from another sentence
- Mathematically: $\alpha \vDash \beta$
- $\alpha \vDash \beta$ iff, in every model in which α is true, β is also true:

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

- Example: $\alpha: x = 0$ entails the sentence $\beta: xy = 0$
  - In any model where x is zero, xy is also zero (regardless of the value of y)

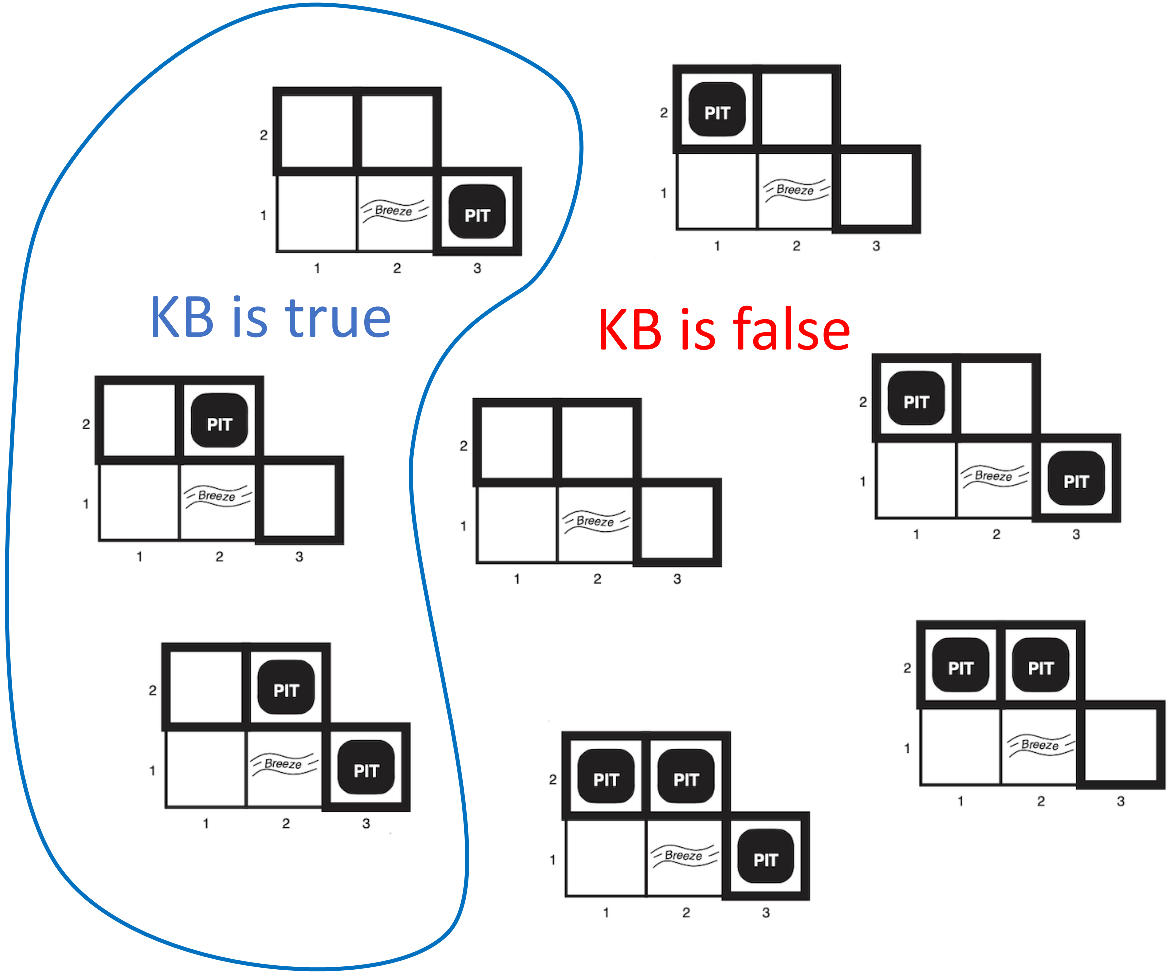
Wumpus world models



Possible models for the presence of pits in squares [1,2], [2,2], and [3,1]

$2^3 = 8$ possible models

Wumpus world models



Possible models for the presence of pits in squares [1,2], [2,2], and [3,1]

$2^3 = 8$ possible models

KB = percepts + rules of the Wumpus world

Wumpus World

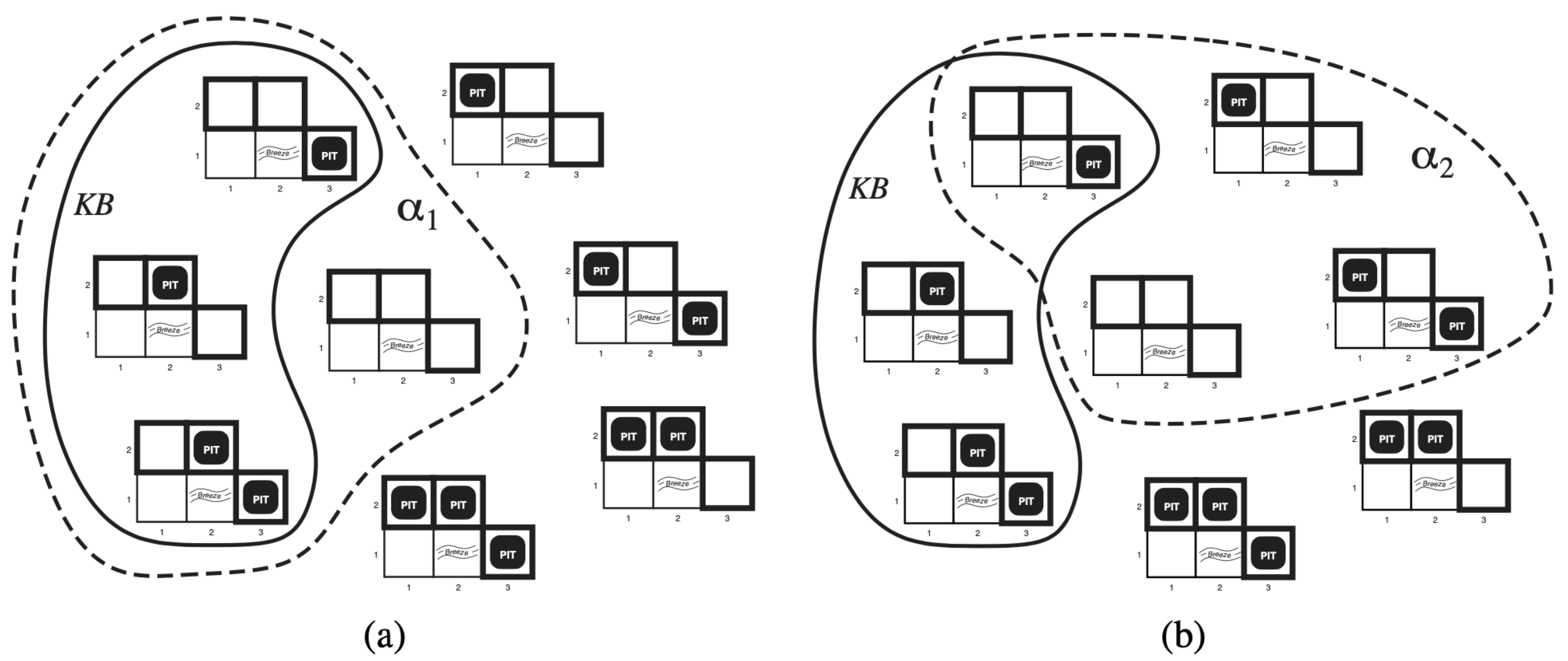


Figure 7.5 Possible models for the presence of pits in squares [1,2], [2,2], and [3,1]. The KB corresponding to the observations of nothing in [1,1] and a breeze in [2,1] is shown by the solid line. (a) Dotted line shows models of α_1 (no pit in [1,2]). (b) Dotted line shows models of α_2 (no pit in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

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

$KB \vDash \alpha_1$ in every model in which KB is T, α_1 is also T

$KB \nvDash \alpha_2$ in some models in which KB is T, α_2 is F
The agent *cannot* conclude that there is no pit in [2,2]

Wumpus World inference

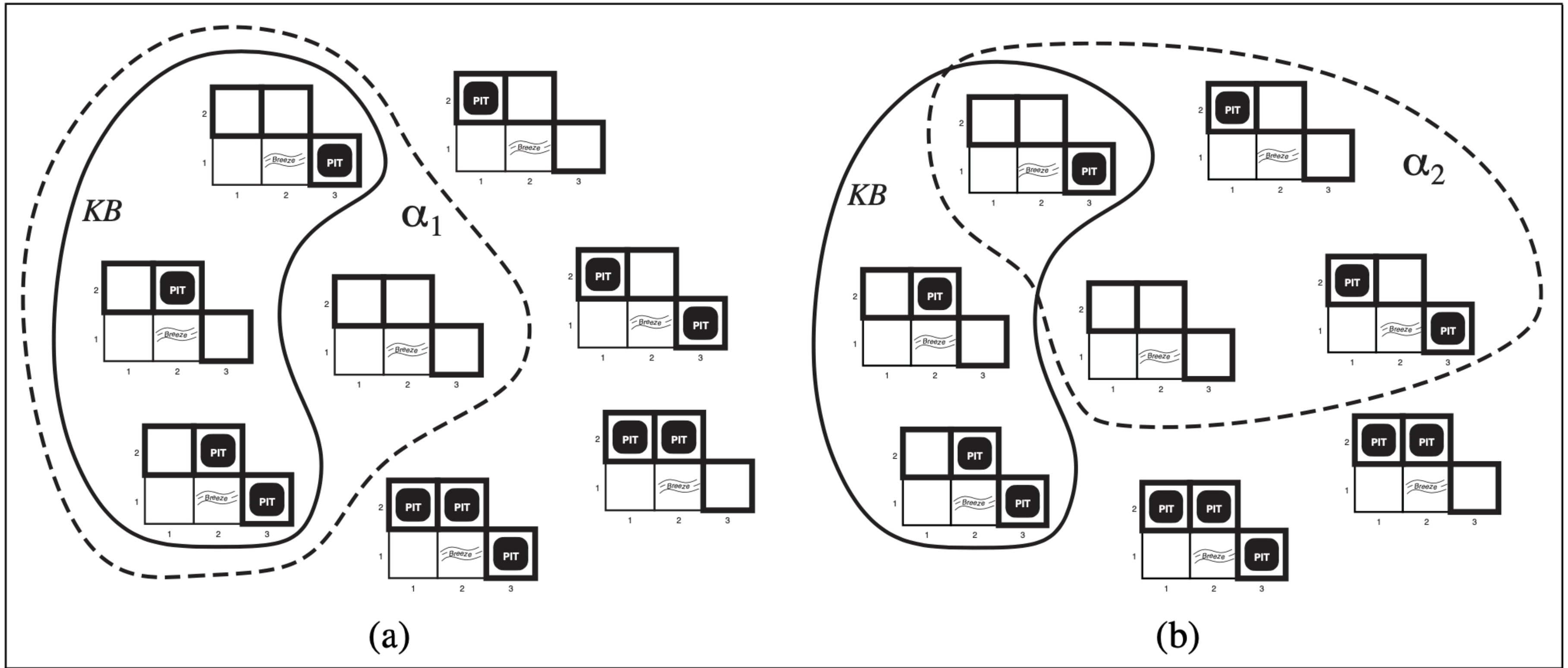


Figure 7.5 Possible models for the presence of pits in squares [1,2], [2,2], and [3,1]. The KB corresponding to the observations of nothing in [1,1] and a breeze in [2,1] is shown by the solid line. (a) Dotted line shows models of α_1 (no pit in [1,2]). (b) Dotted line shows models of α_2 (no pit in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

Logical inference:

This inference algorithm is called **model checking**, because it enumerates all possible models to check that α is true in all models in which KB is true:

$M(KB) \subseteq M(\alpha)$

Inference and entailment

- To understand entailment and inference:
  - the set of all consequences of KB is a haystack, α is a needle
- Entailment: The needle being in the haystack
- Inference: Finding the needle

Inference

- If an inference algorithm i can derive α from KB: "α is derived from KB by i"

$$KB \vdash_i \alpha$$

- **Sound** or **truth-preserving**: If an inference algorithm derives only entailed sentences
  - Soundness is highly desirable.
  - Model checking is a sound algorithm
- **Completeness**: If an inference algorithm can derive any sentence that is entailed
  - Also highly desirable

Propositional logic: Syntax

- Propositional logic is the simplest logic
- Syntax of propositional logic defines the allowable sentences
- Atomic sentences consist of a single proposition symbol
- A proposition may be True or False
  - Examples: $P, Q, R, W_{1,3}$ and $North$
- Complex sentences are constructed from atomic sentences, using parentheses and logical connectives.

Propositional logic: Syntax

The proposition symbols P_1, P_2 etc are sentences

If S is a sentence:

- $\neg S$ is a sentence (negation)

If S_1 and S_2 are sentences:

- $S_1 \wedge S_2$ is a sentence (conjunction)
- $S_1 \vee S_2$ is a sentence (disjunction)
- $S_1 \Rightarrow S_2$ is a sentence (implication)
- $S_1 \Leftrightarrow S_2$ is a sentence (biconditional)

A BNF (Backus–Naur Form) grammar of sentences in propositional logic

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

Propositional logic: Semantics

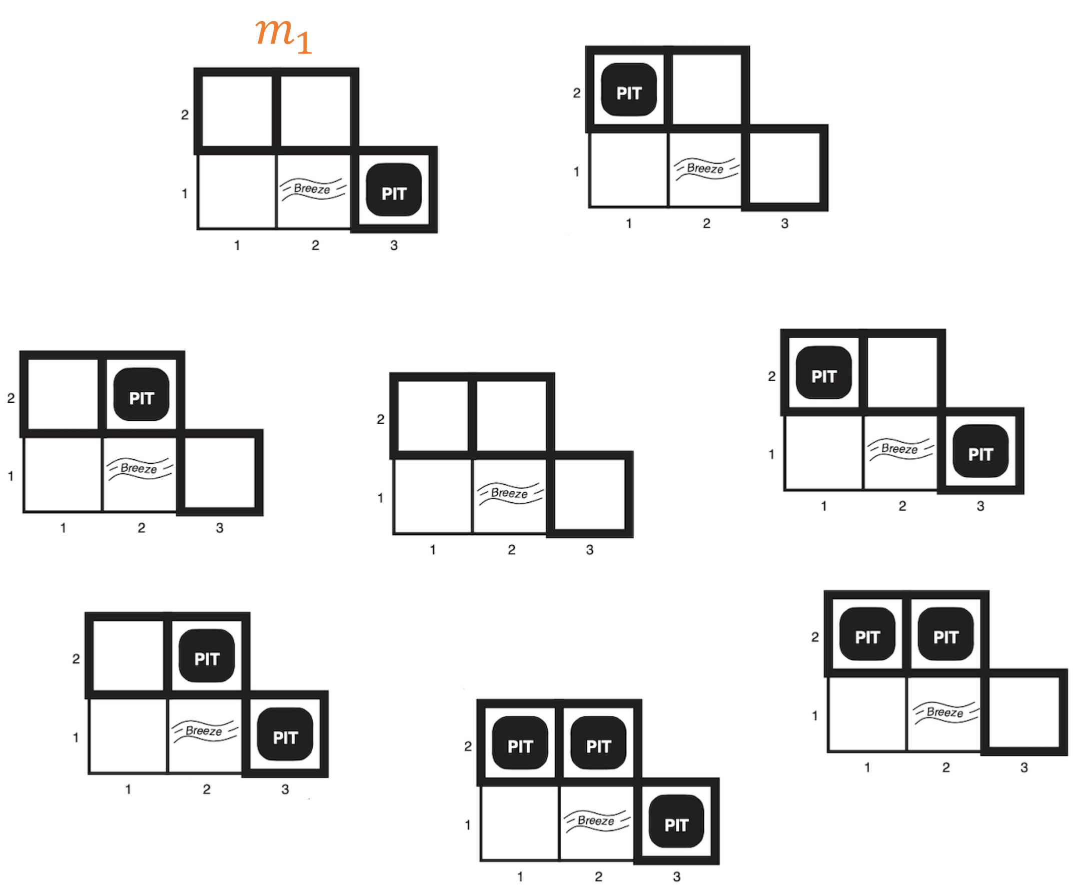
- **Semantics**: define the rules for determining the truth of a sentence with respect to a particular model
- In propositional logic, a model simply fixes the **truth value**—T or F—for every proposition symbol
- Next, compute T or F for all sentences

Propositional logic: Semantics

- Example: if the sentences in the knowledge base make use of the proposition symbols $P_{1,2}$, $P_{2,2}$, and $P_{3,1}$, then one possible model is

$$m_1 = \{P_{1,2} = false, P_{2,2} = false, P_{3,1} = true\}$$

Note: 3 symbols of T or F, 2^3 worlds



Propositional logic: Semantics

Compute T or F for Complex sentences:

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*

Note: $P \implies Q$ (if P then Q)

- $P \implies Q$ says: "If P is true, then I am claiming that Q is true. Otherwise, I am making no claim."
- The only way for this sentence to be *false* is if P is true but Q is false.
- PL does not require any relation of *causation* or *relevance* between P and Q
  - The sentence "5 is odd implies Tokyo is the capital of Japan" is a true sentence even though it is odd
- An implication is true whenever its antecedent is false
  - For example, "5 is even implies Sam is smart" is true, regardless of whether Sam is smart

Propositional logic: Semantics

Compute T or F for Complex sentences:

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*

Note: $P \Leftrightarrow Q$

- True whenever both $P \Rightarrow Q$ and $Q \Rightarrow P$ are true
- Often written as “P if and only if Q.”

Wumpus World KB

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 $[x, y]$.

Wumpus World KB

Sentences: True in all Wumpus worlds

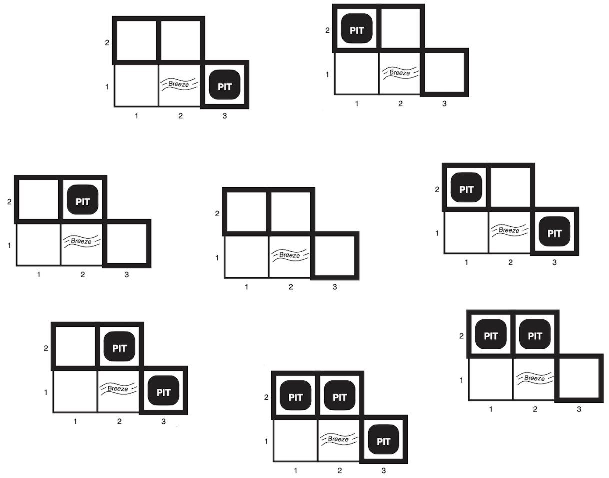
- There is no pit in [1,1]:

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

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

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

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



Wumpus World KB

Sentences: From agent percepts

- Now we include the breeze percepts for the first two squares visited in the specific world the agent is in:

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

R_5: $B_{2,1}$

KB

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: $B_{2,1}$

Model-checking approach

Goal: Inference

- i.e. whether $KB \vDash \alpha$ for some sentence α
- Example: is $\neg P_{1,2}$ entailed by our KB?

First algorithm for inference is a model-checking approach:

- Enumerate the models and check that α is T in every model in which KB is T
- Models are assignments of T or F to every proposition symbol
- Our example symbols: $B_{1,1}$, $B_{2,1}$, $P_{1,1}$, $P_{1,2}$, $P_{2,1}$, $P_{2,2}$, and $P_{3,1}$
- Seven symbols: $2^7 = 128$ possible worlds
- Time complexity: $O(2^n)$

Model-checking approach

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

Pit in $P_{1,2}$ is F

KB is T

Maybe there is a pit in $P_{2,2}$, sometimes it is T and sometimes F

Figure 7.9 A truth table constructed for the knowledge base given in the text. *KB* is true if R_1 through R_5 are true, which occurs in just 3 of the 128 rows (the ones underlined in the right-hand column). In all 3 rows, $P_{1,2}$ is false, so there is no pit in [1,2]. On the other hand, there might (or might not) be a pit in [2,2].

Example: Model-checking approach

A	B	C	**KB** $(A \lor C) \land (B \lor \neg C)$	$\boldsymbol{\alpha}$ $A \lor B$
False	False	False	False	False
False	False	True	False	False
False	True	False	False	True
False	True	True	True	True
True	False	False	True	True
True	False	True	False	True
True	True	False	True	True
True	True	True	True	True

Example: Model-checking approach

A	B	C	**KB** **$(A \lor C) \land (B \lor \neg C)$**	**α** **$A \lor B$**
False	False	False	False	False
False	False	True	False	False
False	True	False	False	True
False	True	True	True	True
True	False	False	True	True
True	False	True	False	True
True	True	False	True	True
True	True	True	True	True

$KB \vDash \alpha$

Theorem Proving approach

Second Algorithm: entailment can be done by theorem proving

- applying rules of inference directly to the sentences in the KB to construct a proof of the desired sentence without consulting models
- Can be more efficient than model checking
- Need some concepts:

1. Logical equivalence
2. Validity
3. Satisfiability

1. Logical equivalence

- Two sentences are **logically equivalent** iff they are T in the same models
- $\alpha \equiv \beta$ iff $\alpha \vDash \beta$ and $\beta \vDash \alpha$

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

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

2. Validity

- A sentence is valid if it is T in *all* models. For example, the sentence $P \lor \neg P$ is valid.
- Valid sentences are also known as tautologies
- Every valid sentence is logically equivalent to T

- Deduction theorem:

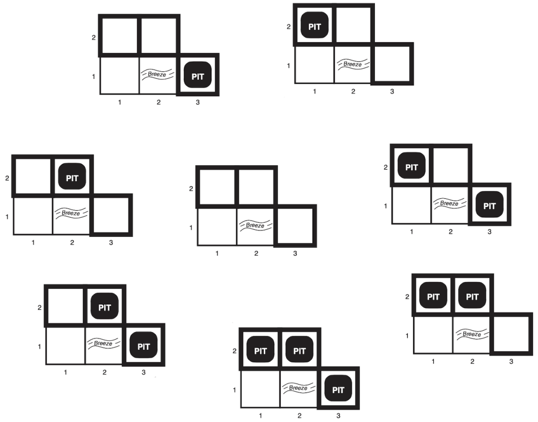
For any sentences α and β, $\alpha \vDash \beta$ iff the sentence $(\alpha \Rightarrow \beta)$ is valid

➢ every valid implication sentence describes a legitimate inference

3. Satisfiability

- A sentence is satisfiable if it is true in *some* model
  - A sentence is **satisfiable** if it is true in **some** model e.g., $A \vee B$
  - A sentence is **unsatisfiable** if it is true in **no** models e.g., $A \wedge \neg A$
- The sentence $(R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5)$ is satisfiable because there are three models in which it is true
- **SAT** problem: the problem of determining the satisfiability of sentences in propositional logic

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: $B_{2,1}$



Validity and satisfiability

- α is valid iff $\neg\alpha$ is unsatisfiable
- Contrapositively: α is satisfiable iff $\neg\alpha$ is not valid
- We also have the following useful result:
- $\alpha \vDash \beta$ *iff the sentence* $(\alpha \wedge \neg\beta)$ *is unsatisfiable.*
  - Proving β from α by checking the unsatisfiability of $(\alpha \wedge \neg\beta)$ corresponds proof by **contradiction**: assume β is F, shows that this leads to a contradiction with α.

Theorem proving

- Rule 1: **Modus Ponens** (Latin for *mode that affirms*)

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

"raining implies soggy courts", "raining"
Infer: "soggy courts"

- Rule 2: **Modus Tollens** (Latin for *mode that denies*)

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

"raining implies soggy courts", "courts not soggy"
Infer: "not raining"

- Rule 3: **And-Elimination**

$$\frac{\alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_n}{\alpha_i}$$

- Rules: Figure 7.11 slide 33
- Later: Resolution Rule

R_1: $\neg P_{1,1}$
R_2: $B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$
R_3: $B_{2,1} \Leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{3,1})$
R_4: $\neg B_{1,1}$
R_5: $B_{2,1}$

Example

- To R_2, apply Biconditional Elimination

R_6: $\left(B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})\right) \land \left((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}\right)$

- And-Elimination:

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

- Logical equivalence for contrapositives:

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

- Modus Ponens with R_8 and the percept R_4:

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

- De Morgan's rule:

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

Theorem proving

- Previous example: proof by hand
- Can apply any of the search algorithms in Chapter 3 to find a sequence of steps that constitutes a proof

Define a proof problem as follows:

- **INITIAL STATE**: the initial KB
- **ACTIONS**: 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

Monotonicity

- Logical systems have the **monotonicity property**
- The set of entailed sentences can only *increase* as information is added to KB
- For any sentences α and β:
- if $KB \vDash \alpha$, then $KB \land \beta \vDash \alpha$

Continued example

- The agent returns from [2,1] to [1,1] then [1,2], where it perceives a stench, but no breeze.

R_{11}: $\neg B_{1,2}$, R_{12}: $B_{1,2} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{1,3})$

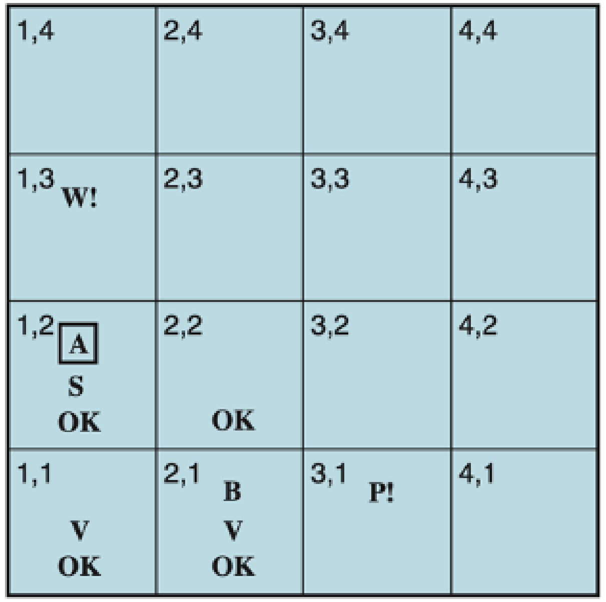
- Using the same process as before, we derive:

R_{13}: $\neg P_{2,2}$, R_{14}: $\neg P_{1,3}$

- Biconditional elimination to R_3, followed by Modus Ponens with R_5, to obtain the fact that there is a pit in [1,1], [2,2], or [3,1]

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

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



Resolution Rule

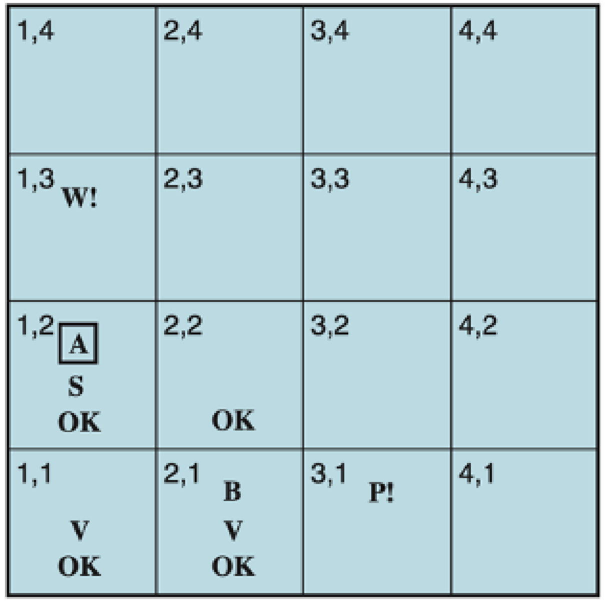
- The resolution Rule: the literal $\neg P_{2,2}$ (R_{13}) *resolves with* the literal $P_{2,2}$ (R_{15}) to give the **resolvent:**

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

- Do the same for R_1:

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

R_1: $\neg P_{1,1}$
R_2: $B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$
R_3: $B_{2,1} \Leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{3,1})$
R_4: $\neg B_{1,1}$
R_5: $B_{2,1}$
R_6: $(B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$
R_7: $(P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}$
R_8: $\neg B_{1,1} \Rightarrow \neg(P_{1,2} \lor P_{2,1})$
R_9: $\neg(P_{1,2} \lor P_{2,1})$
R_{10}: $\neg P_{1,2} \land \neg P_{2,1}$



Unit resolution rule

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

l_i and m are complementary literals

- $l_1 \vee \cdots \vee l_k$ are called **disjunctions** of literals

Full resolution rule

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

l_i and m_j are complementary literals

Special forms

Conjunctive Normal Form (CNF—universal)

$\textit{conjunction}$ of $\underbrace{\textit{disjunctions} \text{ of } \textit{literals}}_{\textit{clauses}}$

E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

Disjunctive Normal Form (DNF—universal)

$\textit{disjunction}$ of $\underbrace{\textit{conjunctions} \text{ of } \textit{literals}}_{\textit{terms}}$

E.g., $(A \wedge B) \vee (A \wedge \neg C) \vee (A \wedge \neg D) \vee (\neg B \wedge \neg C) \vee (\neg B \wedge \neg D)$

Horn Form (restricted)

conjunction of *Horn clauses* (clauses with ≤ 1 positive literal)

E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

Often written as set of implications:

$B \Rightarrow A$ and $(C \wedge D) \Rightarrow B$

Resolution

- Properties of the resolution rule:
  - Sound
  - Complete (yields to a complete inference algorithm)
- The resolution rule forms the basis for a family of complete inference algorithms
- Resolution rule is used to either confirm or refute a sentence, but it cannot be used to enumerate true sentences
- Resolution can be applied only to disjunctions of literals. How can it lead to a complete inference procedure for all propositional logic?
  - Turns out any knowledge base can be expressed as a conjunction of disjunctions (conjunctive normal form, CNF).
  - E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

Inference procedures based on resolution

- Use the principle of proof by contradiction:
- To show that $KB \vDash \alpha$, we show that $(KB \ \wedge \neg\alpha)$ is unsatisfiable

The process:

1. Convert $KB \wedge \neg\alpha$ to CNF
2. Resolution rule is applied to the resulting clauses
3. Process continues until one of two things happens:
   a) There are no new clauses that can be added, in which case $KB \nvDash \alpha$
   b) Two clauses resolve to yield the *empty* clause, in which case $KB \vDash \alpha$

KB

$R_1: B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
$R_2: \neg B_{1,1}$

Example

Want to prove α: $\neg P_{1,2}$

1. $\left(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})\right) \wedge \left((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}\right)$
2. $\left(\neg B_{1,1} \vee (P_{1,2} \vee P_{2,1})\right) \wedge \left(\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1}\right)$
3. $\left(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}\right) \wedge \left((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1}\right)$
4. $\left(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}\right) \wedge \left(\neg P_{1,2} \vee B_{1,1}\right) \wedge \left(\neg P_{2,1} \vee B_{1,1}\right)$

KB $R_1: B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$
$R_2: \neg B_{1,1}$

Want to prove α: $\neg P_{1,2}$

Example

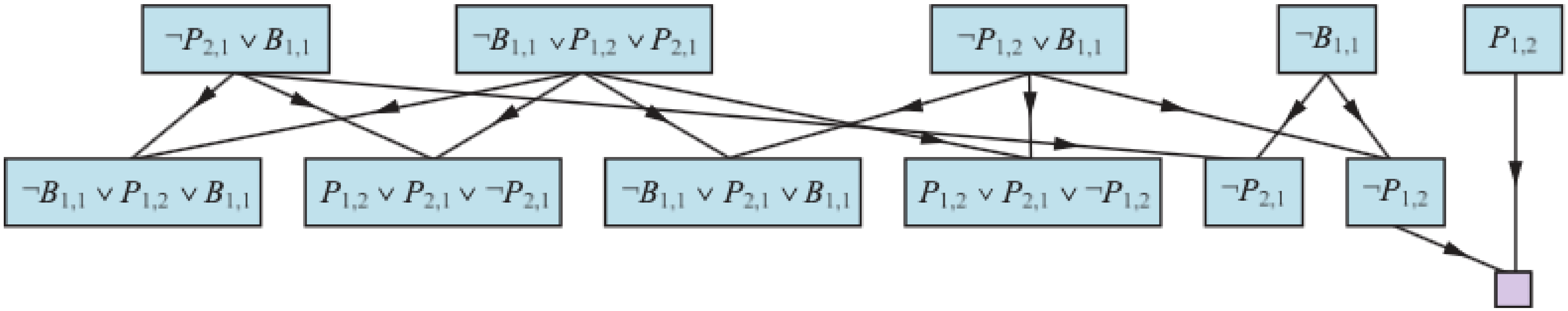


Figure 7.14 Partial application of PL-RESOLUTION to a simple inference in the wumpus world to prove the query $\neg P_{1,2}$. Each of the leftmost four clauses in the top row is paired with each of the other three, and the resolution rule is applied to yield the clauses on the bottom row. We see that the third and fourth clauses on the top row combine to yield the clause $\neg P_{1,2}$, which is then resolved with $P_{1,2}$ to yield the empty clause, meaning that the query is proven.

Resolution: Inference procedure

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

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

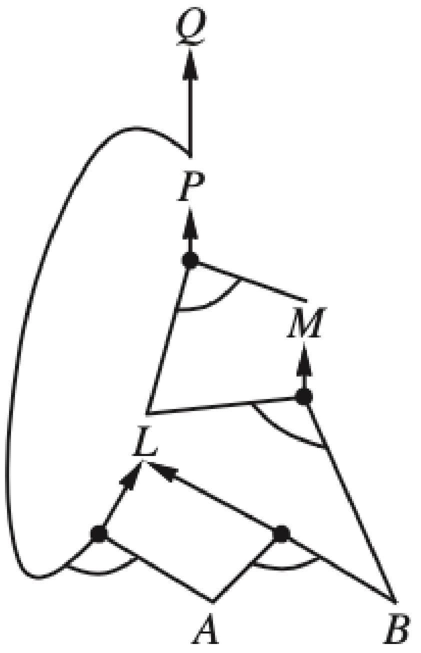
Inference for Horn clauses: Forward chaining

- **Idea**: fire any rule whose premises are satisfied in the KB,
  - add its conclusion to the KB, until query is found

- Forward chaining is sound and complete for Horn KB

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

(a)

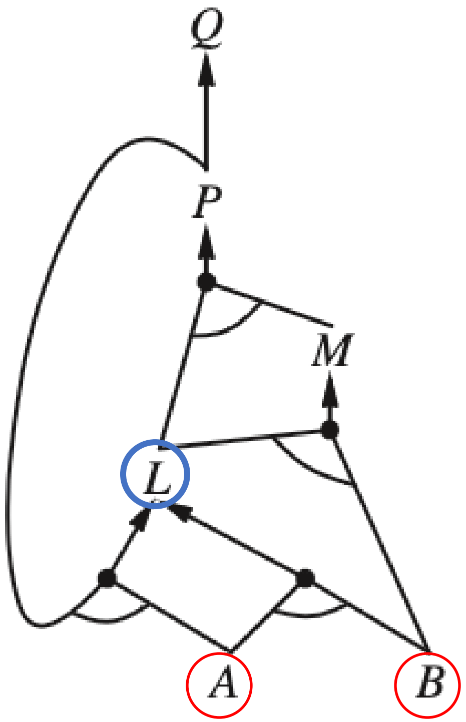


(b)

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

Forward chaining example: Prove Q

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$\boxed{A \wedge B \Rightarrow L}$$
$$A$$
$$B$$



Forward chaining example: Prove Q

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

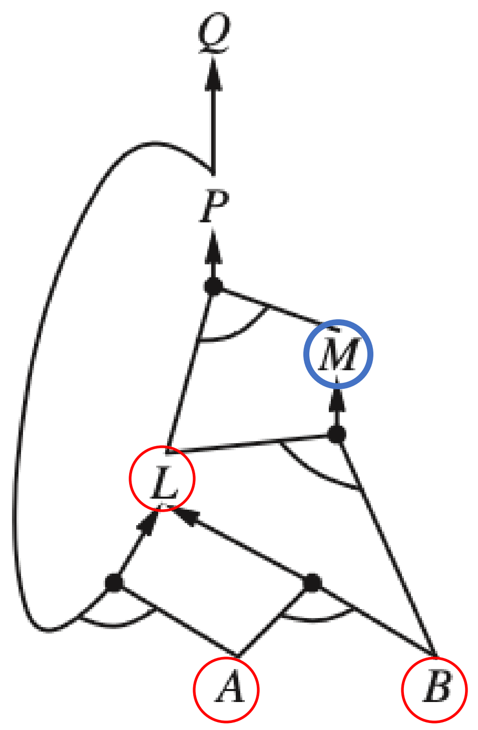
$\boxed{B \wedge L \Rightarrow M}$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

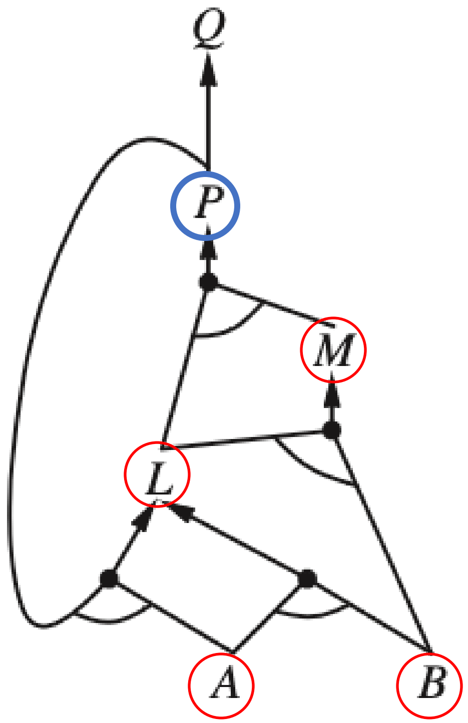
B



Students

Forward chaining example: Prove Q

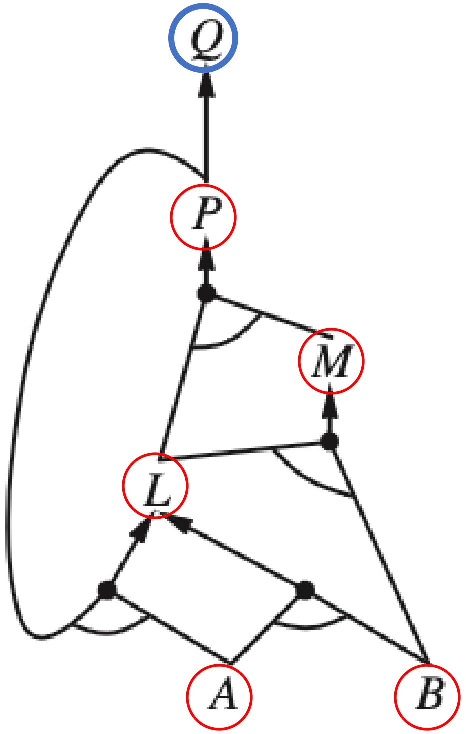
$$P \Rightarrow Q$$
$$\boxed{L \wedge M \Rightarrow P}$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$



Students

Forward chaining example: Prove Q

$$\boxed{P \Rightarrow Q}$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$



Inference for Horn clauses: Backward Chaining

- **Idea**: work backwards from the query Q
- 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

Students

Backward Chaining example: Prove Q

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

Facts	Goals	Clauses
A, B	Q	

Students

Backward Chaining example: Prove Q

$P \Rightarrow Q$
$L \wedge M \Rightarrow P$
$B \wedge L \Rightarrow M$
$A \wedge P \Rightarrow L$
$A \wedge B \Rightarrow L$
A
B

Facts	Goals	Clauses
A, B	Q	$P \Rightarrow Q$
A, B	P, Q	

Students

Backward Chaining example: Prove Q

$P \Rightarrow Q$
$L \wedge M \Rightarrow P$
$B \wedge L \Rightarrow M$
$A \wedge P \Rightarrow L$
$A \wedge B \Rightarrow L$
A
B

Facts	Goals	Clauses
A, B	Q	$P \Rightarrow Q$
A, B	P, Q	$L \wedge M \Rightarrow P$
A, B	L, M, P, Q	

Students

Backward Chaining example: Prove Q

$P \Rightarrow Q$
$L \wedge M \Rightarrow P$
$B \wedge L \Rightarrow M$
$A \wedge P \Rightarrow L$
$A \wedge B \Rightarrow L$
A
B

Facts	Goals	Clauses
A, B	Q	$P \Rightarrow Q$
A, B	P, Q	$L \wedge M \Rightarrow P$
A, B	L, M, P, Q	$A \wedge B \Rightarrow L$
A, B, L	M, P, Q	

Students

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

Backward Chaining example: Prove Q

Facts	Goals	Clauses
A, B	Q	$P \Rightarrow Q$
A, B	P, Q	$L \wedge M \Rightarrow P$
A, B	L, M, P, Q	$A \wedge B \Rightarrow L$
A, B, L	M, P, Q	$B \wedge L \Rightarrow M$
A, B, L, M	P, Q	

Students

Backward Chaining example: Prove Q

$P \Rightarrow Q$
$L \wedge M \Rightarrow P$
$B \wedge L \Rightarrow M$
$A \wedge P \Rightarrow L$
$A \wedge B \Rightarrow L$
A
B

Facts	Goals	Clauses
A, B	Q	$P \Rightarrow Q$
A, B	P, Q	$L \wedge M \Rightarrow P$
A, B	L, M, P, Q	$A \wedge B \Rightarrow L$
A, B, L	M, P, Q	$B \wedge L \Rightarrow M$
A, B, L, M	P, Q	$L \wedge M \Rightarrow P$
A, B, L, M, P	Q	

Students

Backward Chaining example: Prove Q

$P \Rightarrow Q$
$L \wedge M \Rightarrow P$
$B \wedge L \Rightarrow M$
$A \wedge P \Rightarrow L$
$A \wedge B \Rightarrow L$
A
B

Facts	Goals	Clauses
A, B	Q	$P \Rightarrow Q$
A, B	P, Q	$L \wedge M \Rightarrow P$
A, B	L, M, P, Q	$A \wedge B \Rightarrow L$
A, B, L	M, P, Q	$B \wedge L \Rightarrow M$
A, B, L, M	P, Q	$L \wedge M \Rightarrow P$
A, B, L, M, P	Q	$P \Rightarrow Q$
A, B, L, M, P, Q	-	-

Forward vs. backward chaining

- FC is **data-driven**, automatic, unconscious processing,
  - e.g., object recognition, routine decisions
- May do lots of work that is irrelevant to the goal
- BC is **goal-driven**, appropriate for problem-solving,
  - e.g., Where are my keys? How do I get into a PhD program?
- Complexity of BC can be **much less** than linear in size of KB