Logic Agents and Propositional Logic

Knowledge-Based Agents

- Humans, it seems, know things; and what they know helps them do things.

- Above statements make strong claim about how the intelligence of humans is achieved - not by purely reflex mechanisms but by processes of **reasoning** that operate on internal **representations** of knowledge.

- In AI, this approach to intelligence is embodied in **knowledge-based agents.**

Knowledge-Based Agents

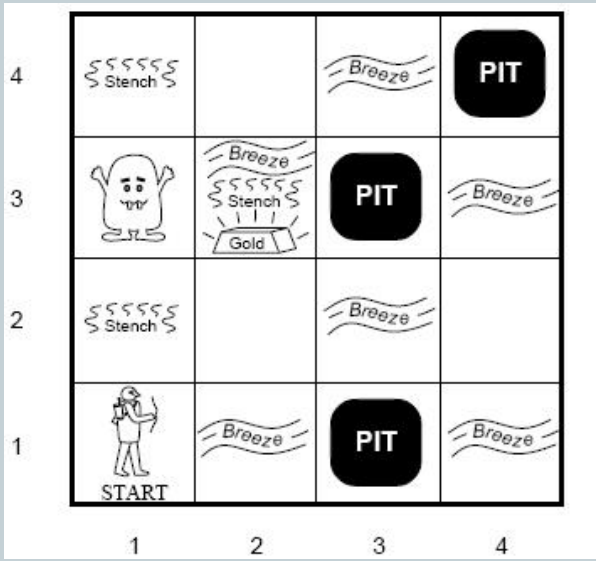
- The central component of a knowledge-based agent is its knowledge base, or KB.
- A knowledge base is a set of sentences. Each sentence is expressed in a language called a knowledge representation language and represents some assertion about the world.
- There must be a way to add new sentences to the knowledge base and a way to query what is known. The standard names for these operations are **TELL** and **ASK**, respectively.
- Both operations may involve **inference -** that is, deriving new sentences from old. Inference must obey teh requirement that when one ASK a question of the knowledge base, the answer should follow from what has been told (or TELLED) to the knowledge base previously.

Knowledge-Based Agents

- KB = knowledge base
  - A set of sentences or facts
  - e.g., a set of statements in a logic language
- Inference
  - Deriving new sentences from old
  - e.g., using a set of logical statements to infer new ones
- A simple model for reasoning
  - Agent is told or perceives new evidence
    - E.g., A is true
  - Agent then infers new facts to add to the KB
    - E.g., KB = { A -> (B OR C) }, then given A and not C we can infer that B is true
    - B is now added to the KB even though it was not explicitly asserted, i.e., the agent inferred B

Wumpus World

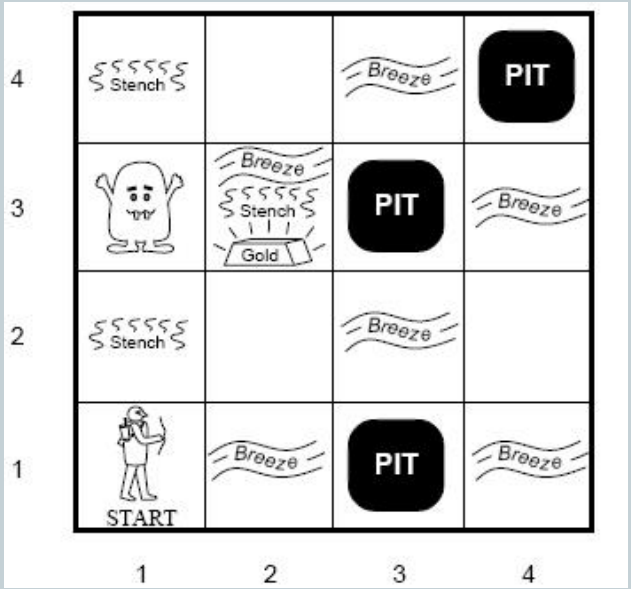
- Environment
  - Cave of 4×4
  - Agent enters in [1,1]
  - 16 rooms
    - Wumpus: A deadly beast who kills anyone entering his room.
    - Pits: Bottomless pits that will trap you forever.
    - Gold



Wumpus World

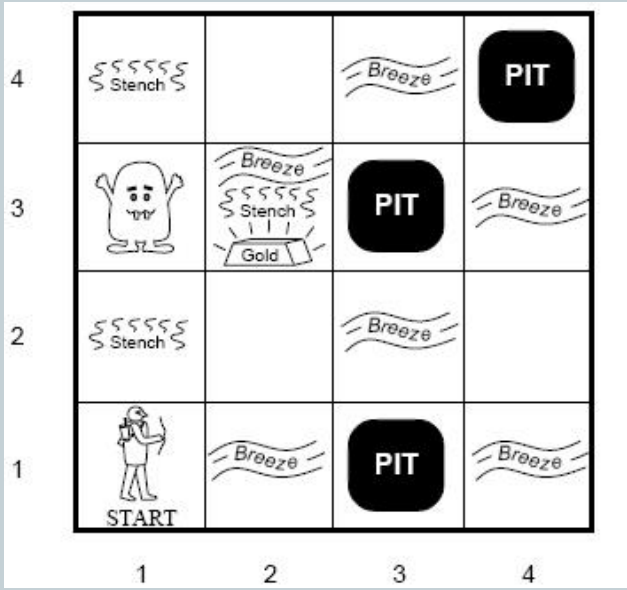
- Agents Sensors:
  - Stench in room adjacent to Wumpus
  - Breeze in room adjacent to pit
  - Glitter in square with gold
  - Bump when agent moves into a wall
  - Scream from wumpus when killed, can be heard anywhere in the cave.

- Agents actions
  - Agent can move forward, turn left or turn right
  - Shoot, one shot
  - Grab, to pick the gold if in same room
  - Climb, to get out of the cave from start square.



Wumpus World

- Performance measure
  - +1000 for picking up gold
  - -1000 got falling into pit or being eaten by wumpus.
  - -1 for each move
  - -10 for using arrow
  - The game ends either when the agent dies or when the agent climbs out of the cave.



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

Reasoning in the Wumpus World

- Agent has initial ignorance about the configuration
  - Agent knows his/her initial location
  - Agent knows the rules of the environment
- Goal is to explore environment, make inferences (reasoning) to try to find the gold.
- Random instantiations of this problem are used to test agent reasoning and decision algorithms

(applications? "intelligent agents" in computer games)

Exploring the Wumpus World

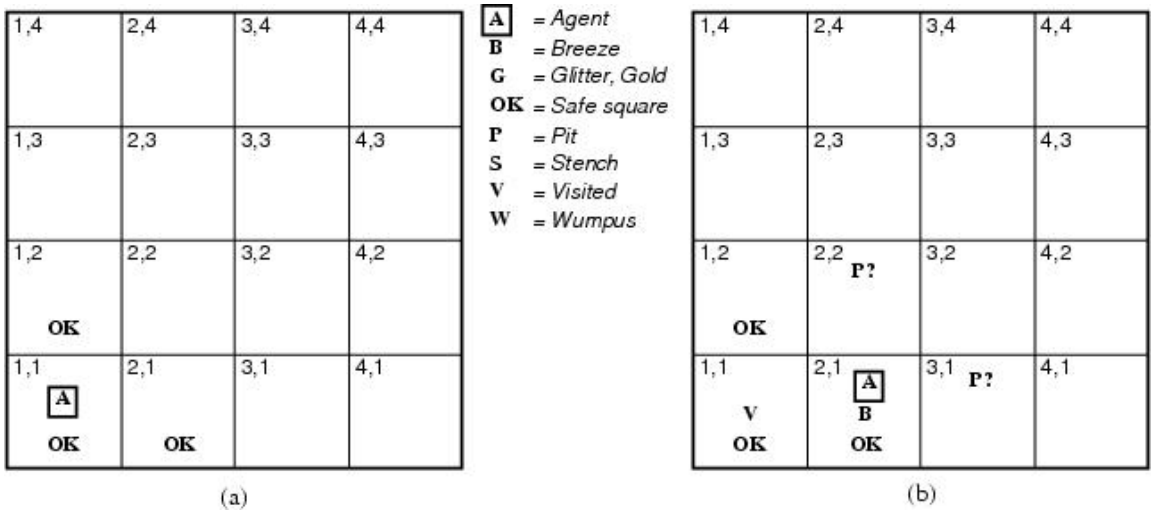
(a) (b)

[1,1] The KB initially contains the rules of the environment.

The first percept is [*none, none,none,none,none*],

move to safe cell e.g. 2,1

Exploring the Wumpus World

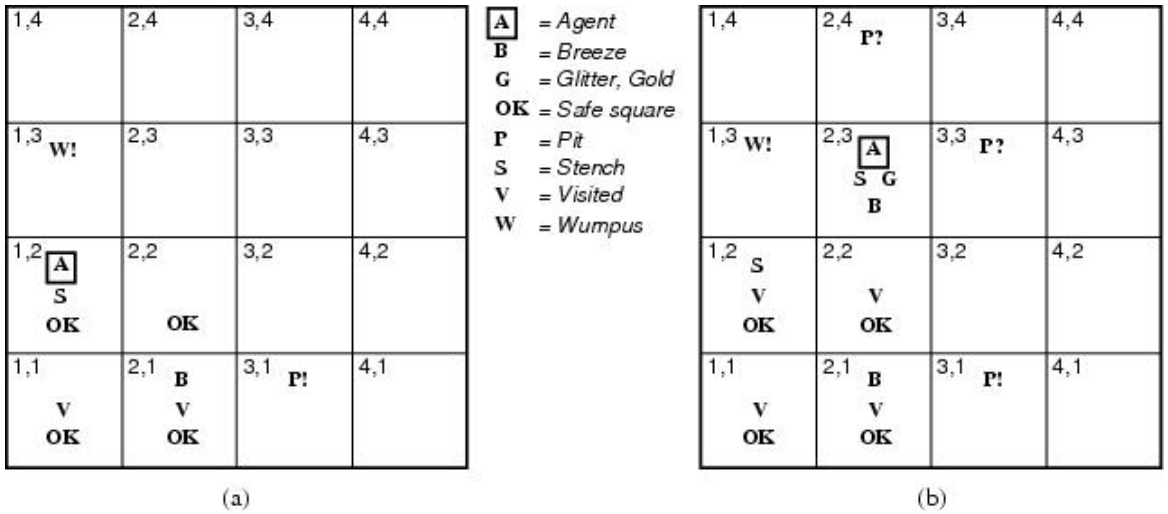


[2,1] = breeze

indicates that there is a pit in [2,2] or [3,1],

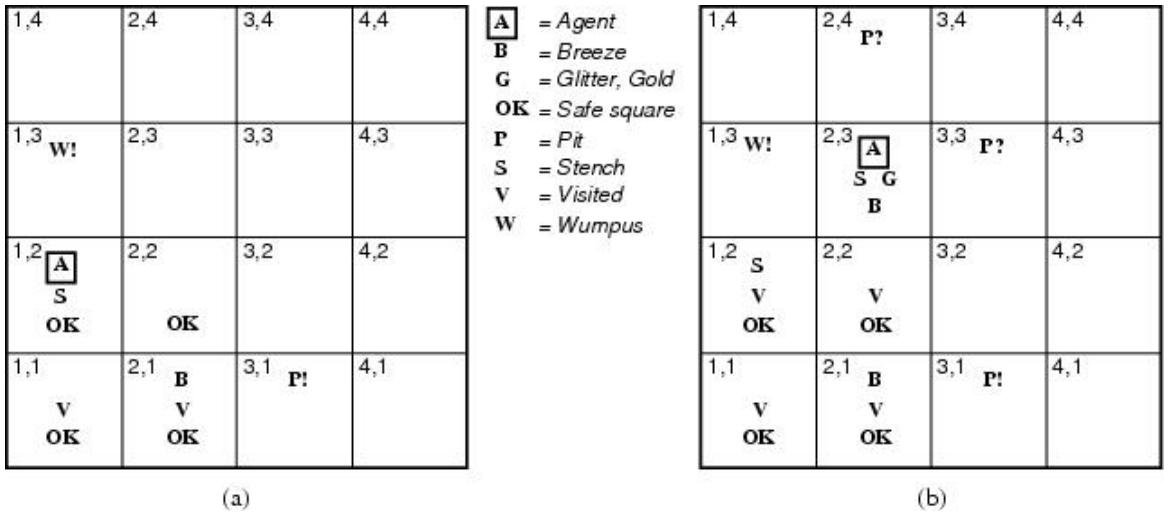
return to [1,1] to try next safe cell

Exploring the Wumpus World



[1,2] Stench in cell which means that wumpus is in [1,3] or [2,2]
YET … not in [1,1]
YET … not in [2,2] or stench would have been detected in [2,1]
(this is relatively sophisticated reasoning!)

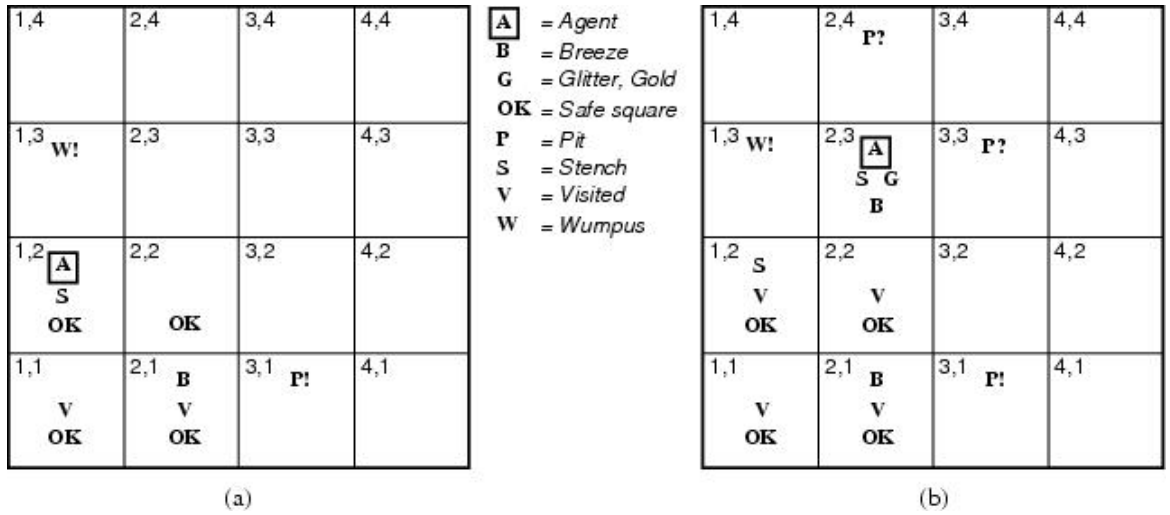
Exploring the Wumpus World



[1,2] Stench in cell which means that wumpus is in [1,3] or [2,2]
YET … not in [1,1]
YET … not in [2,2] or stench would have been detected in [2,1]
(this is relatively sophisticated reasoning!)

THUS … wumpus is in [1,3]
THUS [2,2] is safe because of lack of breeze in [1,2]
THUS pit in [1,3] (again a clever inference)
move to next safe cell [2,2]

Exploring the Wumpus World



[2,2] move to [2,3]

[2,3] detect glitter , smell, breeze
THUS pick up gold
THUS pit in [3,3] or [2,4]

What our example has shown us

- Can represent general knowledge about an environment by a set of rules and facts
- Can gather evidence and then infer new facts by combining evidence with the rules
- The conclusions are guaranteed to be correct if
  - The evidence is correct
  - The rules are correct
  - The inference procedure is correct

    -> logical reasoning
- The inference may be quite complex
  - E.g., evidence at different times, combined with different rules, etc

What is a Logic?

- A formal language
  - KB = set of sentences
- Syntax
  - what sentences are legal (well-formed)
  - E.g., arithmetic
    - X+2 >= y is a well-formed sentence, +x2y is not a well-formed sentence.
- Semantics
  - loose meaning: the interpretation of each sentence
  - More precisely:
    - Defines the truth of each sentence wrt to each possible world
  - e.g,
    - X+2 = y is true in a world where x=7 and y =9
    - X+2 = y is false in a world where x=7 and y =1
  - Note: standard logic – each sentence is T of F wrt each world

Models and possible worlds

- Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated.
- Models are mathematical abstractions, each of which simply fixes the truth or falsehood of every relevant sentence.
- Informally, we may think of a possible world as, for example, having x men and y women sitting at a table playing bridge, and the sentence x + y = 4 is true when there are four people in total.

- *m* is a model of a sentence α if α is true in *m*

- *M(α)* is the set of all models of α

- Possible worlds ~ models
  - Possible worlds: potentially real environments
  - Models: mathematical abstractions that establish the truth or falsity of every sentence

- Example:
  - x + y = 4, where x = #men, y = #women
  - Possible models = all possible assignments of integers to x and y

Entailment

- One sentence follows logically from another

  $\alpha \models \beta$

  α entails sentence β *if and only if* β is true in all worlds where α is true i.e.

  $$\alpha \models \beta \text{ if and only if } M(\alpha) \subseteq M(\beta) .$$

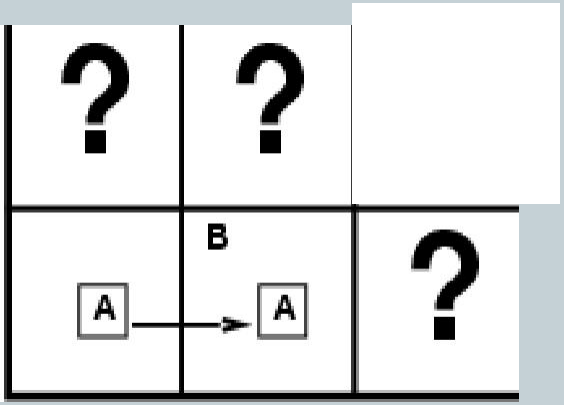
  e.g., x+y=4 |= 4=x+y

- Entailment is a relationship between sentences that is based on semantics.

Entailment in the wumpus world

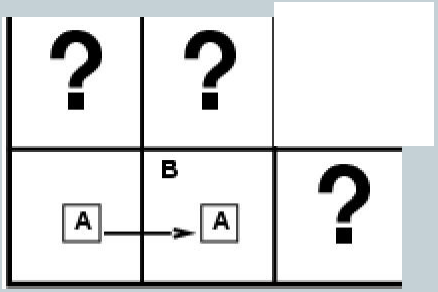
- Consider possible models for *KB* assuming only pits and a reduced Wumpus world

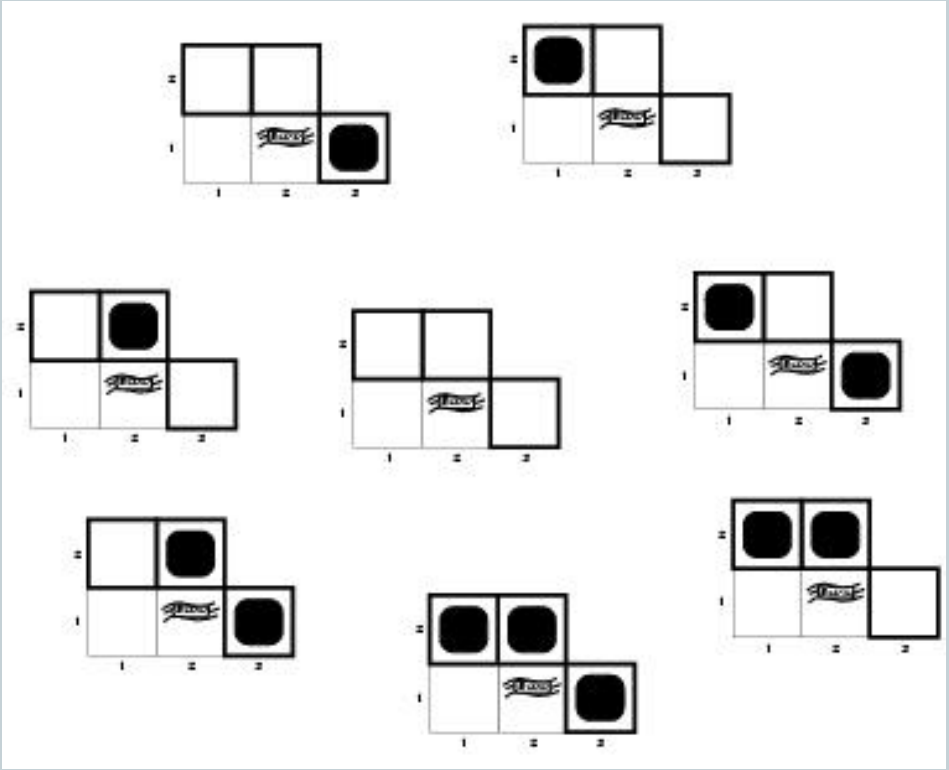
- Situation after detecting nothing in [1,1], moving right, detecting breeze in [2,1]



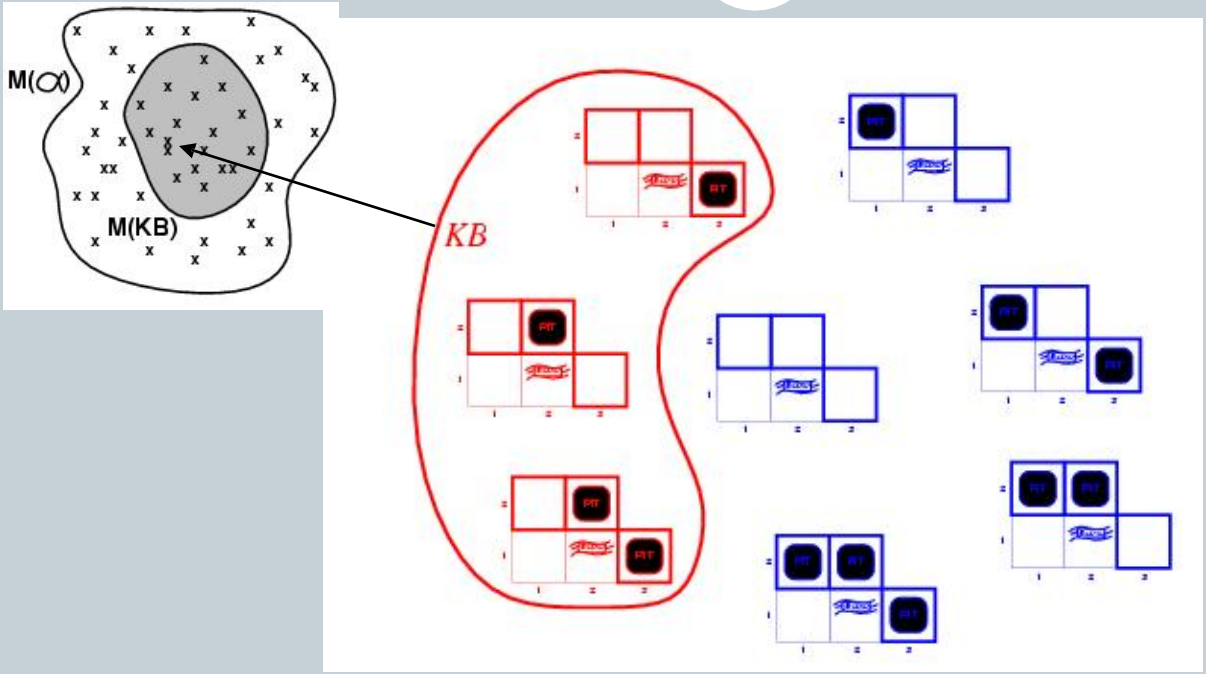
Wumpus models

All possible models in this reduced Wumpus world.





Wumpus models



- *KB* = all possible wumpus-worlds consistent with the observations and the "physics" of the Wumpus world.

Inferring conclusions

- Consider 2 possible conclusions given a KB
  - α_1 = "[1,2] is safe"
  - α_2 = "[2,2] is safe“
- One possible inference procedure
  - Start with KB
  - Model-checking
    - Check if KB $\models \alpha$ by checking if in all possible models where KB is true that α is also true
- Comments:
  - Model-checking enumerates all possible worlds
    - Only works on finite domains, will suffer from exponential growth of possible models

Wumpus World Models

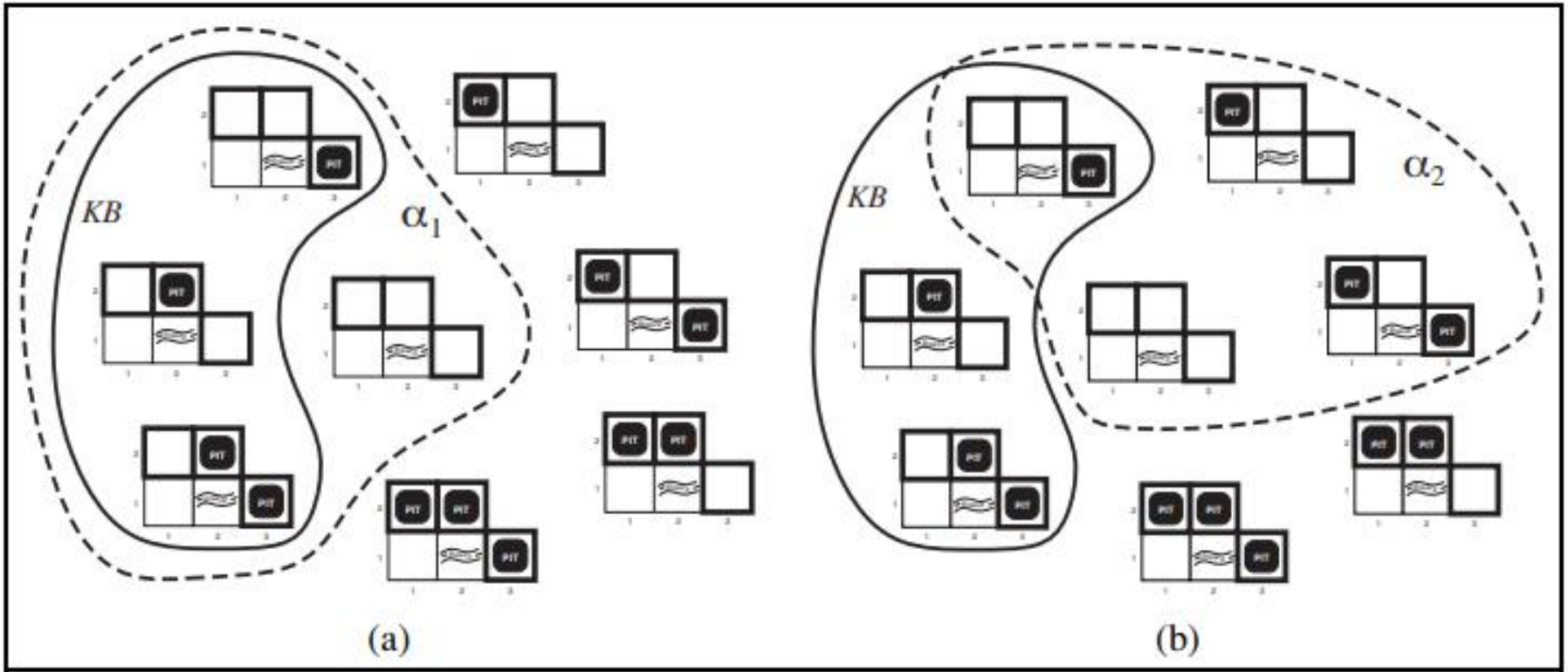
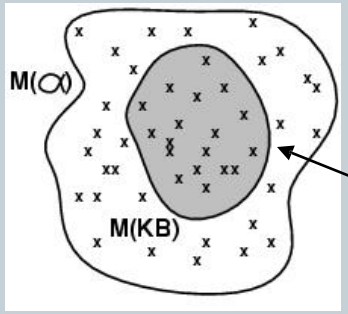
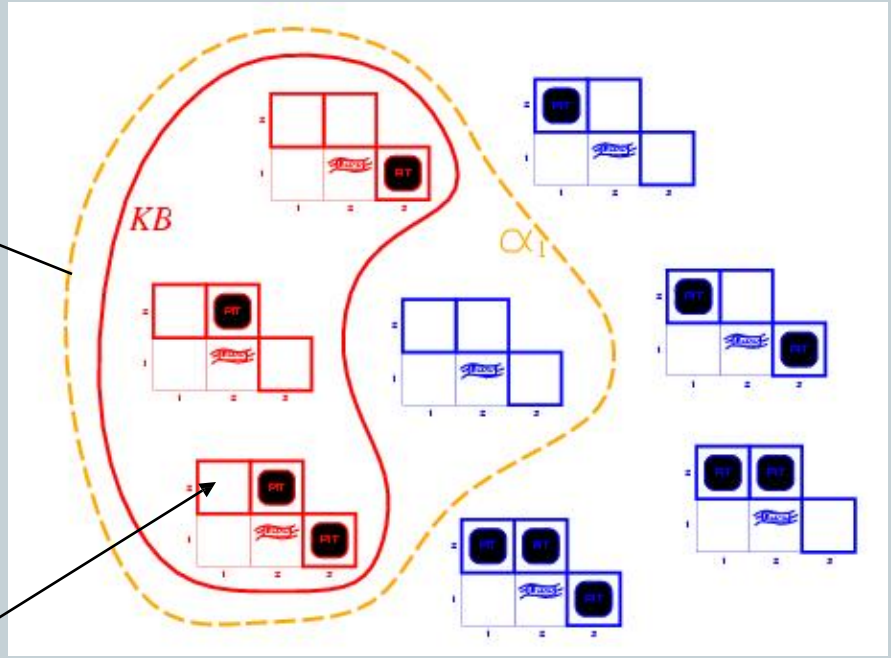


Figure 7.5 Possible models for the presence of pits in squares [1,2], [2,2], and [3,1]. The KB corresponding to the observations of nothing in [1,1] and a breeze in [2,1] is shown by the solid line. (a) Dotted line shows models of α_1 (no pit in [1,2]). (b) Dotted line shows models of α_2 (no pit in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

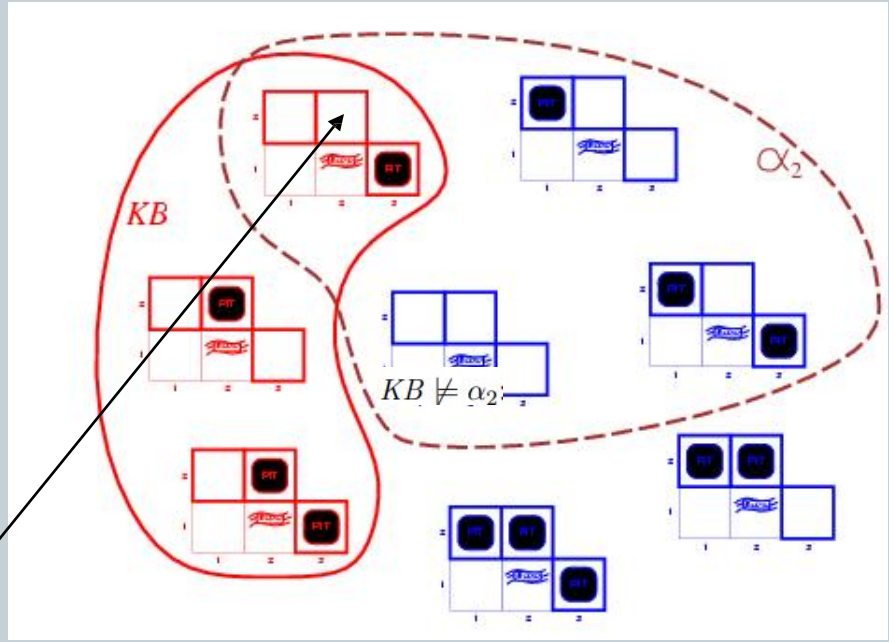
Wumpus models





α_1 = "[1,2] is safe", $KB \models \alpha_1$, proved by model checking

Wumpus models



α_2 = "[2,2] is safe", $KB \not\models \alpha_2$

There are some models entailed by KB where α_2 is false. That means the agent cannot conclude that there is no pit in [2,2]. (Nor can it conclude

that there is a pit in [2,2].)

Logical inference

- The notion of entailment can be used for logical inference.
  - Model checking (see wumpus example): enumerate all possible models and check whether α is true.
- In understanding entailment and inference, it might help to think of the set of all consequences of KB as a haystack and of as a needle. Entailment is like the needle being in the haystack; inference is like finding it.
- If an inference algorithm *i* can derive from KB, we write KB $|--_i$ which is pronounced "α is derived from KB by i" or "i derives α from KB."
- If an inference algorithm only derives entailed sentences it is called *sound* or *truth preserving*.
  - Otherwise it just makes things up.

  i is sound if whenever KB $|-_i \alpha$ *it is also true that KB* $|= \alpha$
  - *E.g., model-checking is sound*
- Completeness : If an inference algorithm can derive any sentence that is entailed.

  i is complete if whenever KB $|= \alpha$ *it is also true that KB* $|-_i \alpha$

Schematic perspective

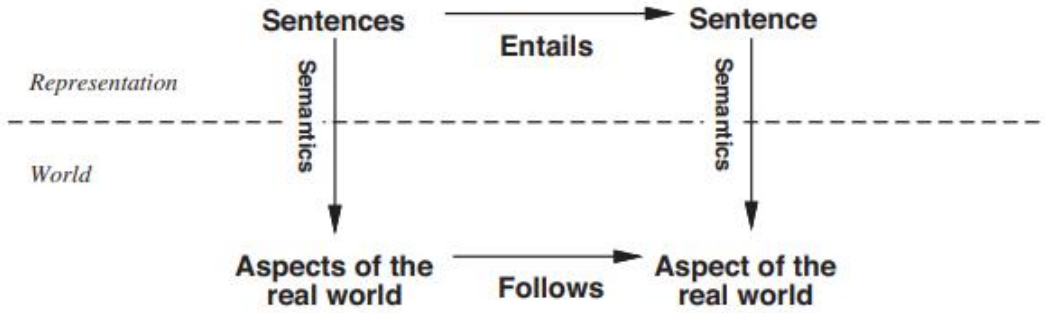


Figure 7.6 Sentences are physical configurations of the agent, and reasoning is a process of constructing new physical configurations from old ones. Logical reasoning should ensure that the new configurations represent aspects of the world that actually follow from the aspects that the old configurations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

If KB is true in the real world, then any sentence α *derived from KB by a sound inference procedure is also true in the real world.*

Propositional logic: Syntax

- Propositional logic is the simplest logic – illustrates basic ideas
- Atomic sentences = single proposition symbols that can be true or false.
  - E.g., P, Q, R
  - Special cases: True = always true, False = always false
- Complex sentences: Constructed from simple 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)
  - If S_1 and S_2 are sentences, $S_1 \vee S_2$ is a sentence (disjunction)
  - If S_1 and S_2 are sentences, $S_1 \Rightarrow S_2$ is a sentence (implication)
  - If S_1 and S_2 are sentences, $S_1 \Leftrightarrow S_2$ is a sentence (biconditional)

Propositional logic: Semantics

Each model/world specifies true or false for each proposition symbol

E.g. $P_{1,2}$ false $P_{2,2}$ true $P_{3,1}$ false

With these symbols, 8 possible models, can be enumerated automatically.

Rules for evaluating truth with respect to a model *m*:

$\neg S$ is true iff S is false

$S_1 \wedge S_2$ is true iff S_1 is true and S_2 is true

$S_1 \vee S_2$ is true iff $S_1$is true or S_2 is true

$S_1 \Rightarrow S_2$ is true iff S_1 is false or S_2 is true
i.e., is false iff S_1 is true and S_2 is false

$S_1 \Leftrightarrow S_2$ is true iff $S_1 \Rightarrow S_2$ is true and $S_2 \Rightarrow S_1$ is true

Simple recursive process evaluates an arbitrary sentence, e.g.,

$$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = \mathit{true} \wedge (\mathit{true} \vee \mathit{false}) = \mathit{true} \wedge \mathit{true} = \mathit{true}$$

Truth tables for connectives

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

Truth tables for connectives

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

Implication is always true when the premise is false

Why? P=>Q means "if P is true then I am claiming that Q is true otherwise no claim"

Only way for this to be false is if P is true and Q is false

Wumpus World Knowledge Base

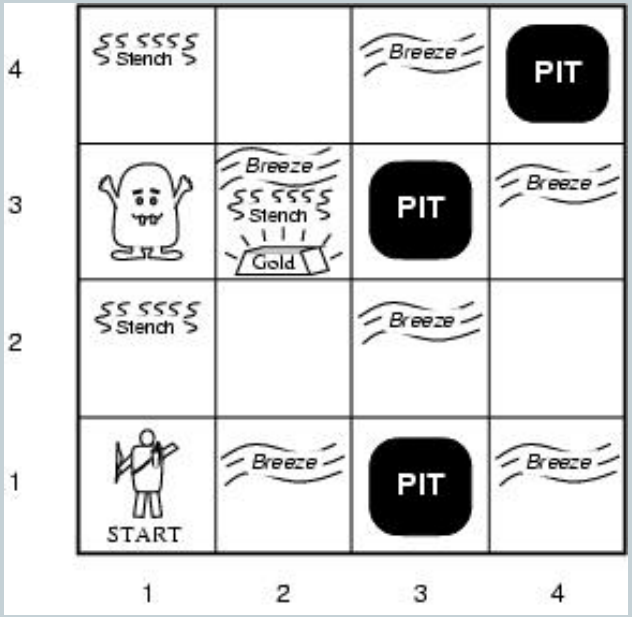
- Let's define following symbols for each [x, y] location -
  - $P_{x,y}$ is true if there is a pit in [x, y]
  - $W_{x,y}$ is true if there is a wumpus in [x, y], dead or alive.
  - $B_{x,y}$ is true if the agent perceives a breeze in [x, y].
  - $S_{x,y}$ is true if the agent perceives a stench in [x, y].
- So we can write following sentences -
  - There is no pit in [1,1]:
    - $R_1 : \neg P_{1,1}$
  - A square is breezy if and only if there is a pit in a neighboring square. This has to be stated for each square; for now, we include just the relevant squares:
    - $R2 : B1,1 \Leftrightarrow (P1,2 \vee P2,1)$.
    - $R3 : B2,1 \Leftrightarrow (P1,1 \vee P2,2 \vee P3,1)$
  - 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:
    - $R4 : \neg B1,1$
    - $R5 : B2,1$

Wumpus world sentences

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].

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

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



- "Pits cause breezes in adjacent squares"

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

- KB can be expressed as the conjunction of all of these sentences

- Note that these sentences are rather long-winded!
  - E.g., breese "rule" must be stated explicitly for each square
  - First-order logic will allow us to define more general relations (later)

A simple inference procedure

- Our goal now is to decide whether KB |= α for some sentence α.
- For example, is ¬P1,2 entailed by our KB?
- Our first algorithm for inference is a model-checking approach that is a direct implementation of the definition of entailment: *enumerate the models, and check that α is true in every model in which KB is true.*
- **Models are assignments of true or false to every proposition symbol.**

Inference by enumeration

- Returning to our wumpus-world example, the relevant proposition symbols are B1,1, B2,1, P1,1, P1,2, P2,1, P2,2, and P3,1.
- With seven symbols, there are $2^7 = 128$ possible models and in three of these, KB is true.
- In those three models, ¬P1,2 is true, hence there is no pit in [1,2].
- On the other hand, P2,2 is true in two of the three models and false in one, so we cannot yet tell whether there is a pit in [2,2].

Truth tables for the Wumpus KB

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

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].

Inference by Enumeration

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

  $symbols \leftarrow$ a list of the proposition symbols in KB and α
  return TT-CHECK-ALL($KB, \alpha, symbols, \{\}$)

---

function TT-CHECK-ALL($KB, \alpha, symbols, model$) **returns** *true* or *false*
  if EMPTY?($symbols$) **then**
    if PL-TRUE?($KB, model$) **then return** PL-TRUE?($\alpha, model$)
    else return *true* // *when KB is false, always return true*
  else do
    $P \leftarrow$ FIRST($symbols$)
    $rest \leftarrow$ REST($symbols$)
    return (TT-CHECK-ALL($KB, \alpha, rest, model \cup \{P = true\}$)
      and
      TT-CHECK-ALL($KB, \alpha, rest, model \cup \{P = false\}$))

Figure 7.10 A truth-table enumeration algorithm for deciding propositional entailment. (TT stands for truth table.) PL-TRUE? returns *true* if a sentence holds within a model. The variable *model* represents a partial model—an assignment to some of the symbols. The keyword "**and**" is used here as a logical operation on its two arguments, returning *true* or *false*.

Inference by enumeration

- We want to see if α is entailed by KB
- Enumeration of all models is **sound** (because it implements directly the definition of entailment, and **complete** because it works for any KB and always terminates as there are only finitely many models to examine.
- If KB and α contain n symbols in all, then there are 2^n models. Thus, the time complexity of the algorithm is $O(2^n)$. (The space complexity is only O(n) because the enumeration is depth-first.)
- We need a more efficient way to do inference
  - But worst-case complexity will remain exponential for propositional logic.

Propositional Theorem Proving

- So far, we have studied how to determine entailment by **model checking:** *enumerating models and showing that the sentence must hold in all models.*

- Now we will talk about how entailment can be done by **theorem proving** - *applying rules of inference directly to the sentences in our knowledge base to construct a proof of the desired sentence without consulting models.*

- If the number of models is large but the length of the proof is short, then theorem proving can be more efficient than model checking.

Logical Equivalence

- Two sentences α and β are **logically equivalent** if they are true in the same set of models. We write this as α ≡ β.

- An alternative definition of equivalence is as follows: *any two sentences α and β are equivalent only if each of them entails the other*:

  α ≡ β if and only if α |= β and β |= α

Logical equivalence

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

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

Validity

- **Validity:** A sentence is valid if it is true in all models.
- For example, the sentence $P \vee \neg P$ is valid.
- Valid sentences are also known as tautologies—they are necessarily true.
- Because the sentence **True** is true in all models, every valid sentence is logically equivalent to **True**.
- What good are valid sentences?
- From our definition of entailment, we can derive the deduction theorem, which was known to the ancient Greeks:

For any sentences α and β, α \|= β if and only if the sentence (α ⇒ β) is valid.

Satisfiability

- A sentence is satisfiable if it is true in, or satisfied by, some model i.e. there should be at least one model in which sentence is true otherwise it's unsatisfiable.
- Satisfiability can be checked by enumerating the possible models until one is found that satisfies the sentence.
- **α is valid iff ¬α is unsatisfiable; contrapositively, α is satisfiable iff ¬α is not valid.**
- **α |= β if and only if the sentence (α ∧ ¬β) is unsatisfiable.**
- Proving β from α by checking the unsatisfiability of (α ∧ ¬β) corresponds exactly to the standard mathematical proof technique called **proof by refutation** or **proof by contradiction**.
- One assumes a sentence is β to be false and shows that this leads to a contradiction with known axioms α. This contradiction is exactly what is meant by saying that the sentence (α ∧ ¬β) is unsatisfiable.

Inference Rules

- Modus Ponens

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

- And-Elimination

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

- Bi-conditional Elimination

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

Inference Rules

- Let us see how these inference rules and equivalences can be used in the wumpus world.
- We start with the knowledge base containing R1 through R5 and show how to prove ¬P1,2, that is, there is no pit in [1,2].
- First, we apply biconditional elimination to R2 to obtain - R6 : $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
- Then we apply And-Elimination to R6 to obtain - R7 : $((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
- Logical equivalence of contrapositives gives - R8 : $(\neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1}))$
- Now we can apply Modus Ponens with R8 and the percept R4 (i.e., $\neg B_{1,1}$), to obtain - R9 : $\neg(P_{1,2} \vee P_{2,1})$
- Finally, we apply De Morgan's rule, giving the conclusion - R10 : $\neg P_{1,2} \wedge \neg P_{2,1}$
- That is, neither [1,2] nor [2,1] contains a pit.

Proof by Resolution

- We begin by using a simple version of the resolution rule in the wumpus world. Let us consider the steps leading up to Figure 7.4(a): the agent returns from [2,1] to [1,1] and then goes to [1,2], where it perceives a stench, but no breeze. We add the following facts to the knowledge base:
  - $R_{11} : \neg B_{1,2}$.
  - $R_{12} : B_{1,2} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{1,3})$.
- By the same process that led to R10 earlier, we can now derive the absence of pits in [2,2] and [1,3] (remember that [1,1] is already known to be pitless):
  - $R_{13} : \neg P_{2,2}$.
  - $R_{14} : \neg P_{1,3}$.
- We can also apply biconditional elimination to R3, followed by Modus Ponens with R5, to obtain the fact that there is a pit in [1,1], [2,2], or [3,1]:
  - $R_{15} : P_{1,1} \vee P_{2,2} \vee P_{3,1}$
- Now comes the first application of the resolution rule: the literal $\neg P_{2,2}$ in R13 resolves with the literal $P_{2,2}$ in R15 to give the resolvent
  - $R_{16} : P_{1,1} \vee P_{3,1}$
- In English; if there's a pit in one of [1,1], [2,2], and [3,1] and it's not in [2,2], then it's in [1,1] or [3,1]. Similarly, the literal $\neg P_{1,1}$ in R1 resolves with the literal $P_{1,1}$ in R16 to give
  - $R_{17} : P_{3,1}$.

Proof by Resolution

the **unit resolution rule** takes a clause—a disjunction of literals—and a literal and produces a new clause.

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

where each l is a literal and l_i and $\boldsymbol{m}$ are complementary literals (i.e., one is the negation of the other).

The unit resolution rule can be generalized to the full resolution rule,

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

where $\boldsymbol{l_i}$ and $\boldsymbol{m_j}$ are complementary literals. This says that resolution takes two clauses and produces a new clause containing all the literals of the two original clauses except the two complementary literals.

Conjunctive Normal Form

- The resolution rule applies only to clauses (that is, disjunctions of literals), so it would seem to be relevant only to knowledge bases and queries consisting of clauses.
- *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

Conjuctive Normal Form

- Let's convert $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$ into CNF. The steps are as follows -
- Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.
  - $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
- Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$:
  - $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$
- CNF requires $\neg$ to appear only in literals, so we "move $\neg$ inwards"
  - $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$
- Above sentence, can be written as -
  - **$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$**

Normal Form

We want to prove: $KB \models \alpha$

equivalent to : $KB \wedge \neg\alpha$ *unsatifiable*

We first rewrite $KB \wedge \neg\alpha$ into conjunctive normal form (CNF).

A "conjunction of disjunctions"

literals

$(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

Clause Clause

- Any KB can be converted into CNF
- k-CNF: exactly k literals per clause

Example: Conversion to CNF

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

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

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

3. Move $\neg$ inwards using de Morgan's rules and double-negation:
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$

4. Apply distributive law ($\wedge$ over $\vee$) and flatten:
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

Resolution Inference Rule for CNF

$(A \vee B \vee C)$

$(\neg A)$

$\therefore (B \vee C)$

"If A or B or C is true, but not A, then B or C must be true."

$(A \vee B \vee C)$

$(\neg A \vee D \vee E)$

$\therefore (B \vee C \vee D \vee E)$

"If A is false then B or C must be true,
or if A is true then D or E must be true,
hence since A is either true or false, B or C or D or E must be true."

$(A \vee B)$

$(\neg A \vee B)$

$\therefore (B \vee B) \equiv B$

Simplification

Resolution Algorithm

- The resolution algorithm tries to prove: $KB \models \alpha$ *equivalent to* $KB \wedge \neg\alpha$ *unsatisfiable*

- Generate all new sentences from KB and the query.
- One of two things can happen:

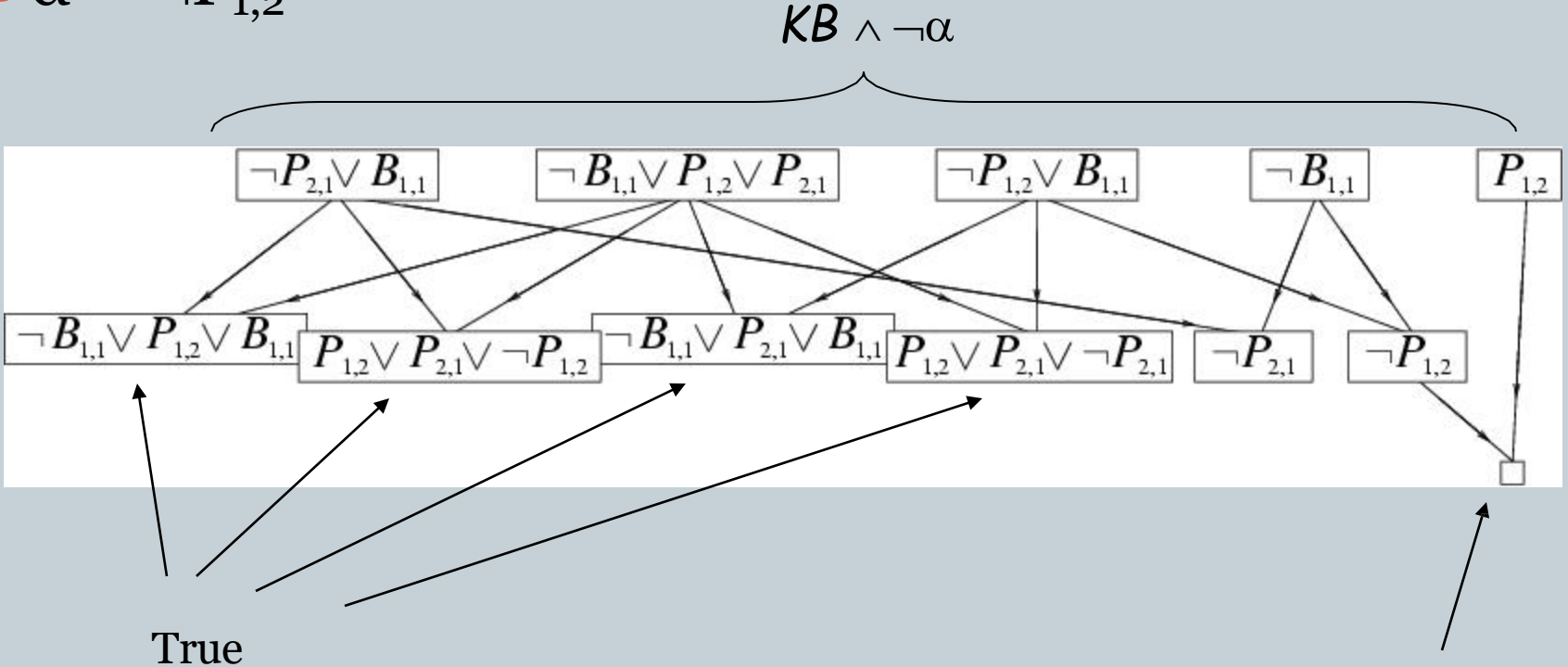
1. We find $P \wedge \neg P$ which is unsatisfiable,
   i.e. we can entail the query.

2. We find no contradiction: there is a model that satisfies the Sentence (non-trivial) and hence we cannot entail the query.

$KB \wedge \neg\alpha$

Resolution example

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



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
- Resolution is complete for propositional logic.
- Forward, backward chaining are linear-time, complete for Horn clauses **(Optional)**
- Propositional logic lacks expressive power.

Knowledge Representation using First-Order Logic

Pros and cons of propositional logic

- ☺ Propositional logic is declarative
  - -programming languages lack general mechanism for deriving facing from other facts
  - Update to data structure is domain specific
  - Knowledge and inference are separate
- ☺ Propositional logic allows partial/disjunctive/negated information
  - unlike most programming languages and databases
- ☺ Propositional logic is compositional:
  - meaning of $B_{1,1} \wedge P_{1,2}$ is derived from meaning of $B_{1,1}$ and of $P_{1,2}$
- ☺ Meaning in propositional logic is context-independent
  - unlike natural language, where meaning depends on context
  - Look, here comes superman.
- ☹ Propositional logic has limited expressive power
  - unlike natural language
  - E.g., cannot say "pits cause breezes in adjacent squares“
    - except by writing one sentence for each square

Wumpus World and propositional logic

- Find Pits in Wumpus world
  - $B_{x,y} \Leftrightarrow (P_{x,y+1} \vee P_{x,y-1} \vee P_{x+1,y} \vee P_{x-1,y})$ (Breeze next to Pit) 16 rules
- Find Wumpus
  - $S_{x,y} \Leftrightarrow (W_{x,y+1} \vee W_{x,y-1} \vee W_{x+1,y} \vee W_{x-1,y})$ (stench next to Wumpus) 16 rules
- At least one Wumpus in world
  - $W_{1,1} \vee W_{1,2} \vee \ldots \vee W_{4,4}$ (at least 1 Wumpus) 1 rule
- At most one Wumpus
  - $\neg W_{1,1} \vee \neg W_{1,2}$ (155 RULES)
- Keep track of location
  - $L_{x,y} \wedge FacingRight \wedge Forward \Rightarrow L_{x+1,y}$

First-Order Logic

- Propositional logic assumes the world contains facts,
- First-order logic (like natural language) assumes the world contains
  - Objects: people, houses, numbers, colors, baseball games, wars, …
  - Relations: red, round, prime, brother of, bigger than, part of, comes between, …
  - Functions: father of, best friend, one more than, plus, …

Syntax of FOL: Basic elements

- **Constant Symbols:**
  - Stand for objects
  - e.g., KingJohn, 2, UCI,…
- **Predicate Symbols**
  - Stand for relations
  - E.g., Brother(Richard, John), greater_than(3,2)…
- **Function Symbols**
  - Stand for functions
  - E.g., Sqrt(3), LeftLegOf(John),…

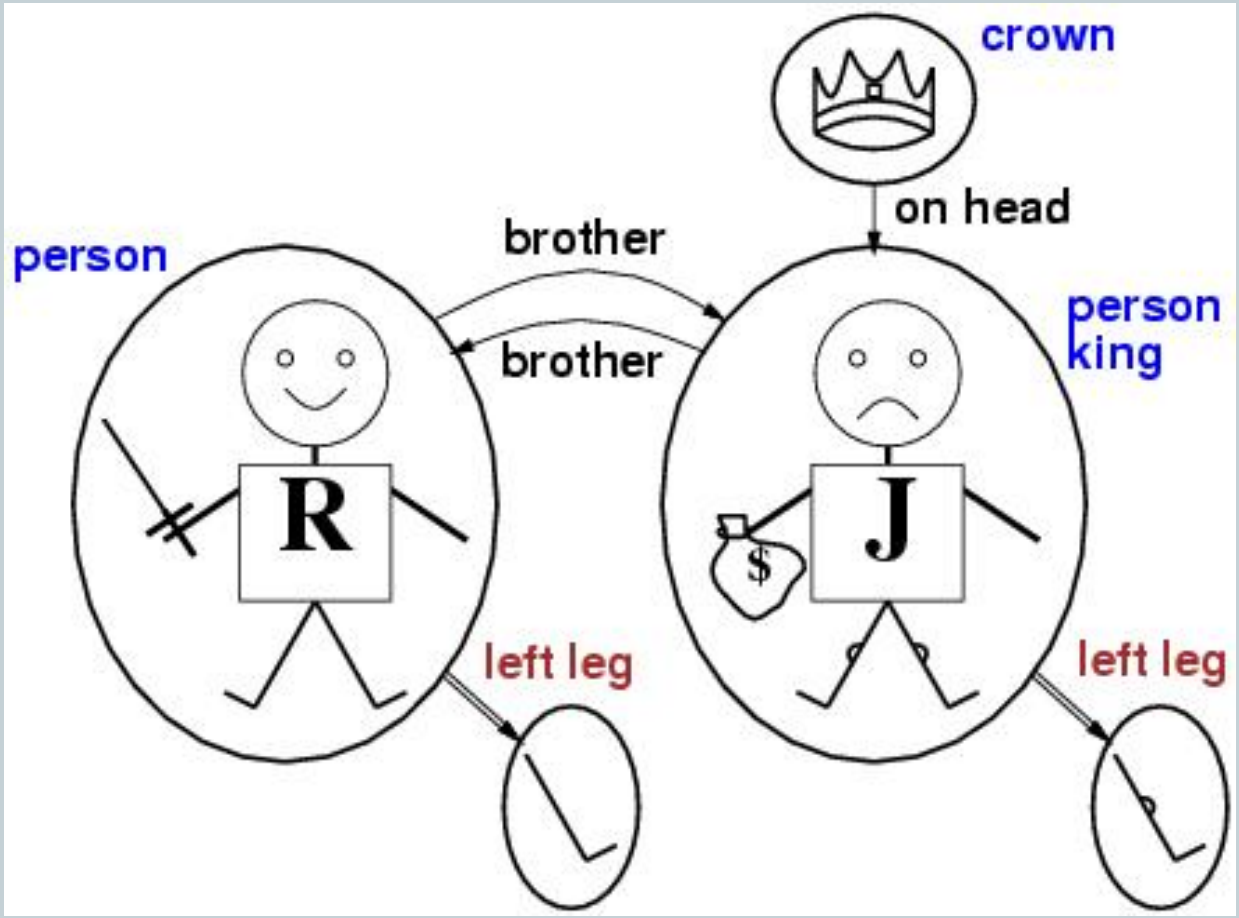
Syntax of FOL: Basic elements

- Constants KingJohn, 2, UCI,...
- Predicates Brother, >,...
- Functions Sqrt, LeftLegOf,...
- Variables x, y, a, b,...
- Connectives $\neg, \Rightarrow, \wedge, \vee, \Leftrightarrow$
- Equality =
- Quantifiers $\forall, \exists$

Relations

- Some relations are properties: they state some fact about a single object: Round(ball), Prime(7).
- n-ary relations state facts about two or more objects: Married(John,Mary), LargerThan(3,2).
- Some relations are functions: their value is another object: Plus(2,3), Father(Dan).

Models for FOL: Example



Terms

- Term = logical expression that refers to an object.

- There are 2 kinds of terms:
  - constant symbols: Table, Computer
  - function symbols: LeftLeg(Pete), Sqrt(3), Plus(2,3) etc

Atomic Sentences

- Atomic sentences state facts using terms and predicate symbols
  - P(x,y) interpreted as "x is P of y"

- Examples:

  LargerThan(2,3) is false.

  Brother_of(Mary,Pete) is false.

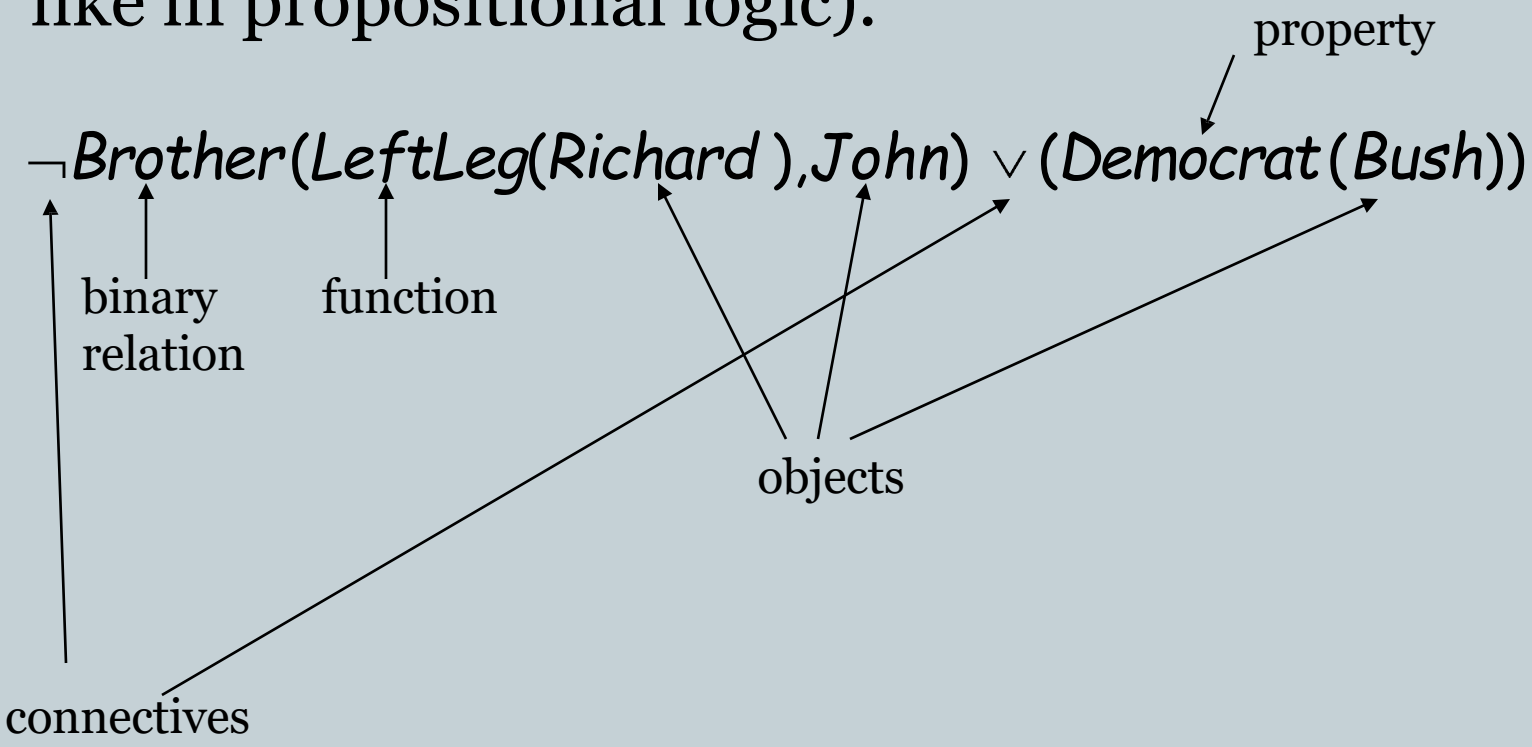
  Married(Father(Richard), Mother(John)) could be true or false

- Note: Functions do not state facts and form no sentence:
  - Brother(Pete) refers to John (his brother) and is neither true nor false.

- Brother_of(Pete,Brother(Pete)) is True.

Complex Sentences

- We make complex sentences with connectives (just like in propositional logic).



More Examples

- Brother(Richard, John) $\wedge$ Brother(John, Richard)
- King(Richard) $\vee$ King(John)
- King(John) => $\neg$ King(Richard)
- LessThan(Plus(1,2) ,4) $\wedge$ GreaterThan(1,2)

(Semantics are the same as in propositional logic)

Variables

- Person(John) is true or false because we give it a single argument “John”

- We can be much more flexible if we allow variables which can take on values in a domain. e.g., all persons x, all integers i, etc.
  - E.g., can state rules like Person(x) => HasHead(x)
    or Integer(i) => Integer(plus(i,1))

Universal Quantification ∀

- ∀ means “for all”
- Allows us to make statements about all objects that have certain properties
- Can now state general rules:

∀ x King(x) => Person(x)

∀ x Person(x) => HasHead(x)

∀ i Integer(i) => Integer(plus(i,1))

Note that
∀ x King(x) ^ Person(x) is not correct!
This would imply that all objects x are Kings and are People

∀ x King(x) => Person(x) is the correct way to say this

Existential Quantification ∃

- ∃ x means "there exists an x such that…." (at least one object x)
- Allows us to make statements about some object without naming it
- Examples:

  ∃ x King(x)

  ∃ x Lives_in(John, Castle(x))

  ∃ i Integer(i) ∧ GreaterThan(i,0)

  Note that ∧ is the natural connective to use with ∃

  (And => is the natural connective to use with ∀)

More examples

For all real x, x>2 implies x>3.

$$\forall x\,[(x > 2) \Rightarrow (x > 3)] \quad x \in R \quad (false)$$

$$\exists x\,[(x^2 = -1)] \quad x \in R \quad (false)$$

There exists some real x whose square is minus 1.

Fun with sentences

Brothers are siblings

Fun with sentences

Brothers are siblings

$\forall\, x, y \;\; Brother(x, y) \;\Rightarrow\; Sibling(x, y).$

"Sibling" is symmetric

Fun with sentences

Brothers are siblings

$$\forall x, y \;\; Brother(x, y) \;\Rightarrow\; Sibling(x, y).$$

"Sibling" is symmetric

$$\forall x, y \;\; Sibling(x, y) \;\Leftrightarrow\; Sibling(y, x).$$

One's mother is one's female parent

Fun with sentences

Brothers are siblings

$$\forall x, y \;\; Brother(x, y) \Rightarrow Sibling(x, y).$$

"Sibling" is symmetric

$$\forall x, y \;\; Sibling(x, y) \Leftrightarrow Sibling(y, x).$$

One's mother is one's female parent

$$\forall x, y \;\; Mother(x, y) \Leftrightarrow (Female(x) \wedge Parent(x, y)).$$

A first cousin is a child of a parent's sibling

Fun with sentences

Brothers are siblings

$$\forall x, y \;\; Brother(x, y) \;\Rightarrow\; Sibling(x, y).$$

"Sibling" is symmetric

$$\forall x, y \;\; Sibling(x, y) \;\Leftrightarrow\; Sibling(y, x).$$

One's mother is one's female parent

$$\forall x, y \;\; Mother(x, y) \;\Leftrightarrow\; (Female(x) \wedge Parent(x, y)).$$

A first cousin is a child of a parent's sibling

$$\forall x, y \;\; FirstCousin(x, y) \;\Leftrightarrow\; \exists p, ps \;\; Parent(p, x) \wedge Sibling(ps, p) \wedge Parent(ps, y)$$

Combining Quantifiers

$\forall x \exists y$ Loves(x,y)

- For everyone ("all x") there is someone ("y") who loves them

$\exists y \forall x$ Loves(x,y)

- there is someone ("y") who loves everyone

Clearer with parentheses: $\exists y (\forall x$ Loves(x,y))

Connections between Quantifiers

- Asserting that all x have property P is the same as asserting that does not exist any x that doesn't have the property P

$$\forall x \ \text{Likes}(x, \text{271 class}) \Leftrightarrow \neg \exists x \ \neg \text{Likes}(x, \text{271 class})$$

In effect:

- $\forall$ is a conjunction over the universe of objects
- $\exists$ is a disjunction over the universe of objects

Thus, DeMorgan's rules can be applied

De Morgan's Law for Quantifiers

$$\forall x\ \neg P \equiv \neg\exists x\ P \qquad \neg(P \vee Q) \equiv \neg P \wedge \neg Q$$
$$\neg\forall x\ P \equiv \exists x\ \neg P \qquad \neg(P \wedge Q) \equiv \neg P \vee \neg Q$$
$$\forall x\ P \equiv \neg\exists x\ \neg P \qquad P \wedge Q \equiv \neg(\neg P \vee \neg Q)$$
$$\exists x\ P \equiv \neg\forall x\ \neg P \qquad P \vee Q \equiv \neg(\neg P \wedge \neg Q)\ .$$

Rule is simple: if you bring a negation inside a disjunction or a conjunction, always switch between them (or →and, and → or).

Inference in First-Order Logic

Outline

- Reducing first-order inference to propositional inference
- Unification
- Generalized Modus Ponens
- Forward chaining
- Backward chaining
- Resolution

Universal instantiation (UI)

- Notation: Subst({v/g}, α) means the result of substituting g for v in sentence α

- Every instantiation of a universally quantified sentence is entailed by it:

$$\frac{\forall v\ \alpha}{\text{Subst}(\{v/g\}, \alpha)}$$

for any variable v and ground term g

- E.g., $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$ yields

$King(John) \wedge Greedy(John) \Rightarrow Evil(John)$, ***{x/John}***

$King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$, ***{x/Richard}***

$King(Father(John)) \wedge Greedy(Father(John)) \Rightarrow Evil(Father(John))$, ***{x/Father(John)}***

Existential instantiation (EI)

- For any sentence α, variable *v*, and constant symbol *k (*that does not appear elsewhere in the knowledge base):

$$\frac{\exists v\ \alpha}{\text{Subst}(\{v/k\}, \alpha)}$$

- E.g., $\exists x\ Crown(x) \wedge OnHead(x,John)$ yields: $Crown(C_1) \wedge OnHead(C_1,John)$
- where C_1 is a new constant symbol, called a Skolem constant

- Existential and universal instantiation allows to "propositionalize" any FOL sentence or KB
  - EI produces one instantiation per EQ sentence
  - UI produces a whole set of instantiated sentences per UQ sentence

Reduction to propositional form

Suppose the KB contains the following:

$\forall x\ \text{King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$
Father(x)
King(John)
Greedy(John)
Brother(Richard,John)

- Instantiating the universal sentence in all possible ways, we have:
  $\text{King}(\text{John}) \wedge \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John})$
  $\text{King}(\text{Richard}) \wedge \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard})$
  King(John)
  Greedy(John)
  Brother(Richard,John)

- The new KB is propositionalized: propositional symbols are
  - King(John), Greedy(John), Evil(John), King(Richard), etc

Reduction continued

- Every FOL KB can be propositionalized so as to preserve entailment
  - A ground sentence is entailed by new KB iff entailed by original KB
- Idea for doing inference in FOL:
  - propositionalize KB and query
  - apply resolution-based inference
  - return result
- Problem: with function symbols, there are infinitely many ground terms,
  - e.g., *Father*(*Father*(*Father*(*John*))), etc

Reduction continued

Theorem: Herbrand (1930). If a sentence α is entailed by a FOL KB, it is entailed by a finite subset of the propositionalized KB

Idea: For $n = 0$ to ∞ do

create a propositional KB by instantiating with depth-n terms
see if α is entailed by this KB

Example

$\forall x$ King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x)
Father(x)
King(John)
Greedy(Richard)
Brother(Richard,John)

Query Evil(X)?

- Depth 0

  Father(John)

  Father(Richard)

  King(John)

  Greedy(Richard)

  Brother(Richard , John)

  $King(John) \wedge Greedy(John) \Rightarrow Evil(John)$

  $King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$

  $King(Father(John)) \wedge Greedy(Father(John)) \Rightarrow Evil(Father(John))$

  $King(Father(Richard)) \wedge Greedy(Father(Richard)) \Rightarrow Evil(Father(Richard))$

- Depth 1

  Depth 0 +

  Father(Father(John))

  Father(Father(John))

  $King(Father(Father(John))) \wedge Greedy(Father(Father(John))) \Rightarrow Evil(Father(Father(John)))$

Problems with Propositionalization

- Problem: works if α is entailed, loops if α is not entailed
- Propositionalization generates lots of irrelevant sentences
  - So inference may be very inefficient
- e.g., from:

  $\forall x\ \text{King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$
  $\text{King}(\text{John})$
  $\forall y\ \text{Greedy}(y)$
  $\text{Brother}(\text{Richard},\text{John})$

- It seems obvious that *Evil*(*John*) is entailed, but propositionalization produces lots of facts such as *Greedy*(*Richard*) that are irrelevant
- With p k-ary predicates and n constants, there are $p \cdot n^k$ instantiations
- Lets see if we can do inference directly with FOL sentences

Unification

- Recall: Subst(θ, p) = result of substituting θ into sentence p

- Unify algorithm: takes 2 sentences p and q and returns a unifier if one exists

  Unify(p,q) = θ where Subst(θ, p) = Subst(θ, q)

- Example:
  p = Knows(John,x)
  q = Knows(John, Jane)

  Unify(p,q) = {x/Jane}

Unification examples

- simple example: query = Knows(John,x), i.e., who does John know?

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}
Knows(John,x)	Knows(y,OJ)	{x/OJ,y/John}
Knows(John,x)	Knows(y,Mother(y))	{y/John,x/Mother(John)}
Knows(John,x)	Knows(x,OJ)	{fail}

- Last unification fails: only because x can't take values John and OJ at the same time
- Problem is due to use of same variable x in both sentences
- Simple solution: Standardizing apart eliminates overlap of variables, e.g., Knows(z,OJ)
-

Unification

- To unify *Knows(John,x)* and *Knows(y,z)*,
- θ = {y/John, x/z } or θ = {y/John, x/John, z/John}
- The first unifier is more general than the second.
- There is a single most general unifier (MGU) that is unique up to renaming of variables.
- MGU = { y/John, x/z }
- General algorithm in Figure 9.1 in the text

Recall our example…

$\forall x\ \text{King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$

King(John)

$\forall y\ \text{Greedy}(y)$

Brother(Richard,John)

And we would like to infer Evil(John) without propositionalization

Generalized Modus Ponens (GMP)

$p_1', p_2', \ldots, p_n', (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)$

$Subst(\theta, q)$

where we can unify p_i' and p_i for all i

Example:

$King(John), Greedy(John), \forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$

$Evil(John)$

p_1' is $King(John)$ p_1 is $King(x)$

p_2' is $Greedy(John)$ p_2 is $Greedy(x)$

θ is $\{x/John\}$ q is $Evil(x)$

$Subst(\theta, q)$ is $Evil(John)$