IBM

Smarter education for smarter planet

IBM Innovation Centre for Education

Welcome to:

Introduction to Artificial Intelligence and Machine Learning

Welcome to:

Unit-2: Logical Approach to AI and Knowledge Based Systems

Introduction

- To solve the complex problems encountered in Artificial Intelligence, large amount of knowledge and mechanisms for manipulating that knowledge are required.

- Many ways are there for Knowledge representation

- Properties of Knowledge Representation Systems
  - Representational adequacy
  - Inferential adequacy
  - Inferential efficiency
  - Acquisitional efficiency

Knowledge Representation

- Representation of knowledge and the reasoning process are central to the entire field of artificial intelligence.

- The primary component is knowledge-base.

- A knowledge-base is a set of sentences.
  - Represent some assertions about the world.

- Inferencing or reasoning

Knowledge Representation using Logic

- Formal Logic is the primary vehicle for representing & reasoning about knowledge.

- Formal logic is
  - Precise and Definite
  - Declarative

- Logic consists of two parts, a language and a method of reasoning.

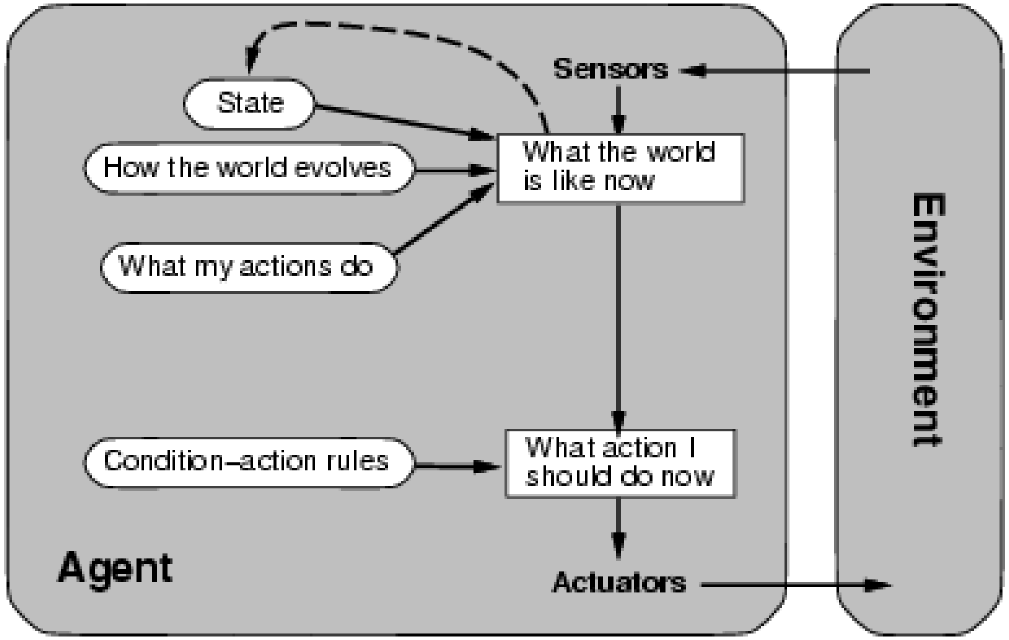
- Language has syntax and semantics

- Logical systems with different syntax and semantics
  - Propositional logic
  - First order predicate logic
  - Temporal logic
  - Modal
  - Higher order logic
  - Non-monotonic

Model-based Agents

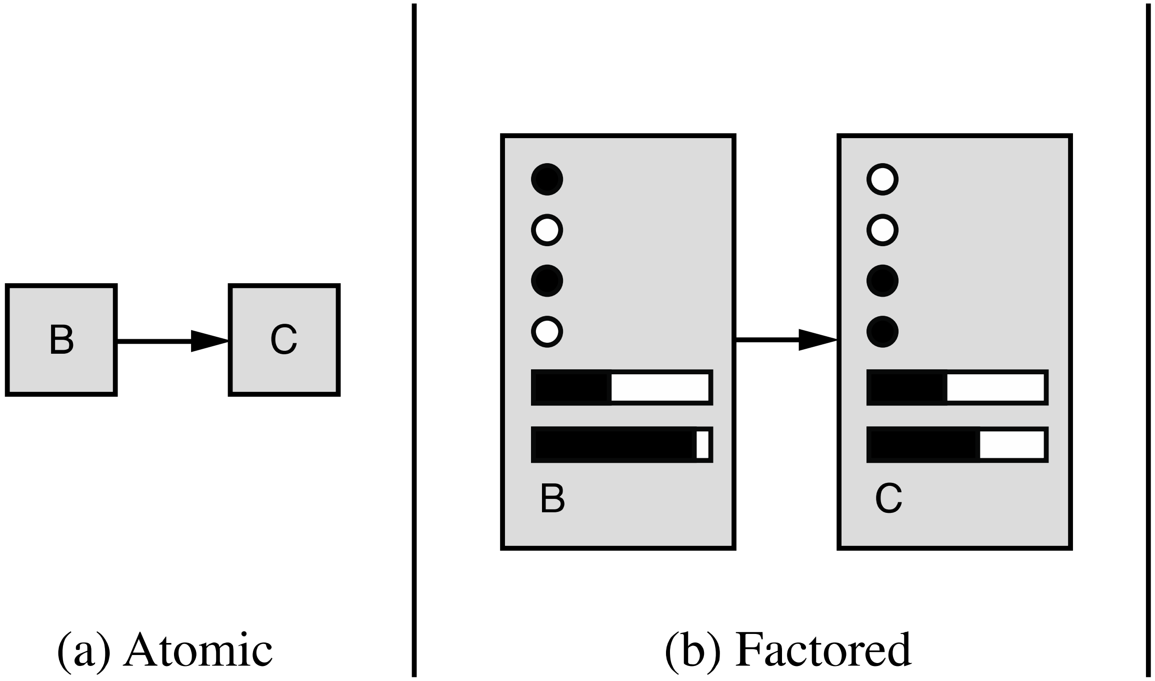
- Know how world evolves
  - Overtaking car gets closer from behind
- How agents actions affect the world
  - Wheel turned clockwise takes you right

- Model base agents update their state.
- Can also add goals and utility/performance measures.



Agent Architecture: Logical Agents

A model is a **structured** representation of the world.



(a) Atomic (b) Factored

- Graph-Based Search: State is **black box**, no internal structure, atomic.
- Factored Representation: State is list or vector of facts.
- Facts are expressed in **formal logic.**

Logic: Motivation

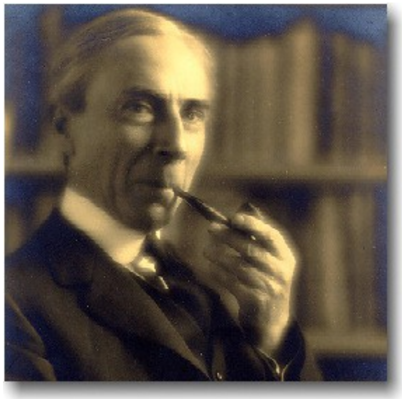
- 1st-order logic is highly expressive.
  - Almost all of known mathematics.
  - All information in relational databases.
  - Can translate much natural language.
  - Can reason about other agents, beliefs, intentions, desires…
- Logic has **complete** inference procedures.
  - All valid inferences can be proven, in principle, by a machine.
- Cook’s fundamental theorem of NP-completeness states that all difficult search problems (scheduling, planning, CSP etc.) can be represented as logical inference problems. (U of T).

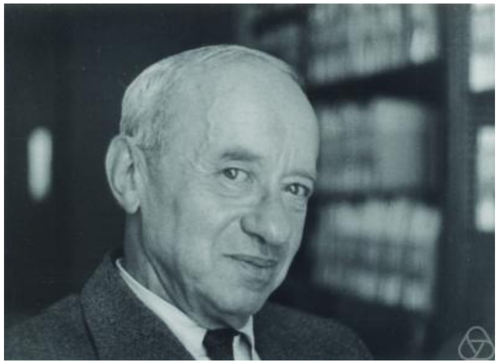
Logic vs. Programming Languages

- Logic is declarative.
- Think of logic as a kind of **language** for expressing knowledge.
  - Precise, computer readable.
- A proof system allows a computer to **infer** consequences of known facts.
- Programming languages lack general mechanism for deriving facts from other facts.

1^{st}-order Logic: Key ideas

- The fundamental question: *What kinds of information do we need to represent?* (Russell, Tarski).
- The world/environment consists of
  - Individuals/entities.
  - Relationships/links among them.



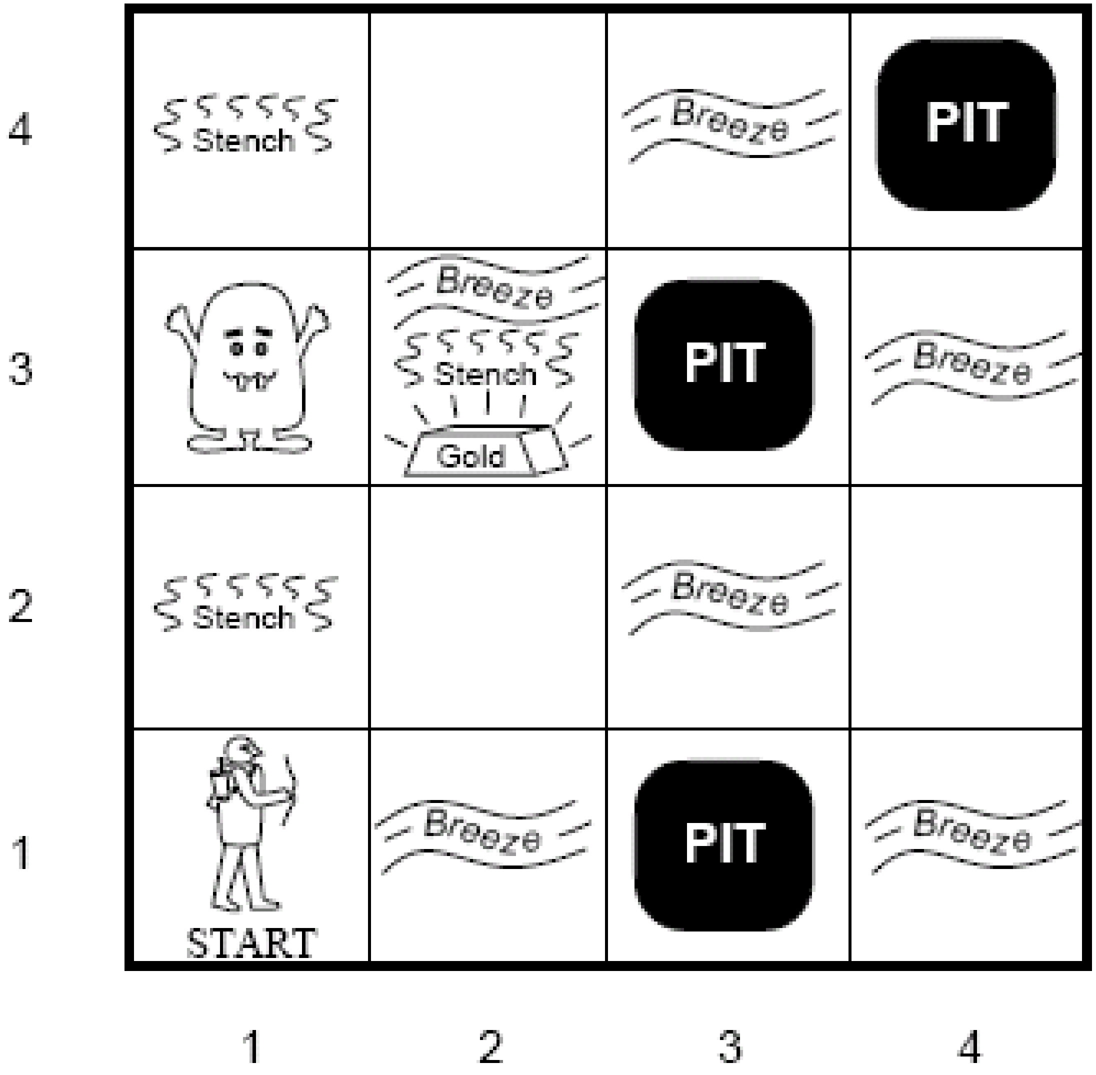


Knowledge-Based Agents

- 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

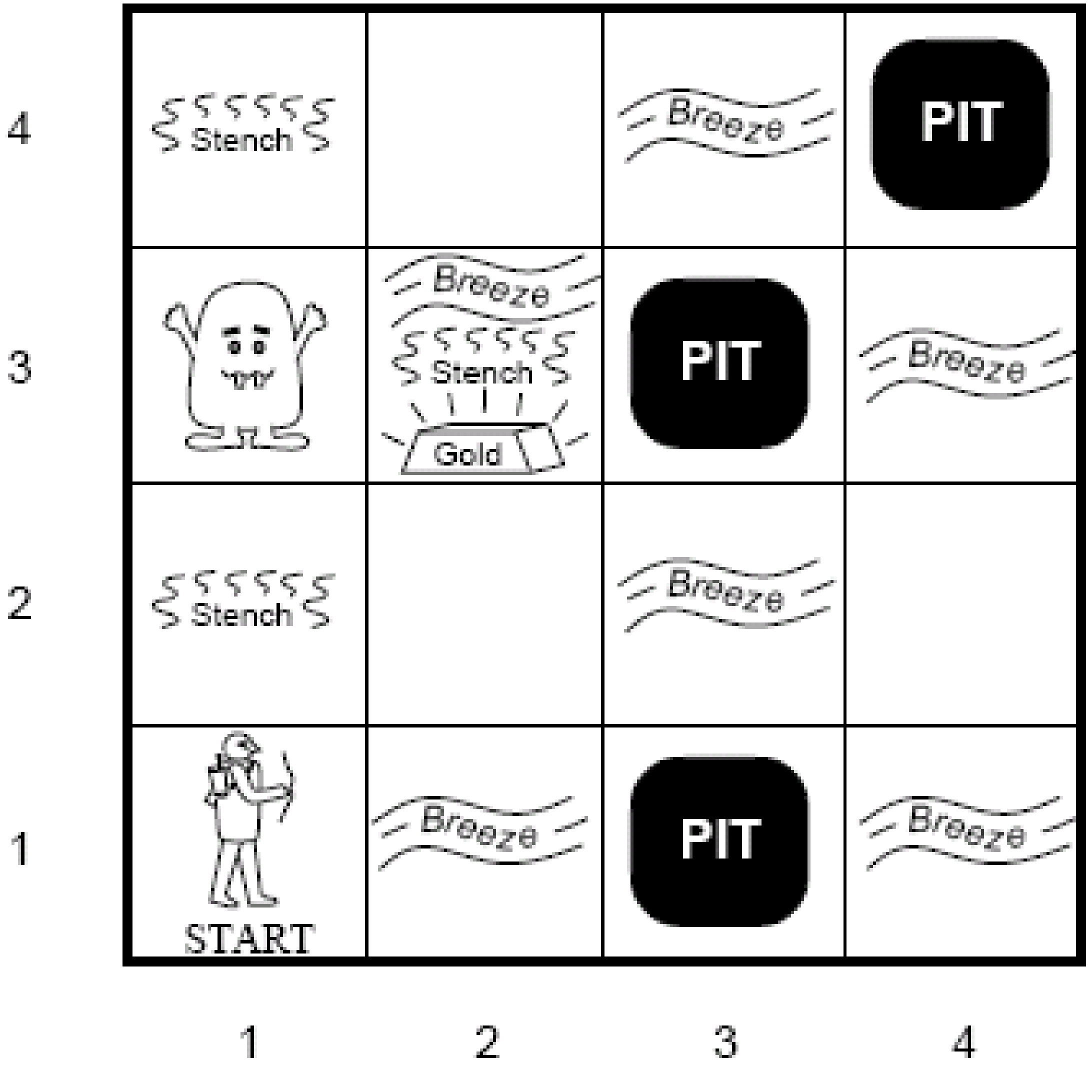
- Environment
  - Cave of 4×4
  - Agent enters in [1,1]
  - 16 rooms
    - Wumpus: A deadly beast who kills anyone entering his room.
    - Pits: Bottomless pits that will trap you forever.
    - Gold



Wumpus World

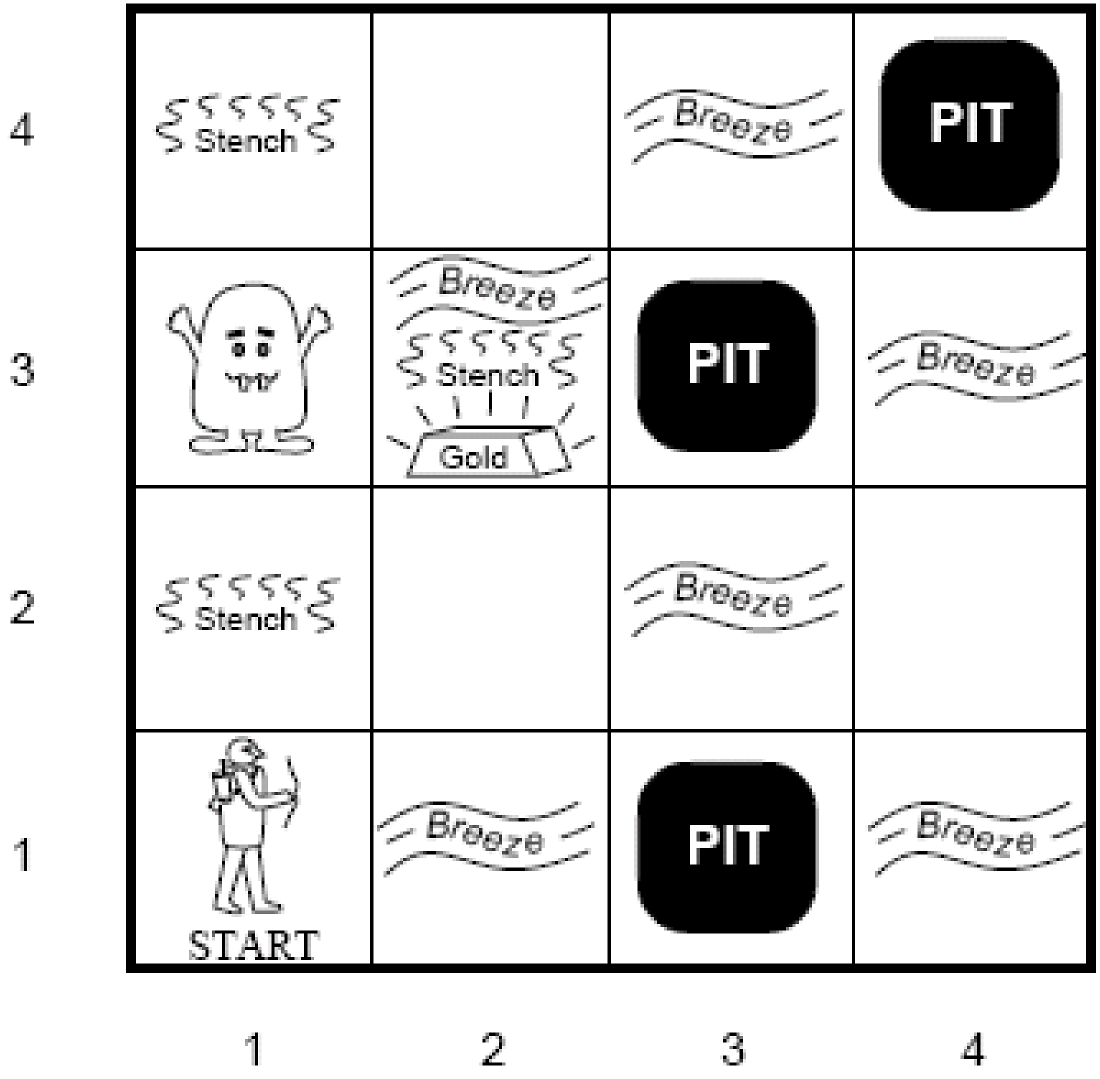
- Agents Sensors:
  - Stench next to Wumpus
  - Breeze next to pit
  - Glitter in square with gold
  - Bump when agent moves into a wall
  - Scream from wumpus when killed

- Agents actions
  - Agent can move forward, turn left or turn right
  - Shoot, one shot



Wumpus World

- Performance measure
  - +1000 for picking up gold
  - -1000 got falling into pit
  - -1 for each move
  - -10 for using arrow



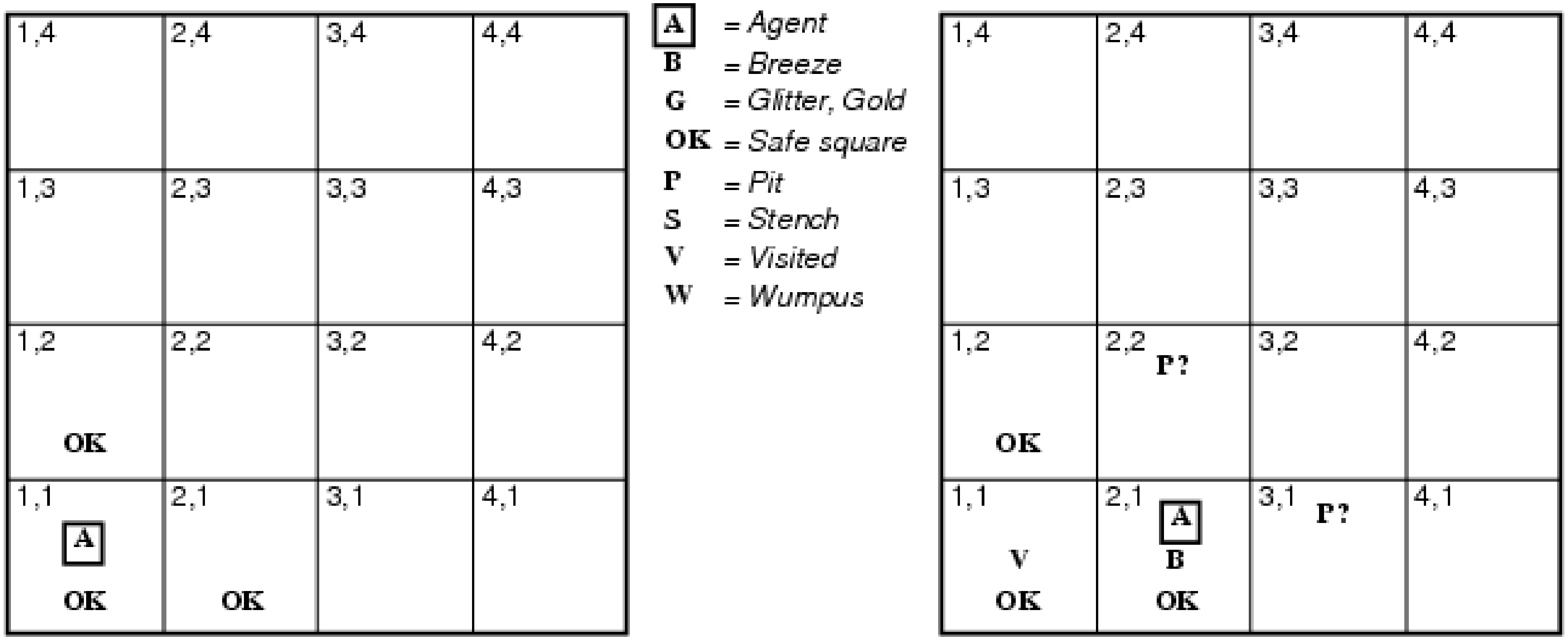
Reasoning in the Wumpus World

- 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 used to test agent reasoning and decision algorithms.

Exploring the Wumpus World



(a)

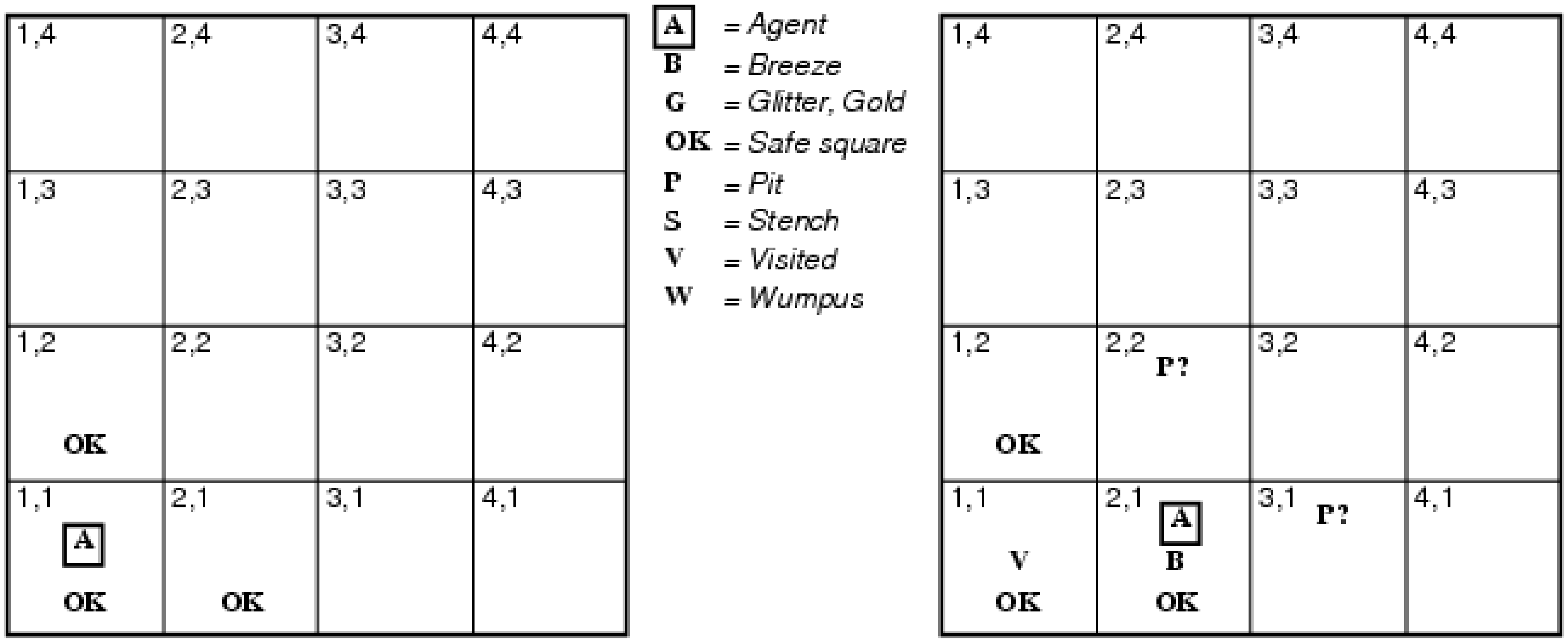
(b)

[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



(a)

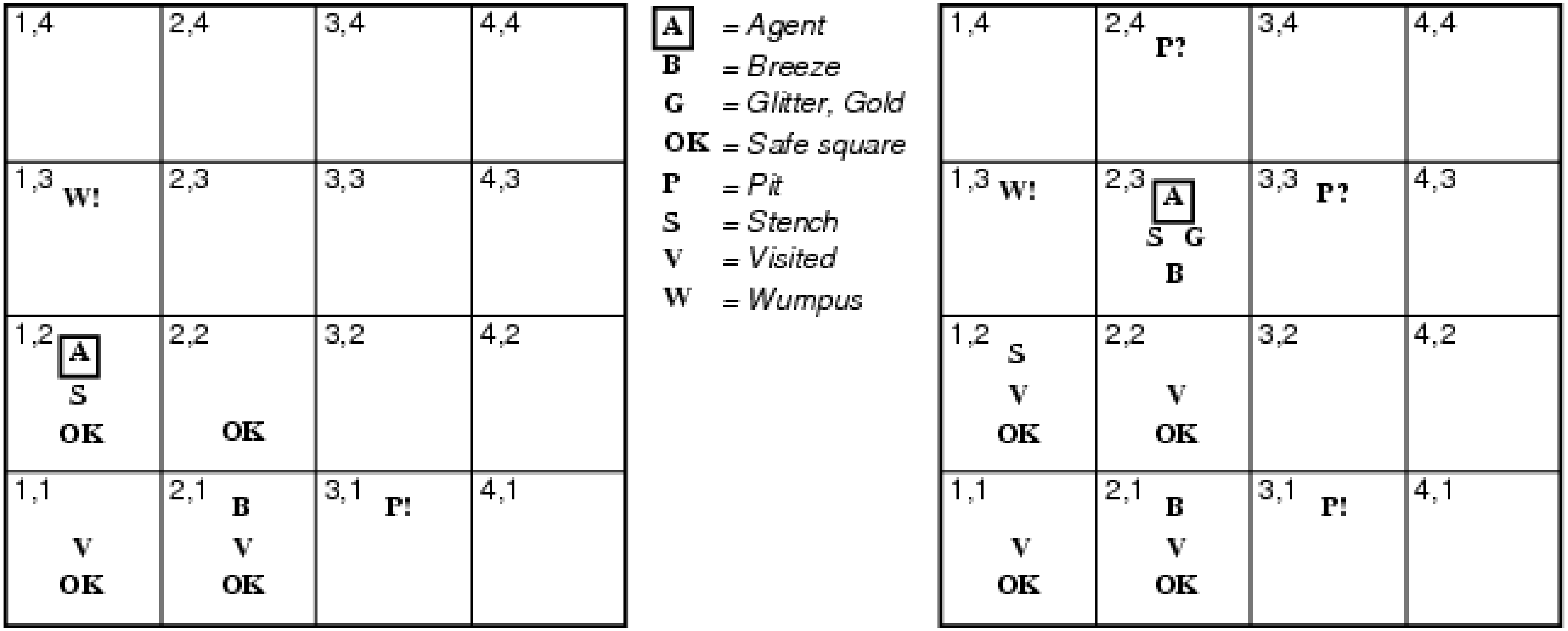
(b)

[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



(a) (b)

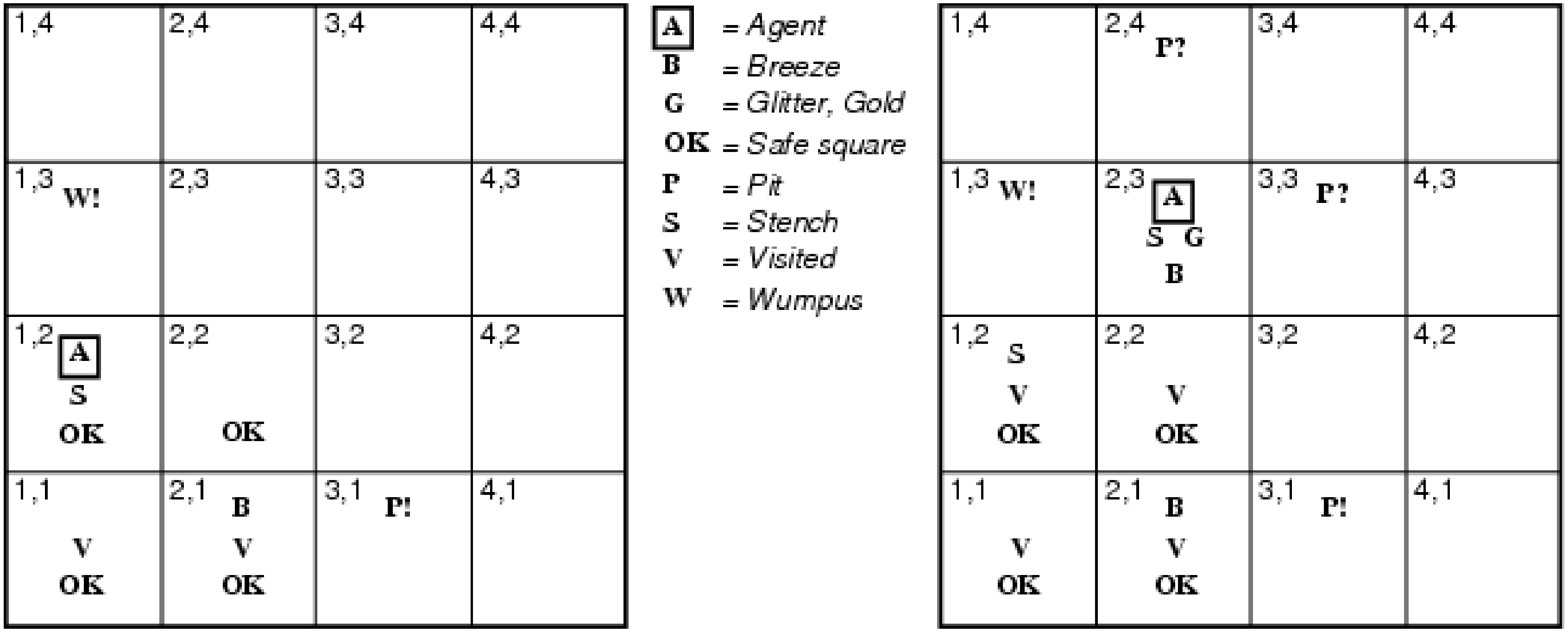
[1,2] Stench in cell which means that wumpus is in [1,3] or [2,2]

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



(a) (b)

[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!)

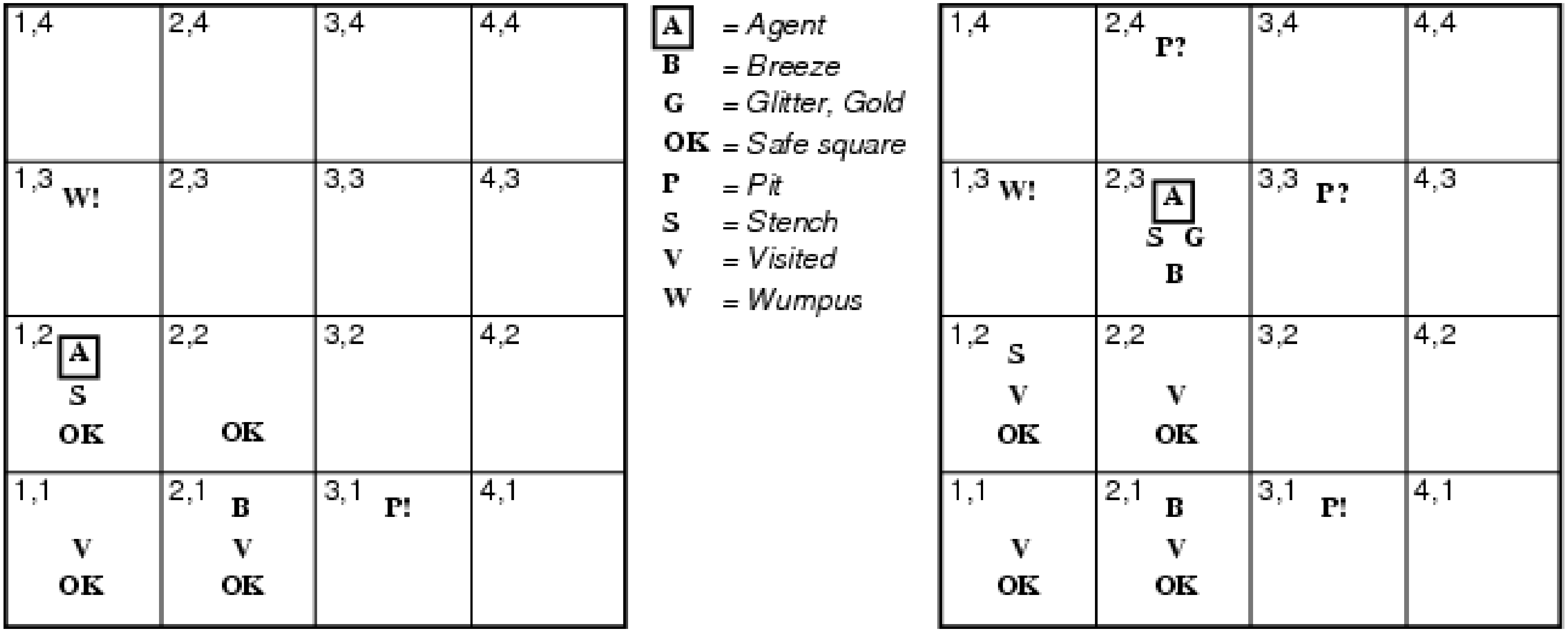
THUS … wumpus is in [1,3]

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



1,4	2,4	3,4	4,4
1,3 W!	2,3	3,3	4,3
1,2 A S OK	2,2 OK	3,2	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1

(a)

A	= Agent
B	= Breeze
G	= Glitter, Gold
OK	= Safe square
P	= Pit
S	= Stench
V	= Visited
W	= Wumpus

1,4	2,4 P?	3,4	4,4
1,3 W!	2,3 A S G B	3,3 P?	4,3
1,2 S V OK	2,2 V OK	3,2	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1

(b)

[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 logical language?

- A formal language
  - KB = set of sentences

- Syntax
  - What sentences are legal (well-formed)
  - E.g., arithmetic
    - X+2 >= y is a wf sentence, +x2y is not a wf 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
    - Fuzzy logic – allows for degrees of truth.

Propositional logic: Syntax

- Propositional logic is the simplest logic – illustrates basic ideas

- Atomic sentences = single proposition symbols
  - E.g., P, Q, R
  - Special cases: True = always true, False = always false

- Complex sentences:
  - If S is a sentence, □S is a sentence (negation)
  - If S_1 and S_2 are sentences, S_1 □ S_2 is a sentence (conjunction)
  - If S_1 and S_2 are sentences, S_1 □ S_2 is a sentence (disjunction)
  - If S_1 and S_2 are sentences, S_1 □ S_2 is a sentence (implication)
  - If S_1 and S_2 are sentences, S_1 □ S_2 is a sentence (biconditional)

Wumpus world sentences

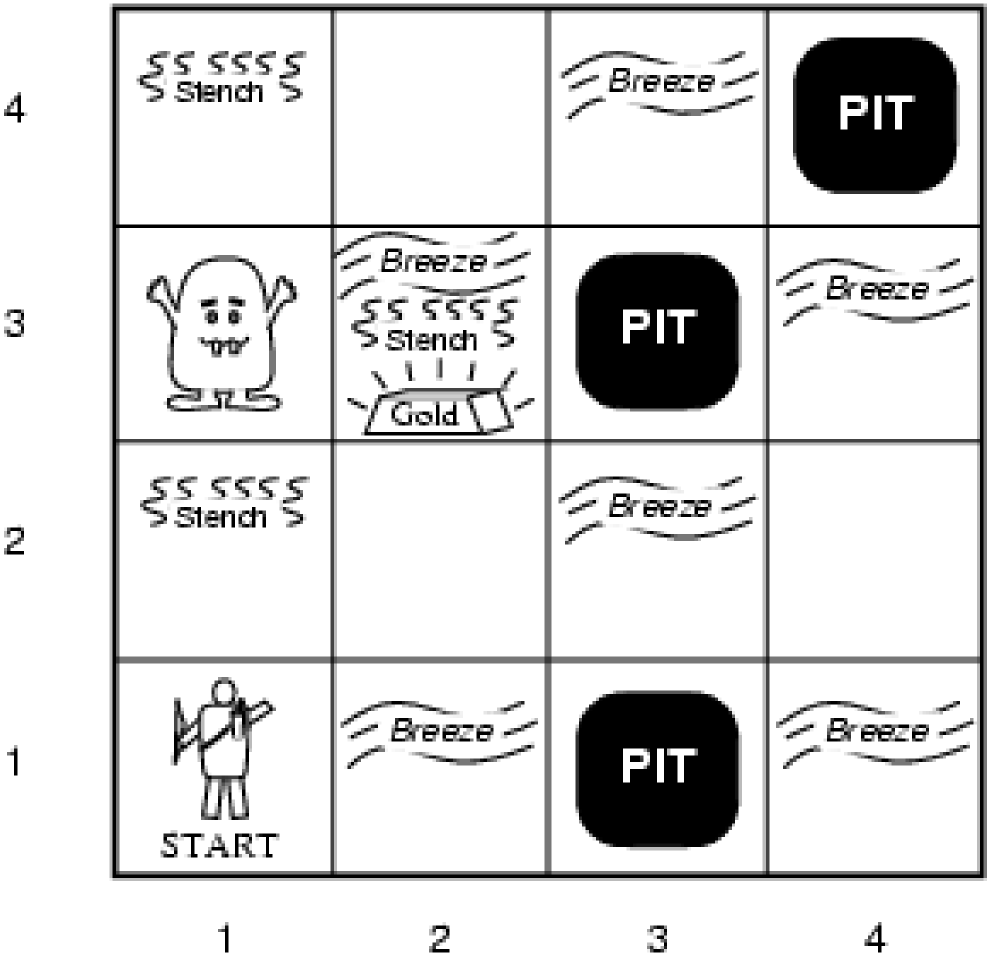
Let $P_{i,j}$ be true if there is a pit in [i, j].

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"

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

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

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

- Note that these sentences are rather long-winded!
  - E.g., breeze “rule” must be stated explicitly for each square
  - First-order logic will allow us to define more general patterns.

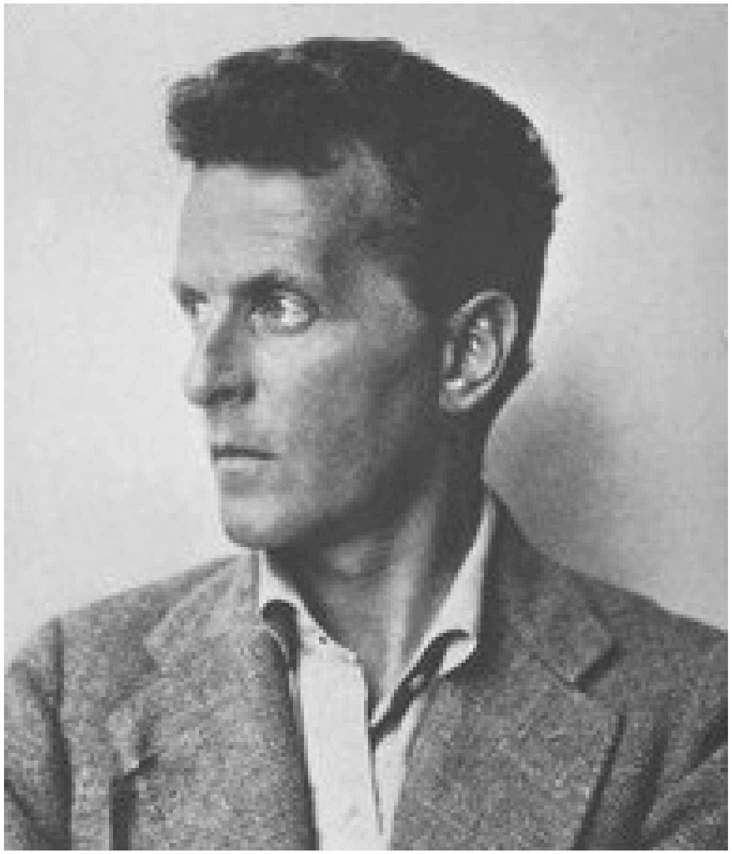
Propositional logic: Semantics

- A sentence is interpreted in terms of **models**, or **possible worlds**.

- These are formal structures that specify a truth value for **each sentence** in a consistent manner.

Ludwig Wittgenstein (1918):

1. The world is everything that is the case.
    1. The world is the complete collection of facts, not of things.
        1. The world is determined by the facts, and by being the *complete* collection of facts.



Ludwig Josef Johann Wittgenstein:
(26 April 1889 – 29 April 1951) was an Austrian-British philosopher who worked primarily in logic, the philosophy of mathematics, the philosophy of mind, and the philosophy of language.

More on Possible Worlds

- *m* is a model of a sentence □ if □ is true in *m*
- *M(□)* is the set of all models of □
- 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.
  - Wumpus Example:
    - P[x,y] is true if there is a pit in square [x,y].
    - W[x,y] is true if there is a wumpus in [x,y].
    - S[x,y] is true if the agent perceives a stench when in square [x,y].

Propositional logic: Formal Semantics

IBM ICE (Innovation Centre for Education)

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

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

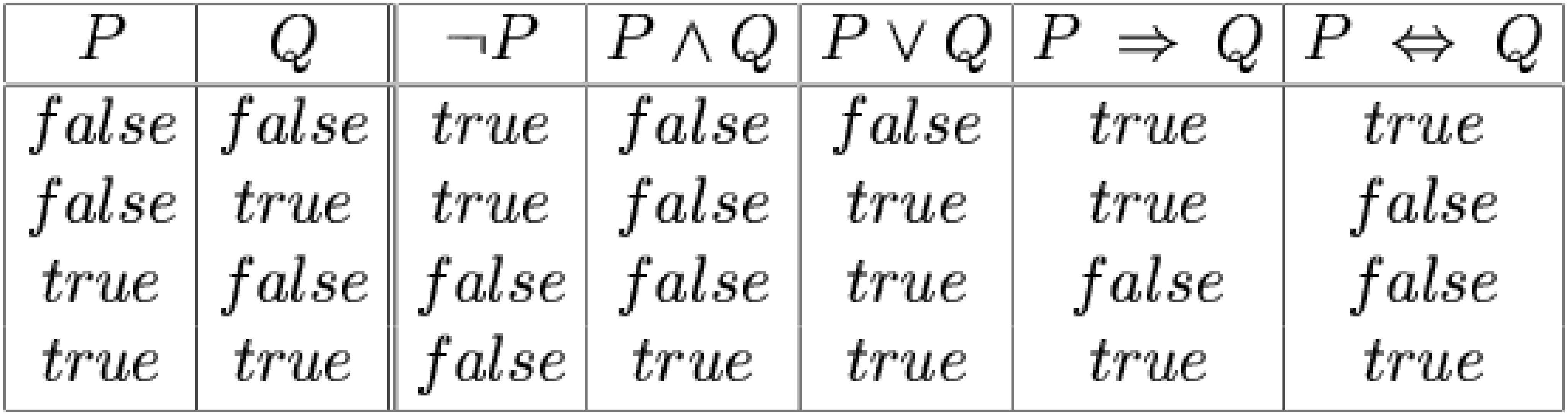
$S_1 \Rightarrow S_2$ is true iff S_1 is false or S_2 is true
i.e., is false iff S_1 is true and S_2 is false

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

Simple recursive process evaluates **every** sentence, e.g.,

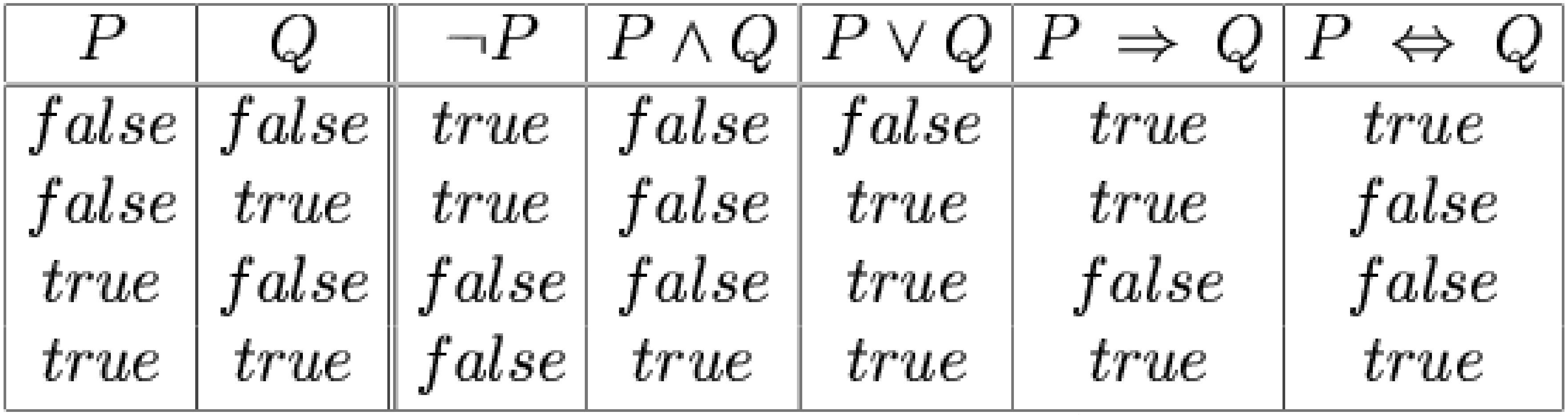
$$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = \mathit{true} \wedge (\mathit{true} \vee \mathit{false}) = \mathit{true} \wedge \mathit{true} = \mathit{true}$$

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

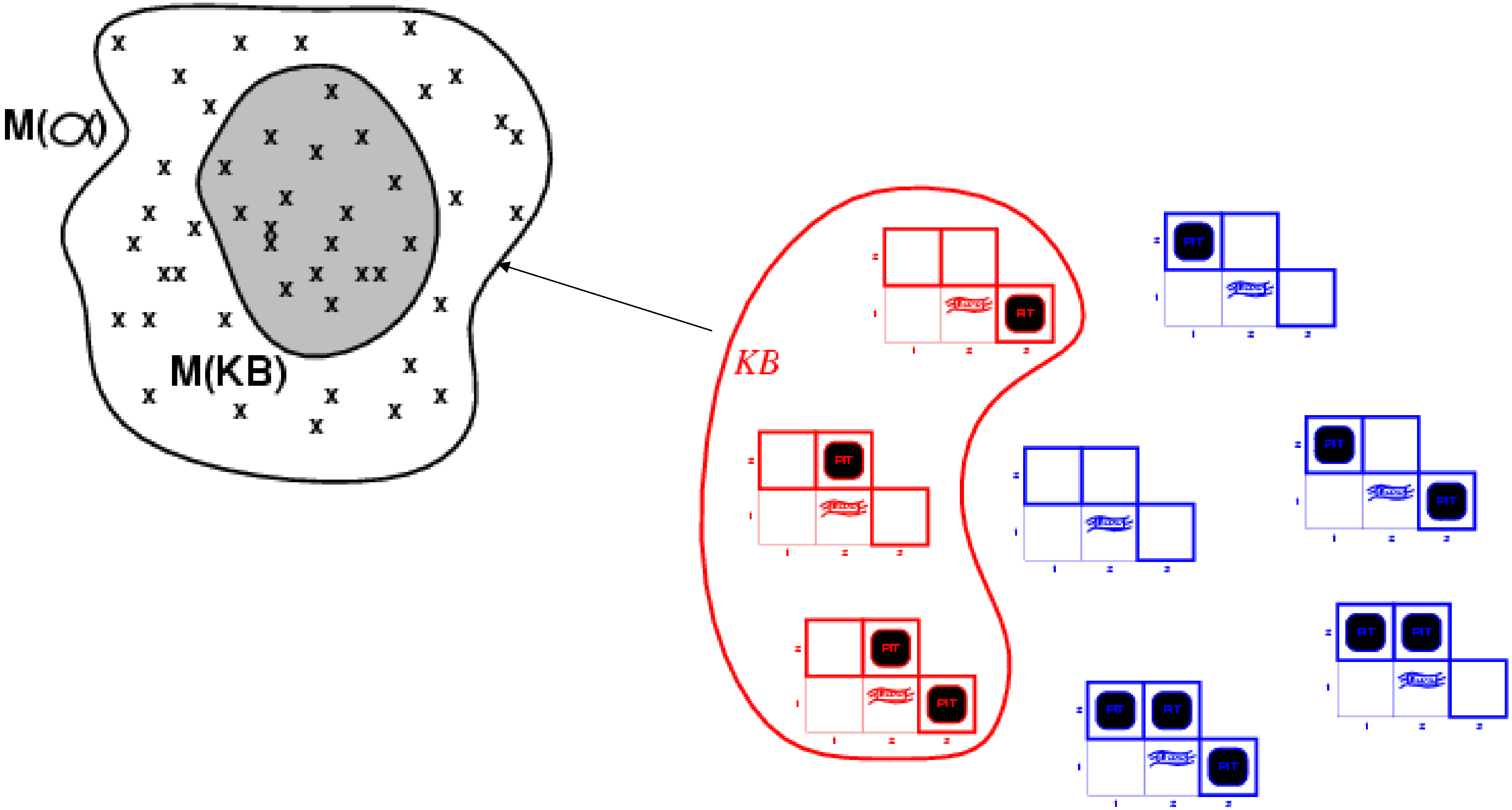
Implication is always true when the premise is false

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 models

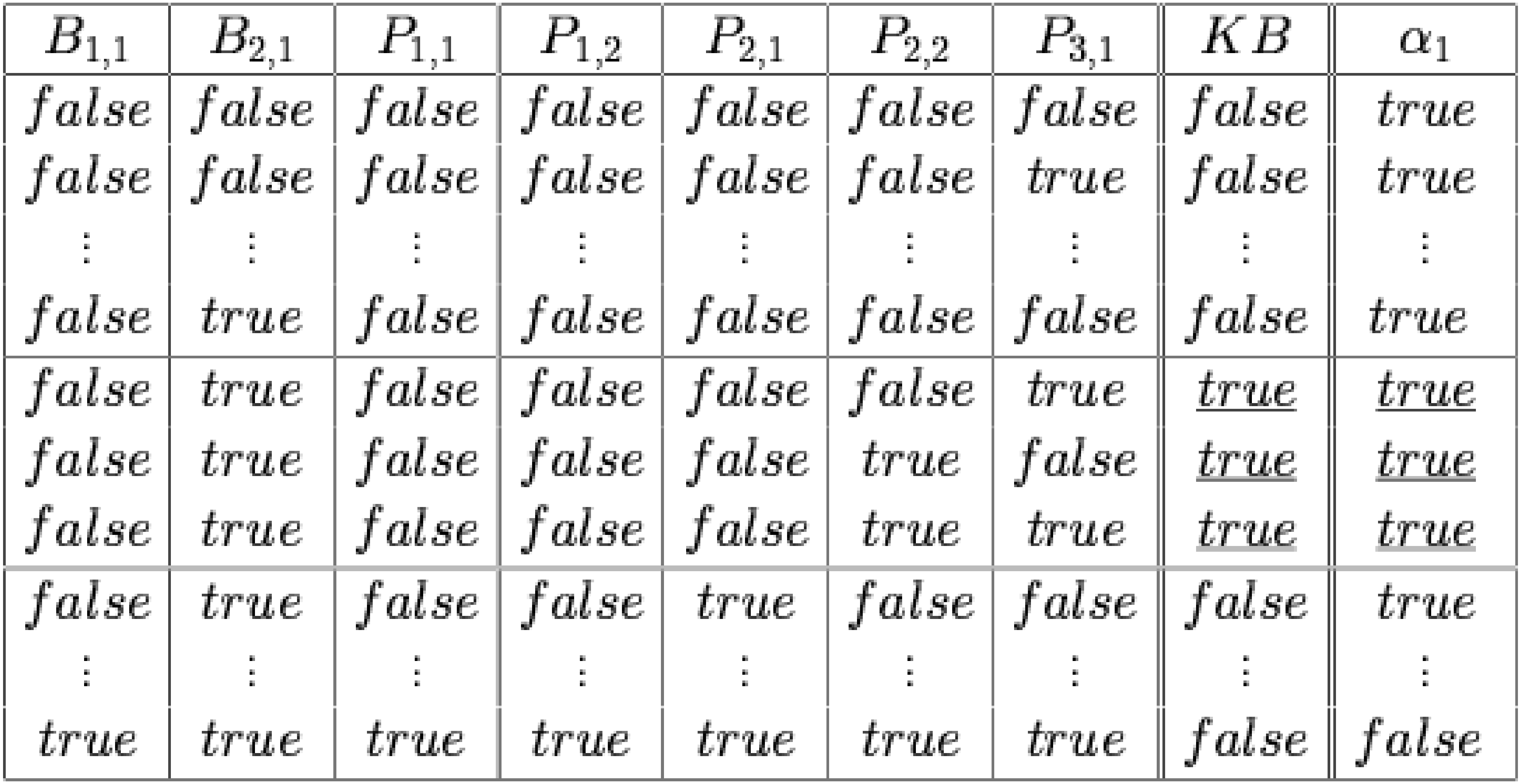
- *KB* = all possible wumpus-worlds consistent with the observations and the “physics” of the Wumpus world.



Listing of possible worlds for the Wumpus KB

α_1 = ”square [1,2] is safe”.

KB = detect nothing in [1,1], detect breeze in [2,1]



$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB	α_1
false	*false*	*false*	*false*	*false*	*false*	*false*	*false*	*true*
false	*false*	*false*	*false*	*false*	*false*	*true*	*false*	*true*
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
false	*true*	*false*	*false*	*false*	*false*	*false*	*false*	*true*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*false*	*true*
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*false*

Entailment

- One sentence follows logically from another

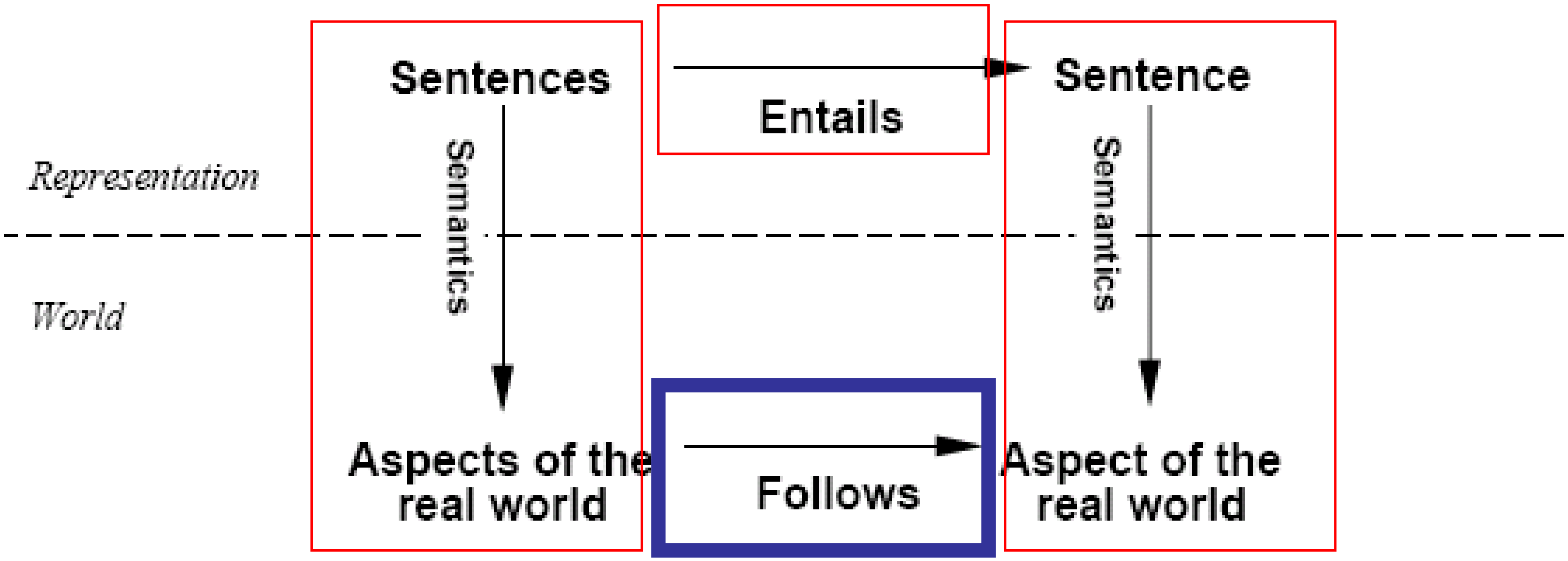
  a |= *b*

  a entails sentence b *if and only if* b is true in all worlds where a is true.

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

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

Schematic Perspective

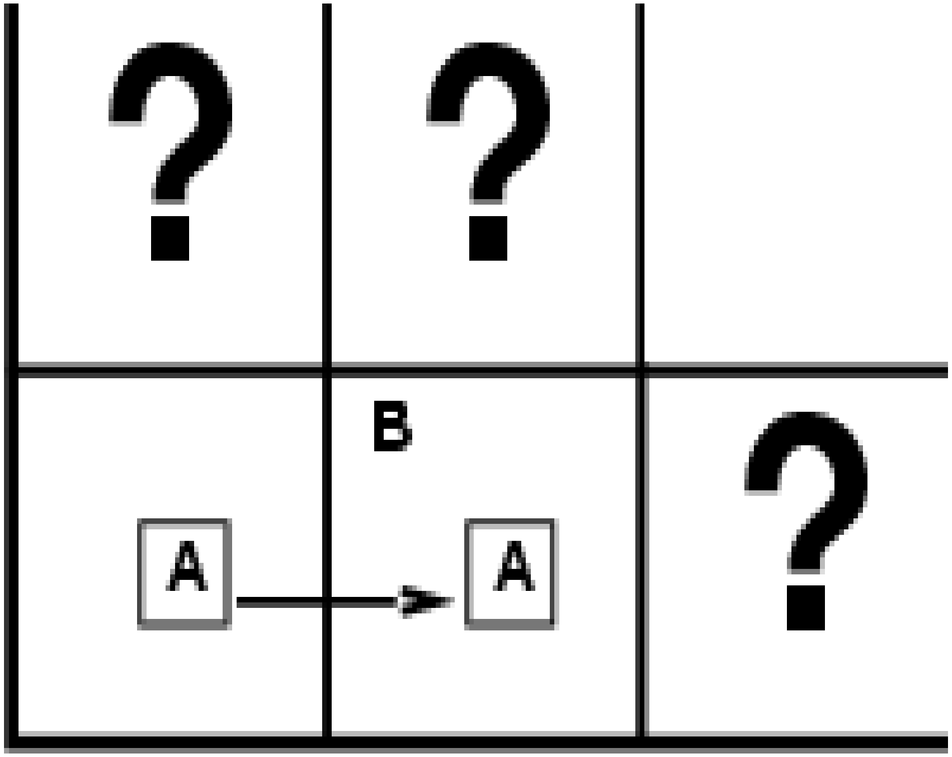


If KB is true in the real world, then any sentence □ *derived from KB by a sound inference procedure is also true in the real world.*

Entailment in the wumpus world

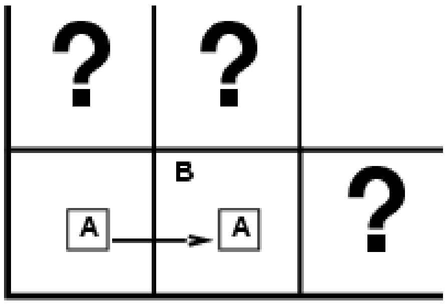
- Consider possible models for *KB* assuming only pits and a reduced Wumpus world

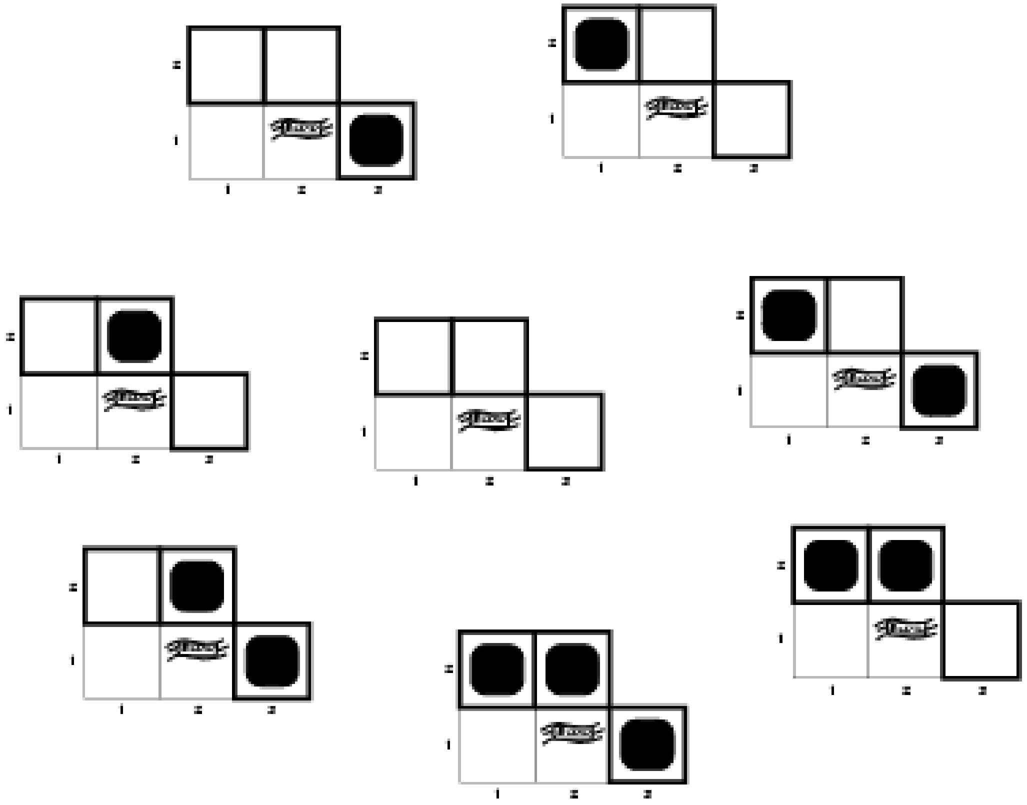
- Situation after detecting nothing in [1,1], moving right, detecting breeze in [2,1]



Wumpus models

All possible models in this reduced Wumpus world.





Inferring conclusions

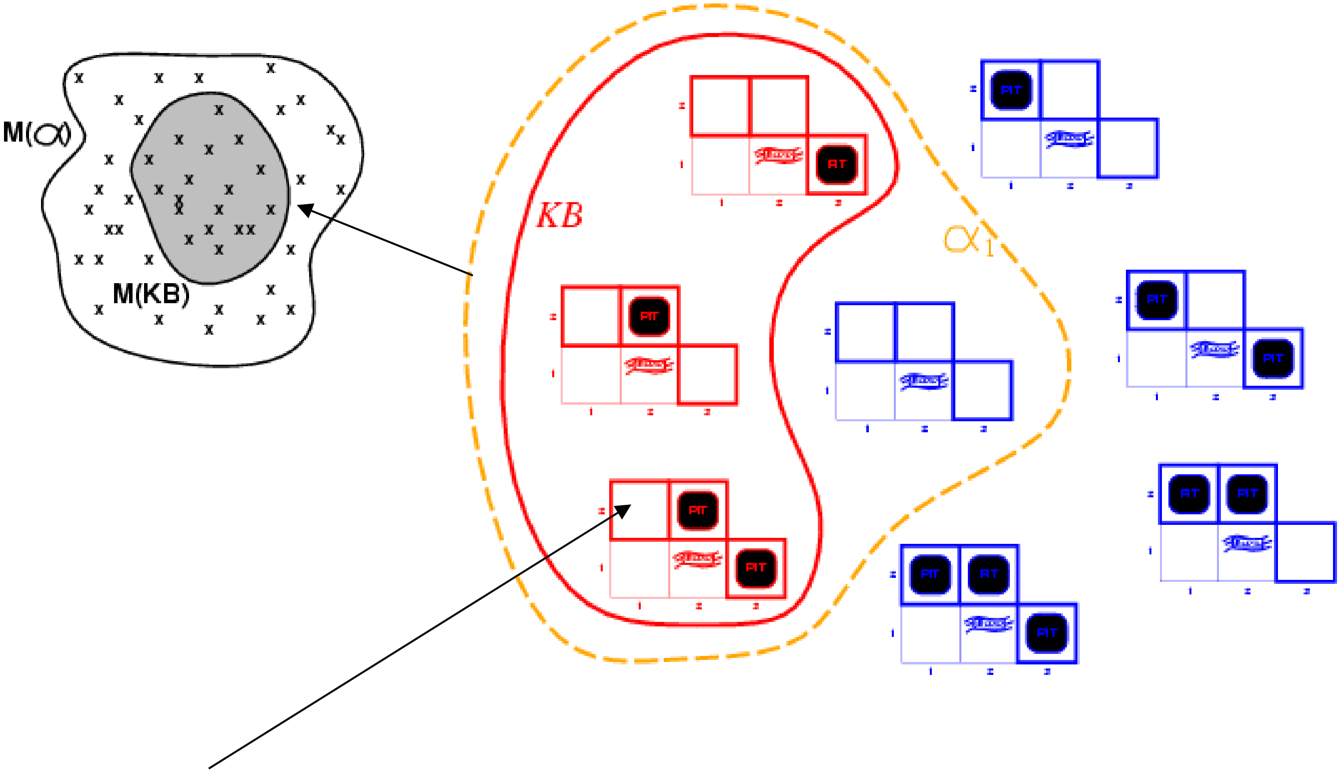
- Consider 2 possible conclusions given a KB
  - α_1 = "[1,2] is safe"
  - α_2 = "[2,2] is safe"

- One possible inference procedure
  - Start with KB
  - Model-checking
    - Check if KB $\models$ a by checking if in all possible models where KB is true that a is also true

- Comments:
  - Model-checking enumerates all possible worlds
    - Only works on finite domains, will suffer from exponential growth of possible models

Wumpus models

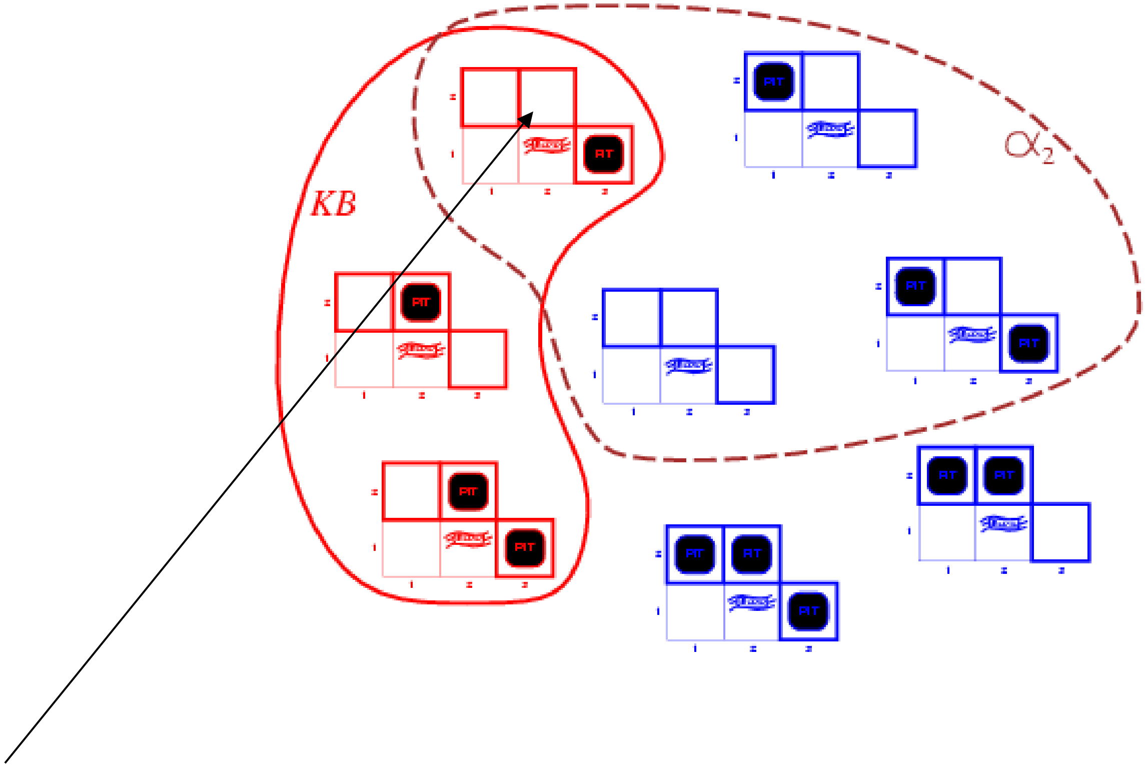
α_1 = "[1,2] is safe", $KB \models \alpha_1$, proved by model checking



Wumpus models

α_2 = "[2,2] is safe", $KB \models \alpha_2$

- There are some models entailed by KB where a_2 is false.



Logical inference

- The notion of entailment can be used for inference.
  - Model checking (see wumpus example): enumerate all possible models and check whether □ is true.

- If an algorithm only derives entailed sentences it is called *sound* or *truth preserving*.
- A proof system is **sound** if whenever the system derives □ from KB, it is also true that KB|= □
  - *E.g., model-checking is sound*

- Completeness : the algorithm can derive any sentence that is entailed.
- A proof system is **complete** if whenever KB|= □, the system derives □ from KB.

Inference by Enumeration

- We want to see if a is entailed by KB

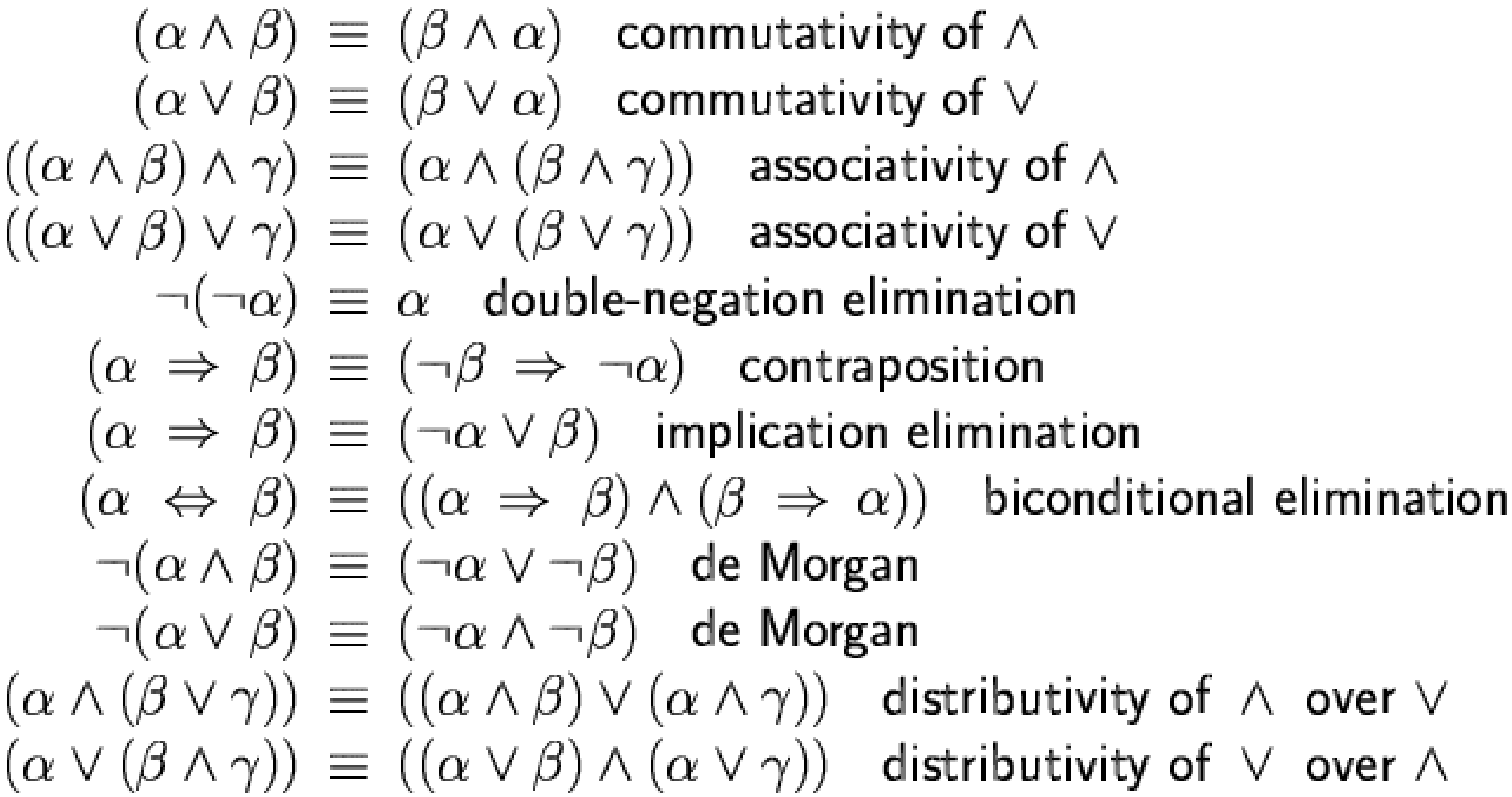
- Enumeration of all models is sound and complete.

- But…for *n* symbols, time complexity is $O(2^n)$...

- We need a more efficient way to do inference
  - But worst-case complexity will remain exponential for propositional logic

Logical Equivalence

- To manipulate logical sentences we need some rewrite rules.
- Two sentences are logically equivalent iff they are true in same models: α ≡ ß iff α ╞ β and β ╞ α



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

Exercises

- Show that *P implies Q* is logically equivalent to *(not P) or Q*.
  That is, one of these formulas is true in a model just in case the other is true.
- A **literal** is a formula of the form P or of the form not P, where P is an atomic formula. Show that the formula *(P or Q) and (not R)* has an equivalent formula that is a disjunction of a conjunction of literals.
  Thus the equivalent formula looks like this:
  [literal 1 and literal 2 and ….] or [literal 3 and …]

Normal Clausal Form

Eventually we want to prove:

Knowledge base KB entails sentence α

We first rewrite into conjunctive normal form (CNF).

A “conjunction of disjunctions”

literals

$$\underbrace{(A \lor \neg B)}_{\text{Clause}} \land \underbrace{(B \lor \neg C \lor \neg D)}_{\text{Clause}}$$

- **Theorem: Any KB can be converted into an equivalent CNF**.
- k-CNF: exactly k literals per clause

Example: Conversion to CNF

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

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta)\land(\beta \Rightarrow \alpha)$.

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

2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\lnot\alpha\lor \beta$.

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

3. Move $\lnot$ inwards using de Morgan's rules and double-negation:

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

4. Apply distributive law ($\lor$ over $\land$) and flatten:

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

Horn Clauses

Horn Clause = A clause with at most 1 positive literal.

e.g. *A ��B ��C*

- Every Horn clause can be rewritten as an implication with a conjunction of positive literals in the premises and at most a single positive literal as a conclusion.

  e.g. *B �C � A*

- 1 positive literal: definite clause

- 0 positive literals: Fact or integrity constraint:

  e.g. *(��A��غ��B��* *(A B False)*

- Psychologically natural: a condition implies (causes) a single fact.

- The basis of **logic programming** (the prolog language).

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

Limitations of Propositional Logic

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
  - $B_{x,y} \square (P_{x,y+1} \square P_{x,y-1} \square P_{x+1,y} \square P_{x-1,y})$ (Breeze next to Pit) 16 rules

- Find Wumpus
  - $S_{x,y} \square (W_{x,y+1} \square W_{x,y-1} \square W_{x+1,y} \square W_{x-1,y})$ (stench next to Wumpus) 16 rules

- At least one Wumpus in world
  - $W_{1,1} \square W_{1,2} \square \ldots \square W_{4,4}$ (at least 1 Wumpus) 1 rule

- At most one Wumpus
  - $\square W_{1,1} \square \square W_{1,2}$ (155 RULES)

First-Order Logic

- Propositional logic assumes that the world contains facts.

- First-order logic (like natural language) assumes the world contains

  – Objects: people, houses, numbers, colors, baseball games, wars, …

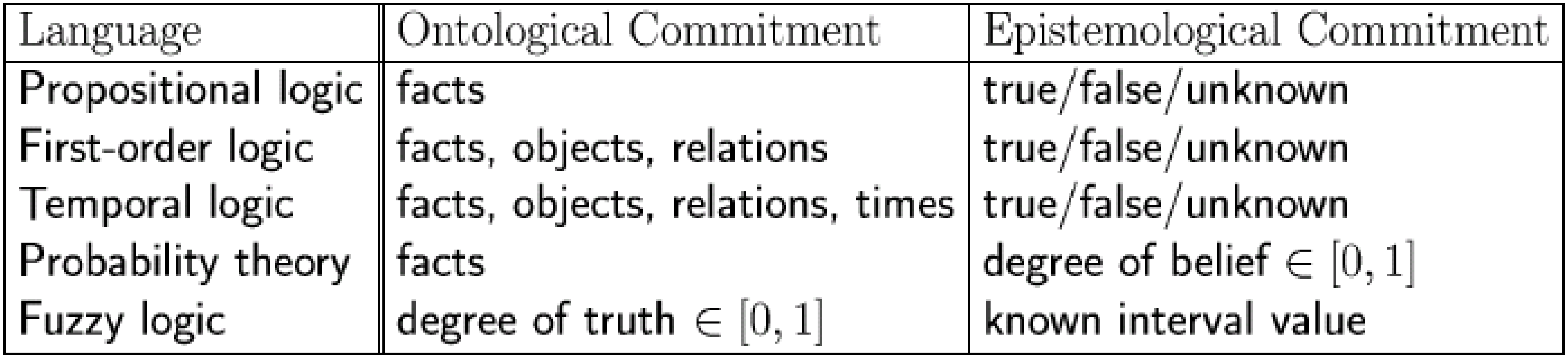
  – Relations: red, round, prime, brother of, bigger than, part of, comes between, …

  – Functions: father of, best friend, one more than, plus, …

Logics in General

- Ontological Commitment:
  - What exists in the world — TRUTH
  - PL : facts hold or do not hold.
  - FOL : objects with relations between them that hold or do not hold

- Epistemological Commitment:
  - What an agent believes about facts — BELIEF
- *Epistemology is the branch of philosophy concerned with the theory of knowledge. Epistemology is the study of the nature of knowledge, justification, and the rationality of belief.*



Language	Ontological Commitment	Epistemological Commitment
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	degree of truth $\in [0,1]$	known interval value

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 □, □, □, □, □
- Equality =
- Quantifiers □, □

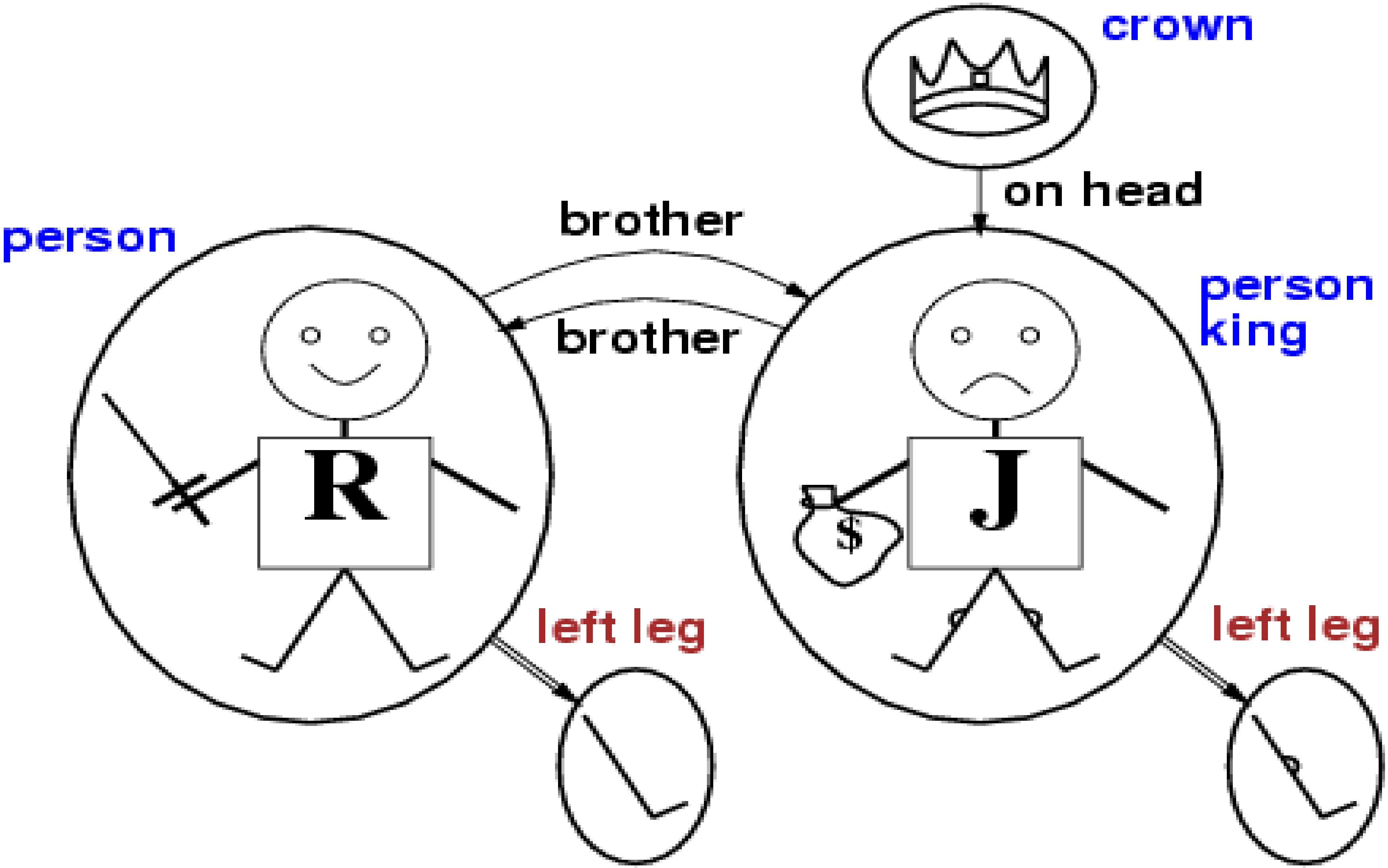
Relations

- 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).

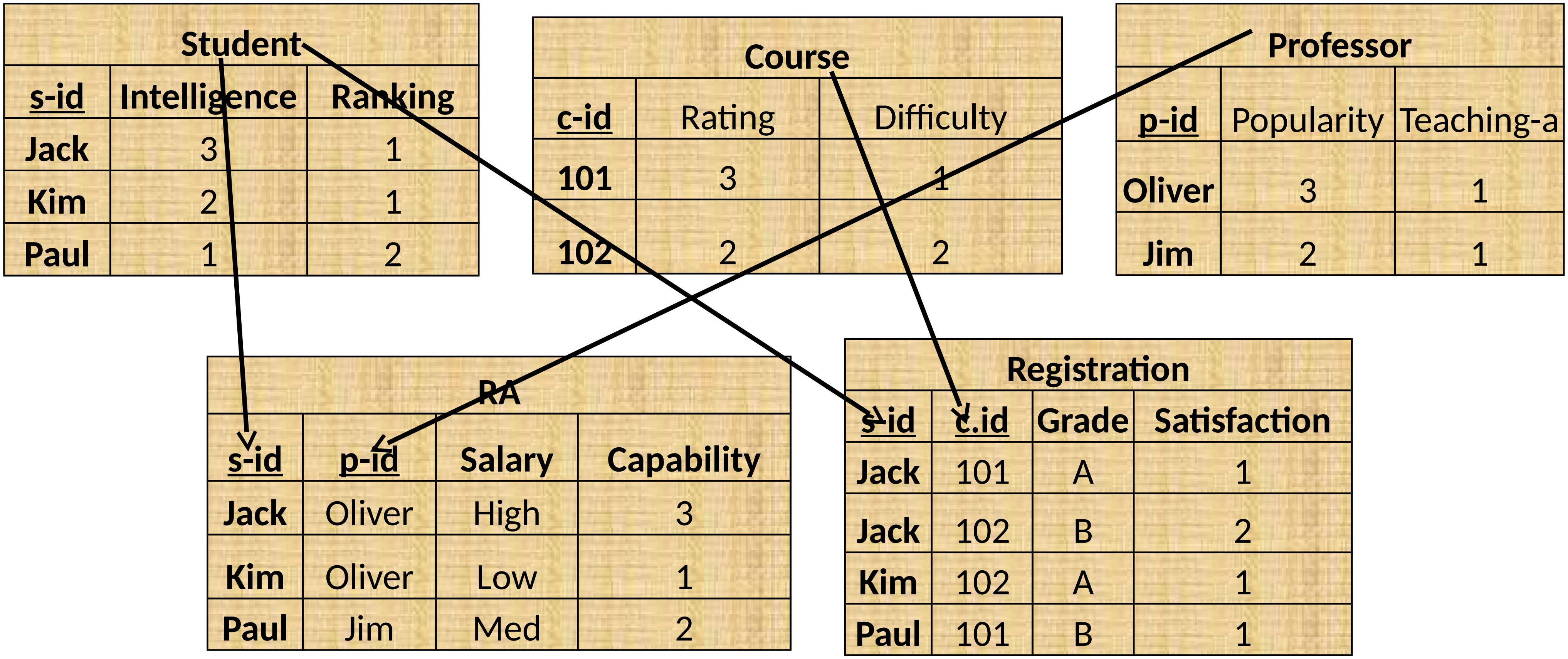
Models for FOL: Graphical Example



Tabular Representation

IBM ICE (Innovation Centre for Education)

- A FOL model is basically equivalent to a relational database instance.
- Historically, the relational data model comes from FOL.



Student

s-id	Intelligence	Ranking
Jack	3	1
Kim	2	1
Paul	1	2

Course

c-id	Rating	Difficulty
101	3	1
102	2	2

Professor

p-id	Popularity	Teaching-a
Oliver	3	1
Jim	2	1

RA

s-id	p-id	Salary	Capability
Jack	Oliver	High	3
Kim	Oliver	Low	1
Paul	Jim	Med	2

Registration

s-id	c.id	Grade	Satisfaction
Jack	101	A	1
Jack	102	B	2
Kim	102	A	1
Paul	101	B	1

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
- Functions can be nested:
  - Pat_Grandfather(x) = father(father(x))
- Terms can contain variables.
- No variables = **ground term**.

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.

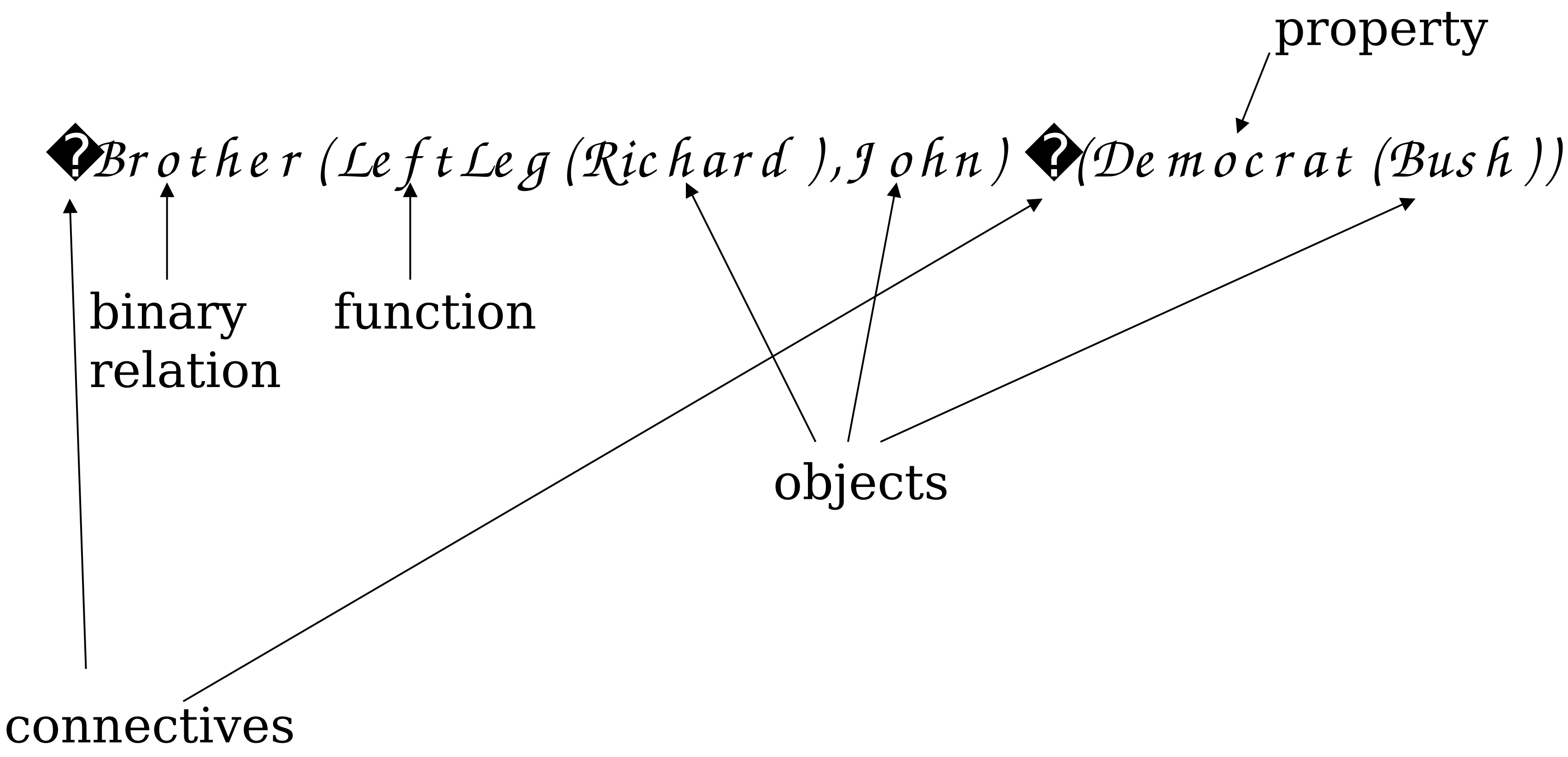
  Brother_of(Mary,Pete) is false.

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

- Note: Functions do not state facts and form no sentence.

Complex Sentences

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



More Examples

- Brother(Richard, John) □ Brother(John, Richard)
- King(Richard) □ King(John)
- King(John) => □ King(Richard)
- LessThan(Plus(1,2) ,4) □ 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

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.

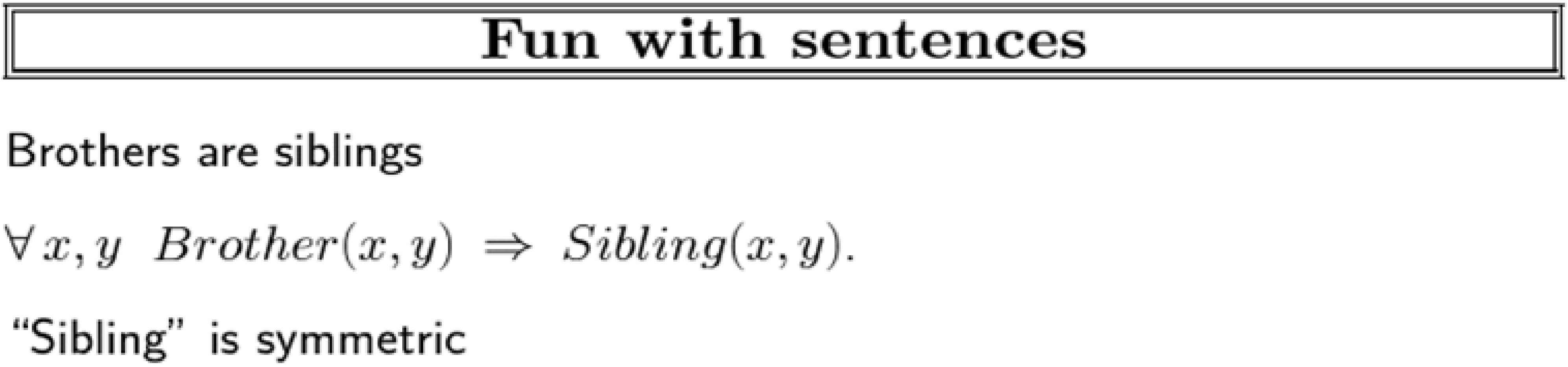
" $x\,[(x>2)$ � $(x>3)]\quad x$ �$\mathcal{R}\quad (false)$

$\$x\,[(x^{2}=-1)]\quad x$ �$\mathcal{R}\quad (false)$

There exists some real x whose square is minus 1.

Fun with sentences

Brothers are siblings

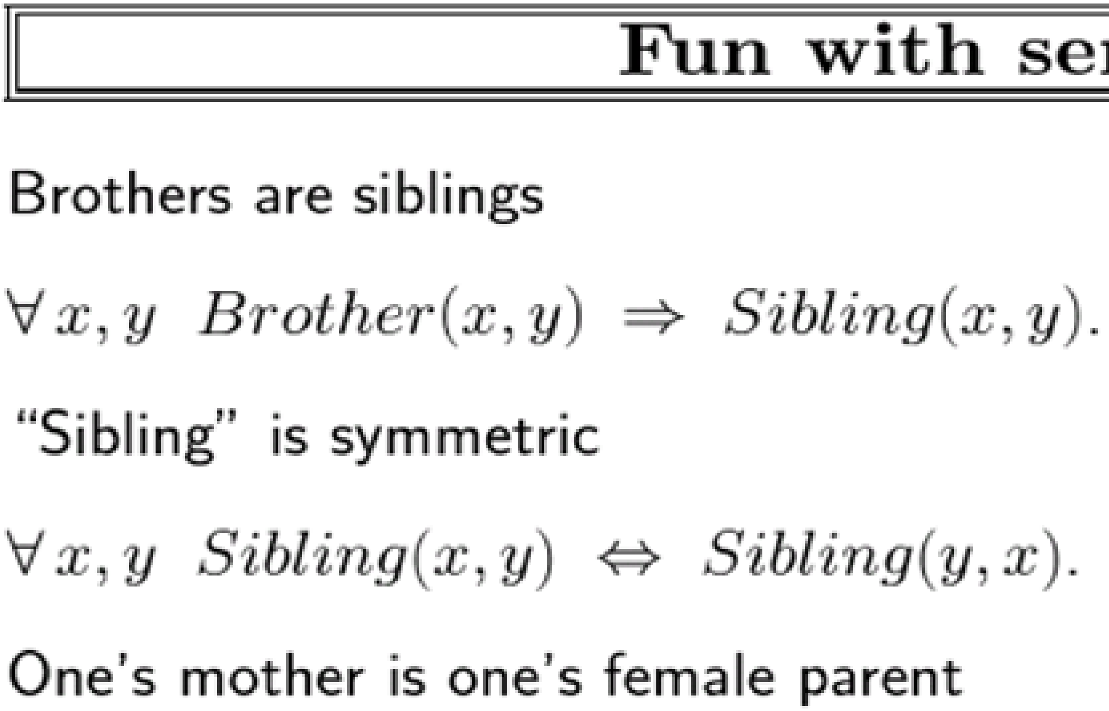


Fun with sentences

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

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

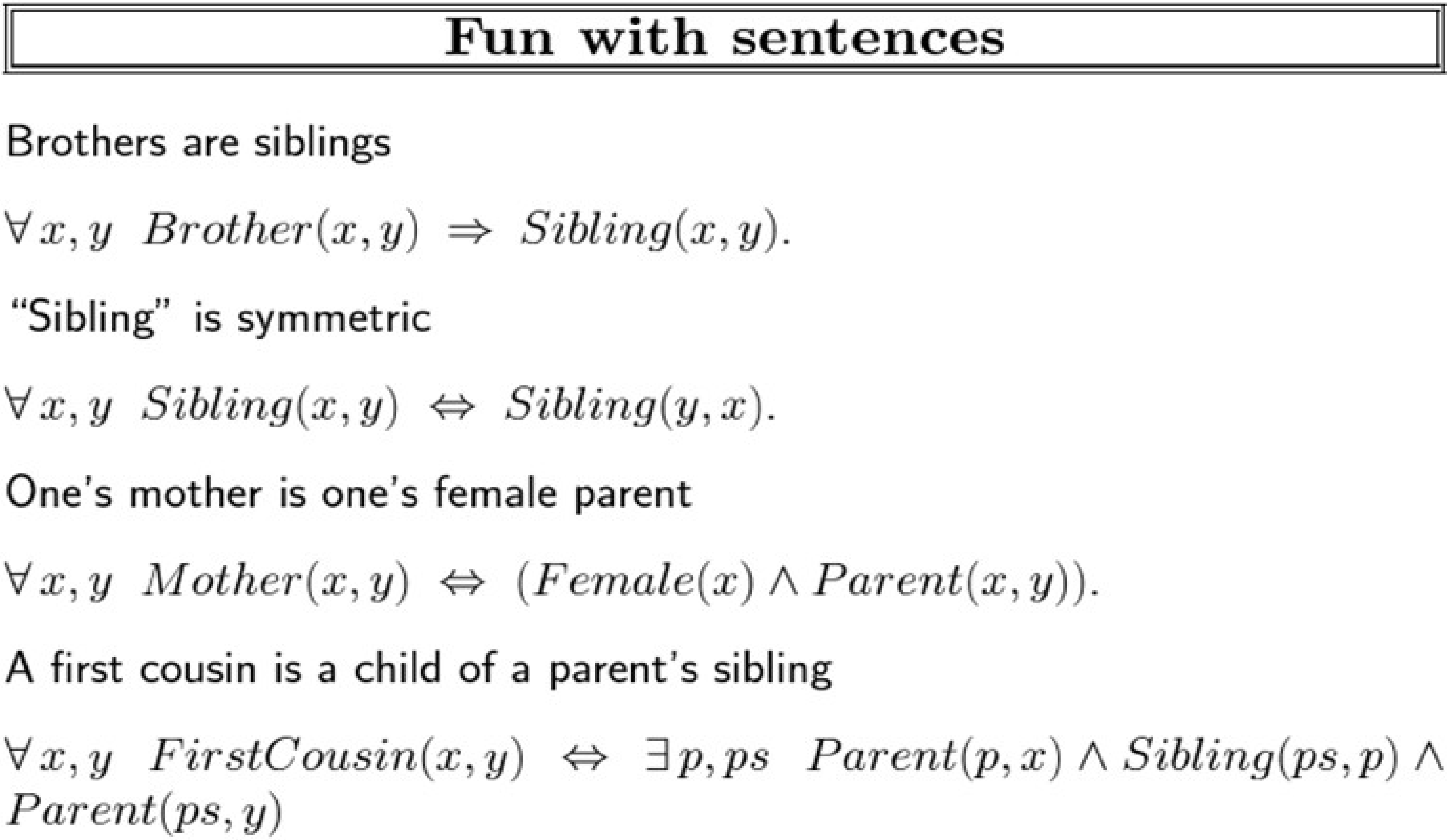
"Sibling" is symmetric

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

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

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

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

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

Combining Quantifiers

□ x □ y Loves(x,y)

- For everyone (“all x”) there is someone (“y”) that they love.

$ y □ x Loves(x,y)

- there is someone (“y”) who is loved by everyone

Clearer with parentheses: □ y (□ x   Loves(x,y))

De Morgan’s Law for Quantifiers

De Morgan’s Rule

P ��*Q*���(ث*P* *Q*)

P ��*Q*���(ں*P* *Q*)

��(*P*��*Q*ث)� *P* *Q*

��(*P*��*Q*ں)� *P* *Q*

Generalized De Morgan’s Rule

" *x* *P* ��$*x* (�*P*)

$*x* *P* ��" *x* (�*P*)

�" *x* *P* �$*x* (�*P*)

�$*x* *P* �" *x* (�*P*)

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

Using FOL

- We want to TELL things to the KB, e.g.
  TELL(KB, " *x, King(x)* � *Person(x)*)
  TELL(KB, *King(John)*)
  These sentences are assertions

- We also want to ASK things to the KB,
  ASK(KB,$*x, Person(x)*)

These are queries or goals

The KB should output x where Person(x) is true:

{x/John,x/Richard,...}

FOL Planning in Wumpus World

- Typical percept sentence:
  Percept([Stench, Breeze, Glitter, None, None], 5)

- Actions:
  Turn(Right), Turn(Left), Forward, Shoot, Grab, Release, Climb

- To determine best action, construct query:
  ∃ a BestAction(a,5).

- ASK solves this and returns {a/Grab}
  - And TELL about the action.

- Using logical queries is a general way to do planning.

Knowledge Base for Wumpus World

- Perception
  - □s,g,t Percept([s, Breeze,g],t) □ Breeze(t)
  - □s,b,t Percept([s,b,Glitter],t) □ Glitter(t)

- Reflex
  - □t Glitter(t) □ BestAction(Grab,t)

- Reflex with internal state
  - □t Glitter(t) □□Holding(Gold,t) □ BestAction(Grab,t)

  Holding(Gold,t) is not a percept: keep track of change.

Deducing hidden properties

Environment definition:

□x,y,a,b *Adjacent*([x,y],[a,b]) □ [a,b] □ {[x+1,y], [x-,y],[x,y+1],[x,y-1]}

Properties of locations:

□s,t *At*(Agent,s,t) □ Breeze(t) □ Breezy(s)

Location *s* and time *t*

Squares are breezy near a pit:

– Diagnostic rule---infer cause from effect

□s Breezy(s) □ □ r Adjacent(r,s) □ Pit(r)

– Causal rule---infer effect from cause.

□r Pit(r) □ [□s Adjacent(r,s) □ Breezy(s)]

Knowledge Engineering in FOL

Identify the task

Assemble the relevant knowledge

Decide on a vocabulary of predicates, functions, and constants

Encode general knowledge about the domain

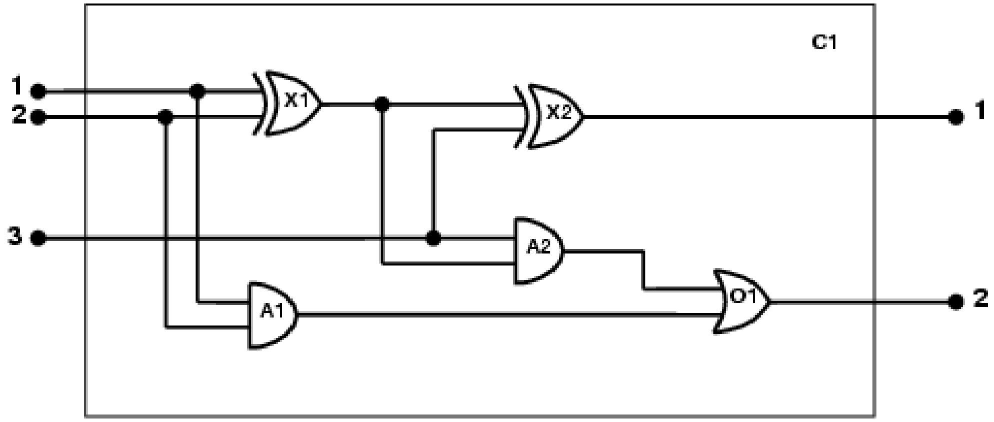
Encode a description of the specific problem instance

Pose queries to the inference procedure and get answers

- Debug the knowledge base.
- See text for full example: electric circuit knowledge base.

The electronic circuits domain

One-bit full adder



Possible queries:

- does the circuit function properly?
- what gates are connected to the first input terminal?
- what would happen if one of the gates is broken?

and so on

The electronic circuits domain

Identify the task

- Does the circuit actually add properly?
- Assemble the relevant knowledge
- Composed of wires and gates; Types of gates (AND, OR, XOR, NOT)
- Irrelevant: size, shape, color, cost of gates
- Decide on a vocabulary
- Alternatives:
- - Type(X1) = XOR (function)
  - Type(X_1, XOR)   (binary predicate)
  - XOR(X_1)
  - (unary predicate)

The electronic circuits domain

Encode general knowledge of the domain

- □t_1,t_2 Connected(t_1, t_2) □ Signal(t_1) = Signal(t_2)
- □t Signal(t) = 1 □ Signal(t) = 0
- 1 ≠ 0
- □t_1,t_2 Connected(t_1, t_2) □ Connected(t_2, t_1)
- □g Type(g) = OR □ Signal(Out(1,g)) = 1 □ □n Signal(In(n,g)) = 1
- □g Type(g) = AND □ Signal(Out(1,g)) = 0 □ □n Signal(In(n,g)) = 0
- □g Type(g) = XOR □ Signal(Out(1,g)) = 1 □ Signal(In(1,g)) ≠ Signal(In(2,g))
- □g Type(g) = NOT □ Signal(Out(1,g)) ≠ Signal(In(1,g))

The electronic circuits domain

Encode the specific problem instance

Type(X_1) = XOR

Type(A_1) = AND

Type(O_1) = OR

Type(X_2) = XOR

Type(A_2) = AND

Connected(Out(1,X_1),In(1,X_2))

Connected(Out(1,X_1),In(2,A_2))

Connected(Out(1,A_2),In(1,O_1))

Connected(Out(1,A_1),In(2,O_1))

Connected(Out(1,X_2),Out(1,C_1))

Connected(Out(1,O_1),Out(2,C_1))

Connected(In(1,C_1),In(1,X_1))

Connected(In(1,C_1),In(1,A_1))

Connected(In(2,C_1),In(2,X_1))

Connected(In(2,C_1),In(2,A_1))

Connected(In(3,C_1),In(2,X_2))

Connected(In(3,C_1),In(1,A_2))

The electronic circuits domain

Pose queries to the inference procedure

What are the possible sets of values of all the terminals for the adder circuit?

$\exists i_1,i_2,i_3,o_1,o_2\ \mathrm{Signal}(\mathrm{In}(1,C\_1)) = i_1 \wedge \mathrm{Signal}(\mathrm{In}(2,C_1)) = i_2 \wedge \mathrm{Signal}(\mathrm{In}(3,C_1)) = i_3 \wedge \mathrm{Signal}(\mathrm{Out}(1,C_1)) = o_1 \wedge \mathrm{Signal}(\mathrm{Out}(2,C_1)) = o_2$

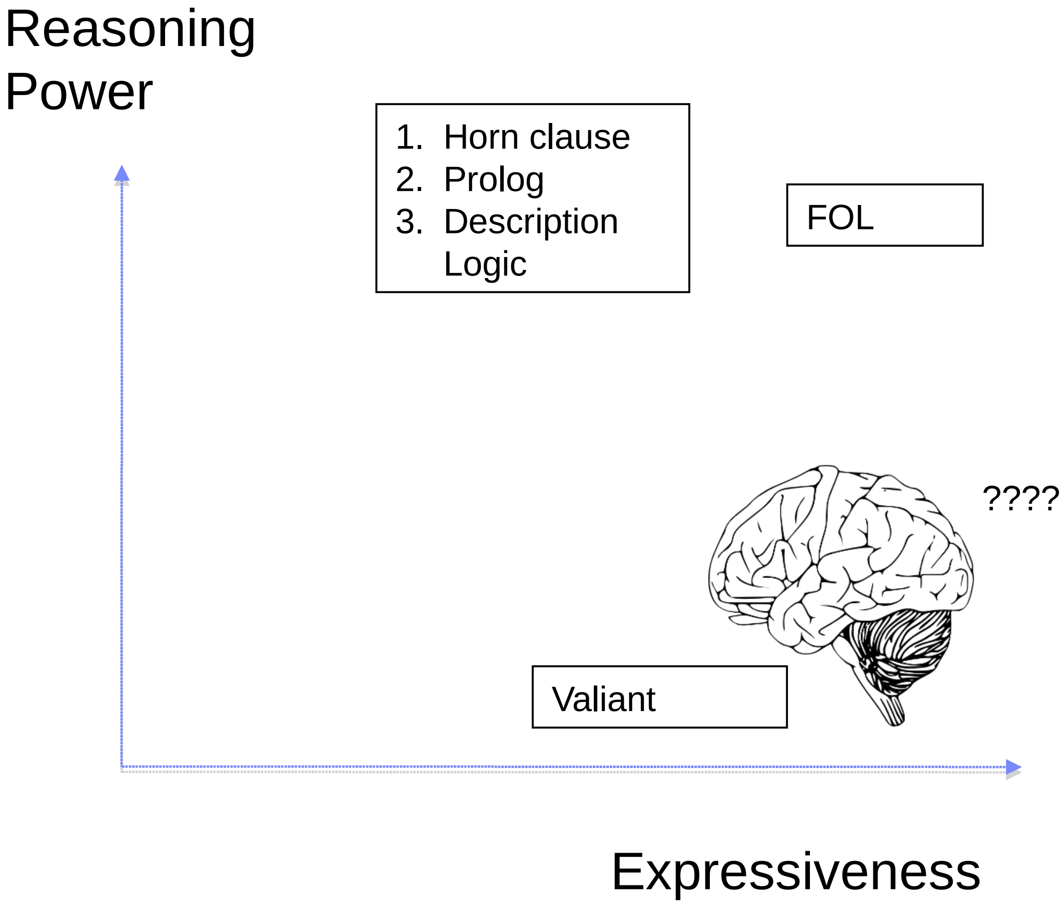
Debug the knowledge base

- May have omitted assertions like $1 \neq 0$.

Expressiveness vs. Tractability

- There is a fundamental trade-off between **expressiveness** and **tractability** in Artificial Intelligence.

- Similar, even more difficult issues with probabilistic reasoning (later).



Reasoning Power

1. Horn clause
2. Prolog
3. Description Logic

FOL

????

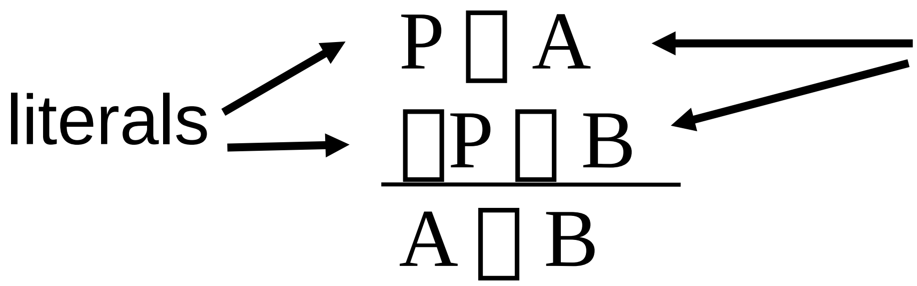
Valiant

Expressiveness

Summary

- First-order logic:
  - Much more expressive than propositional logic
  - Allows objects and relations as semantic primitives
  - Universal and existential quantifiers
  - syntax: constants, functions, predicates, equality, quantifiers
- Knowledge engineering using FOL
  - Capturing domain knowledge in logical form

Resolution Rule



clauses: A and B will always be disjunctions of literals or just a literal or possibly missing.

This says: In two disjunctive clauses, if we have complementary literals, we can discard them and “OR” the remaining clauses.

Resolution is Valid

P	A	B	(P □ A)	□	(□P	□ B)	□	A □ B
T	T	T	T	T	F	T	T	T
T	T	F	T	F	F	F	T	T
T	F	T	T	T	F	T	T	T
T	F	F	T	F	F	F	T	F
F	T	T	T	T	T	T	T	T
F	T	F	T	T	T	T	T	T
F	F	T	F	F	T	T	T	T
F	F	F	F	F	T	T	T	F

Resolution Subsumes “the big 3” Inference Rules

Modus ponens

$$\begin{array}{l} P \\ P \square Q \\ \hline Q \end{array}$$

$$\begin{array}{l} P \square F \\ \square P \square Q \\ \hline Q \square F \end{array}$$

Modus tollens

$$\begin{array}{l} \square P \\ Q \square P \\ \hline \square Q \end{array}$$

$$\begin{array}{l} \square P \square F \\ \square Q \square P \\ \hline \square Q \square F \end{array}$$

Hypothetical syllogism

$$\begin{array}{l} P \square Q \\ Q \square R \\ \hline P \square R \end{array}$$

$$\begin{array}{l} \square P \square Q \\ \square Q \square R \\ \hline \square P \square R \end{array}$$

We now have one rule to rule them all!

Example – 1

If P □ Q, Q □ R, P then R.

1. Negate the conclusion (□R becomes another premise).
2. Convert to CNF: (□P □ Q) □ (□Q □ R) □ P □ □R
3. Write the premises as the first lines of the proof.
4. Do resolution.

1.	□P □ Q	premise
2.	□Q □ R	premise
3.	P	premise
4.	□R	premise
5.	□P □ R	resolution 1,2
6.	R	resolution 3,5
7.		resolution 4,6

} Sometimes called the support.

empty = F

Example – 2

If P □ (Q □ R), □R □ Q then □P.

1. Negate conclusion: □□P □ P
2. Convert to CNF: (□P □ Q) □ (□P □ R) □ □R □ Q □ P

1.	□P □ Q	premise (not used □ could discard)
2.	□P □ R	premise
3.	□R	premise
4.	Q	premise (not used □ could discard)
5.	P	premise
6.	R	resolution 2,5
7.	F	resolution 3,6

Also, resolution 1,5 yields Q, which need not be added to the derivation □ already there.

Do we always need to use all the premises? If not, we can discard them from the statement to be proved.

Example – 3

If P □ Q, Q □ P, P □ Q then P □ Q.

1. Negate conclusion: □(P □ Q) □ (□P □ □Q)
2. CNF: (P □ Q) □ (□Q □ P) □ (□P □ Q) □ (□P □ □Q)

1.	P □ Q	premise
2.	□Q □ P	premise
3.	□P □ Q	premise
4.	□P □ □Q	premise
5.	P	resolution 1,2 (idemp. P □ P □ P)
6.	Q	resolution 3,5
7.	□Q	resolution 4,5
8.	F	resolution 6,7

Example – 4

If (P □ Q) □ (P □ R), P then Q □ R.

1. Negate conclusion: □(Q □ R) □ (□Q □ □R)
2. CNF: ((□P □ Q) □ (□P □ R)) □ P □ (□Q □ □R)
   □ (□P □ Q □ R) □ P □ □Q □ □R

1. □P □ Q □ R premise
2. P premise
3. □Q premise
4. □R premise
5. Q □ R resolution 1,2
6. R resolution 3,5
7. resolution 4,6

Resolution Algorithm

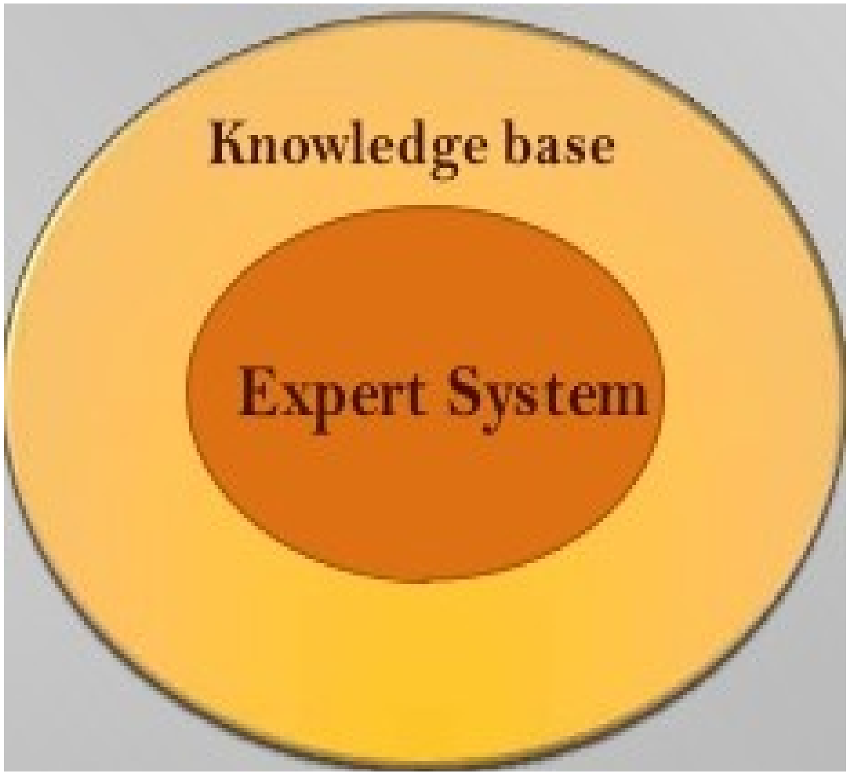
- Resolution works by using the principle of proof by contradiction.
- Negate the conclusion so as to find the conclusion.
- Apply the resolution rule to the resulting clauses.
- Each clause contains complementary literals.
- They are resolved and produce 2 new clause and they are be added to the set of facts if they are not already exist.
- This process continues until any one of the following occur:
  - No more new clauses that can be added
  - An application of the resolution rule derives the empty clause
- An empty clause shows that the negation of the conclusion is a complete contradiction, hence the negation of the conclusion is invalid or false or the assertion is completely valid or true.

Knowledgebase Systems

- A knowledgebase system is a program that uses AI to solve problems within a specialized domain that ordinarily requires human expertise.

- Typical tasks of expert systems include classification, diagnosis, monitoring, design, scheduling and planning for specialized tasks.

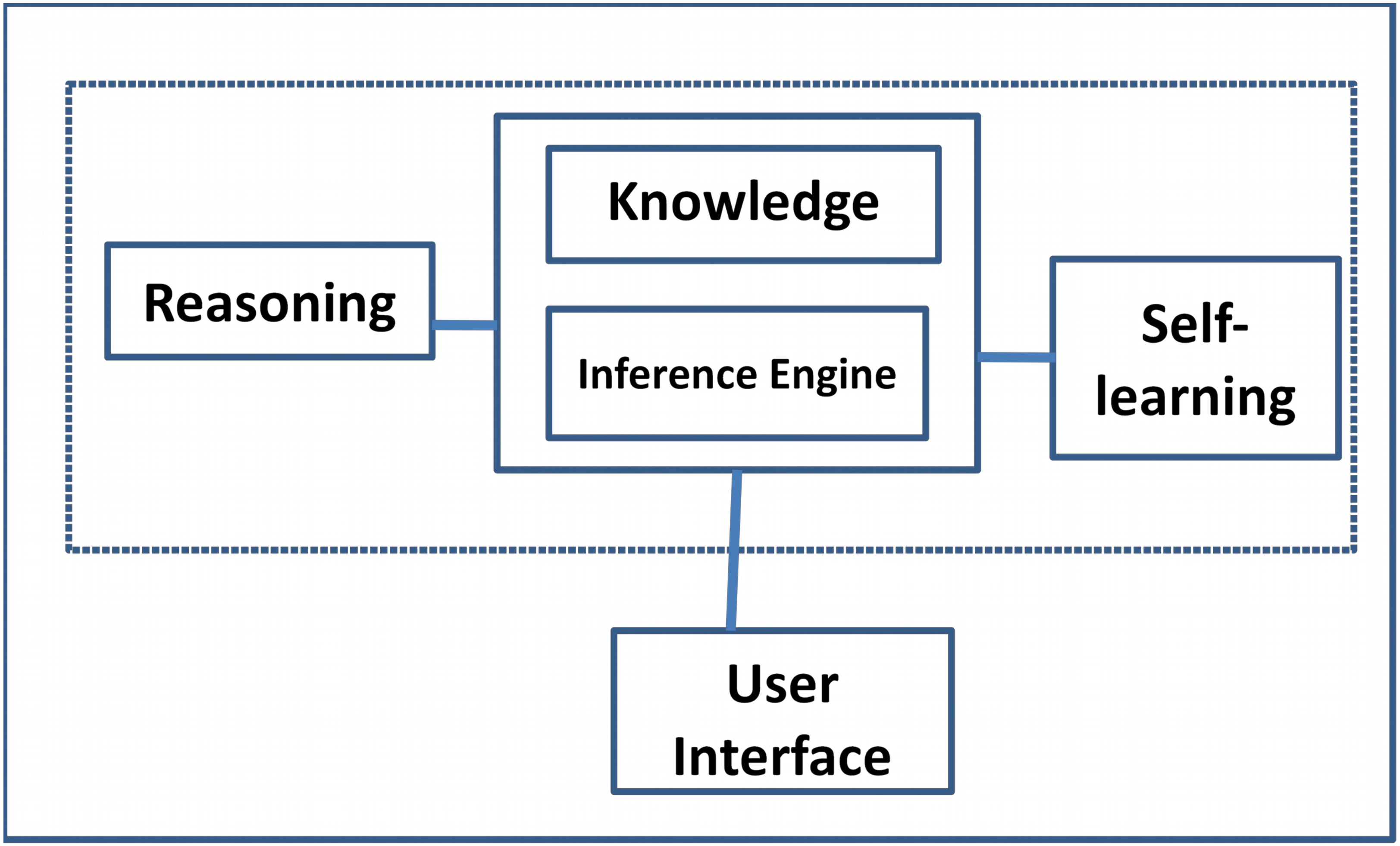
- Knowledgebase is a more general than expert system



Characteristics of Knowledge Based Systems

- Expansion of Knowledge Base for the domain
- Aid for Heuristics Analysis
- It makes use of Search Techniques.
- It is having ability to induce new knowledge from existing knowledge.
- Symbol processing is followed in these systems
- It can explain the reasoning.

Structure of a Knowledge Based System.



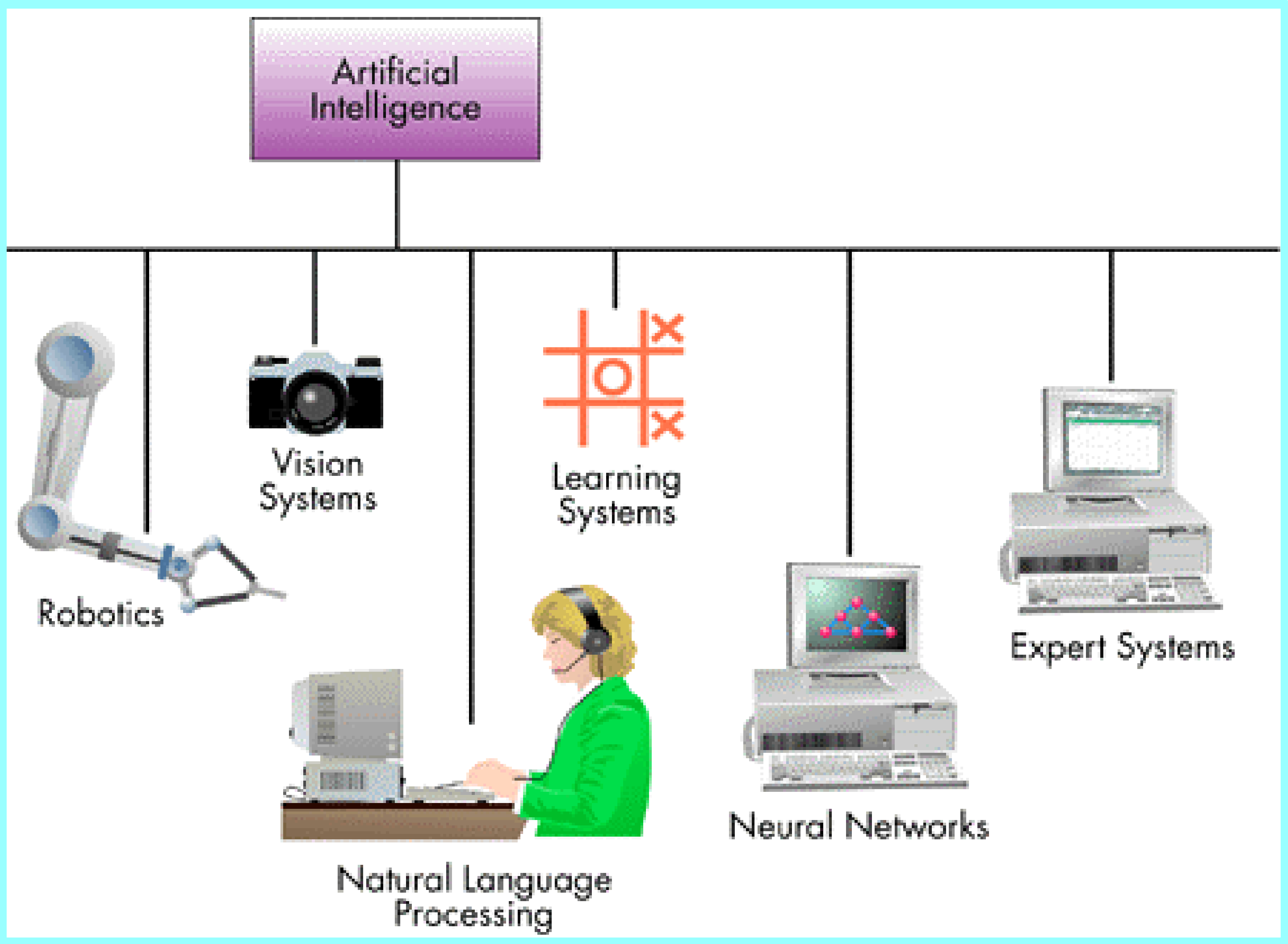
Recall: Artificial Intelligence

- Artificial Intelligence (AI)
  - Computers with the ability to mimic or duplicate the functions of the human brain.

- Artificial Intelligence Systems
  - The people, procedures, hardware, software, data, and knowledge needed to develop computer systems and machines that demonstrate the characteristics of intelligence.

- Intelligent Behavior
  - Learn from experience, Apply knowledge acquired from experience, Handle complex situations, Solve problems when important information is missing, Determine what is important, React quickly and correctly to a new situation, Understand visual images, Process and manipulate symbols, Be creative and imaginative, Use heuristics

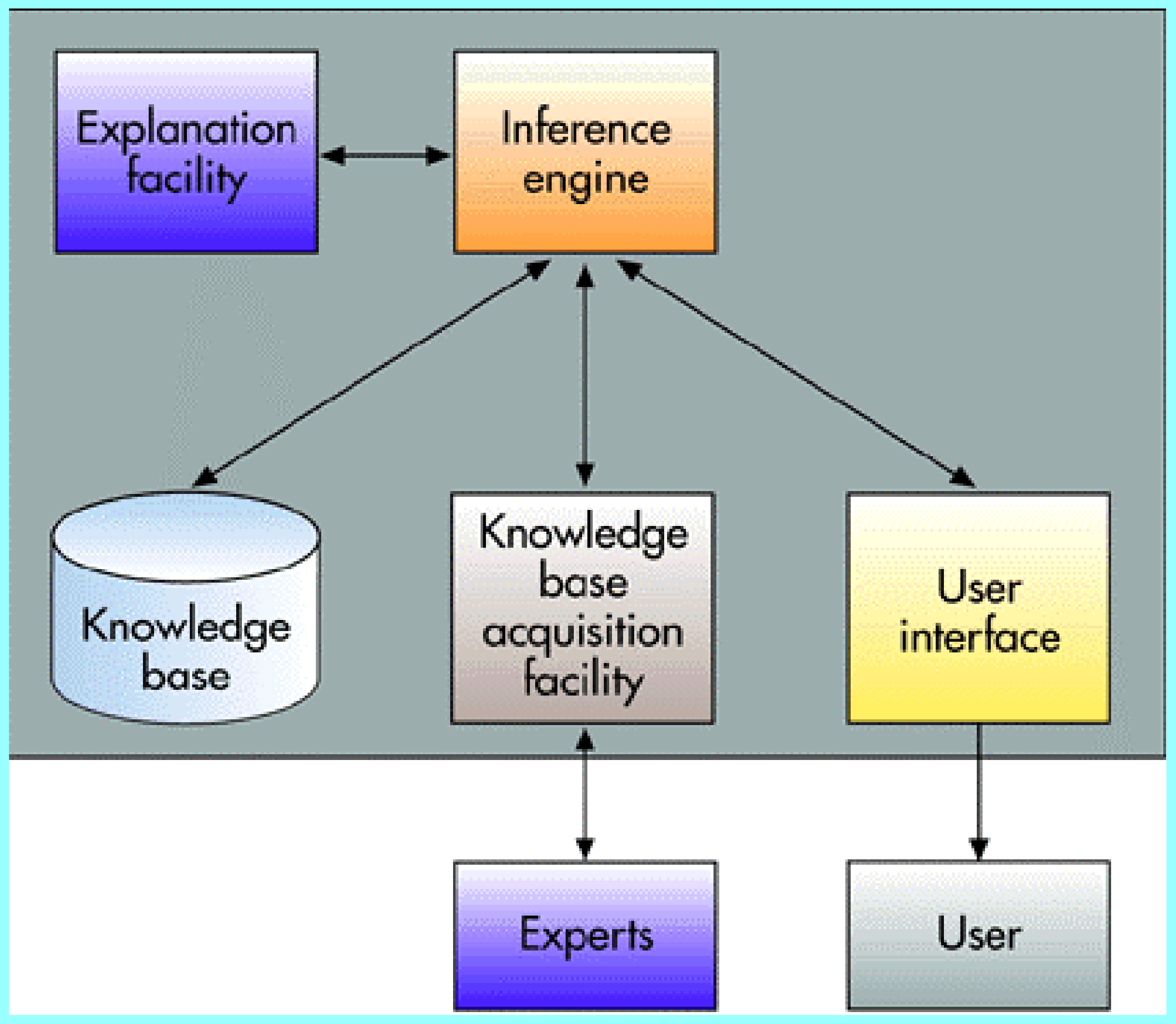
Major Branches of Artificial Intelligence



Overview of Expert Systems

- Can explain their reasoning or suggested decisions.
- Can display intelligent behavior.
- Can draw conclusions from complex relationships.
- Can provide portable knowledge.
- Expert Systems – Limitations
  - Not widely used or tested.
  - Limited to relatively narrow problems.
  - Cannot readily deal with “mixed” knowledge.
  - Possibility of error.
  - Cannot refine own knowledge base.
  - Difficult to maintain.
  - May have high development costs.
  - Raise legal and ethical concerns.

Components of Expert Systems



Rules for a Credit Application

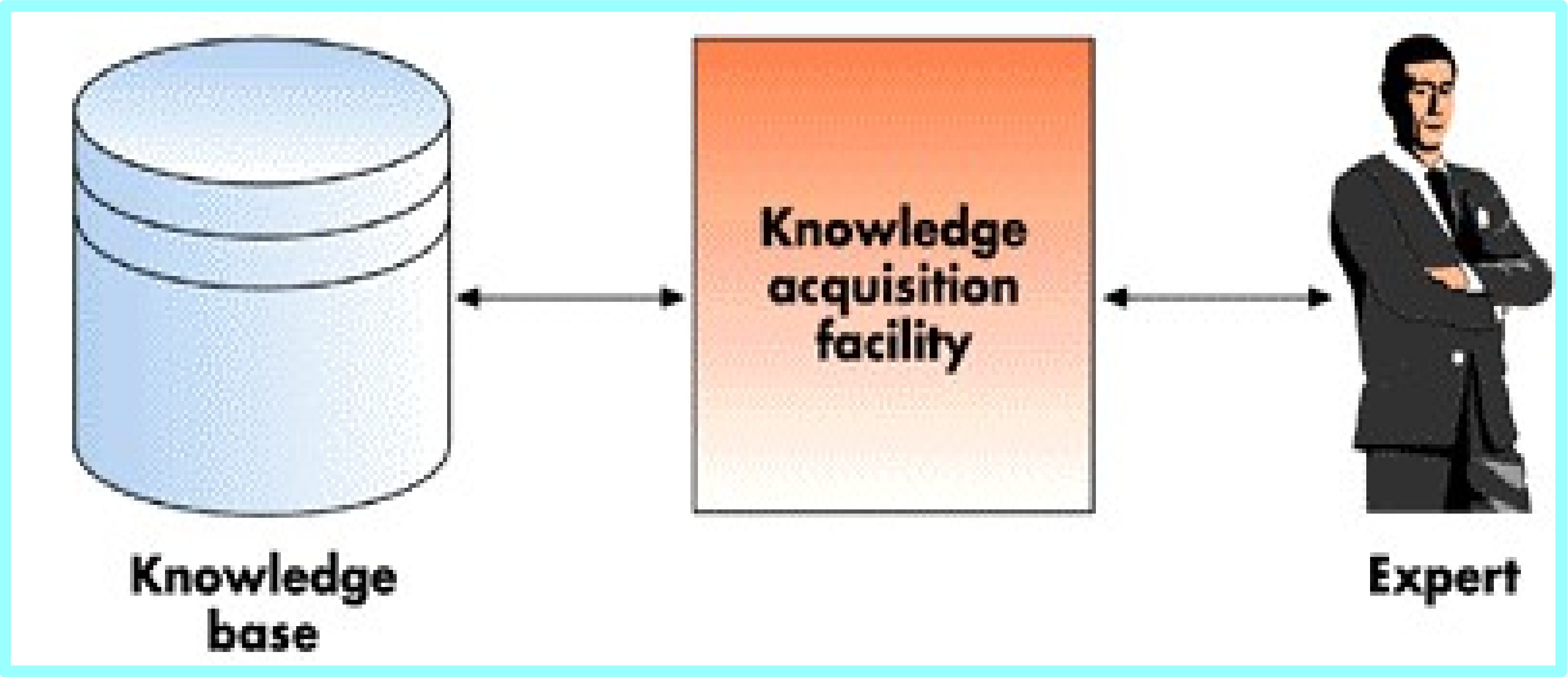
Car Loan application for an amount of Rs. 10,00,000 to Rs. 20,00,000

- If there are no previous credits problems, and
- If month net income is greater than 4x monthly loan payment, and
- If down payment is 15% of total value of property, and
- If net income of borrower is > Rs. 50,000, and
- If employment is > 3 years at same company

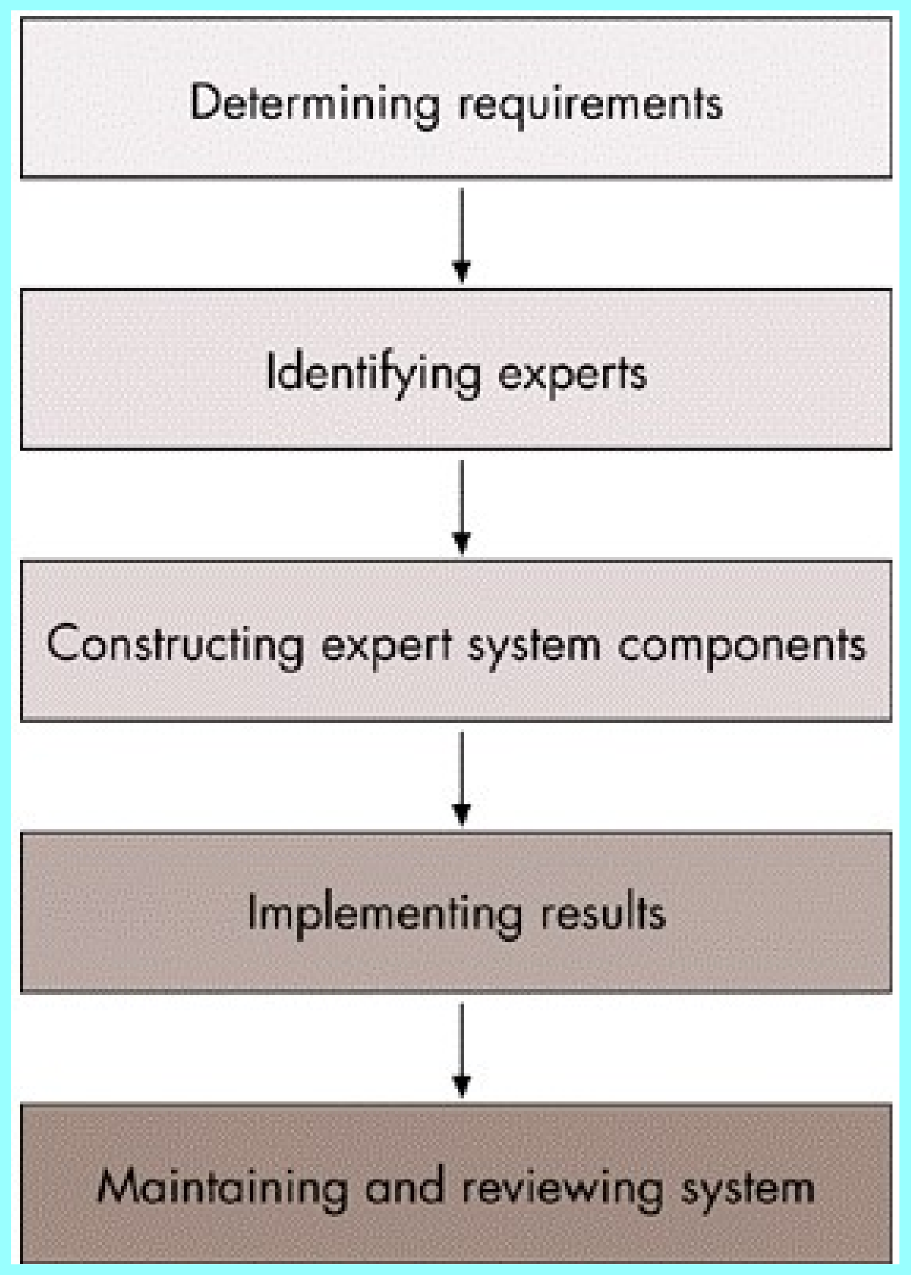
⟹ Then accept the applications

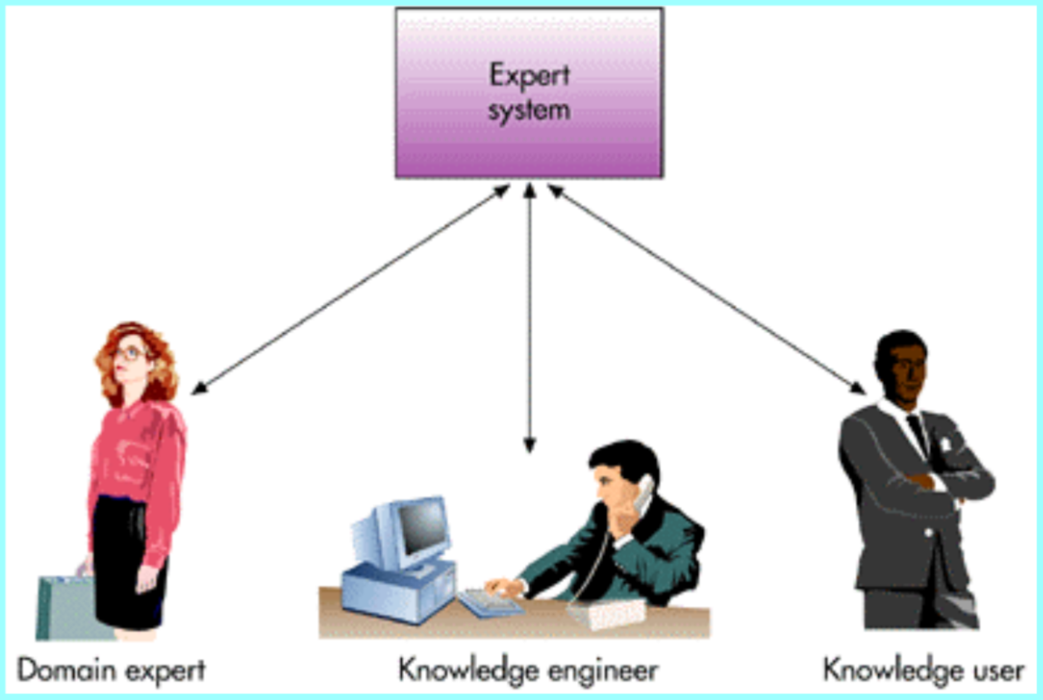
⟹ Else check other credit rules

Knowledge Acquisition Facility



Expert Systems Development


Determining requirements
Identifying experts
Constructing expert system components
Implementing results
Maintaining and reviewing system



Applications of Expert Systems

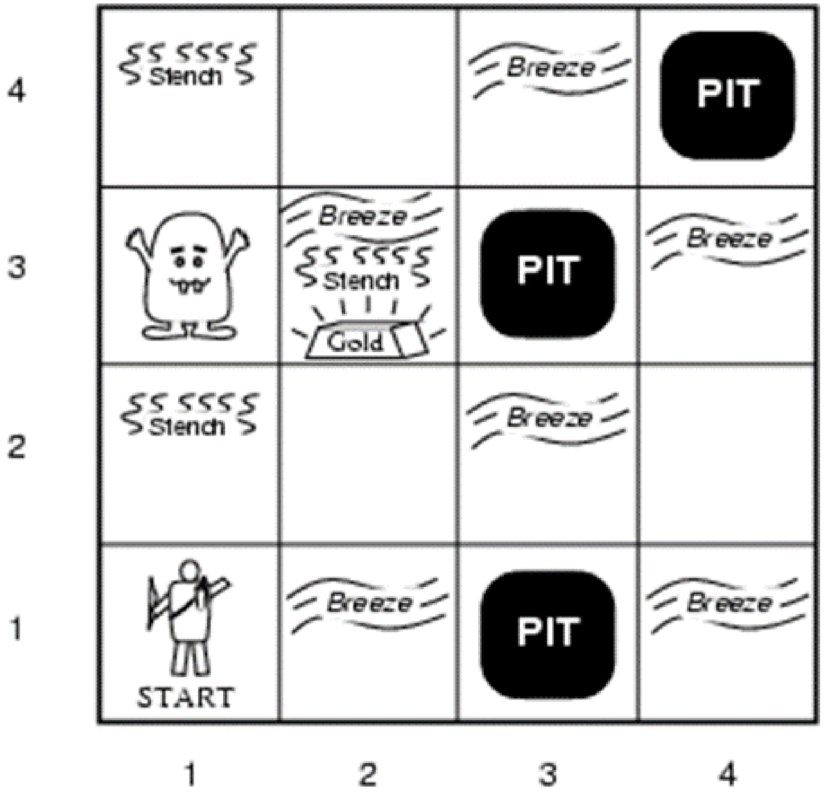
- Credit granting
- Information management and retrieval
- AI and expert systems embedded in products
- Plant layout
- Hospitals and medical facilities
- Help desks and assistance
- Employee performance evaluation
- Loan analysis
- Virus detection
- Repair and maintenance
- Shipping
- Marketing
- Warehouse optimization

Wumpus World

- Performance measure
  - gold +1000,
  - death -1000

    (falling into a pit or being eaten by the wumpus)
  - -1 per step, -10 for using the arrow
- Environment
  - Squares adjacent to wumpus are smelly
  - Squares adjacent to pit are breezy
  - Glitter iff gold is in the same square
  - Shooting kills wumpus if you are facing it
  - Shooting uses up the only arrow
  - Grabbing picks up gold if in same square
  - Releasing drops the gold in same square
- Sensors: Stench, Breeze, Glitter, Bump, Scream
- Actuators: Left turn, Right turn, Forward,

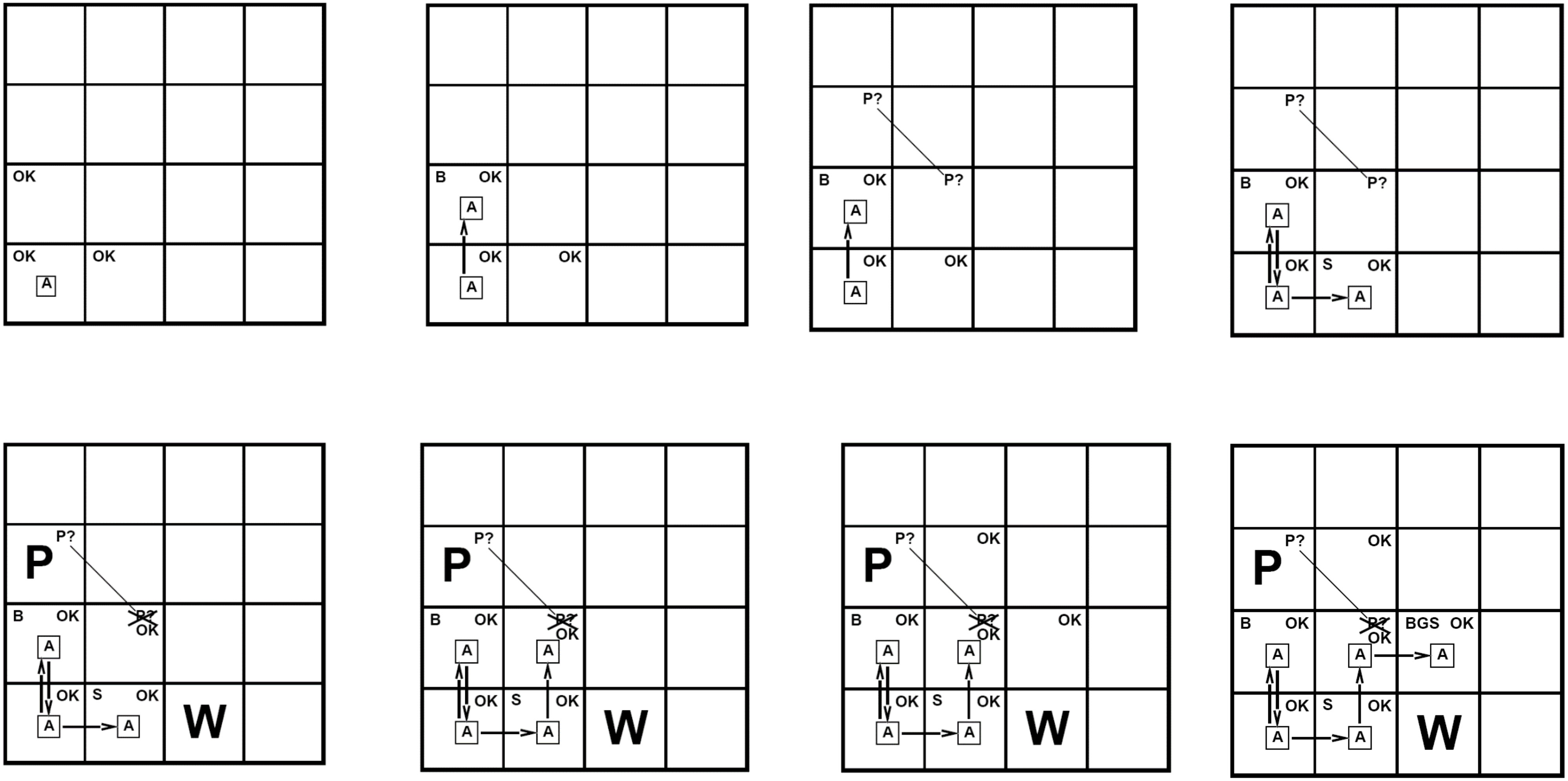
  Grab, Release, Shoot



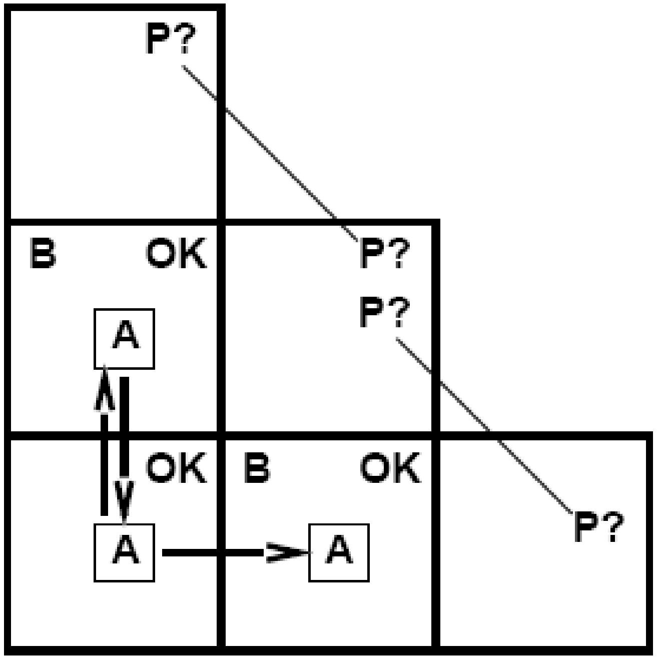
Wumpus World

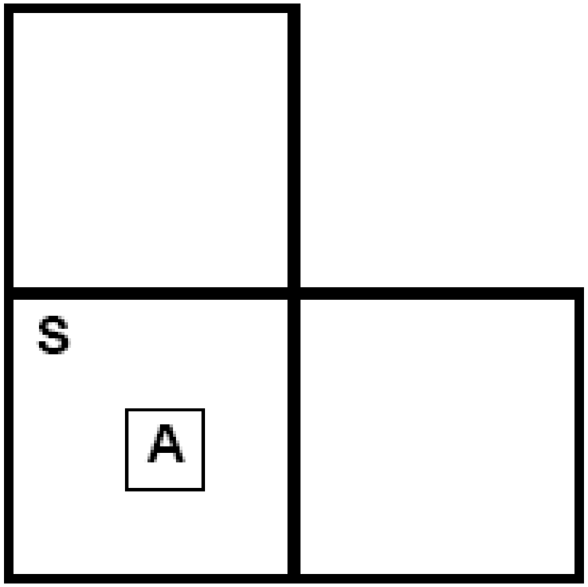
- Characterization of Wumpus World
  - Observable
    - partial, only local perception
  - Deterministic
    - Yes, outcomes are specified
  - Episodic
    - No, sequential at the level of actions
  - Static
    - Yes, Wumpus and pits do not move
  - Discrete
    - Yes
  - Single Agent
    - Yes

Wumpus World



Other Sticky Situations





- Breeze in (1,2) and (2,1)
  - – No safe actions
- Smell in (1,1)
  - – Cannot move

Recall - Logic

- Knowledge bases consist of sentences in a formal language
  - Syntax
    - Sentences are well formed
  - Semantics
    - The “meaning” of the sentence
    - ***The truth of each sentence with respect to each possible world (model)***

Example:

x + 2 >= y is a sentence

x2 + y > is not a sentence

x + 2 >= y is true iff x + 2 is no less than y

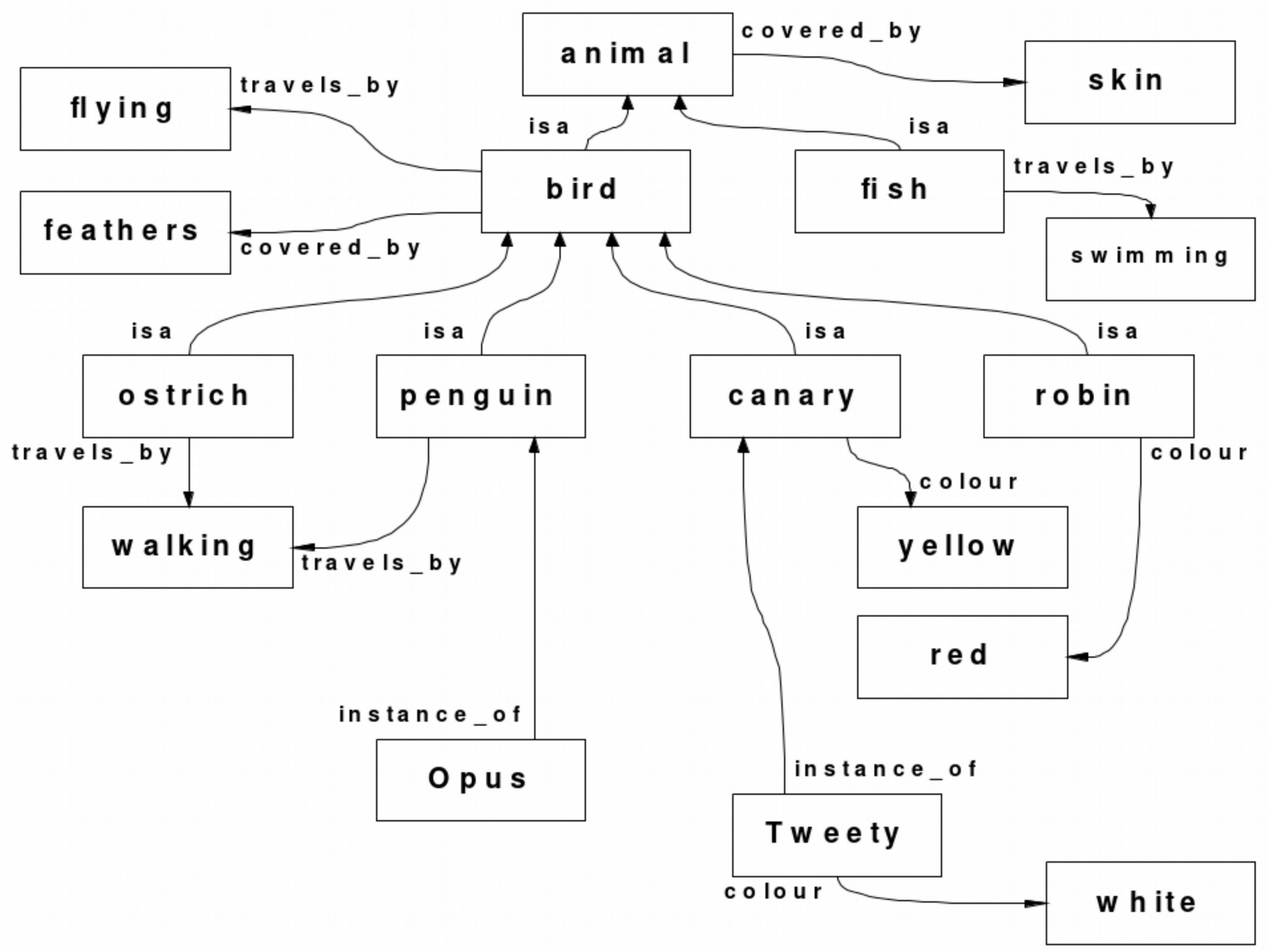
x + 2 >= y is true in a world where x = 7, y=1

x + 2 >= y is false in world where x = 0, y =6

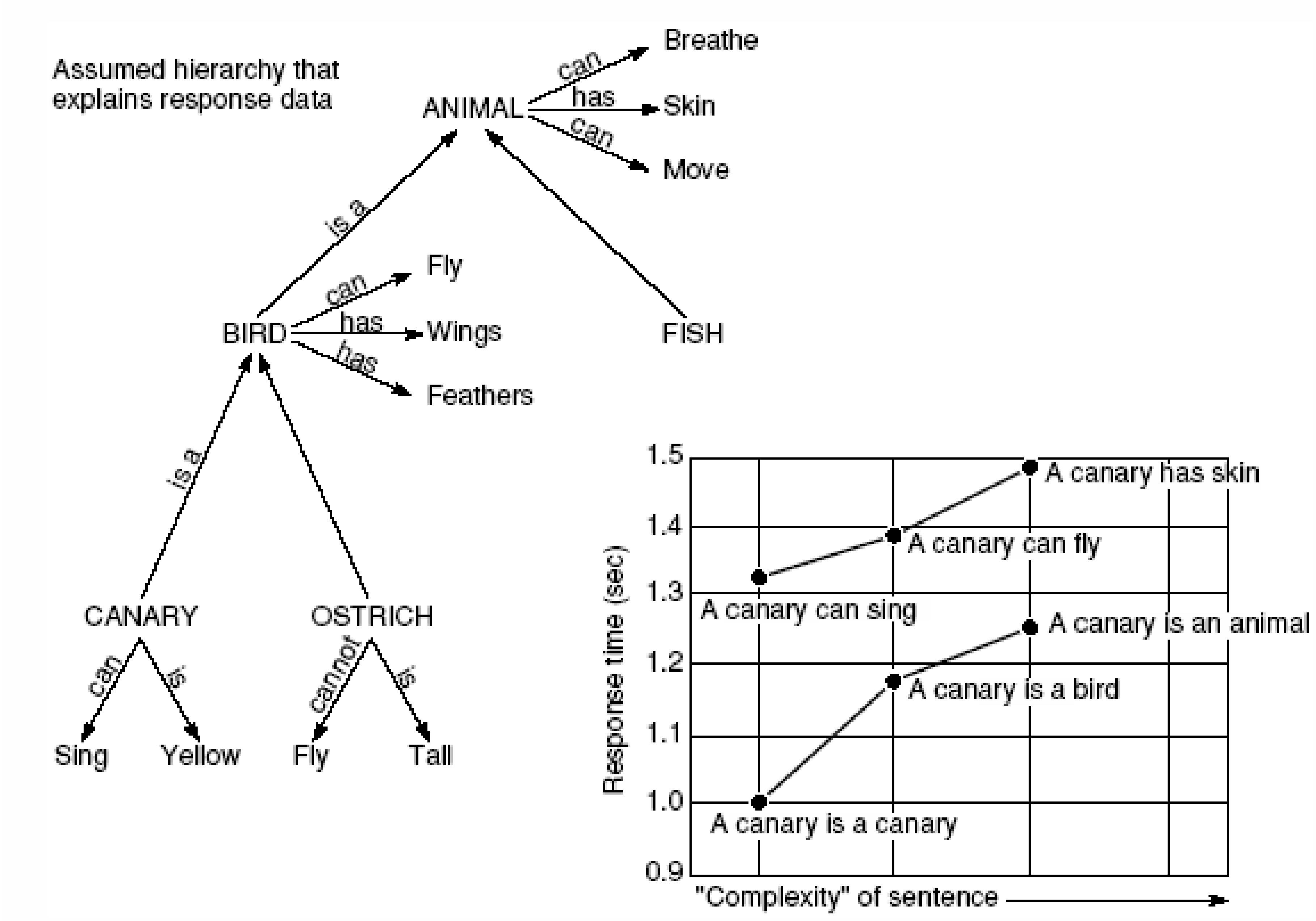
Semantic Net

- A semantic net is a labeled directed graph, where each node represents an object (a proposition), and each link represents a relationship between two objects.
- Semantic nets represent propositional information.
- Relations between propositions are of primary interest because they provide the basic structure for organizing knowledge.
- Some important relations are:
  - "IS-A" (is an instance of). Refers to a member of a **class,** where a class is a group of objects with one or more common attributes (properties). For example, "Tom IS-A bird".
  - "A-KIND-OF". Relates one class to another, for example "Birds are A-KIND-OF animals".
  - "HAS-A". Relates attributes to objects, for example "Mary HAS-A cat".
  - "CAUSE". Expresses a causal relationship, for example "Fire CAUSES smoke".

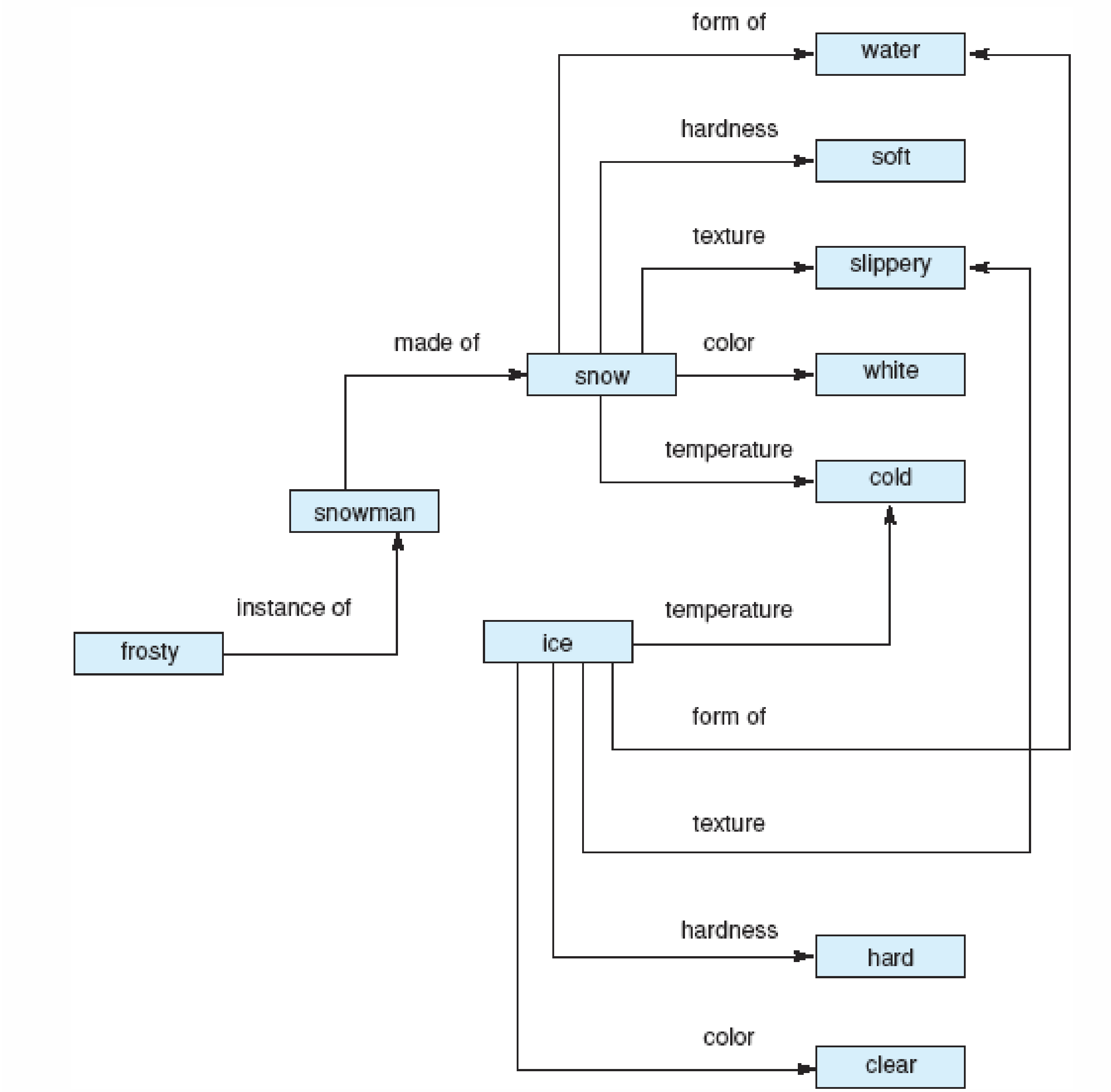
Example – 1


animal
covered_by
skin
flying
travels_by
isa
isa
bird
fish
travels_by
feathers
covered_by
swimming
isa
isa
isa
isa
ostrich
penguin
canary
robin
travels_by
colour
colour
walking
travels_by
yellow
red
instance_of
Opus
instance_of
Tweety
colour
white

Example – 2

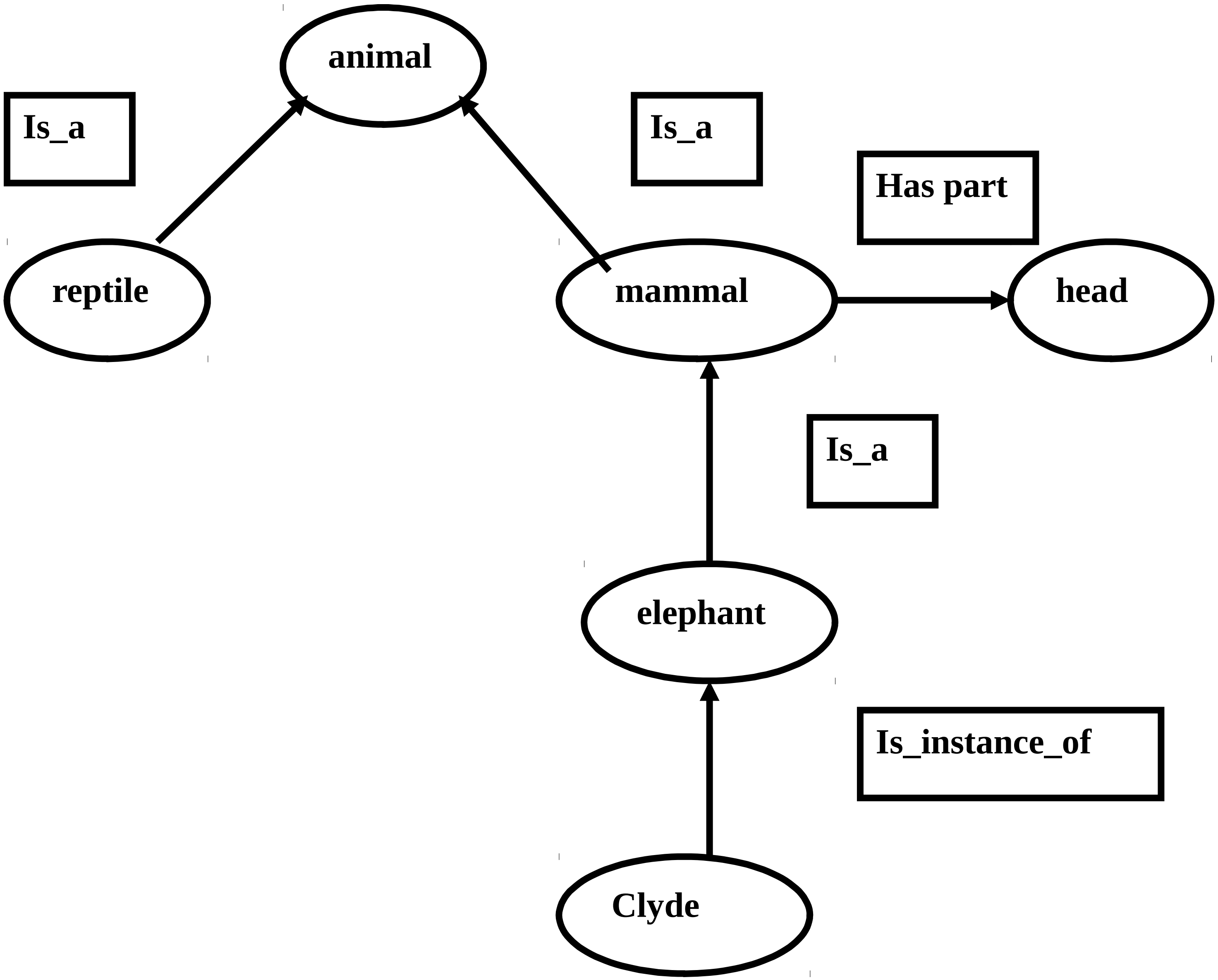

Assumed hierarchy that explains response data
ANIMAL
can
Breathe
has
Skin
can
Move
is a
BIRD
can
Fly
has
Wings
has
Feathers
FISH
is a
CANARY
can
Sing
is
Yellow
OSTRICH
cannot
Fly
is
Tall
Response time (sec)
1.5
1.4
1.3
1.2
1.1
1.0
0.9
A canary has skin
A canary can fly
A canary can sing
A canary is an animal
A canary is a bird
A canary is a canary
"Complexity" of sentence

Example – 3



A Semantic Network Representation of Properties of Snow and Ice

Example – 4


animal
Is_a
Is_a
Has part
reptile
mammal
head
Is_a
elephant
Is_instance_of
Clyde

Inference in Semantic Networks

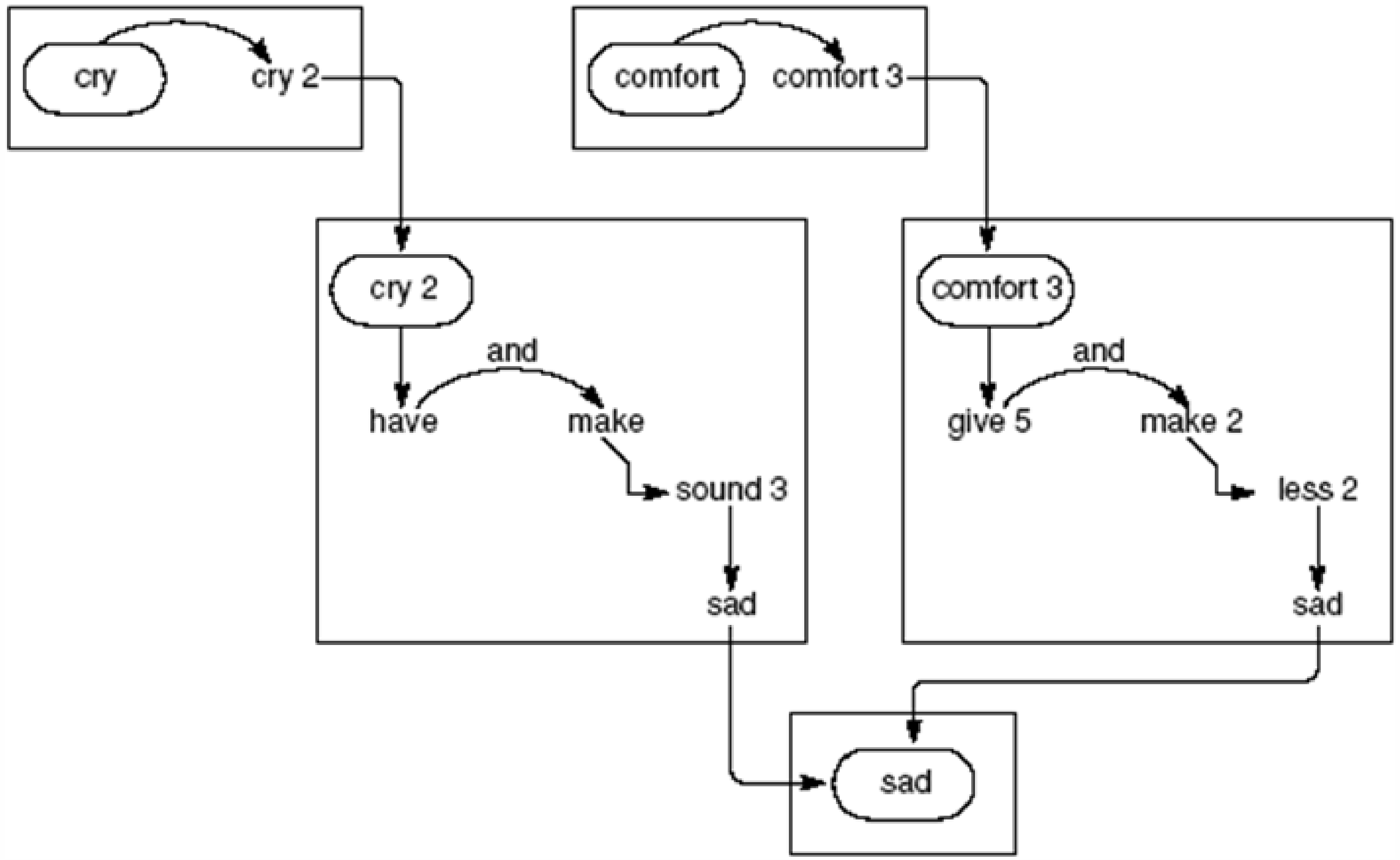
- The inference procedure in semantic nets is called **inheritance**, and it allows one node's characteristics to be duplicated by a descendent node.

- <u>Example:</u>
- Consider a class "aircraft", and assume that "balloons", "propeller-driven objects" and "jets" are subclasses of it, i.e.
  - "Balloons are A-KIND-OF aircrafts"
  - "Propeller-driven objects are A-KIND-OF aircrafts", etc.
- Assume that the following attributes are assigned to aircrafts: "Aircraft IS-A flying object", "Aircraft HAS-A wings", "Aircraft HAS-A engines"

All properties assigned to the superclass, "aircraft", will be inherited by its subclasses, unless there is an "exception" link capturing a non-monotonic inference relation.

Inference in Semantic Networks

- Find relationships between pairs of words
  - Search graphs outward from each word in a breath-first fashion
  - Search for a common concept or intersection node
  - The path between the two given words passing by this intersection node is the relationship being looked for



Semantic Networks: Types and Components

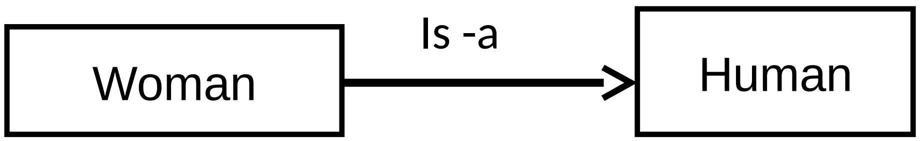
Six types of semantic networks are:

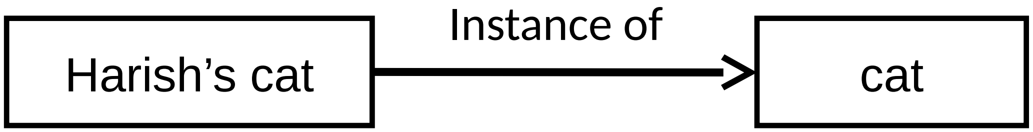
- Definitional network
- Assertional network
- Implicational network
- Executable network
- Learning network
- Hybrid network

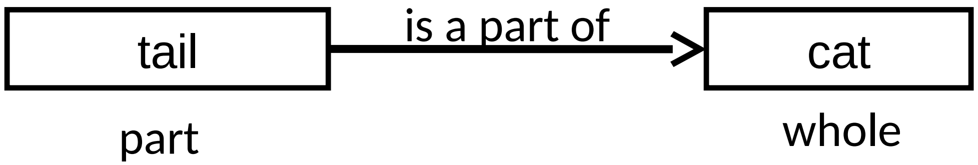
Semantic network components

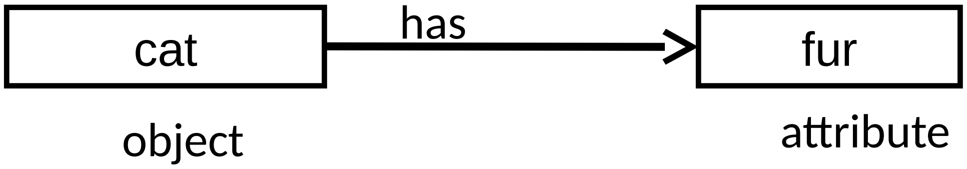
- Lexical component
- Structural components
- Semantic component
- Procedural part

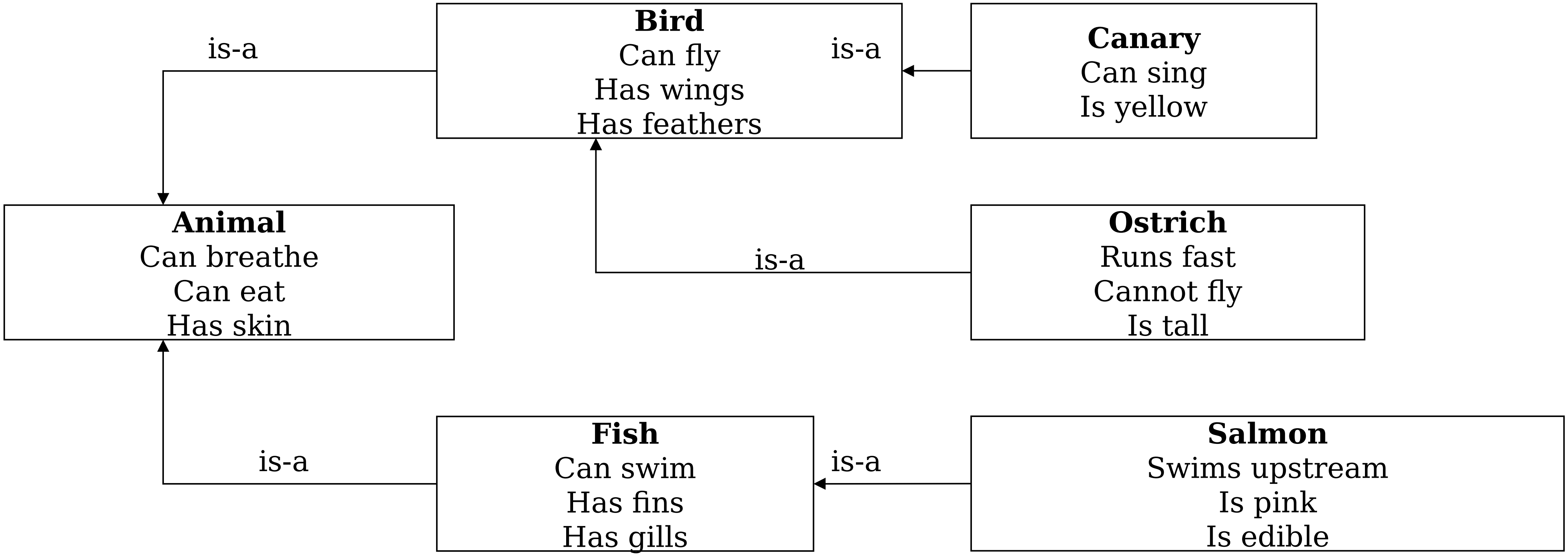
Types of relationships in semantic network











Pros and Cons of Semantic Nets

- Advantages of Semantic Nets
  - Semantic nets are capable to represent default values for categories. For example Jack has one leg while he is a person and all persons have two legs. Hence the default status of persons with two legs is allowed to override by a specific value.
  - Meaning is expressed in a translucent manner.
  - The nets are simple and easy to understand.
  - It simplifies the translation into PROLOG.
- Disadvantage of Semantic Nets
  - logical inadequacy - vagueness about what types and tokens really mean.
  - heuristic inadequacy – finding a specific piece of information could be chronically inefficient.
  - trying to establish negation is likely to lead to a combinatorial explosion.
  - "spreading activation" search is very inefficient, because it is not knowledge-guided.

Frames

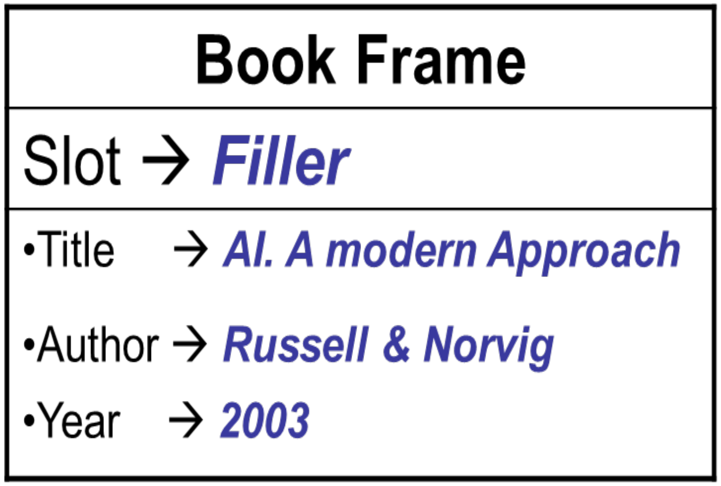
- Devised by Marvin Minsky, 1974.
- Incorporates certain valuable human thinking characteristics:
  - Expectations, assumptions, stereotypes. Exceptions. Fuzzy boundaries between classes.
- The essence of this form of knowledge representation is typicality, with exceptions, rather than definition.
- The idea of frame hierarchies is very similar to the idea of class hierarchies found in object-orientated programming.
- A frame system is a hierarchy of frames
- Each frame has:
  - a name.
  - slots: these are the properties of the entity that has the name, and they have values. A particular value may be:
    - a default value
    - an inherited value from a higher frame
    - a procedure, called a daemon, to find a value
    - a specific value, which might represent an exception.

How frames are organised

- In the higher levels of the frame hierarchy, typical knowledge about the class is stored.
  - The value in a slot may be a range or a condition.
- In the lower levels, the value in a slot may be a specific value, to overwrite the value which would otherwise be inherited from a higher frame.
- An instance of an object is joined to its class by an 'instance_of' relationship.
- A class is joined to its superclass by a 'subclass_of' relationship.
- Frames may contain both procedural and declarative knowledge.
  - Slot values normally amount to declarative knowledge, but a daemon is in effect a small program. So a slot with a daemon in it amounts to procedural knowledge.
- Note that a frames system may allow multiple inheritance but, if it does so, it must make provision for cases when inherited values conflict.

Frames: Some Examples

- Frames – semantic net with properties
- A frame represents an entity as a set of slots (attributes) and associated values
- A frame can represent a specific entry, or a general concept
- Frames are implicitly associated with one another because the value of a slot can be another frame
- The three components of a frame include:
  - frame name; attributes (slots); values (fillers: list of values, range, string, etc.)



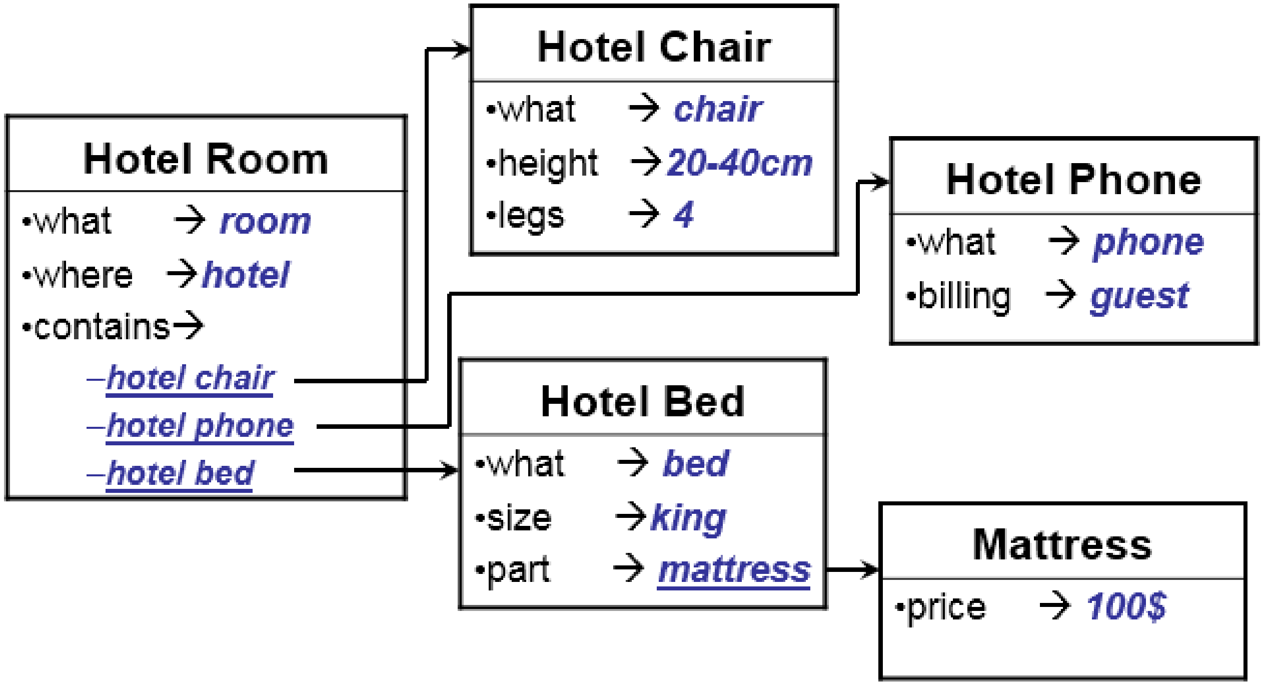
Book Frame

Slot → *Filler*

•Title → *AI. A modern Approach*

•Author → *Russell & Norvig*

•Year → *2003*



Hotel Room

•what → *room*

•where → *hotel*

•contains →

–*hotel chair*

–*hotel phone*

–*hotel bed*

Hotel Chair

•what → *chair*

•height → *20-40cm*

•legs → *4*

Hotel Phone

•what → *phone*

•billing → *guest*

Hotel Bed

•what → *bed*

•size → *king*

•part → *mattress*

Mattress

•price → *100$*

Planning using frames

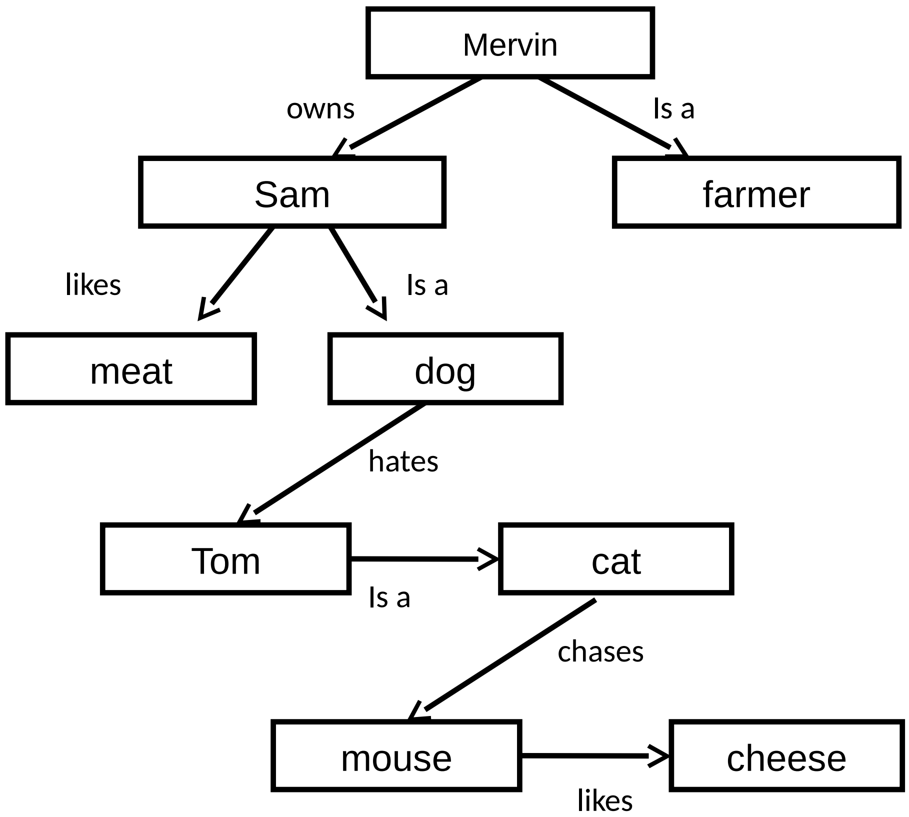


Figure: Semantic Net

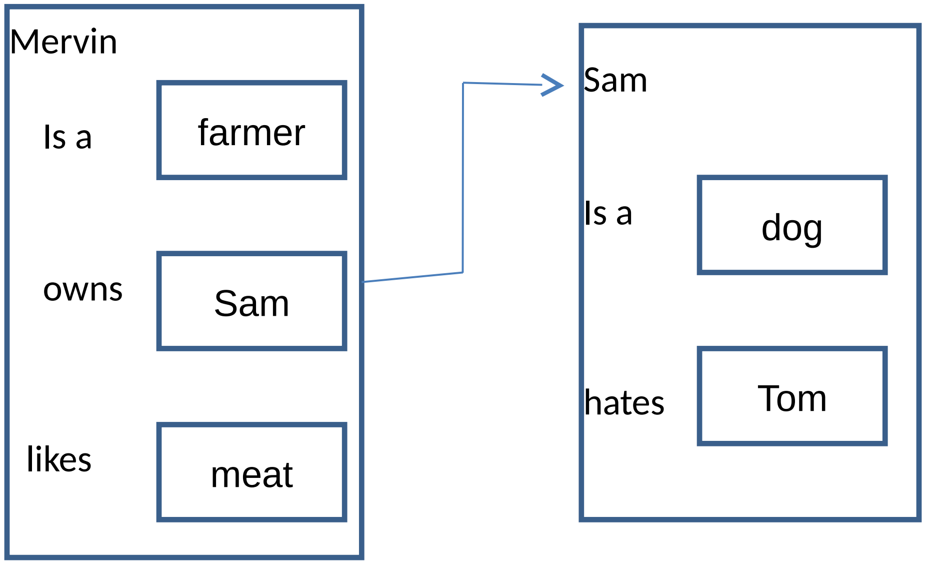


Figure: Diagrammatic form of frame based representation

Pros and Cons of Frames

Advantages of frame knowledge representation forms

- The frame based knowledge representation makes the programming simpler by grouping related data
- Compare to the knowledge representation method described in the form of production rules, the frame is flexible and intuitive in many application areas
- The frame representation is easily understood and used by people who are neither programmer nor designer of a system;
- It is not hard to add slots for new attributes and relations
- It is simple to include default data and to discover the missing values.

- Disadvantages:
  - It is difficult to use the frame system in a program, so the algorithm is required in the process of using the frame in the program
  - The lack of low-priced computer software
  - Inference mechanism is not easily processed in a frame system.

Checkpoints (1 of 2)

Fill in the blanks:

1. A compound proposition that is always ___ is called a tautology.
2. A → (A v q) is a _________
3. The truth value of statement ∃n (4n = 3n) if the domain consists of all integers is _____

State True or False:

4. p v (p→q) is the propositions is tautology.
5. ˥(P → Q) is equivalent to P ^ ˥Q.
6. The truth value of given statement is ‘4+3=7 or 5 is not prime’.

Checkpoint Solutions (1 of 2)

Fill in the blanks:

1. A compound proposition that is always **TRUE** is called a tautology.
2. A → (A v q) is a **TAUTOLOGY**
3. The truth value of statement ∃n (4n = 3n) if the domain consists of all integers is **TRUE**

State True or False:

4. p v (p→q) is the propositions is tautology – True.
5. ˥(P → Q) is equivalent to P ^ ˥Q – True
6. The truth value of given statement is ‘4+3=7 or 5 is not prime’ – False.

Checkpoints (2 of 2)

Multiple Choice Questions

1. Which of the following statements is the contrapositive of the statement, “You win the game if you know the rules but are not overconfident.”

a. If you lose the game then you don’t know the rules or you are overconfident.
b. A sufficient condition that you win the game is that you know the rules or you are not overconfident.
c. If you do not know the rules or are overconfident you lose the game.
d. If you know the rules and are overconfident then you win the game.

2. Which of the following is the negation of the statement, “For all odd primes $p < q$ there exists positive non-primes $r < s$ such that $p^2 + q^2 = r^2 + s^2$.”

e. For all odd primes $p < q$ there exists positive non-primes $r < s$ such that $p^2+q^2 \neq r^2 + s^2$
f. There exists odd primes $p < q$ such that for all positive non-primes $r < s$, $p^2+q^2 = r^2 + s^2$
g. There exists odd primes $p < q$ such that for all positive non-primes $r < s$, $p2+q2 \neq r2 + s2$
h. For all odd primes $p < q$ and for all positive non-primes $r < s$, $p^2 + q^2 \neq r^2 + s^2$

3. Consider the following assertion:

“The two statements (1) $\exists x \in D,(P(x) \wedge Q(x))$ and (2) $(\exists x \in D, P(x)) \wedge (\exists x \in D, Q(x))$ have the same truth value.” Which of the following is correct?

i. This assertion is false. A counterexample is D = N, P(x) = “x is divisible by 6,” Q(x) = “x is divisible by 3.”
j. This assertion is true. The proof follows from the distributive law for $\wedge$.
k. This assertion is false. A counterexample is D = Z, P(x) = “$x < 0$,” Q(x) = “$x \geq 0$.”
l. This assertion is false. A counterexample is D = N, P(x) = “x is a square,” Q(x) = “x is odd.”

Checkpoint Solutions (2 of 2)

Multiple Choice Questions

1. Which of the following statements is the contrapositive of the statement, “You win the game if you know the rules but are not overconfident.”

a. If you lose the game then you don’t know the rules or you are overconfident.

b. A sufficient condition that you win the game is that you know the rules or you are not overconfident.

c. If you do not know the rules or are overconfident you lose the game.

d. If you know the rules and are overconfident then you win the game.

2. Which of the following is the negation of the statement, “For all odd primes $p < q$ there exists positive non-primes $r < s$ such that $p^2 + q^2 = r^2 + s^2$.”

e. For all odd primes $p < q$ there exists positive non-primes $r < s$ such that $p^2+q^2 \neq r^2 + s^2$

f. There exists odd primes $p < q$ such that for all positive non-primes $r < s$, $p^2+q^2 = r^2 + s^2$

g. There exists odd primes $p < q$ such that for all positive non-primes $r < s$, $p^2+q^2 \neq r^2 + s^2$

h. For all odd primes $p < q$ and for all positive non-primes $r < s$, $p^2 + q^2 \neq r^2 + s^2$

3. Consider the following assertion:

“The two statements (1) $\exists x \in D, (P(x) \wedge Q(x))$ and (2) $(\exists x \in D, P(x)) \wedge (\exists x \in D, Q(x))$ have the same truth value.” Which of the following is correct?

i. This assertion is false. A counterexample is D = N, P(x) = “x is divisible by 6,” Q(x) = “x is divisible by 3.”

j. This assertion is true. The proof follows from the distributive law for $\wedge$.

k. This assertion is false. A counterexample is D = Z, P(x) = “$x < 0$,” Q(x) = “$x \geq 0$.”

l. This assertion is false. A counterexample is D = N, P(x) = “x is a square,” Q(x) = “x is odd.”

Two Marks Questions

1. Define logic. Give an example.
2. Define propositional logic. Illustrate.
3. Define Tautology. Give an example.
4. Define semantic net. Give an example.
5. Define frame. Illustrate.
6. What is meant by non-monotonic logic? Give an example.
7. Define circumscription.
8. Define default logic. Illustrate.

Four Marks Questions

1. Write short notes on syntax and semantics of propositional logic.
2. What are the properties of propositional logic?
3. What is resolution? How it is used in Knowledge representation?
4. Write a note on Conjunctive normal form.
5. Describe the procedure of expert systems development.
6. Discuss the role of semantic nets in knowledge representation with an example.

Eight Marks Questions

1. Discuss the characteristics of knowledgebase systems.
2. Explain the structure of knowledgebase systems.
3. Discuss the components of expert systems. Illustrate.
4. Explain Wumpus world problem.
5. Name the types and components of semantic networks. Explain with an example.
6. Compare frames to semantic nets. Illustrate with an example.