# Knowledge Reasoning

Inference in People

- People can do logical inference, but are not very good at it.

- Reasoning with negation and disjunction seems to be particularly difficult.

- But, people seem to employ many kinds of reasoning strategies, most of which are neither complete nor sound.

Wason Selection Task

- I have a pack of cards each of which has a letter written on one side and a number written on the other side and I claim the following rule is true:

  If a card has a vowel on one side, then it has an even number on the other side.

- Look at these cards and say which card or cards to turn over in order to decide whether the rule is true or false?

**E 4 T 7**
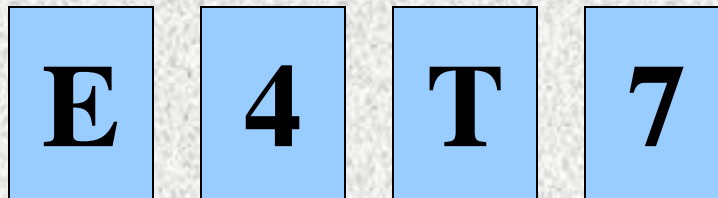
Wason Selection Task

- Wason (1966) showed that people are not very good at this task.

  Wason, P. (1966). Reasoning. In New Horizons in Psychology. Penguin, Hammondsworth, UK.

- To disprove P=>Q, find a situation in which P^~Q

- To disprove vowel => even, find a card with a vowel and an odd number

- Thus, turn over the cards showing vowels and turn over cards showing odd numbers

Wason Selection Task

- This version seems easier for people to do, as was shown by Griggs & Cox, 1982
- Your are the bouncer in a bar. Which of these people do you card given the rule:
  You must be 21 or older to drink beer.

beer	coke	22	20
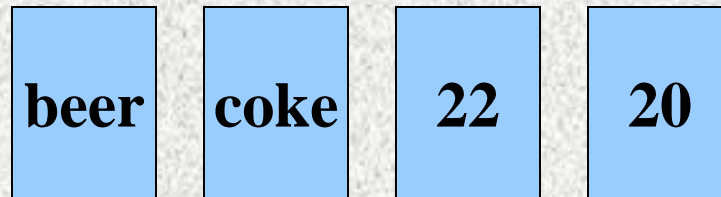
- It may be simpler because it's more familiar or because people have special strategies to reason about certain situations, such as "cheating" in a social situation.

Logic as a Methodology

Even if people do not use formal logical reasoning for solving a problem, logic might be a good approach for AI for a number of reasons

- Airplanes don't need to flap their wings
- Logic may be a good implementation strategy
- Developing a solution in a formal system like logic can offer other benefits, e.g., letting us prove properties of the approach

A knowledge-based agent

- A knowledge-based agent includes a knowledge base and an inference system.
- A knowledge base is a set of representations of facts of the world.
- Each individual representation is called a **sentence**.
- The sentences are expressed in a **knowledge representation language**.
- The agent operates as follows:

  1. It TELLs the knowledge base what it perceives.

  2. It ASKs the knowledge base what action it should perform.

  3. It performs the chosen action.

Architecture of a KB agent

- **Knowledge Level**
  - The most abstract level: describe agent by saying what it knows
  - Ex: A taxi agent might know that the Golden Gate Bridge connects San Francisco with the Marin County
- **Logical Level**
  - The level at which the knowledge is encoded into *sentences*
  - Ex: links(GoldenGateBridge, SanFrancisco, MarinCounty)
- **Implementation Level**
  - The physical representation of the sentences in the logical level
  - Ex: '(links goldengatebridge sanfrancisco marincounty)

Knowledge Representation

- When we use search to solve a problem we must
  - Capture the knowledge needed to formalize the problem
  - Apply a search technique to solve problem
 - Execute the problem solution

Role of KR

- The first step is the role of "knowledge representation" in AI.
- Formally,
  - The intended role of knowledge representation in artificial intelligence is to reduce problems of intelligent action to search problems.
- A good description, developed within the conventions of a good KR, is an open door to problem solving
- A bad description, using a bad representation, is a brick wall preventing problem solving

A Knowledge-Based Agent

- We previously talked about applications of search but not about methods of formalizing the problem.
- Now we look at extended capabilities to general logical reasoning.
- Here is one knowledge representation: logical expressions.
- A knowledge-based agent must be able to
  - Represent states, actions, etc.
  - Incorporate new percepts
  - Update internal representations of the world
  - Deduce hidden properties about the world
  - Deduce appropriate actions
- We will
  - Describe properties of languages to use for logical reasoning
  - Describe techniques for deducing new information from current information
  - Apply search to deduce (or learn) specifically needed information

WW Agent Description

- Performance measure
  - gold +1000, death -1000
  - -1 per step, -10 for using arrow
- Environment
  - Squares adjacent to wumpus are smelly
  - Squares adjacent to pit are breezy
  - Glitter iff gold is in same square
  - Shooting kills wumpus if agent facing it
  - Shooting uses up only arrow
  - Grabbing picks up gold if in same square
  - Releasing drops gold in same square
- Actuators
  - Left turn, right turn, forward, grab, release, shoot
- Sensors
  - Breeze, glitter, smell, bump, scream
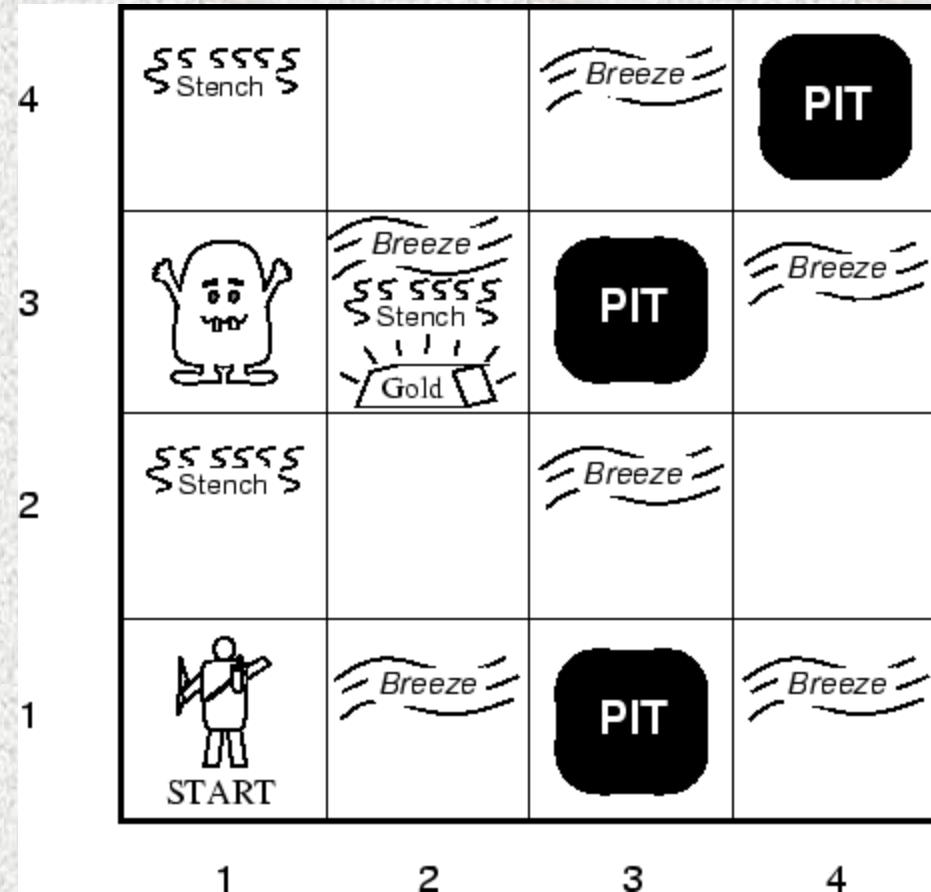
WW Environment Properties

- Observable?
  - Partial
- Deterministic?
  - Yes
- Episodic?
  - Sequential

- Static?
  - Yes (for now), wumpus and pits do not move
- Discrete?
  - Yes
- Single agent?
  - Multi (wumpus, eventually other agents)

Sample Run

1,4	2,4	3,4	4,4
1,3	2,3	2,4	2,5
1,2 OK	2,2	3,2	4,2
1,1 (A) OK	2,1 OK	3,1	4,1

Percept: [None, None, None, None, None]

Deduce: Agent alive, so (1,1) OK.
          No breeze, so (1,2) and (2,1) OK.

Sample Run

1,4	2,4	3,4	4,4
1,3	2,3	2,4	2,5
1,2 OK	2,2	3,2	4,2
1,1 (A) OK	2,1 OK	3,1	4,1

Percept: [None, None, None, None, None]

Deduce: Agent alive, so (1,1) OK
No breeze, so (1,2) and (2,1) OK

Action: Move East (turnright, goforward)

Sample Run

1,4	2,4	3,4	4,4
1,3	2,3	2,4	2,5
1,2 OK	2,2 P?	3,2	4,2
1,1 OK	2,1 Ⓐ OK	3,1 P?	4,1

Percept: [None, Breeze, None, None, None]

Deduce: Pit in (2,2) or (3,1)

Sample Run

1,4	2,4	3,4	4,4
1,3	2,3	2,4	2,5
1,2 OK	2,2 P?	3,2	4,2
1,1 OK	2,1 Ⓐ OK	3,1 P?	4,1

Percept: [None, Breeze, None, None, None]

Deduce: Pit in (2,2) or (3,1)

Action: Back to (1,1) then to (1,2)
(turnleft, turnleft, goforward,
turnright, goforward)

Sample Run

1,4	2,4	3,4	4,4
1,3 W	2,3	2,4	2,5
1,2 Ⓐ OK	2,2 OK	3,2	4,2
1,1 OK	2,1 OK	3,1 P	4,1

Percept: [Stench, None, None, None, None]

Deduce: Wumpus in (1,3)
          No pit in (2,2), pit in (3,1)

Sample Run

1,4	2,4	3,4	4,4
1,3 W	2,3	3,3	4,3
1,2 (A) OK	2,2 OK	3,2	4,2
1,1 OK	2,1 OK	3,1 P	4,1

Percept: [Stench, None, None, None, None]

Deduce: Wumpus in (1,3)
          No pit in (2,2), pit in (3,1)

Action: Move to (2,2) (turnright, goforward)
        Ignore percept for now
        Move to (2,3) (turnleft, goforward)

Sample Run

1,4	2,4 P?	3,4	4,4
1,3 W	2,3 Ⓐ OK G	3,3 P?	4,3
1,2 OK	2,2 OK	3,2	4,2
1,1 OK	2,1 OK	3,1 P	4,1

Percept: [Stench, Breeze, Glitter, None, None]

Deduce: Pit in (2,4) or (3,3)
Gold in (2,3)

Action: Move to (2,2) (turnright, goforward)
Ignore percept for now
Move to (2,3) (turnleft, goforward)

Sample Run

1,4	2,4 P?	3,4	4,4
1,3 OK W	2,3 Ⓐ OK G	3,3 P?	4,3
1,2 OK	2,2 OK	3,2	4,2
1,1 OK	2,1 OK	3,1 P	4,1

Percept: [Stench, Breeze, Glitter, None, None]

Deduce: Pit in (2,4) or (3,3)
Gold in (2,3)

Action: Move to (1,1) through OK locations

Sample Run

1,4	2,4 P?	3,4	4,4
1,3 W	2,3 (A) OK G	3,3 P?	4,3
1,2 OK	2,2 OK	3,2	4,2
1,1 OK	2,1 OK	3,1 P	4,1

Percept: [Stench, Breeze, Glitter, None, None]

Deduce: Pit in (2,4) or (3,3)
Gold in (2,3)

Action: Move to (1,1) through OK locations

Now we look at

- How to represent facts / beliefs
  ✓ "There is a pit in (2,2) or (3,1)"
- How to make inferences
  ✓ "No breeze in (1,2), so pit in (3,1)"

Representation, Reasoning and Logic

- Sentence: Individual piece of knowledge
  - English sentence forms one piece of knowledge in English language
  - Statement in C forms one piece of knowledge in C programming language

- Syntax: Form used to represent sentences
  - Syntax of C indicates legal combinations of symbols
  - a = 2 + 3; is legal
  - a = + 2 3 is not legal
  - Syntax alone does not indicate meaning

- Semantics: Mapping from sentences to facts in the world
  - They define the truth of a sentence in a "possible world"
  - Add the values of 2 and 3, store them in the memory location indicated by variable a

- In the language of arithmetic:

  $x + 2 >= y$ is a sentence
  $x2 + y >$ is not a sentence
  $x + 2 >= y$ is true in all worlds
  where the number $x + 2$ is
  no less than the number y
  $x + 2 >= y$ is true in a world where
  $x = 7, y = 1$
  $x + 2 >= y$ is false in a world where
  $x = 0, y = 6$

Entailment

- There can exist a relationship between items in the language
  - Sentences "entail" sentences (representation level)
  - Facts "follow" from facts (real world)
- Entail / Follow mean the new item is true if the old items are true
- A collection of sentences, or knowledge base (KB), entail a sentence
  - KB |= sentence
 - KB entails the sentence iff the sentence is true in all worlds where the KB is true
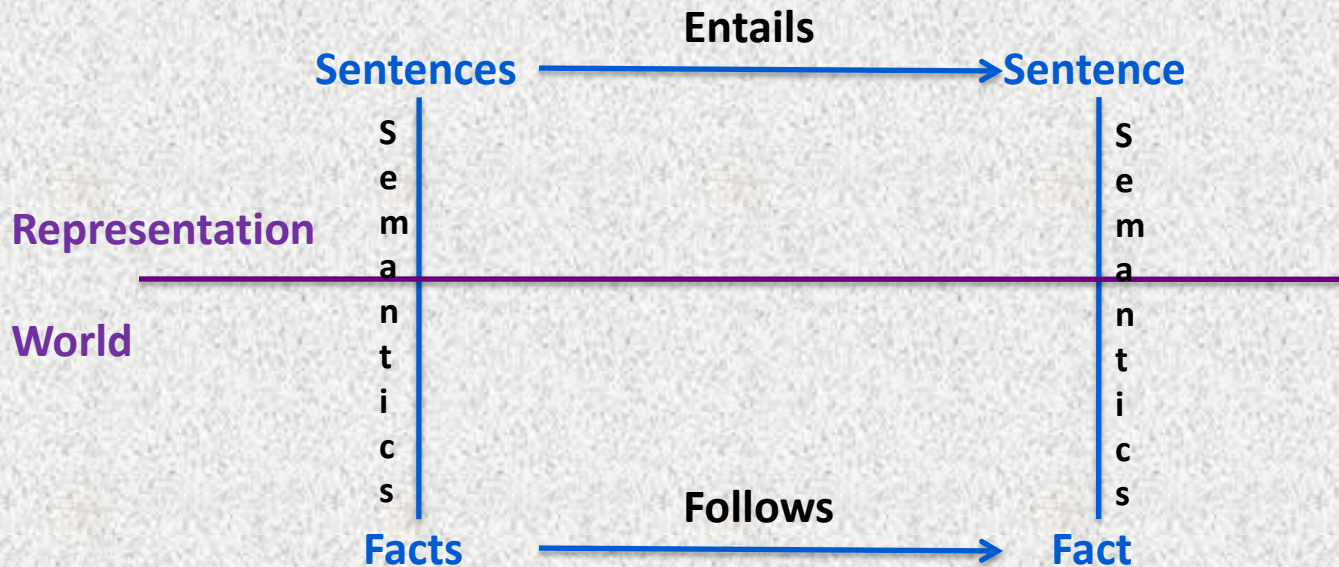
Entailment Examples

- KB
  - The Giants won
  - The Reds won
- Entails
  - Either the Giants won or the Reds won
- KB
  - To get a perfect score your program must be turned in today
  - I always get perfect scores
- Entails
 - I turned in my program today

Models

- Logicians frequently use models, which are formally structured worlds with respect to which truth can be evaluated. These are our "possible worlds".

- M is a model of a sentence s if s is true in M.

- M(s) is the set of all models of s.

- KB entails s (KB |= s) if and only if M(KB) is a subset of M(s)

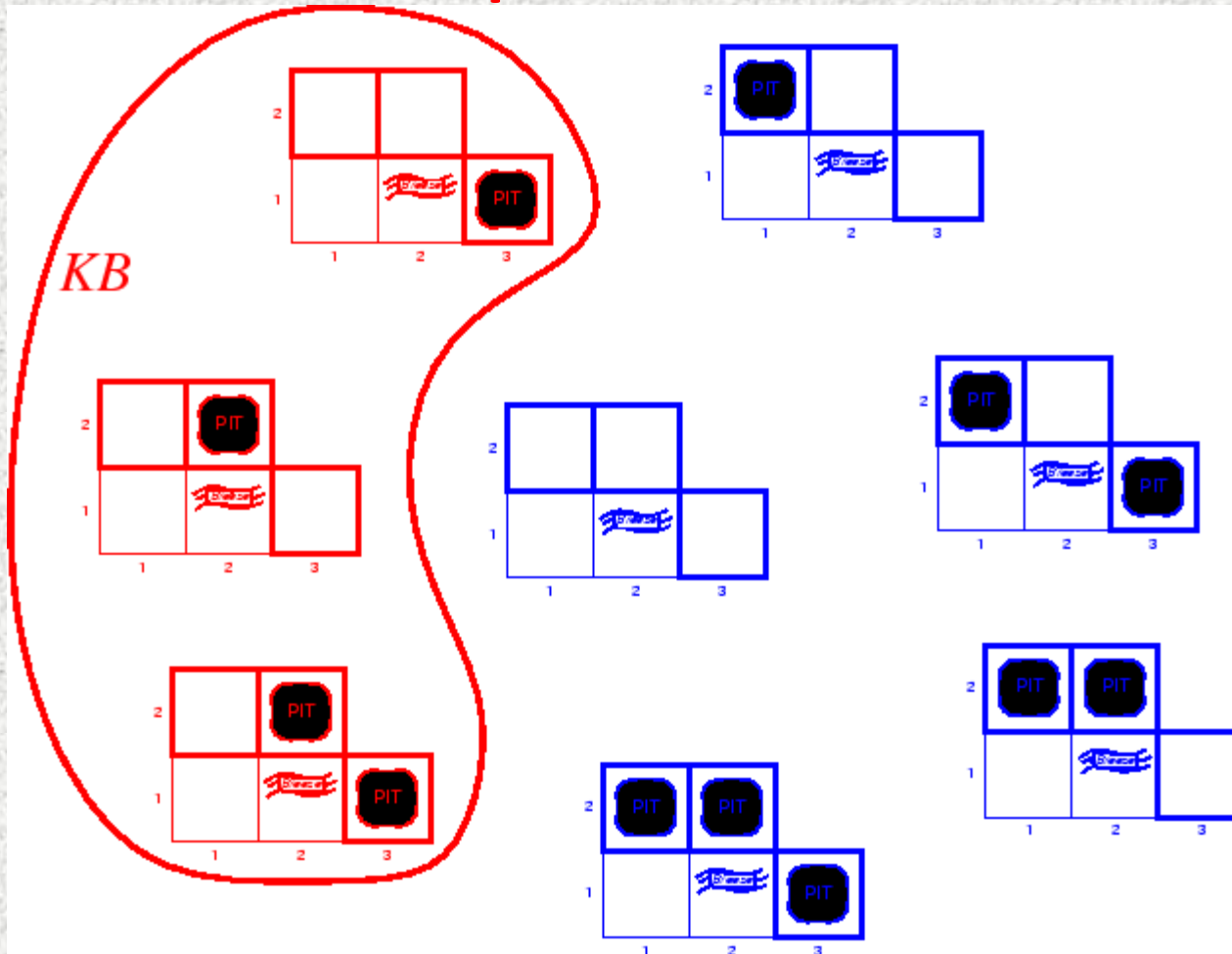# Entailment in the Wumpus World
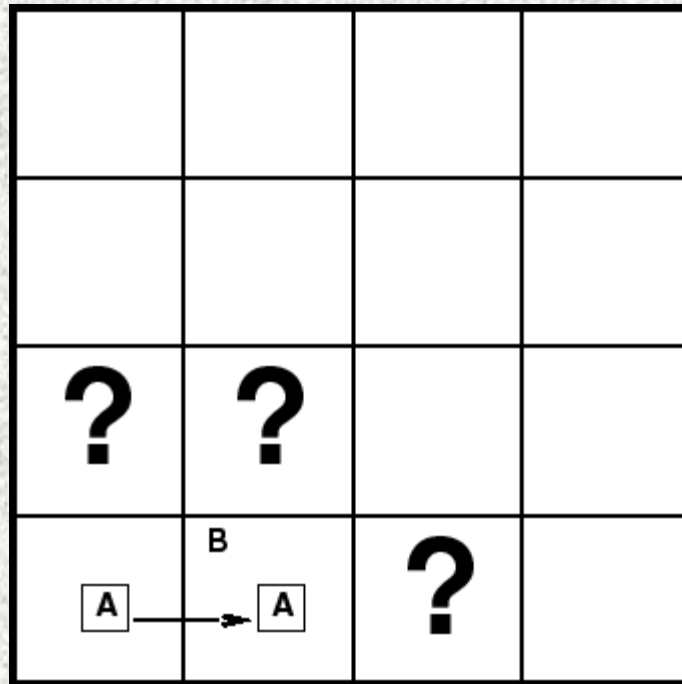
- Situation after detecting nothing in [1,1], moving right, breeze in [2,1]
- Consider possible models for the situation (the region around the visited squares) assuming only pits, no wumpi
- 3 Boolean choices, so there are 8 possible models
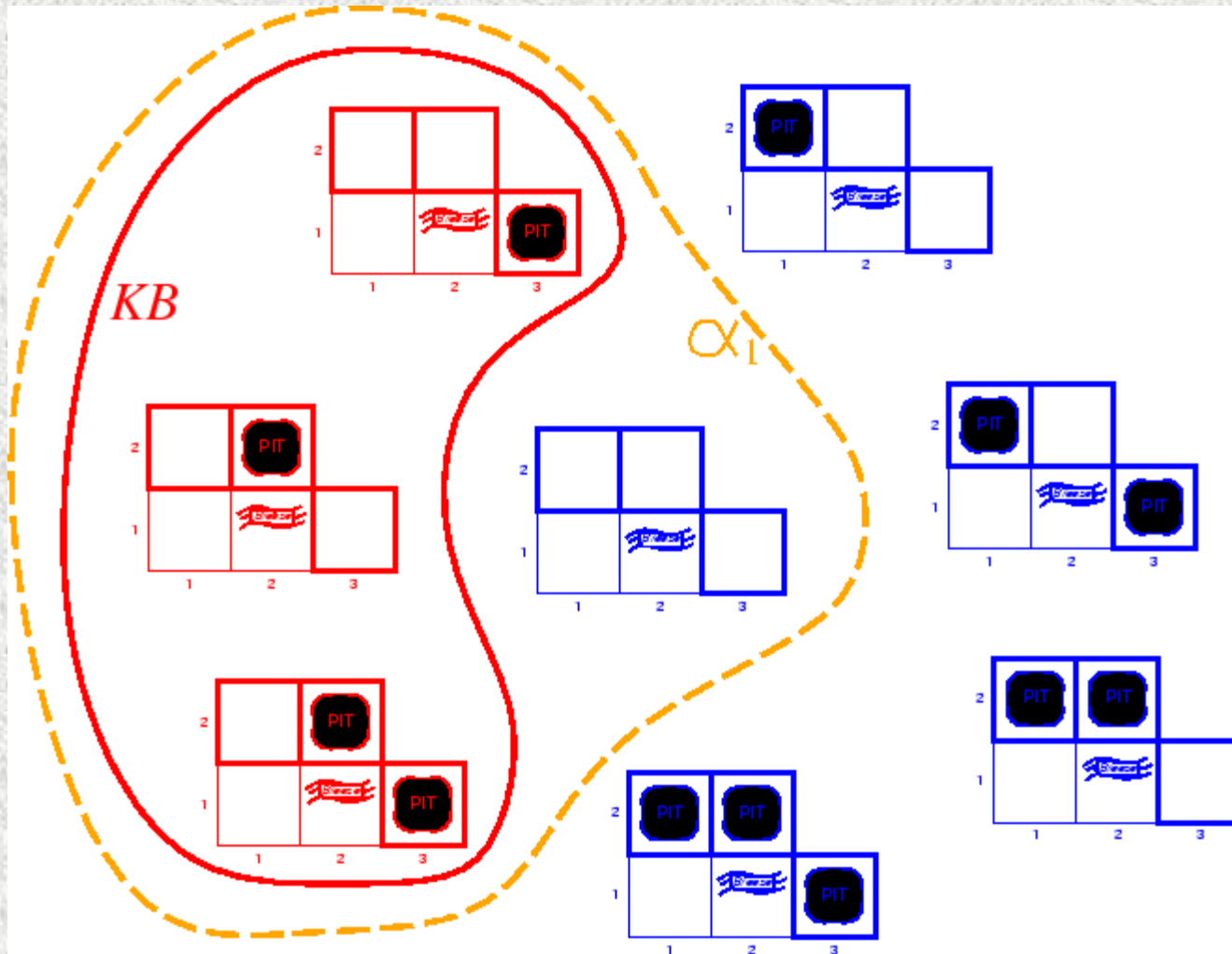
Wumpus Models
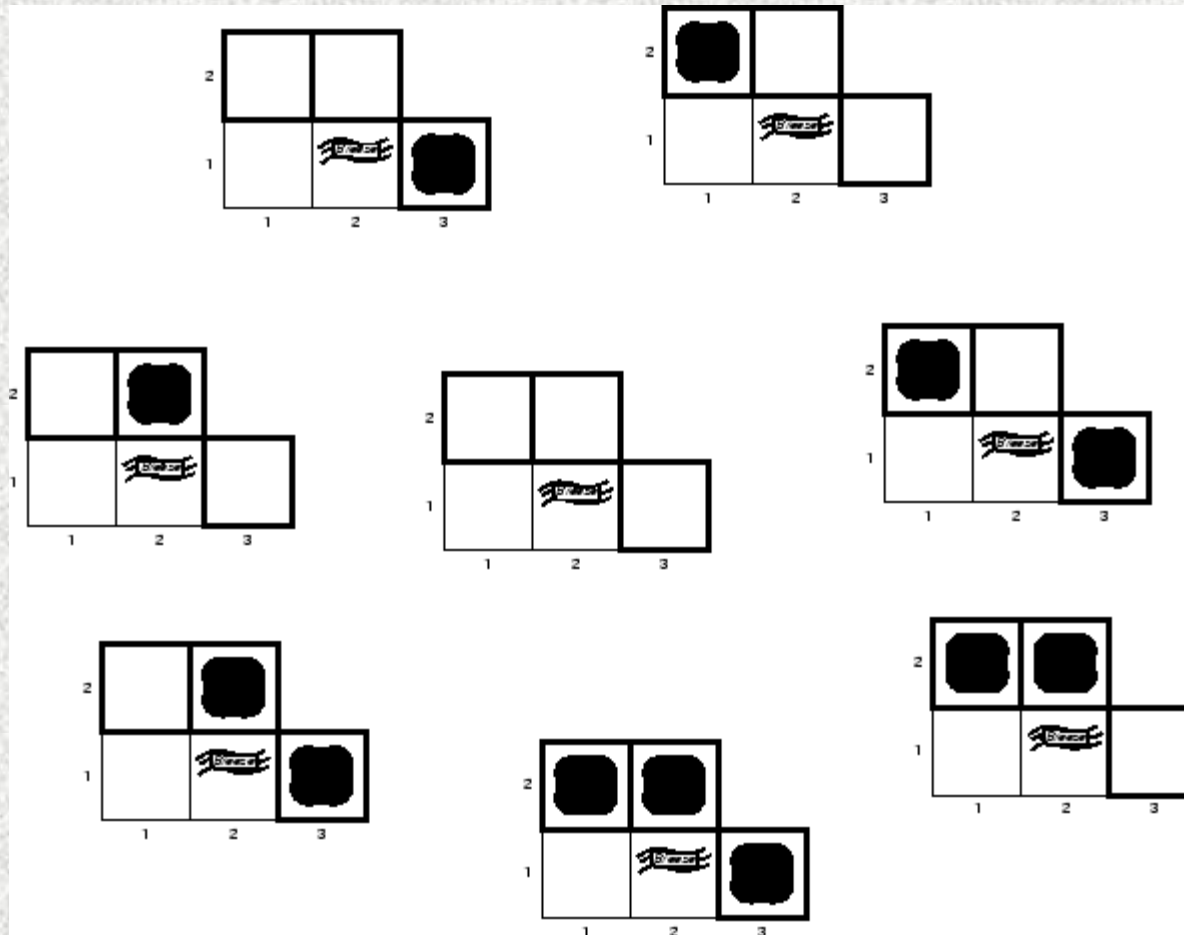
Wumpus Models
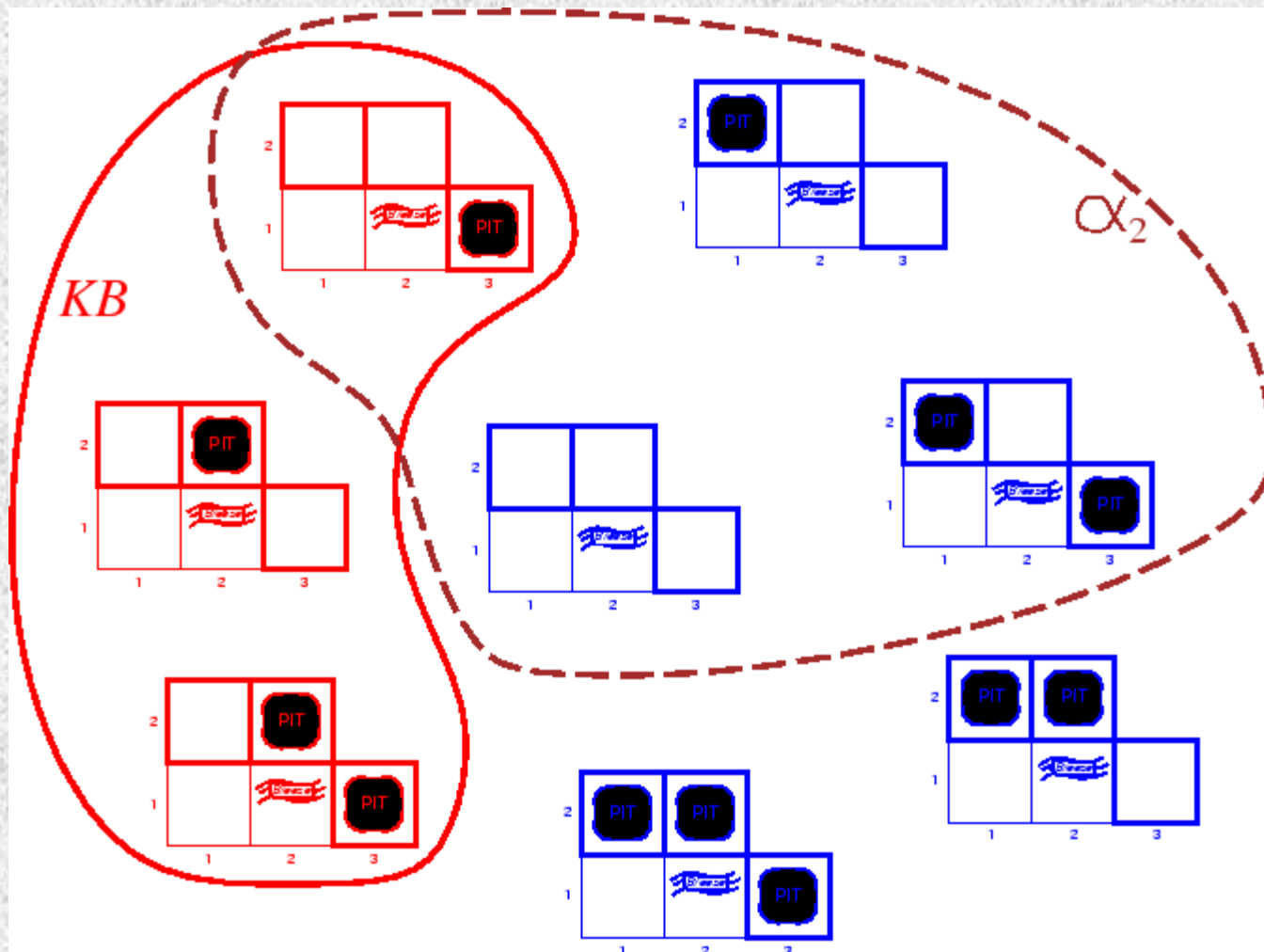

KB = wumpus world rules + observations

Wumpus Models

KB = wumpus world rules + observations
Possible conclusion: alpha1 = "[1,2] is safe"
KB |= alpha1, proved by model checking

Wumpus Models

KB = wumpus world rules + observations

alpha2 = "[2,2] is safe",   KB ⊭ alpha2

Inference, Soundness, Completeness

- $KB \vdash_i \alpha$ = sentence α can be derived from *KB* by procedure *i*

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

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

- Preview: we will define a logic (first-order logic) which is expressive enough to say almost anything of interest, and for which there exists a sound and complete inference procedure. That is, the procedure will answer any question whose answer follows from what is known by the *KB*.

Definitions

- A complete inference procedure can generate all entailed sentences from the knowledge base.

- The meaning of a sentence is a mapping onto the world (a model) and This mapping is an interpretation.

Logics

- Logics are formal languages for representing information such that conclusions can be drawn
- Logics are characterized by their "primitives" commitments
  - Ontological commitment: What exists? Facts? Objects? Time? Beliefs?
  - Epistemological commitment: What are the states of knowledge?

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	value in [0, 1]
Fuzzy logic	degree of truth	known interval value

Disclaimer

"Logic, like whiskey, loses its beneficial effect when taken in too large quantities."

- *Lord Dunsany*

Examples

- Propositional logic
  - Simple logic
  - Symbols represent entire facts
  - Boolean connectives (&, v, ->, <=>, ~)
  - Propositions (symbols, facts) are either TRUE or FALSE
- First-order logic
  - Extend propositional logic to include variables, quantifiers, functions, objects

Propositional logic: Semantics

Each model specifies true/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 an arbitrary sentence, e.g.,

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

Logical equivalence

- Two sentences are logically equivalent} iff true in same models: $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$

-
-

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

Validity and satisfiability

A sentence is valid if it is true in all models,
      e.g., *True*, $A \lor \neg A$, $A \Rightarrow A$, $(A \land (A \Rightarrow B)) \Rightarrow B$

Validity is connected to inference via the Deduction Theorem:
      $KB \models \alpha$ if and only if $(KB \Rightarrow \alpha)$ is valid

A sentence is satisfiable if it is true in some model
      e.g., $A \lor B$, C

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

Satisfiability is connected to inference via the following:
      $KB \models \alpha$ if and only if $(KB \land \neg \alpha)$ is unsatisfiable

Proof methods

- Proof methods divide into (roughly) two kinds:

    - Application of inference rules
    -
        - Legitimate (sound) generation of new sentences from old
        -
        - Proof = a sequence of inference rule applications
                Can use inference rules as operators in a standard search algorithm
        -
        - Typically require transformation of sentences into a normal form

    - Model checking
        - truth table enumeration (always exponential in n)
        -
        - improved backtracking, e.g., Davis--Putnam-Logemann-Loveland (DPLL)
        -
        - heuristic search in model space (sound but incomplete)
                e.g., min-conflicts-like hill-climbing algorithms

Propositional Logic

- For propositional logic, a row in the truth table is one interpretation
- A logic is monotonic as long as entailed sentences are preserved as more knowledge is added

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

Exercises to do…

- $(p \Rightarrow q) \vee (q \Rightarrow p)$
- $p \wedge (p \Rightarrow \neg q) \wedge q$
- *(Smoke $\Rightarrow$ Fire) $\Rightarrow$ ($\neg$Smoke $\Rightarrow$ $\neg$Fire)*
- $(p \Rightarrow (q \wedge r)) \Leftrightarrow (p \Rightarrow q) \wedge (p \Rightarrow r)$
- $(\alpha \vee \beta) \wedge (\beta \vee \gamma) \leftrightarrow (\alpha \vee \gamma)$
- $(P \vee H) \wedge \text{-}H) \Rightarrow P$

Limitations of Propositional Logic

- Propositional logic cannot express general-purpose knowledge succinctly
- We need 32 sentences to describe the relationship between wumpi and stenches
- We would need another 32 sentences for pits and breezes
- We would need at least 64 sentences to describe the effects of actions
- How would we express the fact that there is only one wumpus?
- Difficult to identify specific individuals (Mary, among 3)
- Generalizations, patterns, regularities difficult to represent (all triangles have 3 sides)

First-Order Predicate Calculus

- First-Order Logic includes:
  - Objects: peoples, numbers, places, ideas (atoms)
  - Relations: relationships between objects (predicates, T/F value)
    - Example: father(fred, mary)
    - Properties: properties of atoms (predicates, T/F value) Example: red(ball)
  - Functions: father-of(mary), next(3), (any value in range)
    - Constant: function with no parameters, MARY

Example

- Express "Socrates is a man" in
- Propositional logic
  - MANSOCRATES - single proposition representing entire idea
- First-Order Predicate Calculus
  - Man(SOCRATES) - predicate representing property of constant SOCRATES

FOPL Syntax

- **Constant** symbols (Capitalized, Functions with no arguments)
  Interpretation must map to exactly one object in the world

- **Predicates** (can take arguments, True/False)
  Interpretation maps to relationship or property T/F value

- **Function** (can take arguments)
  Maps to exactly one object in the world

Definitions

- Term
  Anything that identifies an object
  Function(args)
  Constant - function with 0 args

- Atomic sentence
  Predicate with term arguments
  Enemies(WilyCoyote, RoadRunner)
  Married(FatherOf(Alex), MotherOf(Alex))

- Literals
  atomic sentences and negated atomic sentences

- Connectives
  (&), (v), (->), (<=>), (~)
  if connected by , conjunction (components are conjuncts)
  if connected by , disjunction (components are disjuncts)

- Quantifiers
  Universal Quantifier ∀
  Existential Quantifier ∃

Quantifiers

- Universal quantifiers are often used with "implies" to form "rules":

    $(\forall x)$ student(x) $\rightarrow$ smart(x) means "All students are smart"

- Universal quantification is *rarely* used to make blanket statements about every individual in the world:

    $(\forall x)$student(x)$\wedge$smart(x) means "Everyone in the world is a student and is smart"

- Existential quantifiers are usually used with "and" to specify a list of properties about an individual:

    $(\exists x)$ student(x) $\wedge$ smart(x) means "There is a student who is smart"

- A common mistake is to represent this English sentence as the FOL sentence:

    $(\exists x)$ student(x) $\rightarrow$ smart(x)

    – But what happens when there is a person who is *not* a student?

Quantifier Scope

- FOL sentences have structure, like programs
- In particular, the variables in a sentence have a scope
- For example, suppose we want to say
  - "everyone who is alive loves someone"
  - $(\forall x)$ alive$(x) \rightarrow (\exists y)$ loves(x,y)
- Here's how we soce the variables

$$(\forall x)\ \text{alive}(x) \rightarrow (\exists y)\ \text{loves}(x,y)$$

——— Scope of x
——— Scope of y

Quantifier Scope

- **Switching the order of universal quantifiers *does not* change the meaning:**
    - $(\forall x)(\forall y)P(x,y) \leftrightarrow (\forall y)(\forall x) P(x,y)$

- **Similarly, you can switch the order of existential quantifiers:**
    - $(\exists x)(\exists y)P(x,y) \leftrightarrow (\exists y)(\exists x) P(x,y)$

- **Switching the order of universals and existentials *does* change meaning:**
    - Everyone likes someone: $(\forall x)(\exists y) \text{ likes}(x,y)$
    - Someone likes everyone: $(\exists y)(\forall x) \text{ likes}(x,y)$

Connections between All and Exists

- **We can relate sentences involving $\forall$ and $\exists$ using De Morgan's laws:**
  1. $(\forall x)\, \neg P(x) \leftrightarrow \neg(\exists x)\, P(x)$
  2. $\neg(\forall x)\, P \leftrightarrow (\exists x)\, \neg P(x)$
  3. $(\forall x)\, P(x) \leftrightarrow \neg\,(\exists x)\, \neg P(x)$
  4. $(\exists x)\, P(x) \leftrightarrow \neg(\forall x)\, \neg P(x)$

Other Properties

- $(X \rightarrow Y) <-> -X \lor Y$
  - Can prove with truth table
- Not true:
  - $(X \rightarrow Y) <-> (Y \rightarrow X)$
  - This is a type of inference that is not sound (abduction)

Examples

- All men are mortal

Examples

- All men are mortal
  - $\forall x$ [Man(x) $\rightarrow$ Mortal(x)]

Examples

- All men are mortal
    - $\forall x$ [Man(x) -> Mortal(x)]
- Socrates is a man

Examples

- All men are mortal
  - $\forall x \ [Man(x) \rightarrow Mortal(x)]$
- Socrates is a man
  - Man(Socrates)

Examples

- All men are mortal
  - $\forall$ x [Man(x) -> Mortal(x)]
- Socrates is a man
  - Man(Socrates)
- Socrates is mortal
  - Mortal(Socrates)

Examples

- All men are mortal
  - $\forall x$ [Man(x) -> Mortal(x)]
- Socrates is a man
  - Man(Socrates)
- Socrates is mortal
  - Mortal(Socrates)
- All purple mushrooms are poisonous

Examples

- All men are mortal
  - $\forall x \, [Man(x) \rightarrow Mortal(x)]$
- Socrates is a man
  - $Man(Socrates)$
- Socrates is mortal
  - $Mortal(Socrates)$
- All purple mushrooms are poisonous
  - $\forall x \, [(Purple(x) \wedge Mushroom(x)) \rightarrow Poisonous(x)]$

Examples

- All men are mortal
  - $\forall x \ [Man(x) \rightarrow Mortal(x)]$
- Socrates is a man
  - $Man(Socrates)$
- Socrates is mortal
  - $Mortal(Socrates)$
- All purple mushrooms are poisonous
  - $\forall x \ [(Purple(x) \wedge Mushroom(x)) \rightarrow Poisonous(x)]$
- A mushroom is poisonous only if it is purple

Examples

- All men are mortal
  - $\forall x [Man(x) \rightarrow Mortal(x)]$
- Socrates is a man
  - $Man(Socrates)$
- Socrates is mortal
  - $Mortal(Socrates)$
- All purple mushrooms are poisonous
  - $\forall x [(Purple(x) \wedge Mushroom(x)) \rightarrow Poisonous(x)]$
- A mushroom is poisonous only if it is purple

Examples

- All men are mortal
  - $\forall x$ [Man(x) -> Mortal(x)]
- Socrates is a man
  - Man(Socrates)
- Socrates is mortal
  - Mortal(Socrates)
- All purple mushrooms are poisonous
  - $\forall x$ [(Purple(x) ^ Mushroom(x)) -> Poisonous(x)]
- A mushroom is poisonous only if it is purple
  - $\forall x$ [(Mushroom(x) ^ Poisonous(x)) -> Purple(x)]

Examples

- All men are mortal
  - $\forall$ x [Man(x) -> Mortal(x)]
- Socrates is a man
  - Man(Socrates)
- Socrates is mortal
  - Mortal(Socrates)
- All purple mushrooms are poisonous
  - $\forall$ x [(Purple(x) ^ Mushroom(x)) -> Poisonous(x)]
- A mushroom is poisonous only if it is purple
  - $\forall$ x [(Mushroom(x) ^ Poisonous(x)) -> Purple(x)]
- No purple mushroom is poisonous

Examples

- All men are mortal
  - ∀ x [Man(x) -> Mortal(x)]
- Socrates is a man
  - Man(Socrates)
- Socrates is mortal
  - Mortal(Socrates)
- All purple mushrooms are poisonous
  - ∀ x [(Purple(x) ^ Mushroom(x)) -> Poisonous(x)]
- A mushroom is poisonous only if it is purple
  - ∀ x [(Mushroom(x) ^ Poisonous(x)) -> Purple(x)]
- No purple mushroom is poisonous
  - -(∃ x [Purple(x) ^ Mushroom(x) ^ Poisonous(x)])

Translating English to FOL

- **Every gardener likes the sun.**

  $\forall x$ gardener(x) $\rightarrow$ likes(x,Sun)

- **You can fool some of the people all of the time.**

  $\exists x \ \forall t$ person(x) $\wedge$time(t) $\rightarrow$ can-fool(x,t)

- **You can fool all of the people some of the time.**

  $\forall x \ \exists t$ (person(x) $\rightarrow$ time(t) $\wedge$can-fool(x,t))

  $\forall x$ (person(x) $\rightarrow \exists t$ (time(t) $\wedge$can-fool(x,t))

  Every chicken hatched from an egg.

  $\forall x \ \exists y$ ((chicken(x) $\wedge$ egg(y)) $\Rightarrow$ hatchedFrom(x,y))

- *Everybody loves somebody sometime.*

- $\forall x \ \exists y \ \exists t$ ((person(x) $\wedge$ person(y) $\wedge$ time(t)) $\Rightarrow$ loves(x,y,t))

Examples

- There is exactly one mushroom

$$\exists x Mushroom(x) \wedge (\forall y (NEQ(x, y) \rightarrow \neg Mushroom(y)))]$$

  – Because "exactly one" is difficult to express we can use $\exists!$ To denote exactly one of a type of object

Examples

- No human enjoys golf

$$\forall x[Human(x) \rightarrow \neg Enjoys(x, Golf)$$

- Some professor that is not a historian writes programs

$$\exists x[\Pr ofessor(x) \wedge \neg Historian(x) \wedge Writes(x, \Pr ograms)]$$

- Every boy owns a dog

We want to build the arguments

- Solution: Logical Inference

- Converting the sentences
  - Propositions
  - Representing sentences into FOL
  - Use Logical inference Rule

Rules of Inference

Sound rules of inference

- Here are some examples of sound rules of inference
- Each can be shown to be sound using a truth table

RULE	PREMISE	CONCLUSION
Modus Ponens	A, A → B	B
And Introduction	A, B	A ∧ B
And Elimination	A ∧ B	A
Double Negation	¬¬A	A
Unit Resolution	A ∨ B, ¬B	A
Resolution	**A ∨ B, ¬B ∨ C**	**A ∨ C**

Normal Forms

- Other approaches to inference use syntactic operations on sentences, often expressed in standardized forms

- Conjunctive Normal Form (CNF)
  conjunction of disjunctions of literals (conjunction of clauses)
  For example, (A v –B) ^ (B v –C v –D)

- Disjunctive Normal Form (DNF)
  disjunction of conjunctions of literals (disjunction of terms)
  For example, (A ^ B) v (A ^ -C) v (A ^ -D) v (-B ^ -C) v (-B ^ -D)

- Horn Form (restricted)
  conjunction of *Horn clauses* (clauses with <= 1 positive literal)
  For example, (A v –B) ^ (B v –C v –D)

Conversion to CNF

1. Eliminate all $\leftrightarrow$ connectives

   $(P \leftrightarrow Q) \Rightarrow ((P \rightarrow Q) \wedge (Q \rightarrow P))$

2. Eliminate all $\rightarrow$ connectives

   $(P \rightarrow Q) \Rightarrow (\neg P \vee Q)$

3. Reduce the scope of each negation symbol to a single predicate

   $\neg\neg P \Rightarrow P$

   $\neg(P \vee Q) \Rightarrow \neg P \wedge \neg Q$

   $\neg(P \wedge Q) \Rightarrow \neg P \vee \neg Q$

   $\neg(\forall x)P \Rightarrow (\exists x)\neg P$

   $\neg(\exists x)P \Rightarrow (\forall x)\neg P$

4. Standardize variables: rename all variables so that each quantifier has its own unique variable name

Unification

- The process of finding a substitution for predicate parameters is called unification.

- *unification algorithm* helps to find out the most general conditions under which two literals can be made the same.

- Takes two atomic sentences, called literals, as input

  – Returns "Failure" if they do not match and a substitution list, θ, if they do

• That is, unify(p,q) = θ means subst(θ, p) = subst(θ, q) for two atomic sentences, p and q •

- θ is called the most general unifier (mgu)

Quantified inference rules

- Universal instantiation
  - $\forall x\ P(x)\ \therefore\ P(A)$ ← **skolem constant F**
- Universal generalization
  - $P(A) \wedge P(B) \ldots \therefore\ \forall x\ P(x)$
- Existential instantiation
  - $\exists x\ P(x)\ \therefore P(F)$ ← **skolem constant F**
- Existential generalization
  - $P(A)\ \therefore\ \exists x\ P(x)$

Resolution

- **Resolution** is a valid inference rule producing a new clause implied by two clauses containing *complementary literals*
  - A literal is an atomic symbol or its negation, i.e., P, ~P
- Amazingly, this is the only interference rule you need to build a sound and complete theorem prover
  - Based on proof by contradiction and usually called resolution refutation
- The resolution rule was discovered by Alan Robinson (CS, U. of Syracuse) in the mid 60s

Resolution

- A KB is actually a set of sentences all of which are true, i.e., a conjunction of sentences.

- To use resolution, put KB into *conjunctive normal form* (CNF), where each sentence written as a disjunction of (one or more) literals

Example

Tautologies
$$(A \rightarrow B) \leftrightarrow (\sim A \vee B)$$
$$(A \vee (B \wedge C)) \leftrightarrow (A \vee B) \wedge (A \vee C)$$

- KB: $[P \rightarrow Q , Q \rightarrow R \wedge S]$

- KB in CNF: $[\sim P \vee Q , \sim Q \vee R , \sim Q \vee S]$

- Resolve KB(1) and KB(2) producing: $\sim P \vee R$ *(i.e., $P \rightarrow R$)*

- Resolve KB(1) and KB(3) producing: $\sim P \vee S$ *(i.e., $P \rightarrow S$)*

- New KB: $[\sim P \vee Q , \sim Q \vee \sim R \vee \sim S , \sim P \vee R , \sim P \vee S]$

An example

It is not sunny this afternoon and it is colder than yesterday. If we go swimming it is sunny. If we do not go swimming then we will take a canoe trip. If we take a canoe trip then we will be home by sunset. We will be home by sunset

1. It is not sunny this afternoon and it is colder than yesterday.
2. If we go swimming it is sunny.
3. If we do not go swimming then we will take a canoe trip.
4. If we take a canoe trip then we will be home by sunset.
5. We will be home by sunset

p It is sunny this afternoon

q It is colder than yesterday

r We go swimming

s We will take a canoe trip

t We will be home by sunset (the conclusion)

propositions

1. $\neg p \wedge q$

2. $r \rightarrow p$

3. $\neg r \rightarrow s$

4. $s \rightarrow t$

5. t

hypotheses

p It is sunny this afternoon

q It is colder than yesterday

r We go swimming

s We will take a canoe trip

t We will be home by sunset (the conclusion)

1. $\neg p \wedge q$
2. $r \rightarrow p$
3. $\neg r \rightarrow s$
4. $s \rightarrow t$
5. t

	Step	Reason
1.	$\neg p \wedge q$	Hypothesis
2.	$\neg p$	Simplification using (1)
3.	$r \rightarrow p$	Hypothesis
4.	$\neg r$	Modus tollens using (2) and (3)
5.	$\neg r \rightarrow s$	Hypothesis
6.	s	Modus ponens using (4) and (5)
7.	$s \rightarrow t$	Hypothesis
8.	t	Modus ponens using (6) and (7)

Rule of inference	Tautology	Name
$p \rightarrow q$ p $\therefore q$	$[p \wedge (p \rightarrow q)] \rightarrow q$	Modus ponens
$\neg q$ $p \rightarrow q$ $\therefore \neg p$	$[\neg q \wedge (p \rightarrow q)] \rightarrow \neg p$	Modus tollen
$p \rightarrow q$ $q \rightarrow r$ $\therefore p \rightarrow r$	$[(p \rightarrow q) \wedge (q \rightarrow r)] \rightarrow (p \rightarrow r)$	Hypothetical syllogism
$p \vee q$ $\neg p$ $\therefore q$	$((p \vee q) \wedge \neg p) \rightarrow q$	Disjunctive syllogism
p $\therefore p \vee q$	$p \rightarrow (p \vee q)$	Addition
$p \wedge q$ $\therefore p$	$(p \wedge q) \rightarrow p$	Simplification
p q $\therefore p \wedge q$	$((p) \wedge (q)) \rightarrow (p \wedge q)$	Conjunction
$p \vee q$ $\neg p \vee r$ $\therefore q \vee r$	$[(p \vee q) \wedge (\neg p \vee r)] \rightarrow (p \vee r)$	Resolution

Resolution Refutation

- Show by a *resolution refutation* that the following formulas is a tautology:

  $$(P \to Q) \to [(R \vee P) \to (R \vee Q)]$$

- Solution

$$(P \Rightarrow Q) \Rightarrow [(R \vee P) \Rightarrow (R \vee Q)]$$
$$= \neg(\neg P \vee Q) \vee [\neg(R \vee P) \vee (R \vee Q)]$$
$$= (P \wedge \neg Q) \vee (\neg R \wedge \neg P) \vee R \vee Q$$

Now negate the whole expression, ready to perform resolution

$$: \neg((P \wedge \neg Q) \vee (\neg R \wedge \neg P) \vee R \vee Q)$$
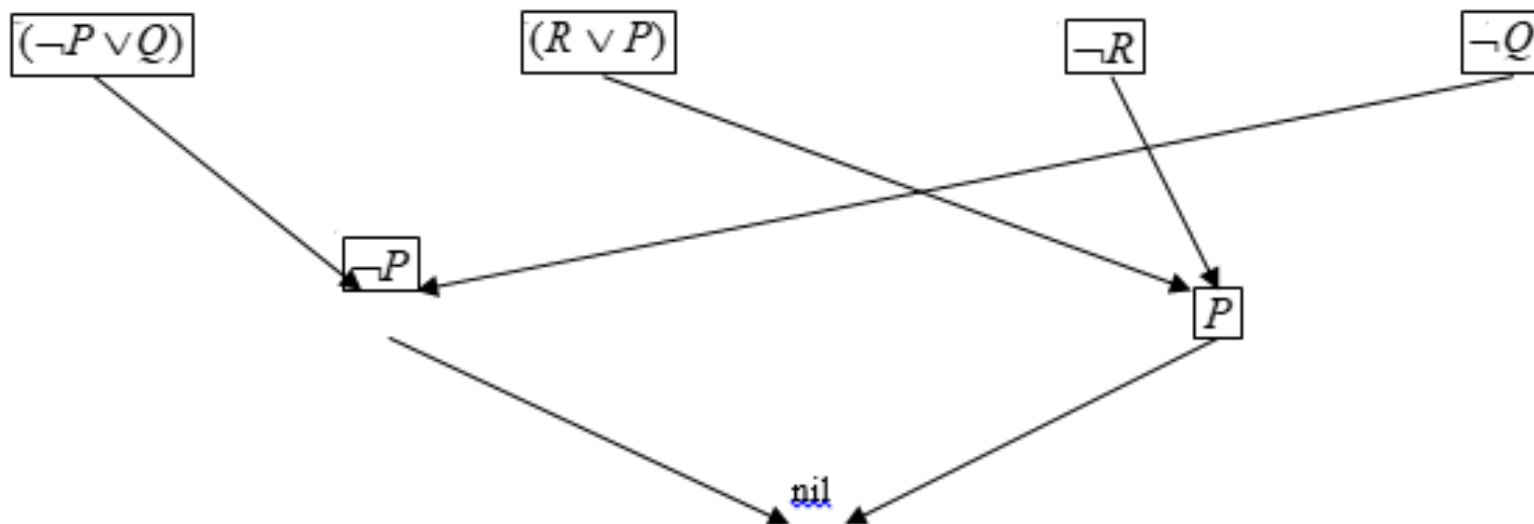$$= (\neg P \vee Q) \wedge (R \vee P) \wedge \neg R \wedge \neg Q$$

Resolution Refutation

- Show by a *resolution refutation* that the following formulas is a tautology:

  (P →Q) → [(R ∨ P) → (R ∨ Q)]

- Solution

Now take each term and perform resolution:

$(\neg P \vee Q)$ $(R \vee P)$ $\neg R$ $\neg Q$

$\neg P$ P

nil

Resolution Refutation

- Anyone passing his or her COMP472 exam and getting an internship is happy. But anyone who studies can pass all his or her exams. Gothe studies and he is lucky. Anyone who is lucky gets an internship. Is Gothe happy? Use *resolution refutation* to prove your answer.

Represent the axioms in predicate calculus:

- $\forall X$ ([pass(X, COMP472) $\wedge$ get (X, internship)] $\Rightarrow$ happy(X))
- $\forall X \forall Y$ (studies(X) $\Rightarrow$ pass(X, Y))
- studies(Gothe)
- lucky(Gothe)
- $\forall X$ (lucky(X) $\Rightarrow$ get(X, Internship))

Resolution Refutation

- ## *Convert axioms to clauses:*

1. Eliminate $\Rightarrow$ $\forall X \neg[\text{pass}(X, \text{COMP472}) \wedge \text{get}(X, \text{Internship})] \vee \text{happy}(X)$
   Convert to CNF $\forall X \neg\text{pass}(X, \text{COMP472}) \vee \neg\text{get}(X, \text{Internship}) \vee \text{happy}(X)$
   Eliminate $\forall$ $\neg\text{pass}(X, \text{COMP472}) \vee \neg\text{get}(X, \text{Internship}) \vee \text{happy}(X)$

2. Eliminate $\Rightarrow$ $\forall X \forall Y (\neg\text{studies}(X)) \vee \text{pass}(X, Y))$
   Convert to CNF $\forall X \forall Y ([\neg\text{studies}(X) \vee \text{pass}(X, Y))$
                        $\forall X \forall Y ([\neg\text{studies}(X) \vee \text{pass}(X, Y)]$
   Eliminate $\forall$ $\neg\text{studies}(X) \vee \text{pass}(X, Y)$

3. Nothing to do. $\text{studies}(\text{Gothe})$

4. Nothing to do. $\text{lucky}(\text{Gothe})$

5. Eliminate $\Rightarrow$ $\forall X (\neg\text{lucky}(X) \vee \text{get}(X, \text{Internship}))$
   Eliminate $\forall$ $\neg\text{lucky}(X) \vee \text{get}(X, \text{Internship})$

Resolution Refutation

To prove: happy(Gothe)

- So negate the goal and try to make a contradiction. Therefore add "¬happy(Gothe)" to the list of axioms.

Final list of axioms: (with variables renamed)

1. ¬pass(X1, COMP472) ∨ ¬get(X1, Internship) ∨ happy(X1)
2. ¬studies(X2) ∨ pass(X2, Y1)
3. studies(Gothe)
4. lucky(Gothe)
5. ¬lucky(X4) ∨ get(X4, Internship)
6. ¬happy(Gothe)

¬pass(X1, COMP472) ∨ ¬get(X1, Internship) ∨ happy(X1)

¬studies(X2) ∨ pass(X2, Y1)

- ¬get(X1, Internship) ∨ happy(X1)V¬studies(X2)

¬lucky(X4) ∨ get(X4, Internship)

studies(Gothe)

¬lucky(X4)V happy(X1) V ¬studies(X2)

lucky(Gothe)

¬happy(Gothe)

NIL