Artificial Intelligence
CS 534

Week 4

Unit 3: Reasoning with logical agents

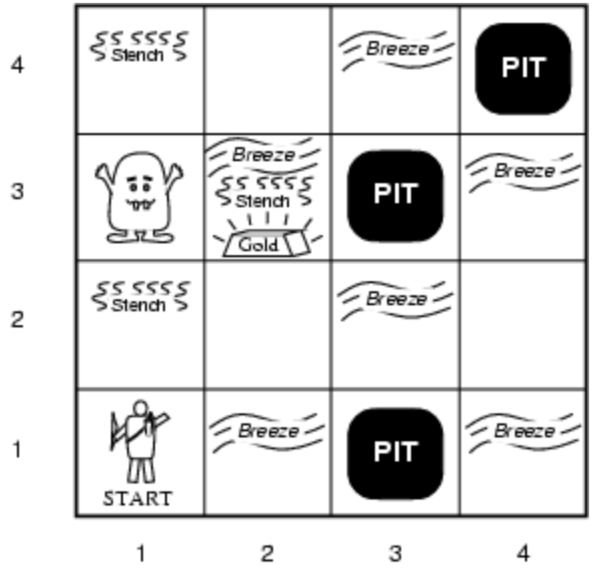
Part I: Logical agents. Propositional logic

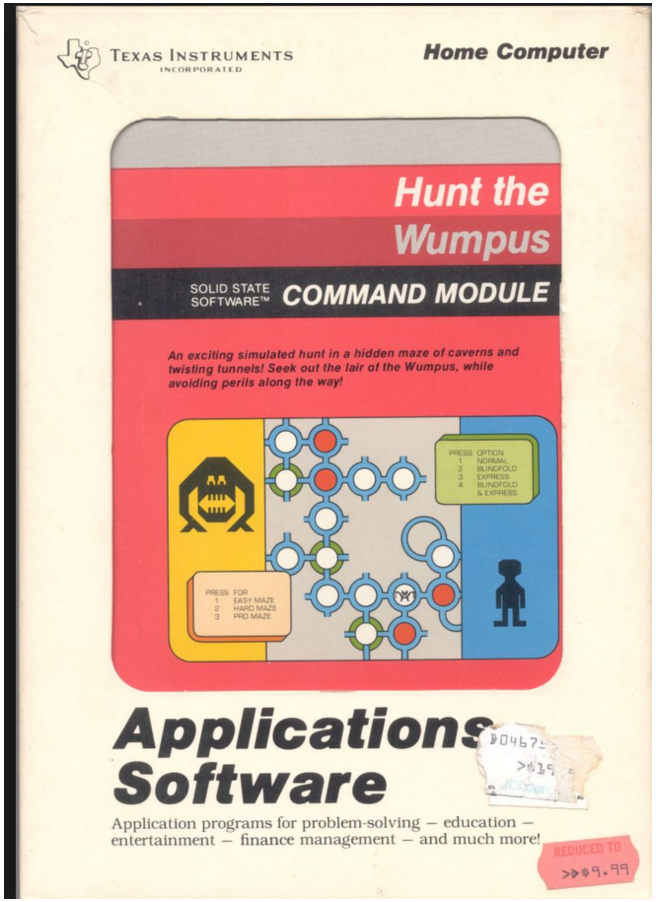
Materials

- Propositional logic:
  - Chapter 7

Wumpus World PEAS description

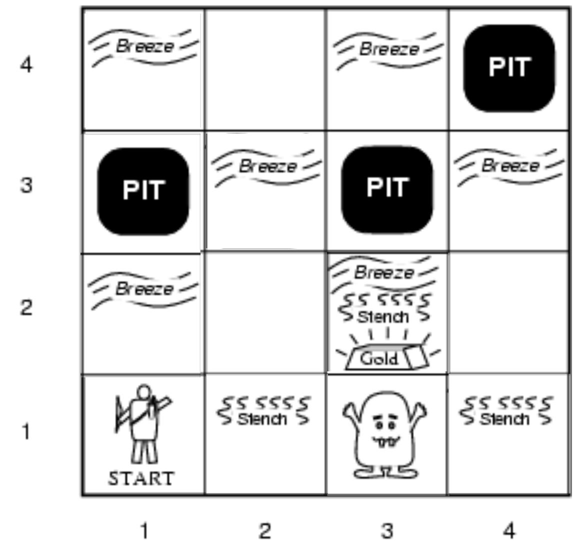
- A cave consisting of rooms connected by passageways
- Wumpus is a beast that eats anyone who enters its room
- It can be shot by the agent
- Agent has only one arrow
- Pit: bottomless trap for the agent
- Agent can dig some gold





Wumpus World PEAS description

- Performance measure
  - gold: +1000, death: -1000 (Pit or Wumpus)
  - -1 per step, -10 for using the arrow
- Environment
  - Each square other than start can be a pit (*p*=0.2)
  - Squares adjacent to wumpus are smelly
  - Squares adjacent to pit are breezy
  - Glitter **iff** gold is in the same square; can grab
  - Shooting kills wumpus if you are facing it
  - Shooting uses up the only arrow
  - Grabbing picks up gold if in same square

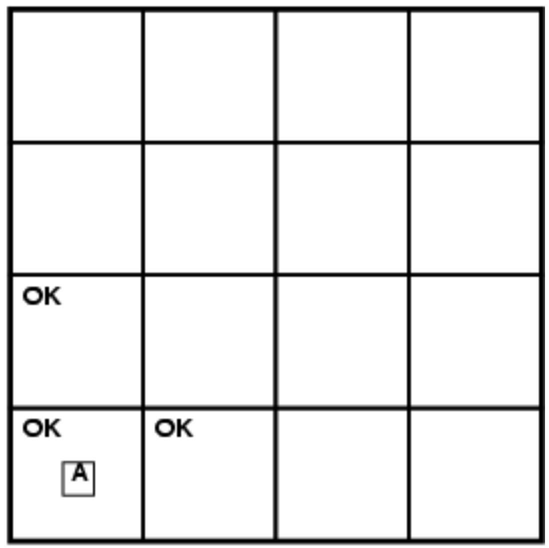


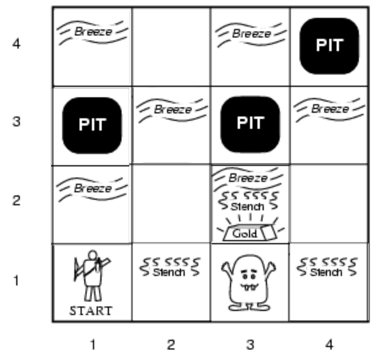
- Actuators: Left turn (90°), Right turn (90°), Forward, Grab, Shoot
- Sensors: outputs are {Stench, Breeze, Glitter, BUmp, ScReam}
  - Percepts: `[s1,s2,s3,s4,s5]`, where `s1= S` or `N (none)`, *etc*

Wumpus world characterization

- Fully Observable? No – only local perception→partially
- Deterministic? Yes – outcomes exactly specified
- Episodic? No – sequential at the level of actions
- Static? Yes – Wumpus and Pits do not move
- Discrete? Yes
- Single-agent? Yes – Wumpus is essentially a natural feature

Exploring a wumpus world

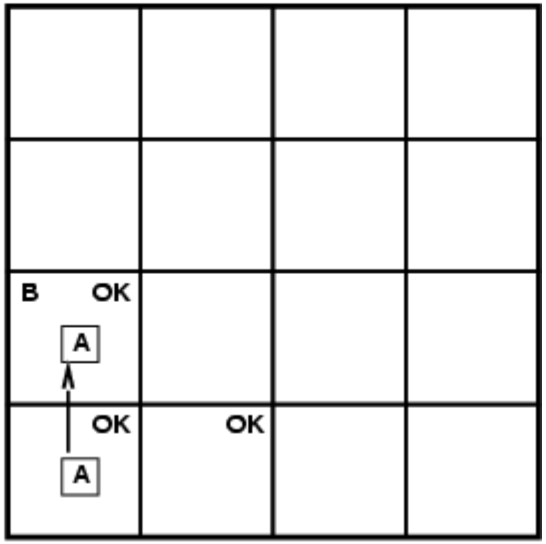


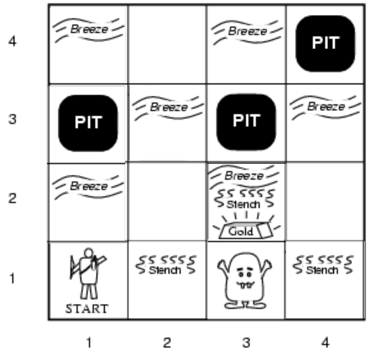


[N,N,N,N,N]

Agent concludes that there its neighboring states, [1,2] and [2,1], are safe squares

Exploring a wumpus world

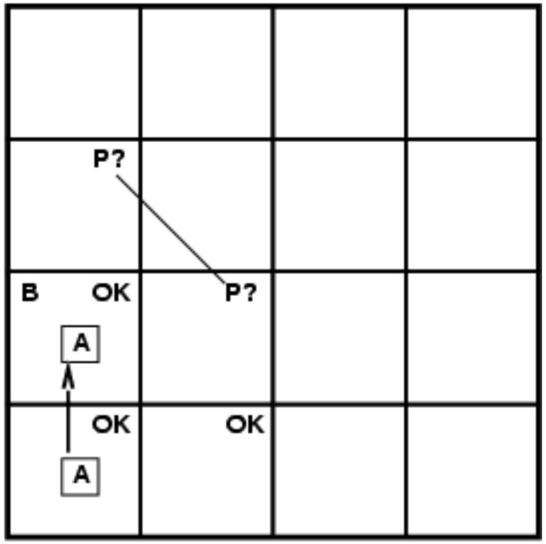


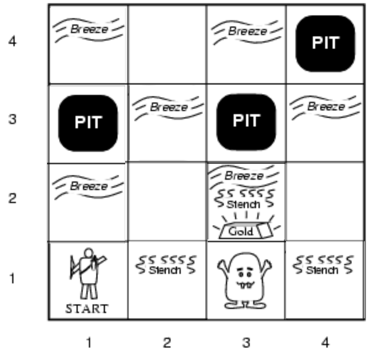


[N,B,N,N,N]

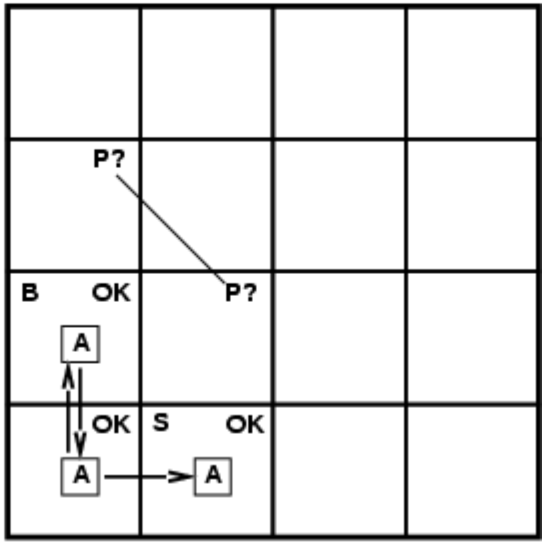
After one move

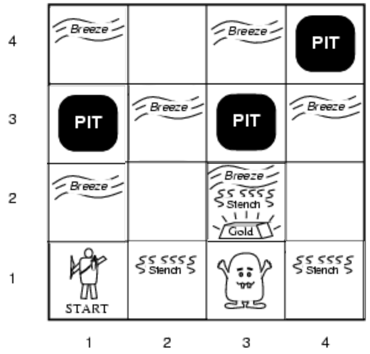
Exploring a wumpus world





Exploring a wumpus world

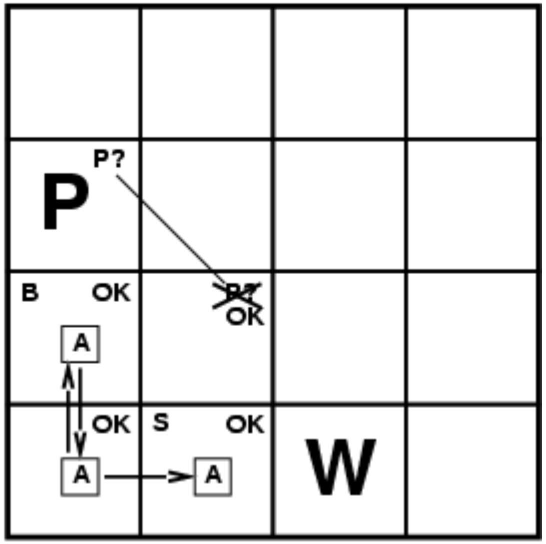


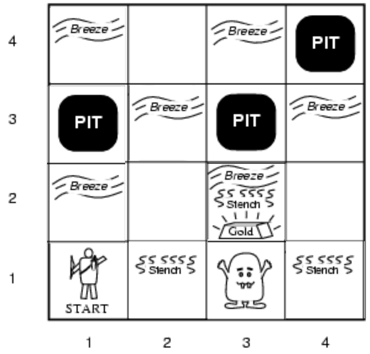


[S,N,N,N,N]

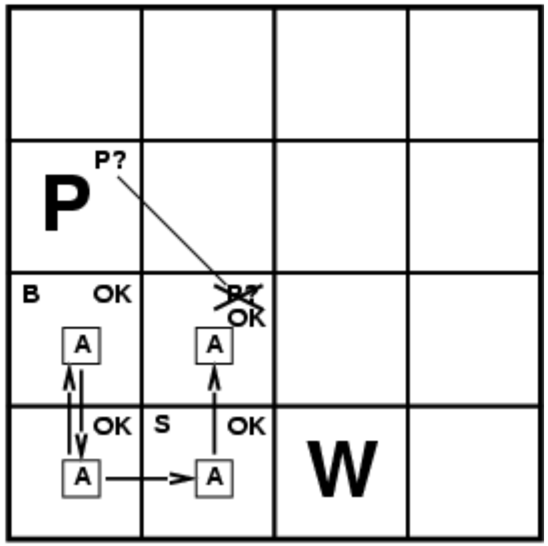
After three move of a cautious agent

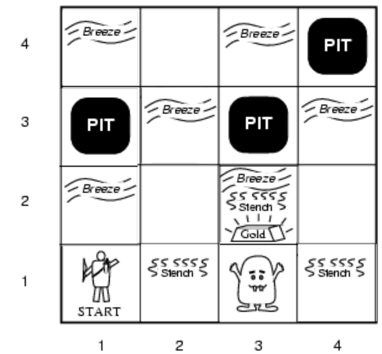
Exploring a wumpus world



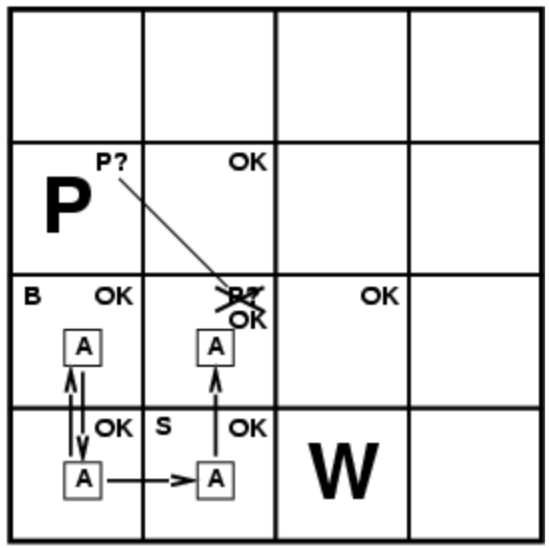


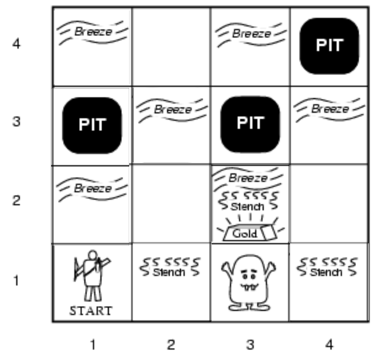
Exploring a wumpus world



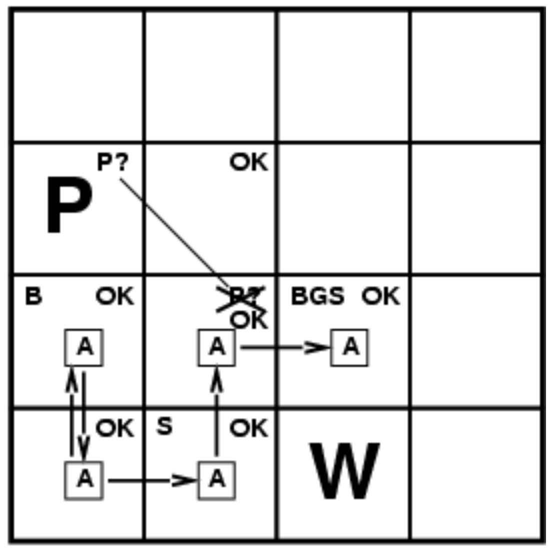


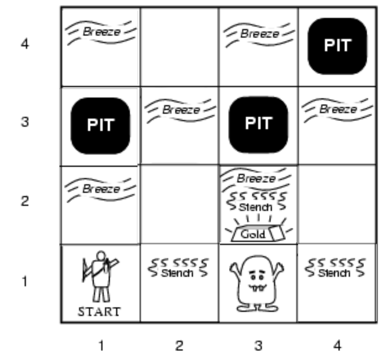
Exploring a wumpus world



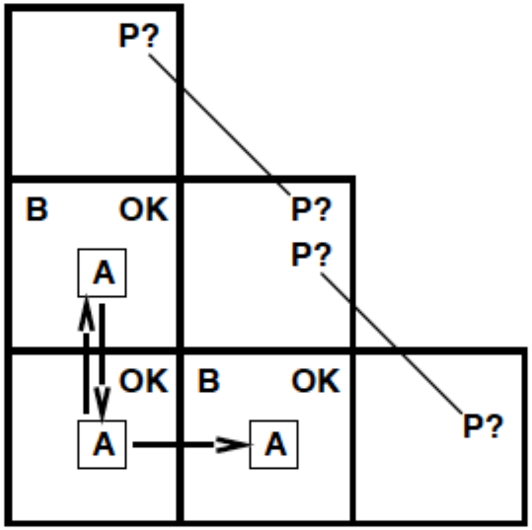


Exploring a wumpus world





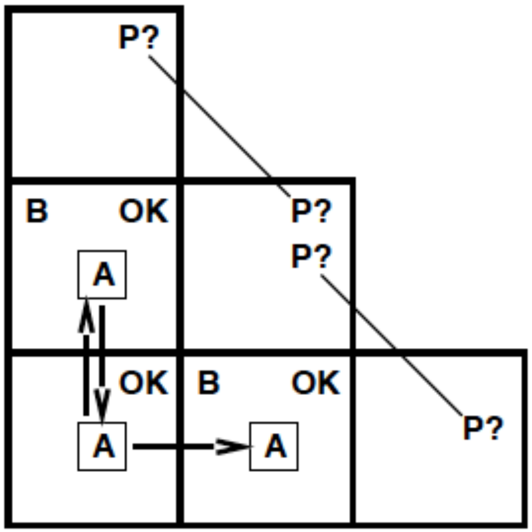
Other tight spots



Breeze in (1,2) and (2,1)
→ no safe actions

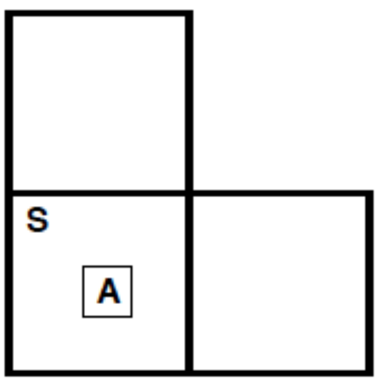
Assuming pits uniformly distributed, (2,2) has pit w/ prob 0.86, vs. 0.31

Other tight spots



Breeze in (1,2) and (2,1)
→ no safe actions

Assuming pits uniformly distributed, (2,2) has pit w/ prob 0.86, vs. 0.31



Smell in (1,1)
→cannot move

Can use a strategy of coercion:
shoot straight ahead
wumpus was there→dead→safe
wumpus wasn't there→safe

Logic in general

- Logics are formal languages for representing information such that conclusions can be drawn
- Syntax defines the sentences in the language
- Semantics define the "meaning" of sentences;
  - i.e., define truth of a sentence in a world
- *E.g.*, the language of arithmetic
  - $x+2 \geq y$ is a sentence; $x2+y > \{\}$ is not a sentence
  - $x+2 \geq y$ is true **iff** the number $x+2$ is no less than the number y
  - $x+2 \geq y$ is true in a world where $x = 7$, $y = 1$
  - $x+2 \geq y$ is false in a world where $x = 0$, $y = 6$

Entailment

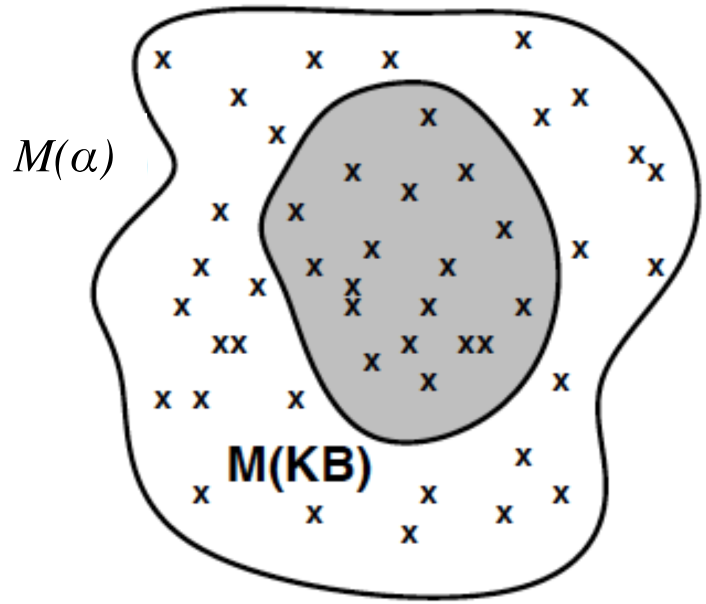
- Entailment means that one thing logically follows from another:

$$KB \models \alpha$$

- Knowledge base *KB* entails sentence α if and only if α is true in all worlds where *KB* is true
  – *E.g.*, the KB containing "the Giants won" and "the Reds won" entails "Either the Giants won or the Reds won"
  – *E.g.*, x+y = 4 entails 4 = x+y
  – Entailment is a relationship between sentences (*i.e.*, syntax) that is based on semantics

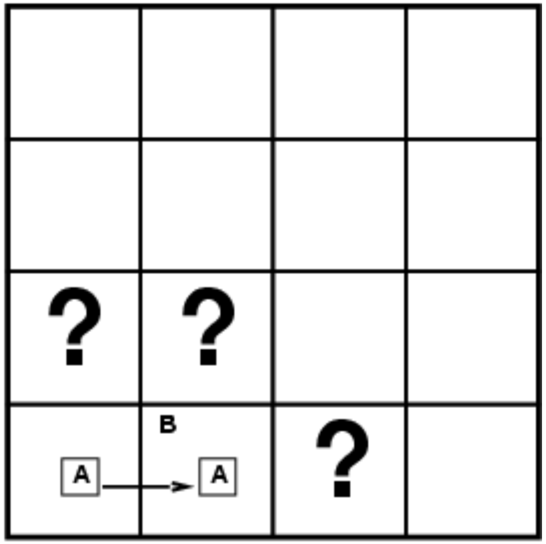
Models

- Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated
- We say m is a model of a sentence α if α is true in m
- $M(\alpha)$ is the set of all models of α
- Then $\mathrm{KB} \vDash \alpha$ **iff** $M(KB) \subseteq M(\alpha)$
  - E.g. *KB* = Giants won and Reds won
    α = Giants won



Entailment in the wumpus world

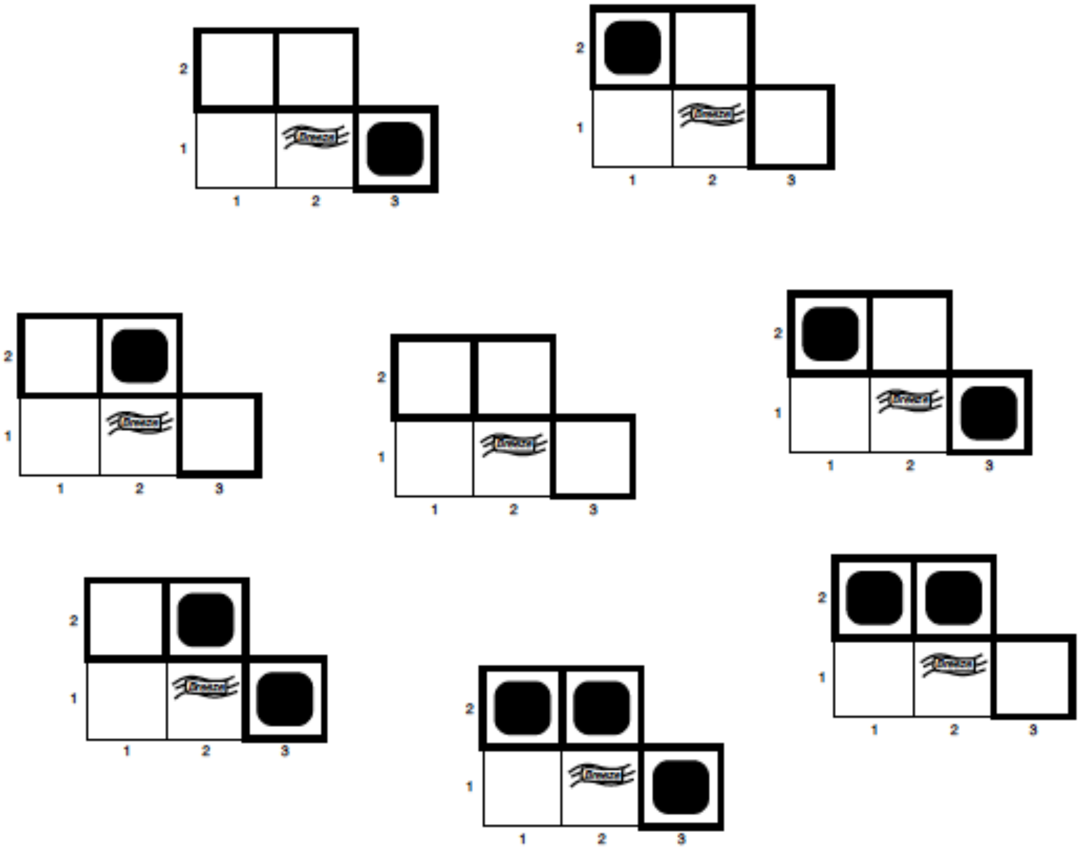
Situation after detecting nothing in [1,1], moving right, breeze in [2,1]



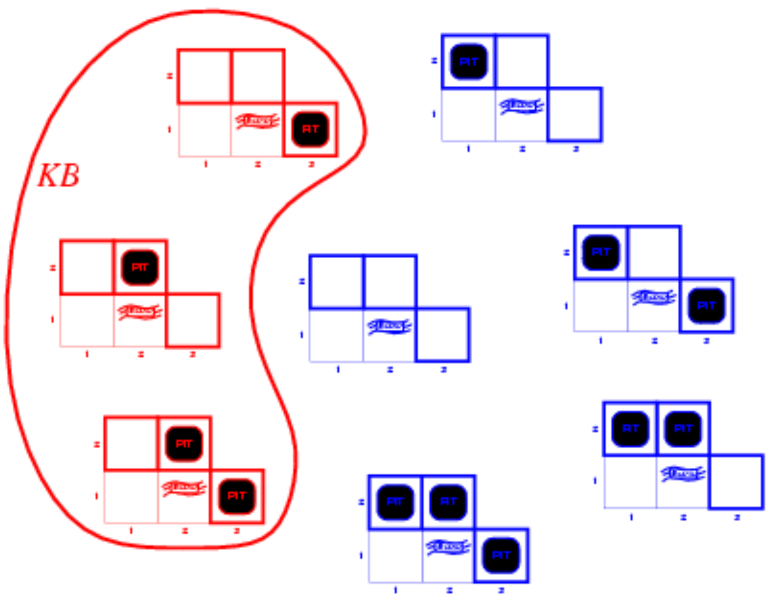
Consider possible models for *KB* assuming only pits

3 Boolean choices $\Rightarrow$ 8 possible models

Wumpus models

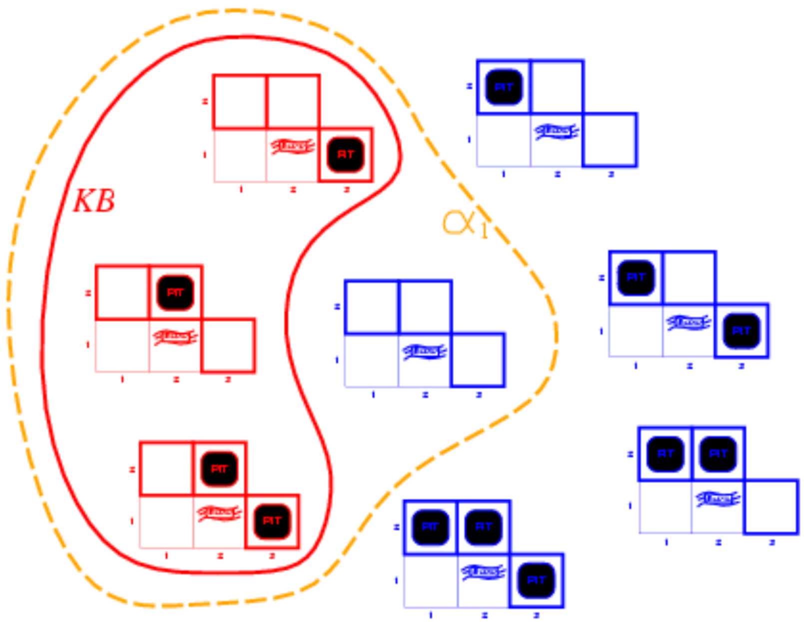


Wumpus models



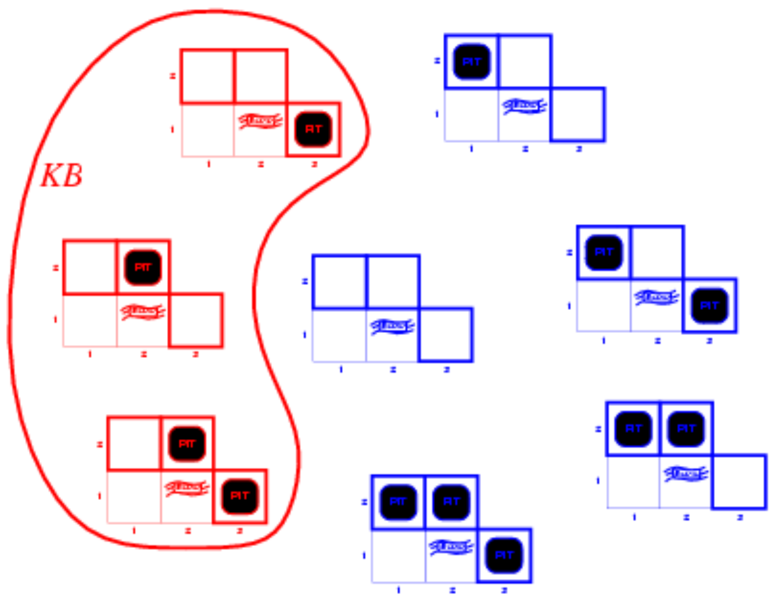
- *KB* = wumpus-world rules + observations

Wumpus models



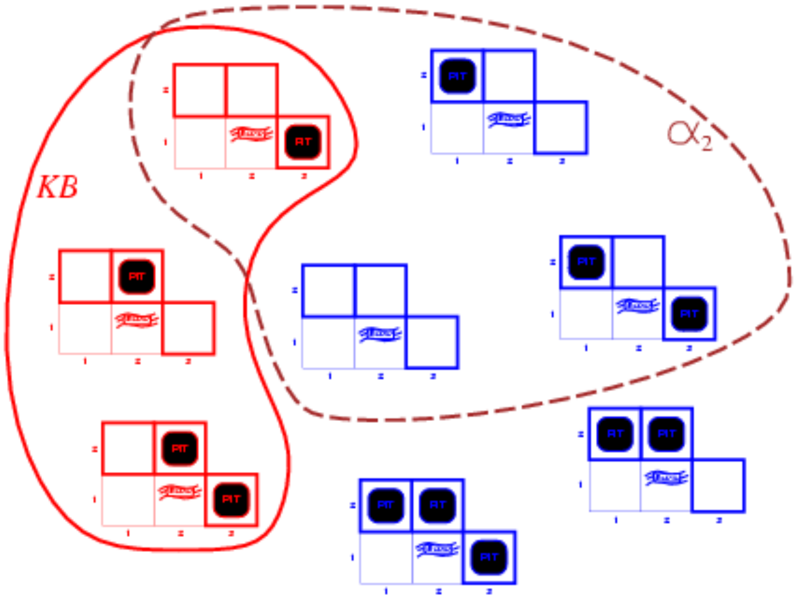
- *KB* = wumpus-world rules + observations
- α_1 = "[1,2] is safe", $KB \models \alpha_1$, proved by model checking

Wumpus models



- *KB* = wumpus-world rules + observations

Wumpus models



- *KB* = wumpus-world rules + observations
- α_2 = "[2,2] is safe", $KB \not\models \alpha_2$

Inference

- $KB \vdash_i \alpha$ = sentence α can be derived from KB by procedure i
- Soundness: i is sound if whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$
  - Also known as truth-preserving
  - An unsound inference "makes things up"
- 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*
- *If KB is true in real world, then any sentences derived from KB by a sound inference procedure is also true in the real world*

Propositional logic: Syntax

- Propositional logic is the simplest logic – illustrates basic ideas. But it is also powerful.

- Atomic sentences = single proposition symbols P_1, P_2 , *etc*
  - *True* is the always-true proposition
  - *False* is the always- false proposition
  - If S is a sentence, $\neg$S is a sentence (negation, NET)
  - If S_1 and S_2 are sentences, $S_1 \wedge S_2$ is a sentence (conjunction, AND)
  - If S_1 and S_2 are sentences, $S_1 \vee S_2$ is a sentence (disjunction, OR)
  - If S_1 and S_2 are sentences, $S_1 \Rightarrow S_2$ is a sentence (implication, IMPLIES)
  - If S_1 and S_2 are sentences, $S_1 \Leftrightarrow S_2$ is a sentence (biconditional, IFF)

Propositional logic: Semantics

Each model specifies true/false for each proposition symbol

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

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

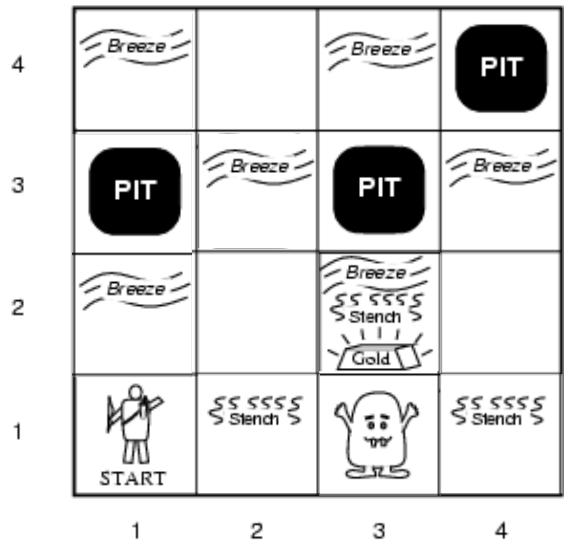
Truth tables for connectives

P	Q	$\neg P$	$P \land Q$	$P \lor Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
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*

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].

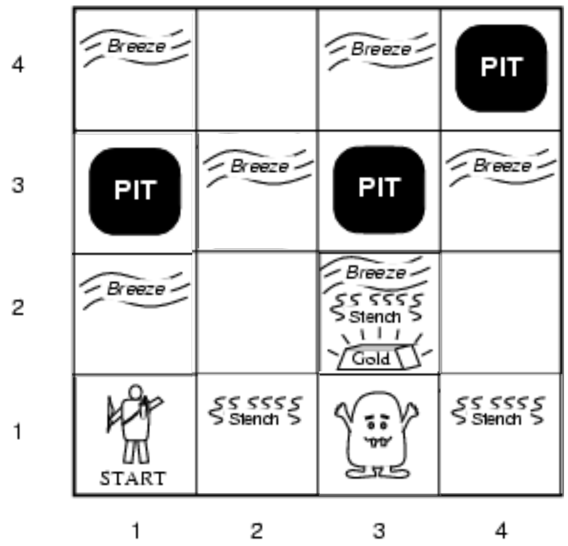
$\neg P_{1,1}$
$\neg B_{1,1}$
$B_{2,1}$



•

Wumpus world sentences

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



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

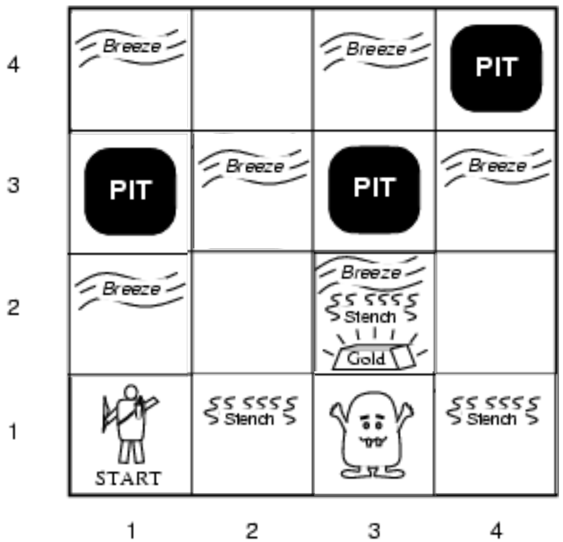
- "Pits cause breezes in adjacent squares"

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

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].

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



- "Pits cause breezes in adjacent squares"

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

“A square is breezy if and only if there is an adjacent pit"

Inference procedure

- Our goal is decide whether $KB \vDash \alpha$ for some sentence α
- A simple algorithm is a model-checking approach that implements directly the definition of entailment
- Idea: enumerate the models and check that α is true in every model KB is true
- Note: models are assignment of **true** or **false** to every propositional symbol
- For N symbols there are 2^N possible models

Truth tables for inference

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*false*	*true*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*false*	*true*	*true*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	<u>*true*</u>
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	<u>*true*</u>
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	<u>*true*</u>
false	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*false*	*true*	*false*

Enumerate rows (different assignments to symbols),
if KB is true in row, check that α is too

Inference by enumeration

- Depth-first enumeration of all models is sound and complete

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

    symbols ← a list of the proposition symbols in KB and α
    return TT-CHECK-ALL(KB, α, symbols, [])
```

```
function TT-CHECK-ALL(KB, α, symbols, model) returns true or false
    if EMPTY?(symbols) then
        if PL-TRUE?(KB, model) then return PL-TRUE?(α, model)
        else return true
    else do
        P ← FIRST(symbols); rest ← REST(symbols)
        return TT-CHECK-ALL(KB, α, rest, EXTEND(P, true, model)) and
                    TT-CHECK-ALL(KB, α, rest, EXTEND(P, false, model))
```

$O(2^n)$ for n symbols; problem is **co-NP-complete**

Logical equivalence

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

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

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

i.e., prove α by *reductio ad absurdum*

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

i.e., prove α by *reductio ad absurdum*

* A common form of argument which seeks to demonstrate that a statement is true by showing that a false, untenable, or absurd result follows from its denial

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 alg.

– Typically require translation of sentences into a normal form

Model checking

truth table enumeration (always exponential in n)

improved backtracking, e.g., Davis–Putnam–Logemann–Loveland

heuristic search in model space (sound but incomplete)

e.g., min-conflicts-like hill-climbing algorithms

Resolution

Conjunctive Normal Form (CNF—universal)

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

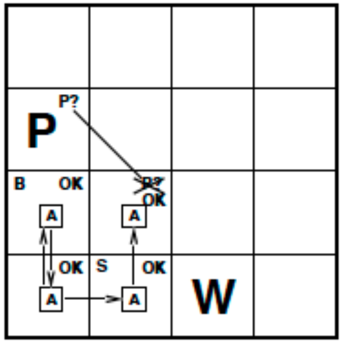
E.g., $(A \lor \neg B) \land (B \lor \neg C \lor \neg D)$

Resolution inference rule (for CNF): complete for propositional logic

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

where ℓ_i and m_j are complementary literals. E.g.,

$$\frac{P_{1,3} \lor P_{2,2}, \qquad \neg P_{2,2}}{P_{1,3}}$$



Resolution is sound and complete for propositional logic

Resolution

Conjunctive Normal Form (CNF—universal)

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

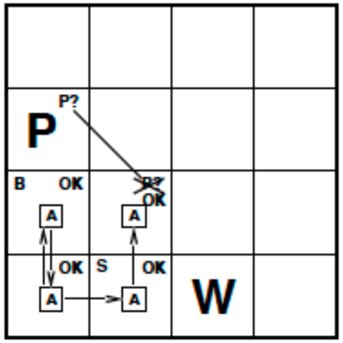
E.g., $(A \lor \neg B) \land (B \lor \neg C \lor \neg D)$

Resolution inference rule (for CNF): complete for propositional logic

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

where ℓ_i and m_j are complementary literals. E.g.,

$$\frac{P_{1,3} \lor P_{2,2}, \qquad \neg P_{2,2}}{P_{1,3}}$$



Resolution is sound and complete for propositional logic

Factoring: The removal of multiple copies of literals

Conversion to CNF

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

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

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

Conversion to CNF

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

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

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

2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$.

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

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 $\neg\alpha \lor \beta$.

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

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

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

Conversion to CNF

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

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

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

2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$.

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

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

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

4. Apply distributivity law ($\vee$ over $\wedge$) and flatten:

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

Resolution algorithm

Proof by contradiction, i.e., show $KB \wedge \neg\alpha$ unsatisfiable

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

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

Clauses resolve to yield the empty clause
↓ (to "then return true")

<—No new clauses that can be added to KB (beside "if new ⊆ clauses then return false")

Example: Using propositional logic to solve the crime

Let’s see how to use Propositional Logic and resolution to solve crime:

1. There are three suspects for a murder: Adams, Brown, and Clark.
2. Adams says “I didn’t do it. The victim was old acquaintance of Brown’s. But Clark hated him.”
3. Brown states “I didn’t do it. I didn’t know the guy. Besides I was out of town all the week.”
4. Clark says “I didn’t do it. I saw both Adams and Brown downtown with the victim that day; one of them must have done it.”

Adapted from materials of CS 188: Artificial Intelligence (UC Berkeley, instructor: Prof. S. Narayan), Spring 2007

Example: Using propositional logic to solve the crime

Let's see how to use Propositional Logic and resolution to solve crime:

1. There are three suspects for a murder: Adams, Brown, and Clark.

2. Adams says "I didn't do it. The victim was old acquaintance of Brown's. But Clark hated him."

3. Brown states "I didn't do it. I didn't know the guy. Besides I was out of town all the week."

4. Clark says "I didn't do it. I saw both Adams and Brown downtown with the victim that day; one of them must have done it."

5. Assume that the two innocent men are telling the truth, but that the guilty man might not be.

Who is the KILLER???

Adapted from materials of CS 188: Artificial Intelligence (UC Berkeley, instructor: Prof. S. Narayan), Spring 2007

Solving the crime

The key to the solution is to make the suspect's statements (true/false) conditional on their innocence. The following rules/axioms are sufficient:

- Let Adams, Brown, and Clark be A, B, and C respectively. Let V be the victim.
- Let I(A) be the proposition that A is innocent, I(B) the proposition that B is innocent, and I(C) the proposition that C is innocent.
- Let F(A, V) indicate that A is a friend (acquaintance) of V and L(A,V) indicate that A liked V.
- Let W(A,V) be the proposition that A was with V on the day of the murder and W(B,V) the proposition that B was with V on that day.
- Let T(B) be the proposition that B was in town on the day of the murder.
- Let K(B,V) be the proposition that B knows V.

Adapted from materials of CS 188: Artificial Intelligence (UC Berkeley, instructor: Prof. S. Narayan), Spring 2007

Rules

1. If A was innocent, then B was V's friend and C did not like V, just as A said.
   1.1. $I(A) \Rightarrow F(B,V)$
   1.2. $I(A) \Rightarrow \neg L(C,V)$

Rules

1. If A was innocent, then B was V's friend and C did not like V, just as A said.
   1.1. $I(A) \Rightarrow F(B,V)$
   1.2. $I(A) \Rightarrow \neg L(C,V)$
2. If B was innocent, then he wasn't in town and he doesn't know V
   2.1. $I(B) \Rightarrow \neg T(B)$
   2.2. $I(B) \Rightarrow \neg K(B, V)$

Adapted from materials of CS 188: Artificial Intelligence (UC Berkeley, instructor: Prof. S. Narayan), Spring 2007

Rules

1. If A was innocent, then B was V's friend and C did not like V, just as A said.
   1.1. $I(A) \Rightarrow F(B,V)$
   1.2. $I(A) \Rightarrow \neg L(C,V)$
2. If B was innocent, then he wasn't in town and he doesn't know V
   2.1. $I(B) \Rightarrow \neg T(B)$
   2.2. $I(B) \Rightarrow \neg K(B, V)$
3. If C was innocent, then A was with V and B was with V
   3.1. $I(C) \Rightarrow W(A,V)$
   3.2. $I(C) \Rightarrow W(B,V)$

Adapted from materials of CS 188: Artificial Intelligence (UC Berkeley, instructor: Prof. S. Narayan), Spring 2007

Rules

1. If A was innocent, then B was V's friend and C did not like V, just as A said.
   1.1. $I(A) \Rightarrow F(B,V)$
   1.2. $I(A) \Rightarrow \neg L(C,V)$
2. If B was innocent, then he wasn't in town and he doesn't know V
   2.1. $I(B) \Rightarrow \neg T(B)$
   2.2. $I(B) \Rightarrow \neg K(B, V)$
3. If C was innocent, then A was with V and B was with V
   3.1. $I(C) \Rightarrow W(A,V)$
   3.2. $I(C) \Rightarrow W(B,V)$
4. Now some general knowledge rules in our KB[1]
   4.1. $W(B,V) \Rightarrow T(B)$
   4.2. $F(B,V) \Rightarrow K(B,V)$
   4.3. $L(C,V) \Rightarrow K(C,V)$

Rules

1. If A was innocent, then B was V's friend and C did not like V, just as A said.
   1.1. $I(A) \Rightarrow F(B,V)$
   1.2. $I(A) \Rightarrow \neg L(C,V)$
2. If B was innocent, then he wasn't in town and he doesn't know V
   2.1. $I(B) \Rightarrow \neg T(B)$
   2.2. $I(B) \Rightarrow \neg K(B, V)$
3. If C was innocent, then A was with V and B was with V
   3.1. $I(C) \Rightarrow W(A,V)$
   3.2. $I(C) \Rightarrow W(B,V)$
4. Now some general knowledge rules in our KB[1]
   4.1. $W(B,V) \Rightarrow T(B)$
   4.2. $F(B,V) \Rightarrow K(B,V)$
   4.3. $L(C,V) \Rightarrow K(C,V)$
5. Now we assert that only one of A, B, and C is the guilty person
   5.1. $I(A) \vee I(B)$
   5.2. $I(A) \vee I(C)$
   5.3. $I(B) \vee I(C)$

Adapted from materials of CS 188: Artificial Intelligence (UC Berkeley, instructor: Prof. S. Narayan), Spring 2007

Converting to CNF form

1. $\{\neg I(A), F(B, V)\}$
2. $\{\neg I(A), \neg L(C, V)\}$
3. $\{\neg I(B), \neg T(B)\}$
4. $\{\neg I(B), \neg K(B, V)\}$
5. $\{\neg I(C), W(A, V)\}$
6. $\{\neg I(C), W(B, V)\}$
7. $\{\neg W(B, V), T(B)\}$
8. $\{\neg F(B, V), K(B, V)\}$
9. $\{\neg L(C, V), K(C, V)\}$
10. $\{I(A), I(B)\}$
11. $\{I(A), I(C)\}$
12. $\{I(B), I(C)\}$

Converting to CNF form

1. $\{\neg I(A), F(B, V)\}$
2. $\{\neg I(A), \neg L(C, V)\}$
3. $\{\neg I(B), \neg T(B)\}$
4. $\{\neg I(B), \neg K(B, V)\}$
5. $\{\neg I(C), W(A, V)\}$
6. $\{\neg I(C), W(B, V)\}$
7. $\{\neg W(B, V), T(B)\}$
8. $\{\neg F(B, V), K(B, V)\}$
9. $\{\neg L(C, V), K(C, V)\}$
10. $\{I(A), I(B)\}$
11. $\{I(A), I(C)\}$
12. $\{I(B), I(C)\}$

Adapted from materials of CS 188: Artificial Intelligence (UC Berkeley, instructor: Prof. S. Narayan), Spring 2007

Converting to CNF form

Initial Rules

1. $\{\neg I(A), F(B, V)\}$
2. $\{\neg I(A), \neg L(C, V)\}$
3. $\{\neg I(B), \neg T(B)\}$
4. $\{\neg I(B), \neg K(B, V)\}$
5. $\{\neg I(C), W(A, V)\}$
6. $\{\neg I(C), W(B, V)\}$
7. $\{\neg W(B, V), T(B)\}$
8. $\{\neg F(B, V), K(B, V)\}$
9. $\{\neg L(C, V), K(C, V)\}$
10. $\{I(A), I(B)\}$
11. $\{I(A), I(C)\}$
12. $\{I(B), I(C)\}$

1.1. $I(A) \Rightarrow F(B,V)$

1.2. $I(A) \Rightarrow \neg L(C,V)$

Adapted from materials of CS 188: Artificial Intelligence (UC Berkeley, instructor: Prof. S. Narayan), Spring 2007

Converting to CNF form

Initial Rules

Logical equivalency

1. $\{\neg I(A), F(B, V)\}$
2. $\{\neg I(A), \neg L(C, V)\}$
3. $\{\neg I(B), \neg T(B)\}$
4. $\{\neg I(B), \neg K(B, V)\}$
5. $\{\neg I(C), W(A, V)\}$
6. $\{\neg I(C), W(B, V)\}$
7. $\{\neg W(B, V), T(B)\}$
8. $\{\neg F(B, V), K(B, V)\}$
9. $\{\neg L(C, V), K(C, V)\}$
10. $\{I(A), I(B)\}$
11. $\{I(A), I(C)\}$
12. $\{I(B), I(C)\}$

1.1. $I(A) \Rightarrow F(B,V)$

1.2. $I(A) \Rightarrow \neg L(C,V)$

$$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$$

Adapted from materials of CS 188: Artificial Intelligence (UC Berkeley, instructor: Prof. S. Narayan), Spring 2007

Performing resolution to prove that B is guilty

Let's try to prove that B is the killer: $\neg$ I (B) (B did it)

Ideas:

- Start with a new clause {I(B)}
- Use the resolution inference rule for CNF:

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

where ℓ_i and m_j are complementary literals.

- Use the principle of proof by contradiction and come up with P and ¬P, which will resolve to an empty clause

Adapted from materials of CS 188: Artificial Intelligence (UC Berkeley, instructor: Prof. S. Narayan), Spring 2007

Resolutions

13. {I (B)}
14. { ¬ I (A), K (B, V)} -- RESOLVING Clauses 1 and 8
15. { ¬ I (C), T (B)} -- RESOLVING Clauses 6 and 7
16. { ¬ I (A), ¬ I (B)} -- RESOLVING 4 and 14
17. { ¬ I (C), ¬ I (B)} -- RESOLVING 3 and 15
18. {I (C), ¬ I (B)} -- RESOLVING 11 and 16
19. {¬ I (B)} – RESOLVING 17 and 18
20. {} – RESOLVING 13 and 19.

Thus, Brown is the KILLER!!!

Adapted from materials of CS 188: Artificial Intelligence (UC Berkeley, instructor: Prof. S. Narayan), Spring 2007

Expressiveness limitation of propositional logic

- KB contains "physics" sentences for every single square

- Need to explicitly specify for every time *t* and every location [*x*,*y*]

- Rapid proliferation of clauses

Summary

- Logical agents apply inference to a knowledge base to derive new information and make decisions
- Basic concepts of logic:
  - syntax: formal structure of sentences
  - semantics: truth of sentences *wrt* models
  - entailment: necessary truth of one sentence given another
  - inference: deriving sentences from other sentences
  - soundness: derivations produce only entailed sentences
  - completeness: derivations can produce all entailed sentences
- Wumpus world requires the ability to represent partial and negated information, reason by cases, *etc*.
- Resolution is complete for propositional logic
- Propositional logic lacks expressive power

Part II: Brief notes on First Order Logic

Basic principles of FOL

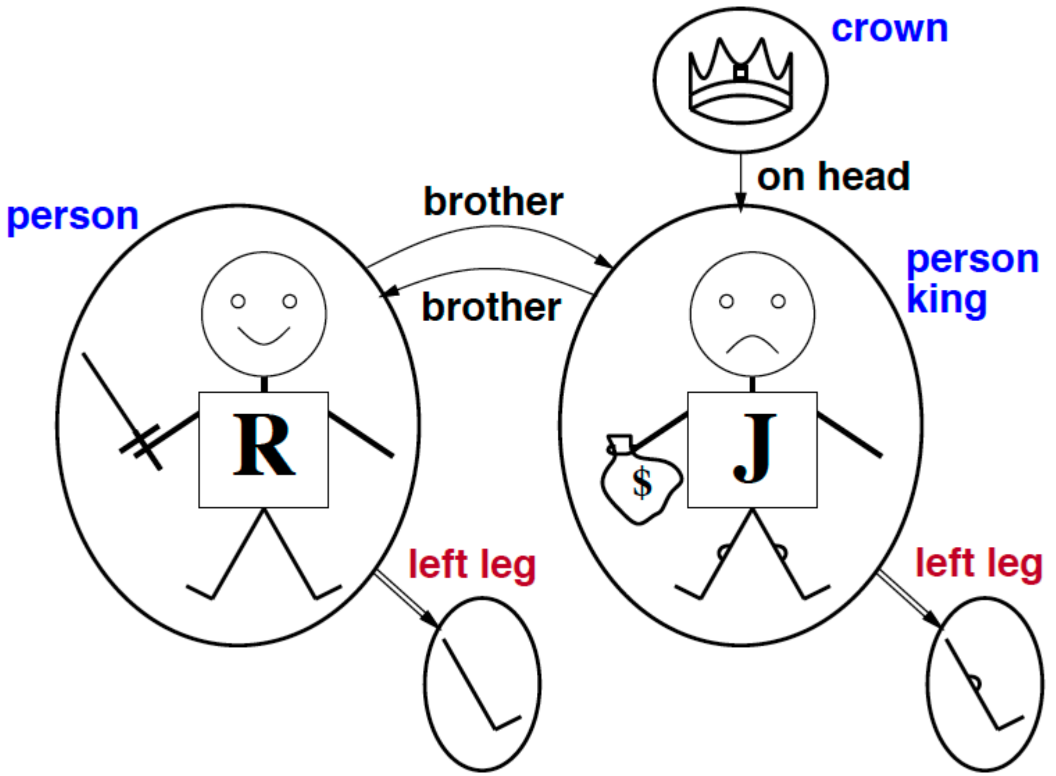
Whereas propositional logic assumes world contains **facts**,
first-order logic (like natural language) assumes the world contains

- Objects: people, houses, numbers, theories, Ronald McDonald, colors, baseball games, wars, centuries . . .
- Relations: red, round, bogus, prime, multistoried . . .,
  brother of, bigger than, inside, part of, has color, occurred after, owns, comes between, . . .
- Functions: father of, best friend, third inning of, one more than, end of . . .

PL and FOL are not the only logics available

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
Fuzzy logic	facts + degree of truth	known interval value

First order logic (FOL): Example



- Five objects: King John, Richard the Lionheart, left legs of Richard and John, and a crown
- Two binary relations: “brother” and “on head”
- Three unary relations: “person”, “king”, and “crown”
- One unary function: “left leg”

Syntax of FOL: Basics

Constants	$KingJohn,\ 2,\ UCB, \ldots$	→ *objects*
Predicates	$Brother,\ >, \ldots$	→ *relations*
Functions	$Sqrt,\ LeftLegOf, \ldots$	→ *functions*
Variables	$x,\ y,\ a,\ b, \ldots$	
Connectives	$\land \quad \lor \quad \neg \quad \Rightarrow \quad \Leftrightarrow$	
Equality	$=$	
Quantifiers	$\forall\ \exists$	

Syntax of FOL: Atomic sentences

Atomic sentence $=$ $predicate(term_1, \ldots, term_n)$
or $term_1 = term_2$

Term $=$ $function(term_1, \ldots, term_n)$
or $constant$ or $variable$

E.g., $Brother(KingJohn, RichardTheLionheart)$
$> (Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))$

Syntax of FOL: Complex sentences

Complex sentences are made from atomic sentences using connectives

$\neg S, \quad S_1 \wedge S_2, \quad S_1 \vee S_2, \quad S_1 \Rightarrow S_2, \quad S_1 \Leftrightarrow S_2$

E.g. $Sibling(KingJohn, Richard) \Rightarrow Sibling(Richard, KingJohn)$
$>(1,2) \vee \leq(1,2)$
$>(1,2) \wedge \neg>(1,2)$

Syntax of FOL: Truth

Sentences are true with respect to a model and an interpretation

Model contains ≥ 1 objects (domain elements) and relations among them

Interpretation specifies referents for

constant symbols $\rightarrow$ objects

predicate symbols $\rightarrow$ relations

function symbols $\rightarrow$ functional relations

An atomic sentence $predicate(term_1, \ldots, term_n)$ is true
iff the objects referred to by $term_1, \ldots, term_n$
are in the relation referred to by $predicate$

Truth Example

Consider the interpretation in which

$Richard \rightarrow$ Richard the Lionheart

$John \rightarrow$ the evil King John

$Brother \rightarrow$ the brotherhood relation

Under this interpretation, $Brother(Richard, John)$ is true just in case Richard the Lionheart and the evil King John are in the brotherhood relation in the model

Materials

- First order logic:
  - Chapter 8 (Basic)
  - Chapter 9 (Advanced)

Basic principles of FOL

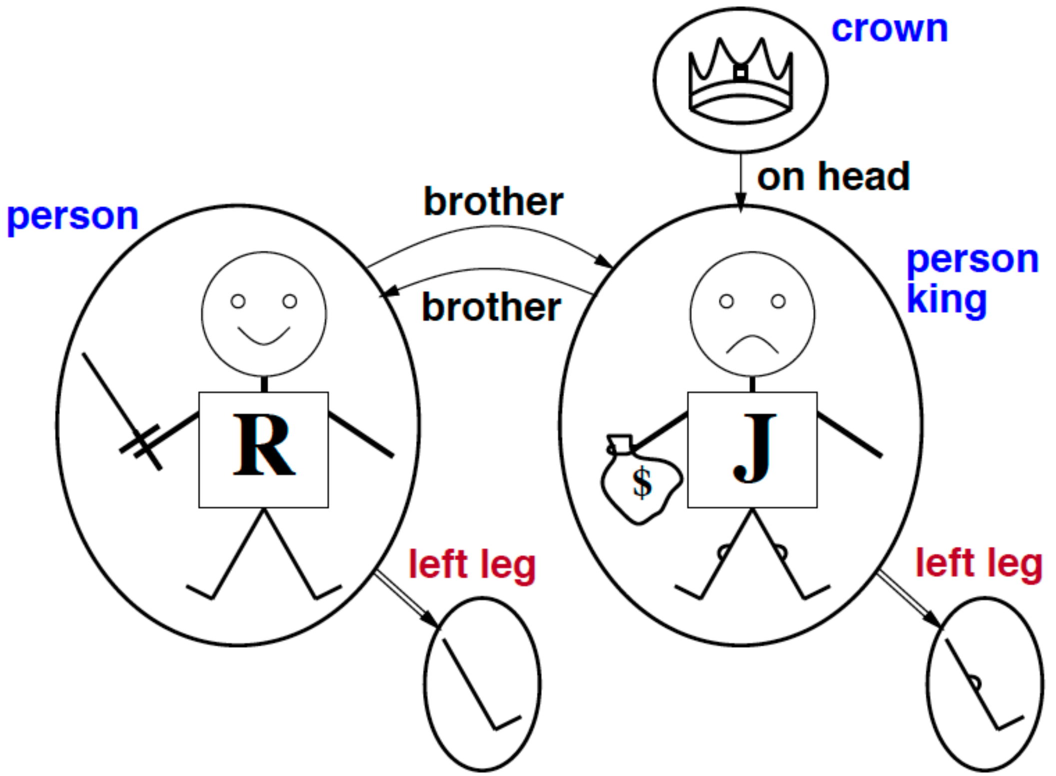
Whereas propositional logic assumes world contains **facts**,
first-order logic (like natural language) assumes the world contains

- Objects: people, houses, numbers, theories, Ronald McDonald, colors, baseball games, wars, centuries . . .
- Relations: red, round, bogus, prime, multistoried . . .,
  brother of, bigger than, inside, part of, has color, occurred after, owns, comes between, . . .
- Functions: father of, best friend, third inning of, one more than, end of . . .

PL and FOL are not the only logics available

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
Fuzzy logic	facts + degree of truth	known interval value

First order logic (FOL): Example



- Five objects: King John, Richard the Lionheart, left legs of Richard and John, and a crown
- Two binary relations: “brother” and “on head”
- Three unary relations: “person”, “king”, and “crown”
- One unary function: “left leg”

Syntax of FOL: Basics

Constants	$KingJohn,\ 2,\ UCB, \ldots$	→ *objects*
Predicates	$Brother,\ >, \ldots$	→ *relations*
Functions	$Sqrt,\ LeftLegOf, \ldots$	→ *functions*
Variables	$x,\ y,\ a,\ b, \ldots$	
Connectives	$\wedge \quad \vee \quad \neg \quad \Rightarrow \quad \Leftrightarrow$	
Equality	$=$	
Quantifiers	$\forall\ \exists$	

Syntax of FOL: Atomic sentences

$$\begin{aligned} \text{Atomic sentence} &= predicate(term_1, \ldots, term_n) \\ &\quad \text{or } term_1 = term_2 \\ \\ \text{Term} &= function(term_1, \ldots, term_n) \\ &\quad \text{or } constant \text{ or } variable \end{aligned}$$

E.g., $Brother(KingJohn, RichardTheLionheart)$
$> (Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))$

Syntax of FOL: Complex sentences

Complex sentences are made from atomic sentences using connectives

$$\neg S, \quad S_1 \wedge S_2, \quad S_1 \vee S_2, \quad S_1 \;\Rightarrow\; S_2, \quad S_1 \;\Leftrightarrow\; S_2$$

E.g. $Sibling(KingJohn, Richard) \;\Rightarrow\; Sibling(Richard, KingJohn)$
$>(1,2) \vee \leq(1,2)$
$>(1,2) \wedge \neg{>}(1,2)$

Syntax of FOL: Truth

Sentences are true with respect to a model and an interpretation

Model contains ≥ 1 objects (domain elements) and relations among them

Interpretation specifies referents for

constant symbols $\rightarrow$ objects

predicate symbols $\rightarrow$ relations

function symbols $\rightarrow$ functional relations

An atomic sentence $predicate(term_1, \ldots, term_n)$ is true
iff the objects referred to by $term_1, \ldots, term_n$
are in the relation referred to by $predicate$

Truth Example

Consider the interpretation in which

$Richard \rightarrow$ Richard the Lionheart

$John \rightarrow$ the evil King John

$Brother \rightarrow$ the brotherhood relation

Under this interpretation, $Brother(Richard, John)$ is true just in case Richard the Lionheart and the evil King John are in the brotherhood relation in the model

Models for FOL

Entailment in propositional logic can be computed by enumerating models

We **can** enumerate the FOL models for a given KB vocabulary:

For each number of domain elements n from 1 to ∞
    For each k-ary predicate P_k in the vocabulary
        For each possible k-ary relation on n objects
            For each constant symbol C in the vocabulary
                For each choice of referent for C from n objects ...

Computing entailment by enumerating FOL models is not easy!

Quantifiers

- Expressing properties of entire collection of objects
- No need to enumerate the objects by name
- In FOL, there are two standard quantifiers: universal and existential

Universal quantification

$\forall \langle variables \rangle \; \langle sentence \rangle$

Everyone at Berkeley is smart:
$\forall x \;\; At(x, Berkeley) \;\Rightarrow\; Smart(x)$

$\forall x \;\; P$ is true in a model m iff P is true with x being **each** possible object in the model

Roughly speaking, equivalent to the conjunction of instantiations of P

$$
\begin{aligned}
& (At(KingJohn, Berkeley) \;\Rightarrow\; Smart(KingJohn)) \\
\wedge\; & (At(Richard, Berkeley) \;\Rightarrow\; Smart(Richard)) \\
\wedge\; & (At(Berkeley, Berkeley) \;\Rightarrow\; Smart(Berkeley)) \\
\wedge\; & \ldots
\end{aligned}
$$

A common mistake

Typically, $\Rightarrow$ is the main connective with $\forall$

Common mistake: using $\wedge$ as the main connective with $\forall$:

$$\forall x \;\; At(x, Berkeley) \wedge Smart(x)$$

means "Everyone is at Berkeley and everyone is smart"

Existential quantification

$\exists \langle variables \rangle \;\; \langle sentence \rangle$

Someone at Stanford is smart:
$\exists x \;\; At(x, Stanford) \land Smart(x)$

$\exists x \;\; P$ is true in a model m iff P is true with x being **some** possible object in the model

Roughly speaking, equivalent to the disjunction of instantiations of P

$$\begin{aligned} & (At(KingJohn, Stanford) \land Smart(KingJohn)) \\ \lor \; & (At(Richard, Stanford) \land Smart(Richard)) \\ \lor \; & (At(Stanford, Stanford) \land Smart(Stanford)) \\ \lor \; & \ldots \end{aligned}$$

Another common mistake

Typically, $\wedge$ is the main connective with $\exists$

Common mistake: using $\Rightarrow$ as the main connective with $\exists$:

$$\exists x \;\; At(x, Stanford) \Rightarrow Smart(x)$$

is true if there is anyone who is not at Stanford!

Properties of quantifiers

$\forall x \;\; \forall y$ is the same as $\forall y \;\; \forall x$ (why??)

$\exists x \;\; \exists y$ is the same as $\exists y \;\; \exists x$ (why??)

$\exists x \;\; \forall y$ is **not** the same as $\forall y \;\; \exists x$

$\exists x \;\; \forall y \;\; Loves(x, y)$
"There is a person who loves everyone in the world"

$\forall y \;\; \exists x \;\; Loves(x, y)$
"Everyone in the world is loved by at least one person"

Quantifier duality: each can be expressed using the other

$\forall x \;\; Likes(x, IceCream) \qquad \neg\exists x \;\; \neg Likes(x, IceCream)$

$\exists x \;\; Likes(x, Broccoli) \qquad \neg\forall x \;\; \neg Likes(x, Broccoli)$

Acknowledgements

- Lecture materials are based on:
  - The textbook
  - Lecture materials by Stuart Russell (the co-author of the textbook) in Berkeley