Logical agents

5

5 Logical Agents

5.1 Knowledge-based agents

5.2 Propositional logic

5.3 SAT problem

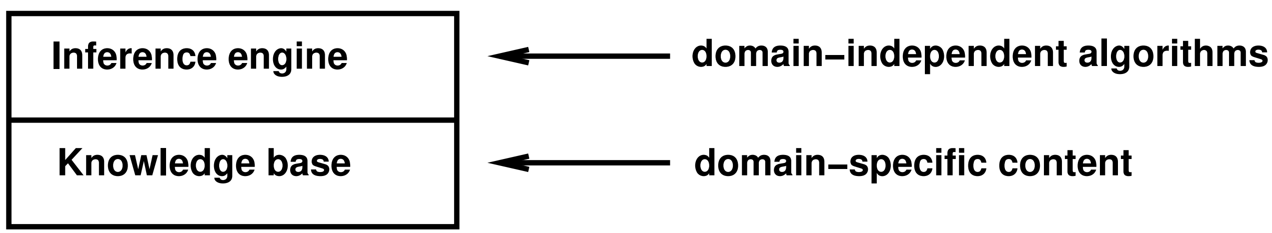
5.4 First-order logic

5.5 Logical foundation of AI

Knowledge-based agents

Logic: a study of thought, rational part of intelligence
Knowledge: power of thinking

Before building knowledge-based systems (agents)
before there can be learning, reasoning, planning, $\cdots$
need to be able to represent knowledge
Need a <u>formal</u> (precise declarative) language $\rightarrow$ logical language



Knowledge base (KB) = set of sentences in a logical language
Inference engine (IE) = algorithms by logical reasoning

Using logic: – no universal language / How about English or Chinese?

knowledge level

Declarative approach to building an agent
TELL it what it needs to know
— into KB

Then it can ASK itself what to do
— from KB

Agents can be viewed at the knowledge level
i.e., **what they know**, regardless of how implemented

Or at the implementation level
i.e., data structure in KB and algorithms that manipulate them

A simple knowledge-based agent

```
def KB-AGENT( percept)
    persistent: KB, a knowledge base
                t, a counter, initially 0, indicating time

    TELL(KB, MAKE-PERCEPT-SENTENCE( percept, t))
    action ← ASK(KB, MAKE-ACTION-QUERY(t))
    TELL(KB, MAKE-ACTION-SENTENCE(action, t))
    t ← t + 1
    return action
```

The agent must be able to

- represent states, actions, etc.
- incorporate new percepts
- update internal representations of the world
- deduce hidden properties of the world
- deduce appropriate actions

Example: Wumpus World

Performance measure

gold +1000, death -1000

-1 per step, -10 for using the arrow

Environment

Squares adjacent to wumpus are smelly

Squares adjacent to the pit are breezy

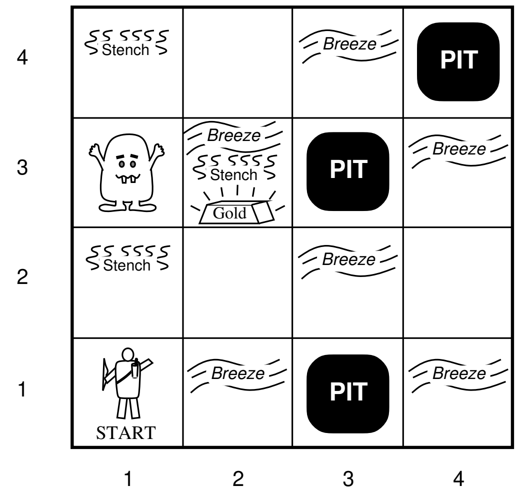
Glitter iff gold is in the samesquare

Shooting kills wumpus if you are facing it

Shooting uses up the only arrow

Grabbing picks up gold if in the square

Releasing drops the gold in the square



Actuators Left turn, Right turn,

Forward, Grab, Release, Shoot

Sensors Breeze, Glitter, Smell

Wumpus world characterization

Observable??

Wumpus world characterization

Observable?? No — only local perception

Deterministic??

Wumpus world characterization

Observable?? No — only local perception

Deterministic?? Yes — outcomes exactly specified

Episodic??

Wumpus world characterization

Observable?? No — only local perception

Deterministic?? Yes — outcomes exactly specified

Episodic?? No — sequential at the level of actions

Static??

Wumpus world characterization

Observable?? No — only local perception

Deterministic?? Yes — outcomes exactly specified

Episodic?? No — sequential at the level of actions

Static?? Yes — Wumpus and Pits do not move

Discrete??

Wumpus world characterization

Observable?? No — only local perception

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??

Wumpus world characterization

Observable?? No — only local perception

Deterministic?? Yes — outcomes exactly specified

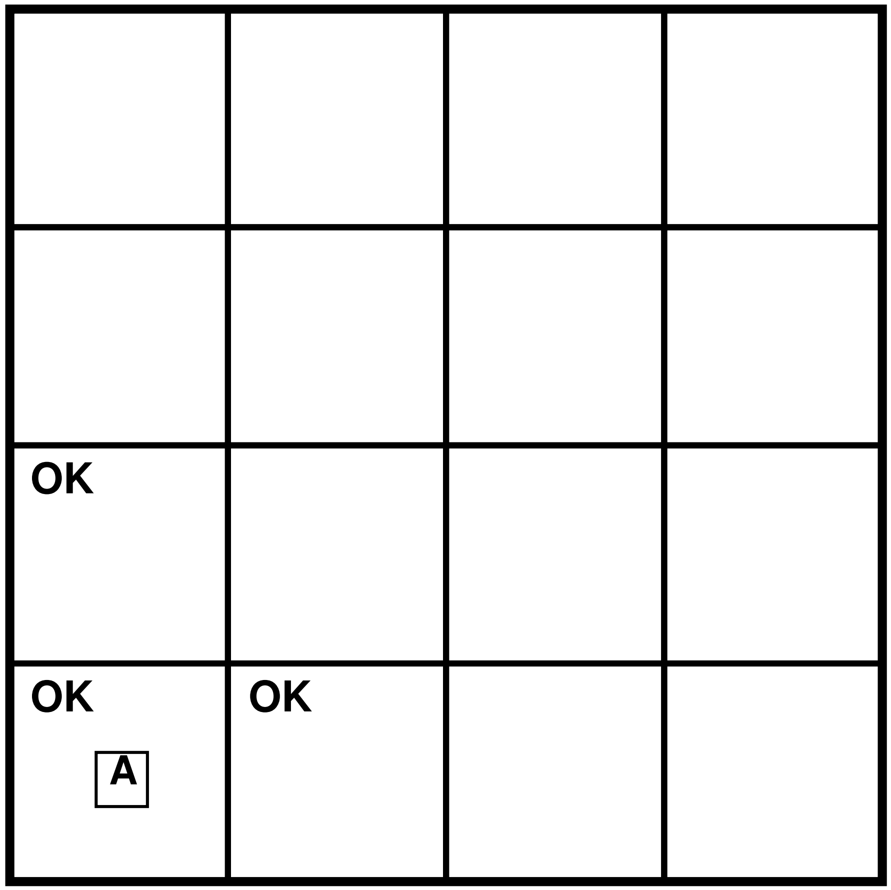
Episodic?? No — sequential at the level of actions

Static?? Yes — Wumpus and Pits do not move

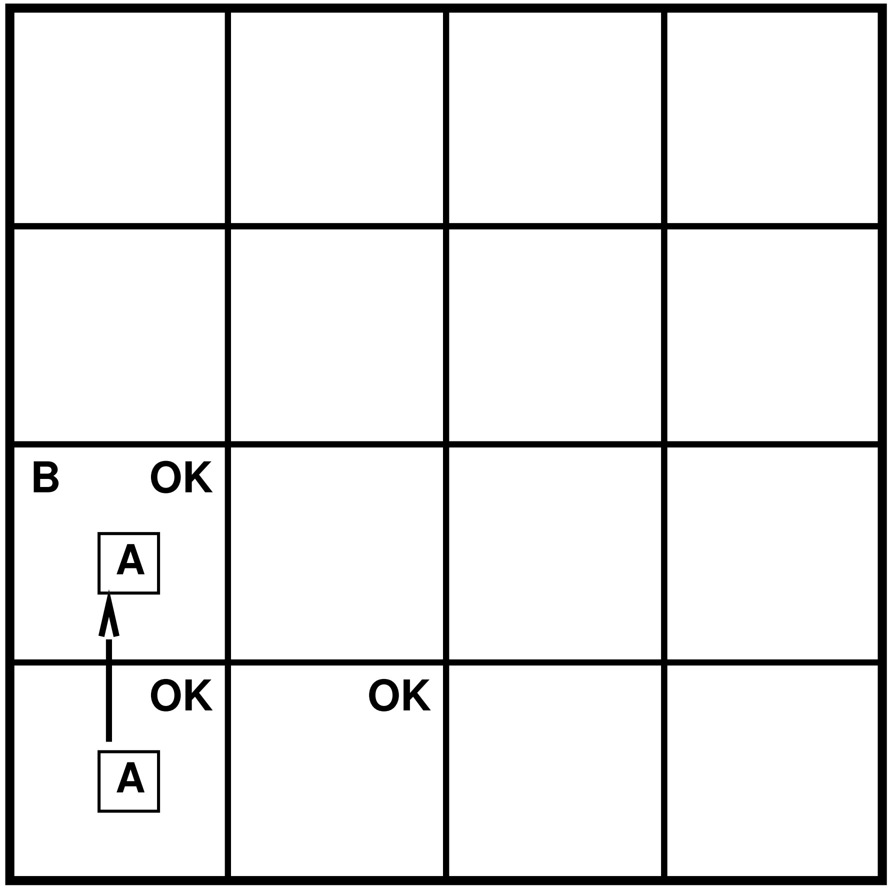
Discrete?? Yes

Single-agent?? Yes — Wumpus is essentially a natural feature

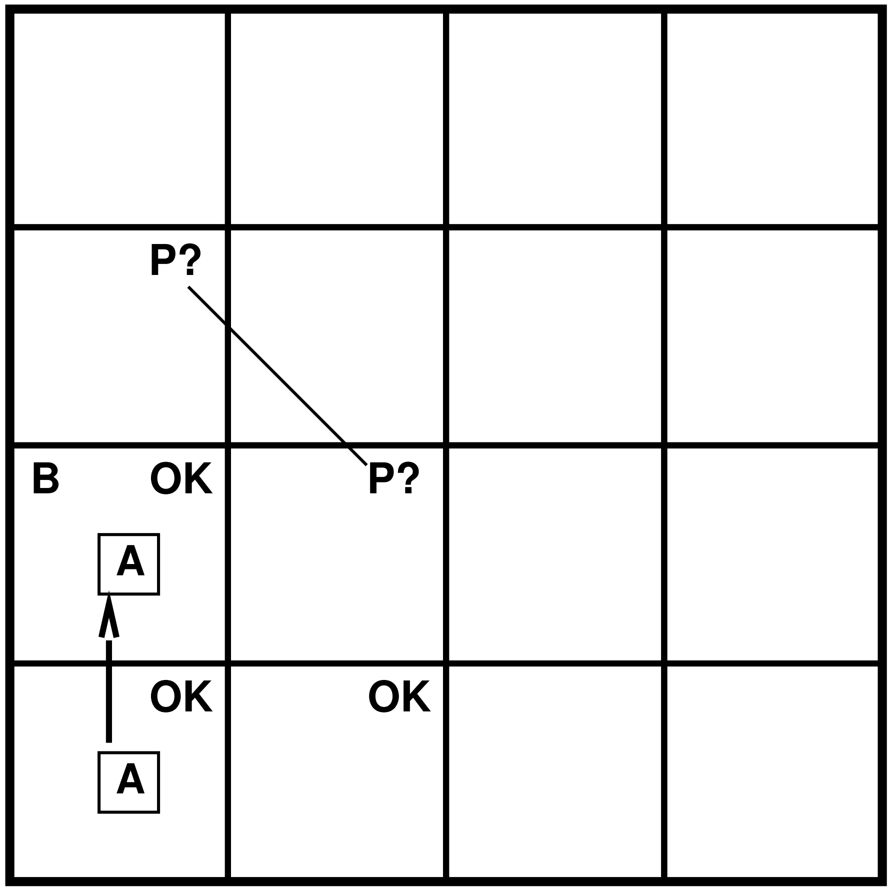
Exploring a wumpus world



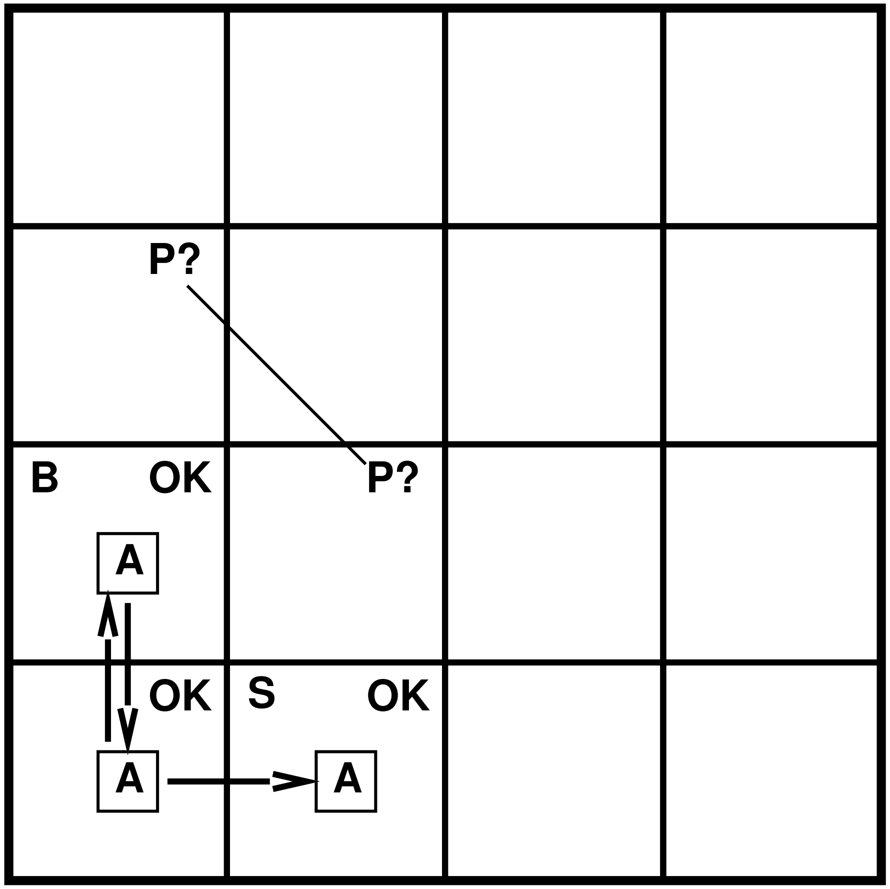
Exploring a wumpus world



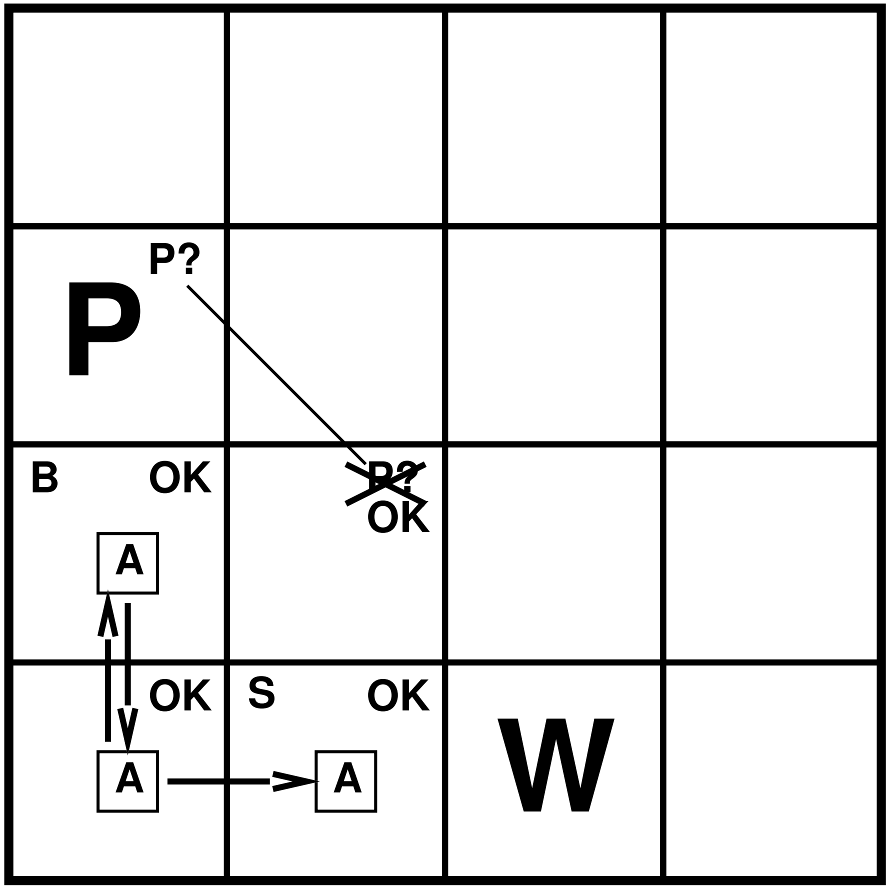
Exploring a wumpus world



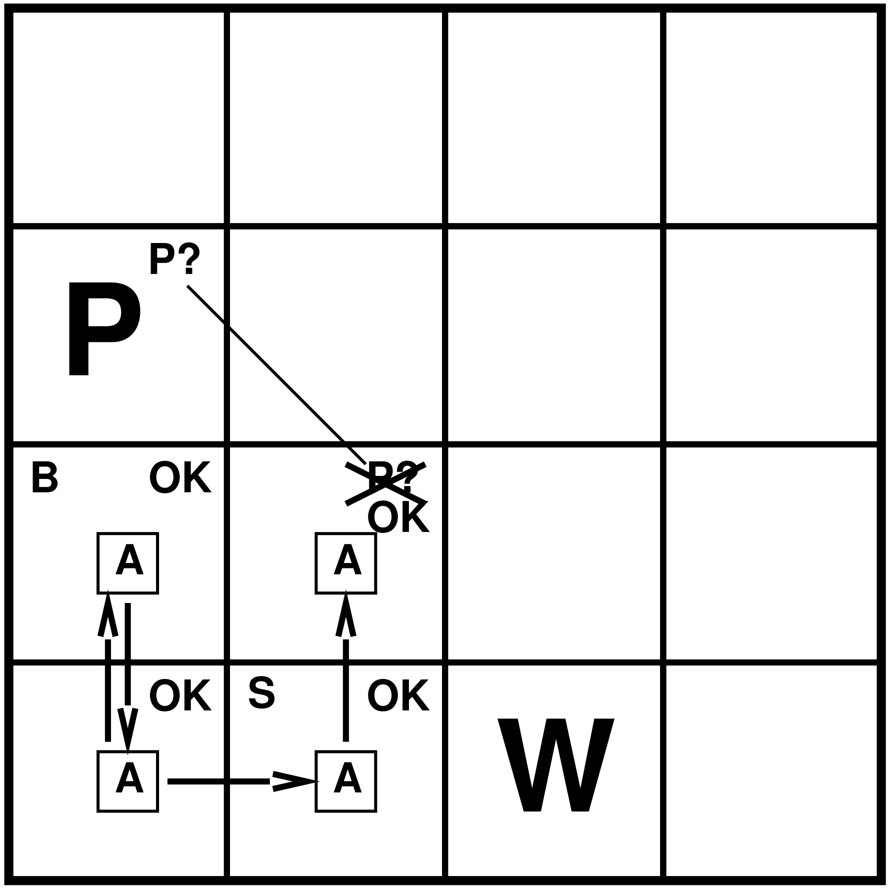
Exploring a wumpus world



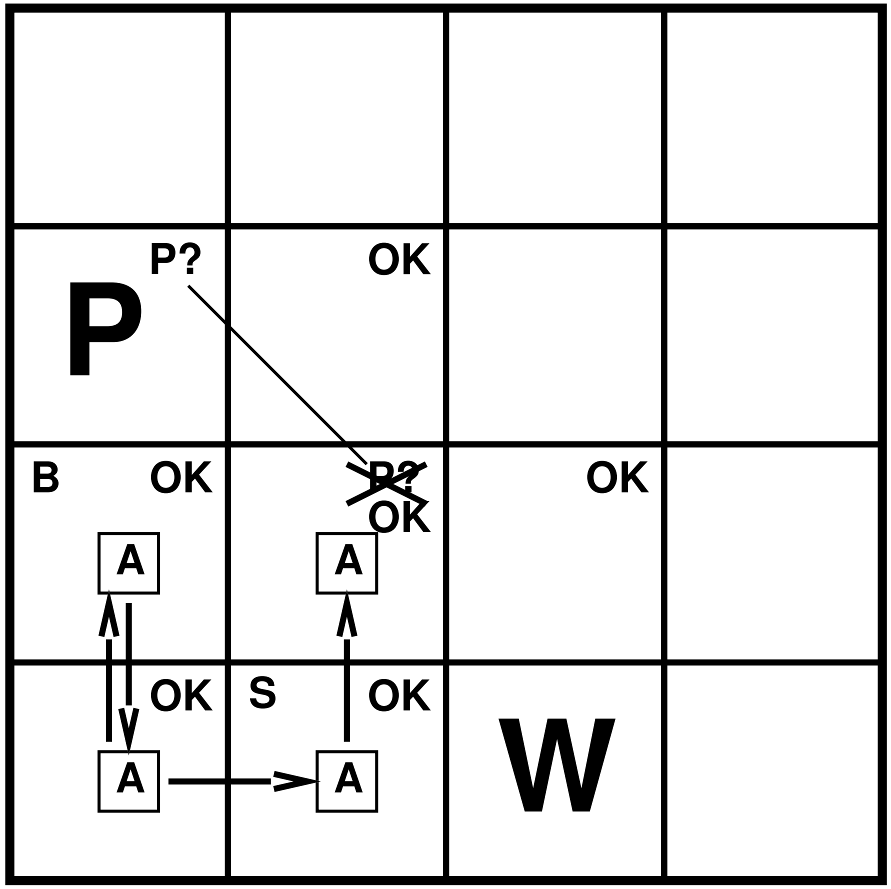
Exploring a wumpus world



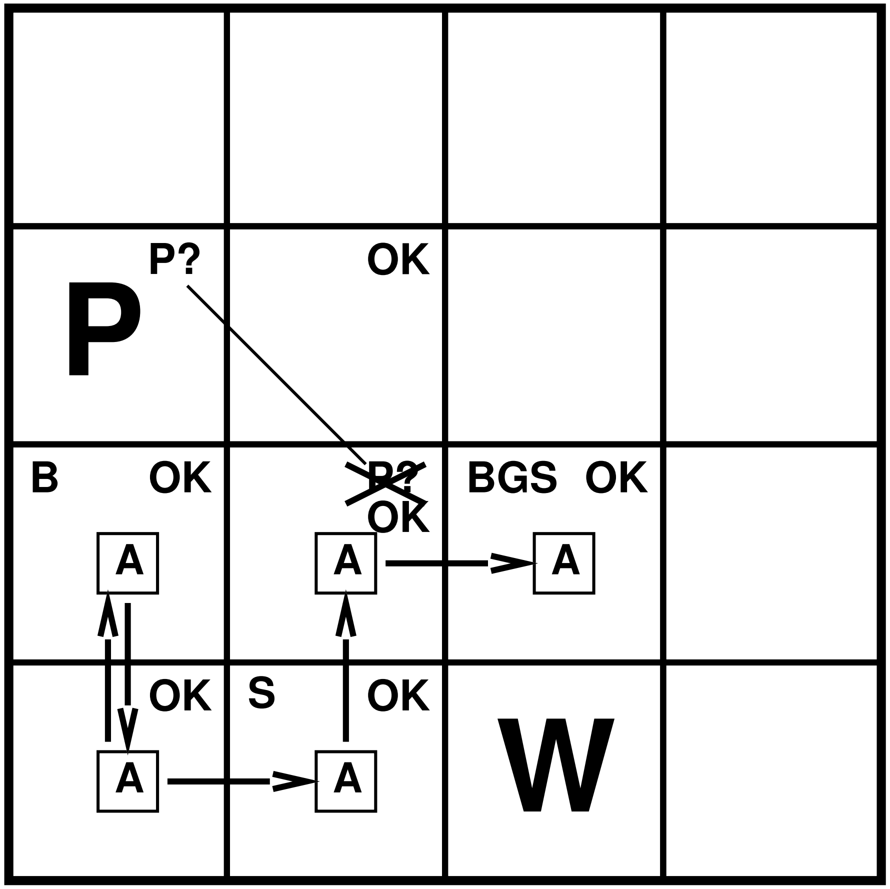
Exploring a wumpus world



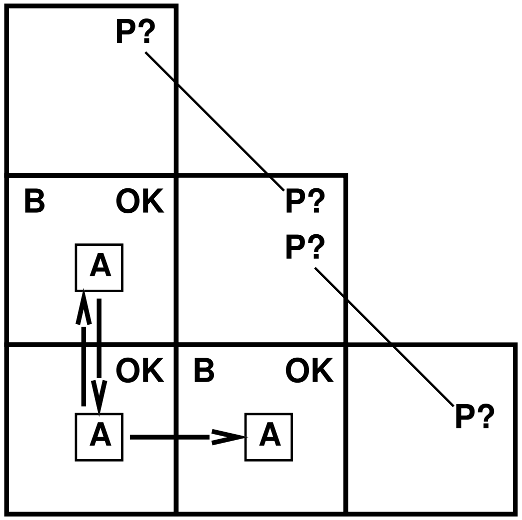
Exploring a wumpus world



Exploring a wumpus world

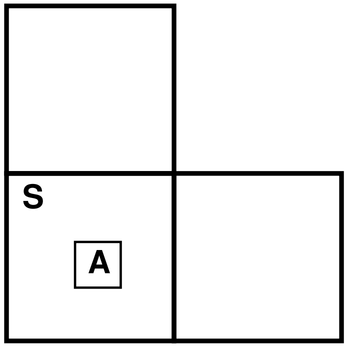


Other tight spots



Breeze in (1,2) and (2,1)
$\Rightarrow$ no safe actions

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



Smell in (1,1)
$\Rightarrow$ cannot move
Can use a strategy of coercion:
shoot straight ahead
wumpus was there $\Rightarrow$ dead $\Rightarrow$ safe
wumpus wasn't there $\Rightarrow$ safe

Logic

Logic (mathematical/symbolic logic) is a formal language for representing knowledge

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 $x = 0,\ y = 6$

Types of logic

Logics are characterized by what they commit to as "primitives"
– Ontological commitment: what exists—facts? objects? time? beliefs?
– Epistemological commitment: what 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	degree of belief $0 \ldots 1$
Fuzzy logic	degree of truth	degree of belief $0 \ldots 1$

Picking a logic has issues at the knowledge level
Start with first-order (predicate calculus) logic (FOL)
consider subsets/supersets and very different looking representation languages – propositional logic as subsets of FOL

Propositional logic

Propositional logic (PL) is the simplest logic but illustrates basic ideas and important applications

- Propositional language
- Syntax

  Proof theory
- Semantics

  Model theory
- Pragmatics

  Reasoning

  Knowledge Representation

Propositional language

A propositional language L_0

- Syntax
  - a set of (possibly infinite) symbols

    $\neg, \ \Rightarrow, \ (, \ ), \ True, \ False, \ P_1, \ P_2, \cdots$
  - a set of (well-formed) formulas (Wffs) or sentences
- Semantics
  - truth evaluations, i.e., truth functions (truth tables)

Syntax

The proposition symbols P_1, P_2, $\cdots$ are sentences (atom)

If S is a sentence, $\neg S$ is a sentence (negation)

If S_1 and S_2 are sentences, $S_1 \wedge S_2$ is a sentence (conjunction)

If S_1 and S_2 are sentences, $S_1 \vee S_2$ is a sentence (disjunction)

If S_1 and S_2 are sentences, $S_1 \Rightarrow S_2$ is a sentence (implication)

If S_1 and S_2 are sentences, $S_1 \Leftrightarrow S_2$ is a sentence (biconditional)

Connectives precedence: $\neg$, $\wedge$ $\vee$, $\Rightarrow$, $\Leftrightarrow$

A literal is a proposition (symbol) or its negation, i.e., P_i, or $\neg P_i$

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 for 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

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

Example: 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"

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"

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

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

Building a (simple) knowledge base for the wumpus world

Entailment

Entailment means that one thing **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**

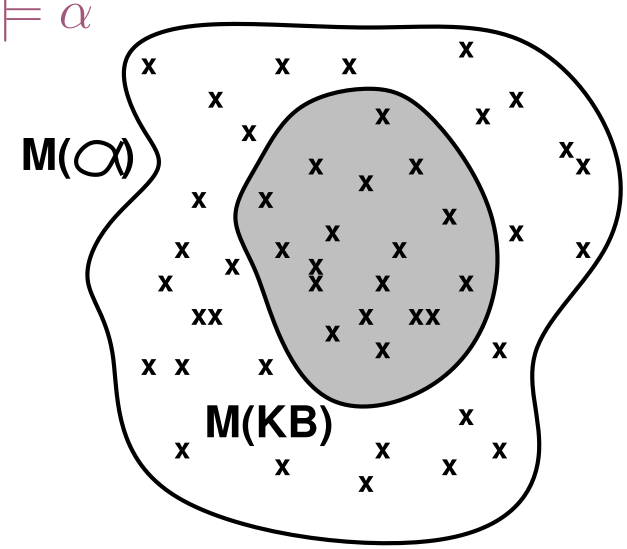
Models

Models are formally structured worlds in which truth can be evaluated

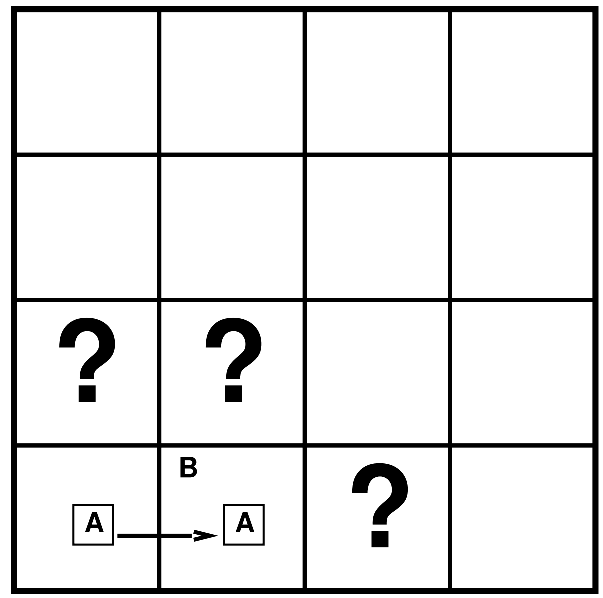
Say m is a model of a sentence α if α is true in m, written as $m \models \alpha$
$M(\alpha)$ is the set of all models of α

Then $KB \models \alpha$ iff $M(KB) \subseteq M(\alpha)$
i.e., for all model m, if $m \models KB$ then $m \models \alpha$

E.g. $KB =$ Giants won and Reds won
$\alpha =$ Giants won



Example: Entailment in the wumpus world

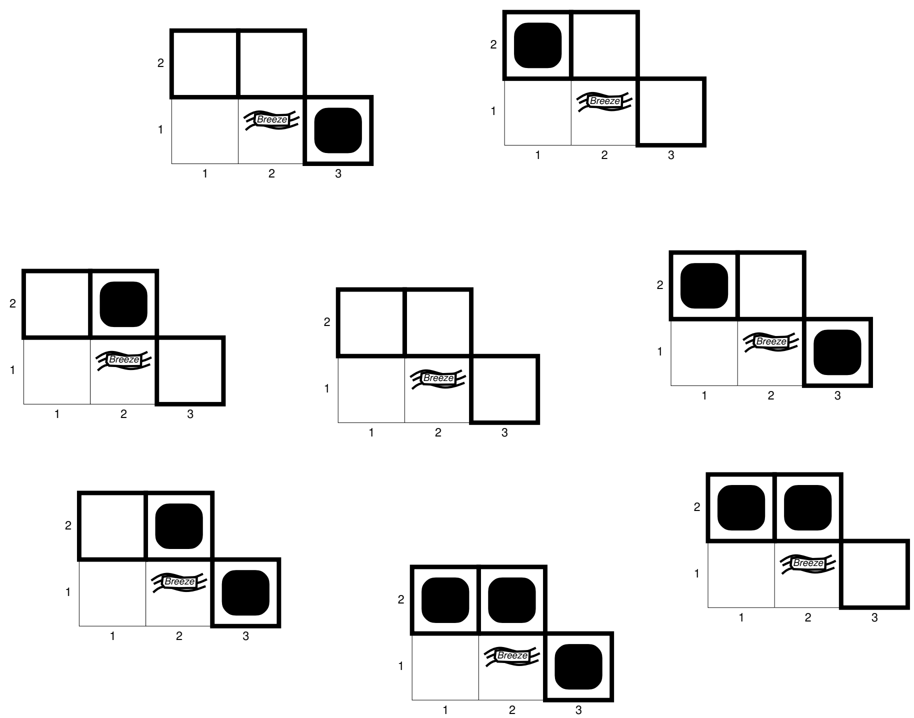


Situation after detecting nothing in [1,1],
moving right, breeze in [2,1]

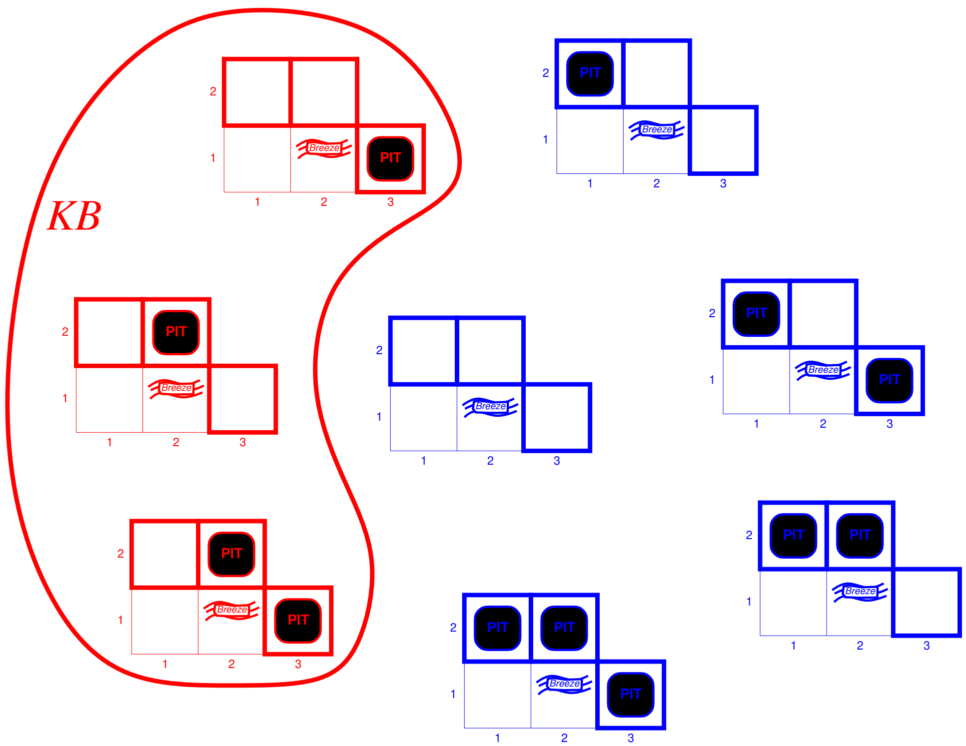
Consider possible models for ?s
assuming only pits

3 Boolean choices ⇒ 8 possible models

Wumpus models

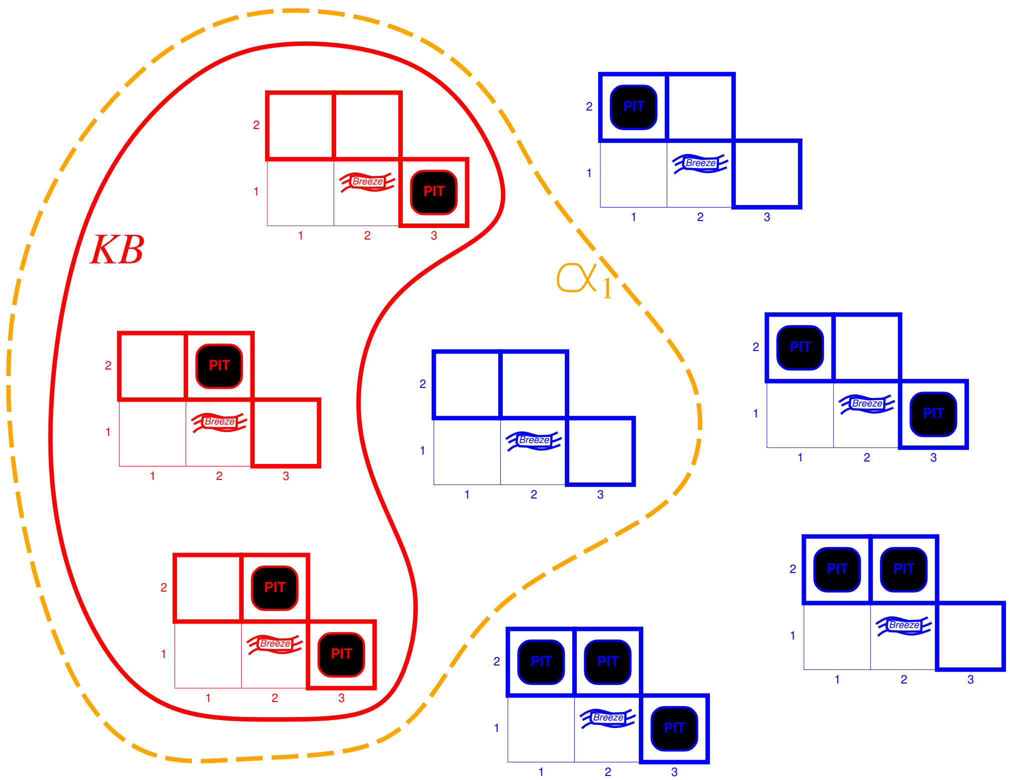


Wumpus models



KB = wumpus-world rules + observations

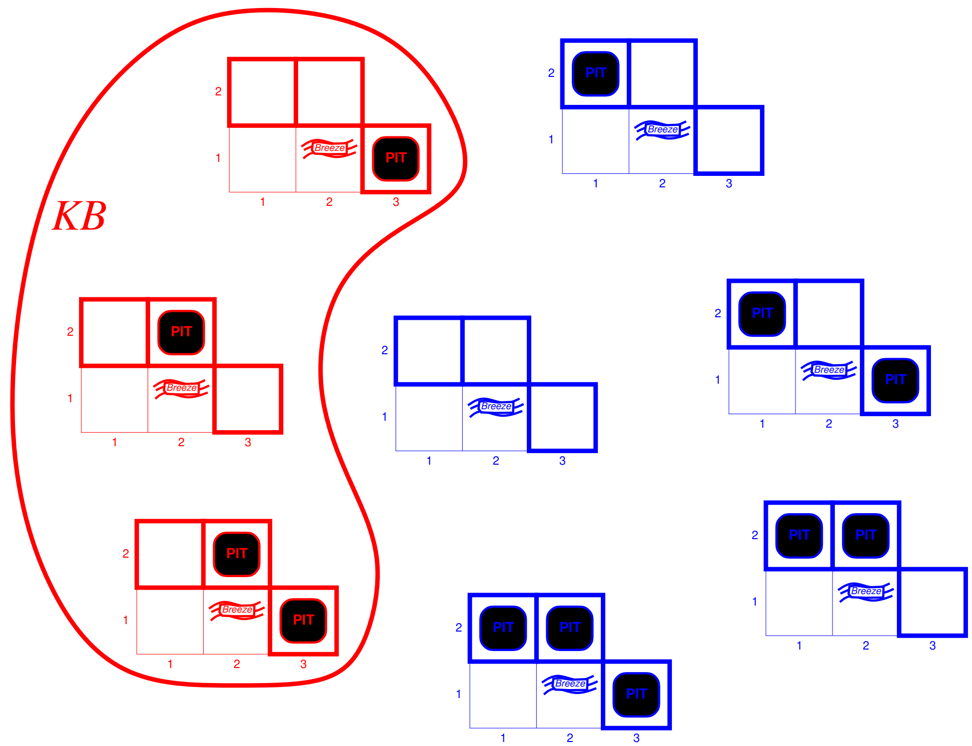
Wumpus models



KB = wumpus-world rules + observations

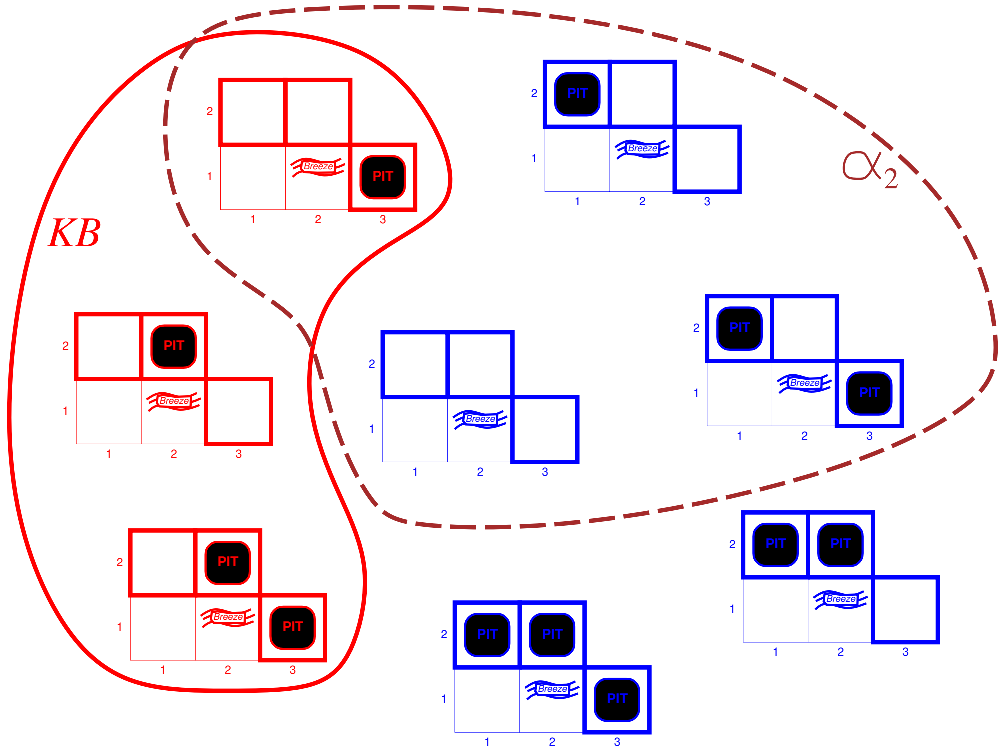
α_1 = "[1,2] is safe", $KB \models \alpha_1$, proved by model checking

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

Consequences of KB are a haystack; α is a needle.
Entailment = needle in a haystack; inference = finding it

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: FOL 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.

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*
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
false	*true*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*false*	*true*	*true*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*true*	*false*
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
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

```
def TT-ENTAILS?(KB, α)
    symbols ← a list of the proposition symbols in KB and α
    return TT-CHECK-ALL(KB, α, symbols, [ ])
def TT-CHECK-ALL(KB, α, symbols, model)
    if EMPTY?(symbols) then
        if PL-TRUE?(KB, model) then return PL-TRUE?(α, model)
        else return true // When KB is false always return true
    else
        P ← FIRST(symbols); rest ← REST(symbols)
        return TT-CHECK-ALL(KB, α, rest, {P = true} ∪ model)
            and TT-CHECK-ALL(KB, α, rest, {P = false} ∪ model)
```

$O(2^n)$ for n symbols, the problem is co-NP-complete

Logical equivalence#

Two sentences are logically equivalent iff true in the same models

$\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$

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

Validity and satisfiability

A sentence α is valid if it is true in **all** models

e.g., $True$, $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$

written as $\models \alpha$

If the procedure i is completeness, $= \vdash_i \alpha$

called α a theorem

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., $A \vee B$, C

A sentence is unsatisfiable if it is true in **no** models

e.g., $A \wedge \neg A$

Satisfiability is connected to inference via the following

$KB \models \alpha$ if and only if $(KB \wedge \neg\alpha)$ is unsatisfiable

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

SAT problem

SAT(isfiability) is the problem of determining the satisfiability of sentences in propositional logic

– The first problem was proved to be NP-complete (Cook Theorem, 1971)

– Many problems in computer science are SAT problems

– E.g., CSPs ask whether the constraints are satisfiable by some assignment

Roughly, any search task where what is searched for can be verified in polynomial time can be recast as a SAT problem

– Recall that $KB \models \alpha$ can be done by testing unsatisfiability of $KB \wedge \neg\alpha$

In general, SAT can be checked by enumerating the possible models until one is found that satisfies the sentences
Most people believe SAT to be unsolvable in polynomial time

Clause form

Conjunctive Normal Form (CNF)

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

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

Clause $=$ disjunction of literals

- proposition symbol; or
- (conjunction of symbols) $\Rightarrow$ symbol
  (i.e., conjunction of literals)

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

Any sentence can be equivalently transformed to CNF (clauses) by the logical equivalence

SAT: the satisfiability of sentences in clause form

3SAT$^+$

Notation $CNF_k(m, n)$: k-CNF sentence with m clauses (disjunction of literals) and n symbols, where the clauses are chosen uniformly, independently, and without replacement from among all clauses with k different literals, which are positive or negative at random

– E.g., 3SAT or $CNF_3(m, n)$

– 3SAT is also NP-complete
(2SAT can be solved in polynomial time)

DPLL algorithm is complete for solving SAT problem

There are a number of SAT solvers, say, WalkSAT, MaxSAT, Satz, RSat, MiniSAT, GSAT, etc.

– WalkSAT algorithm: on every iteration, picking an unsatisfied clause and a symbol to flip at a "random walk" step

WalkSAT algorithm*

```
def WalkSAT(clauses, p, max-flips)
   inputs: clauses, a set of clauses in propositional logic
           p, the probability of choosing to do a "random walk" move, typically 0.5
           max-flips, number of flips allowed before giving up
   model ← a random assignment of true/false to the symbols in clauses
   for i = 1 to max-flips do
      if model satisfies clauses then return model
      clause ← a randomly selected clause from clauses that is false in model
      with probability p flip the value in model of
         a random selected symbol from clause
      else flip whichever symbol in clause maximizes the number of satisfied clauses
   return failure
```

if *max-flips* is infinity and the sentence is unsatisfiable, then the algorithm never terminates

SAT vs. CSP

SAT $\Leftrightarrow$ CSP

SAT problem with the clausal form can be represented by the constraint graph as CSP

– using CSP algorithms to solve SAP problems

CSP with the constraint graph can be translated into the clausal form as SAP problem

– using SAP solvers to solve CSPs

Both benefit from each other

Satisfiability modulo theories*

SMT problem is a decision problem for logical formulas w.r.t. combinations of background theories expressed in FOL, e.g.

– integers, real numbers $\cdots$

– data structures, such as lists, arrays, bit vectors $\cdots$

SMT can be viewed as a form of CSP and solved by applications of SAT solvers

SAT competitions since 2002

Ref: Biere et al., 2009, *Handbook of Satisfiability*

First-Order Logic

- From PL to FOL
- FOL
  - Syntax
  - Semantics
  - Completeness
  - Reduction FOL to PL

From PL to FOL

Why FOL: pros and cons of PL

- PL is **declarative**: pieces of syntax correspond to facts
- PL allows partial/disjunctive/negated information
  (unlike most data structures and databases)
- PL is **compositional**:
  meaning of $B_{1,1} \wedge P_{1,2}$ is derived from meaning of $B_{1,1}$ and of $P_{1,2}$
- Meaning in PL is **context-independent**
  (unlike natural language, where meaning depends on context)
- PL has very limited expressive power
  (unlike natural language)
  E.g., cannot say "pits cause breezes in adjacent squares"
  except by writing one sentence for each square

FOL theses*

Mathematics: Hilbert's Thesis

There is no logic beyond first-order logic

– that when one is forced to make all one's mathematical (extra-logical) assumption explicit, these axioms can always be expressed in FOL, and

– that the informal notion of provable used in mathematics is made precise by the formal notion of provable in FOL

AI: McCarthy's Thesis

There is no declarative knowledge representation beyond first-order logic

FOL is very powerful

can be used as a full programming language

FOL

Whereas PL assumes world contains **facts**,
FOL 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 . . .

Syntax

Let L be a first-order language

	Constants	$kingJohn,\ 2,\ pku, \ldots$
	Predicates	$Brother,\ >, \ldots$
	Functions	$sqrt,\ leftLegOf, \ldots$
Vocabulary:	Variables	$x,\ y,\ a,\ b, \ldots$
	Connectives	$\wedge\ \vee\ \neg\ \Rightarrow\ \ \Leftrightarrow$
	Equality	$=$
	Quantifiers	$\forall\ \exists$

Note: all of vocabulary are symbols (countable infinity)

Syntax

arity: number of arguments

– arity 0 predicates: propositional symbols

– arity 0 functions: constant symbols

$\Leftarrow$ PL as special case of FOL

The predicates and functions are non-logical symbols

– predicate: mixed case capitalized, e.g., OlderThan

– functions: mixed case uncapitalized, e.g., brotherOf

Sometimes no distinction if no confusion

Notation

– occasionally add or omit (,)

– use [,] and {,} also

the parentheses are technical and not necessary (for readability)

Atomic sentences

Sentences (formulas) are defined from the vocabulary
– declarative, compositional and context-independent

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

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

E.g., $Brother(kingJohn, richardTheLionheart)$
$> (Length(leftLegOf(richard)), Length(leftLegOf(kingJohn)))$

Complex sentences

Complex sentences (well-formed formulas, wffs) are inductively defined from atomic sentences using connectives

1. Every atomic sentence is a wff
2. If S_1 and S_1 are wffs, and x is a variable, then

$$\neg S, \quad S_1 \wedge S_2, \quad S_1 \vee S_2, \quad S_1 \Rightarrow S_2, \quad S_1 \Leftrightarrow S_2, \quad \forall x S_1(x), \quad \exists x S_1(x)$$

are wffs

E.g. $Sibling(kingJohn, richard) \Rightarrow Sibling(richard, kingJohn)$
$>(1,2) \vee \leq(1,2)$

Note: PL as FOL subset: no terms, no quantifiers
$\forall \mathbf{x} S(\mathbf{x}) : \forall x_1 \ldots \forall x_n S(x_1, \ldots, x_n)$
$\mathbf{x} = (x_1, \cdots, x_n)$ stands for a tuple of variables (also terms)
Higher-order (second-order) logic, e.g., $\forall predicates S(predicates)$

Universal quantification

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

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

$\forall x \;\; P$ 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

$$\begin{aligned} & (At(kingJohn, beida) \Rightarrow Smart(kingJohn)) \\ \wedge \;\; & (At(richard, beida) \Rightarrow Smart(richard)) \\ \wedge \;\; & (At(lin, Beida) \Rightarrow Smart(lin)) \\ \wedge \;\; & \ldots \end{aligned}$$

Existential quantification

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

Someone at Qinghua is smart
$\exists x \ \ At(x, qinghua) \land Smart(x)$

$\exists x \ \ P$ 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

$$
\begin{array}{ll}
 & (At(kingJohn, Qinghua) \land Smart(kingJohn)) \\
\lor & (At(richard, Qinghua) \land Smart(richard)) \\
\lor & (At(wang, qinghua) \land Smart(wang)) \\
\lor & \ldots
\end{array}
$$

Mistakes to avoid

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

Mistake: using $\wedge$ as the main connective with $\forall$

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

means "Everyone is at Beida and everyone is smart"

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

Mistake: using $\Rightarrow$ as the main connective with $\exists$

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

is true if there is anyone who is not at Qinghua

Properties of quantifiers

$\forall x \ \forall y$ is the same as $\forall y \ \forall x$

$\exists x \ \exists y$ is the same as $\exists y \ \exists x$

$\exists x \ \forall y$ 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

$\forall x \ Likes(x, iceCream) \qquad \neg\exists x \ \neg Likes(x, iceCream)$

$\exists x \ Likes(x, broccoli) \qquad \neg\forall x \ \neg Likes(x, broccoli)$

Variable scope

The variables have a scope determined by the quantifiers

$P(x) \wedge \forall x(P(x) \vee Q(x))$

$\quad\uparrow \qquad\quad \uparrow \quad \uparrow \qquad\quad \uparrow$

free **bound** occurrences of variables

- sentences: wffs with no **free** variables (i.e., **closed** wffs)
- usually, free variables assumed to be universally quantified
- use dot "." for the scope, e.g., $\forall x.P(x) \vee Q(x)$ for $\forall x(P(x) \vee Q(x))$

Substitution:

– $\alpha[x/t]$ means α with all free occurrences of the x replaced by term t

– also, $\alpha[t_1, ..., t_n]$ means $\alpha[x_1/t_1, \cdots, x_n/t_n]$, or simple $\alpha[\mathbf{x}/\mathbf{t}]$

Semantics

Consider how to interpret sentences

– what do sentences claim about the world?

– or, what does believing one amount to?

Without meaning, sentences cannot be used to represent knowledge

Compared with PL, cannot fully specify the interpretation of sentences

because non-logical symbols reach outside

Logical interpretation

– specification of how to understand predicate and function symbols

Problem: cannot realistically expect to specify **once and for all** what a sentence means

the non-logical symbols are used in an application-dependent way

E.g., Happy(lin), who's lin, even if we were to agree on what "Happy" means

Semantics

Abstract structure to specify interpretation

1. There are objects (in the world)
2. For any predicate P (of arity 1), some of the objects will satisfy P and some will not

   – each interpretation settles extension of P

   – each interpretation assigns to function f a mapping from objects to objects

   functions always well-defined and single-valued
3. No other aspects of the world matter

The FOL assumption

This is all you need to know about the non-logical symbols to understand which sentences of FOL are true or false

Models

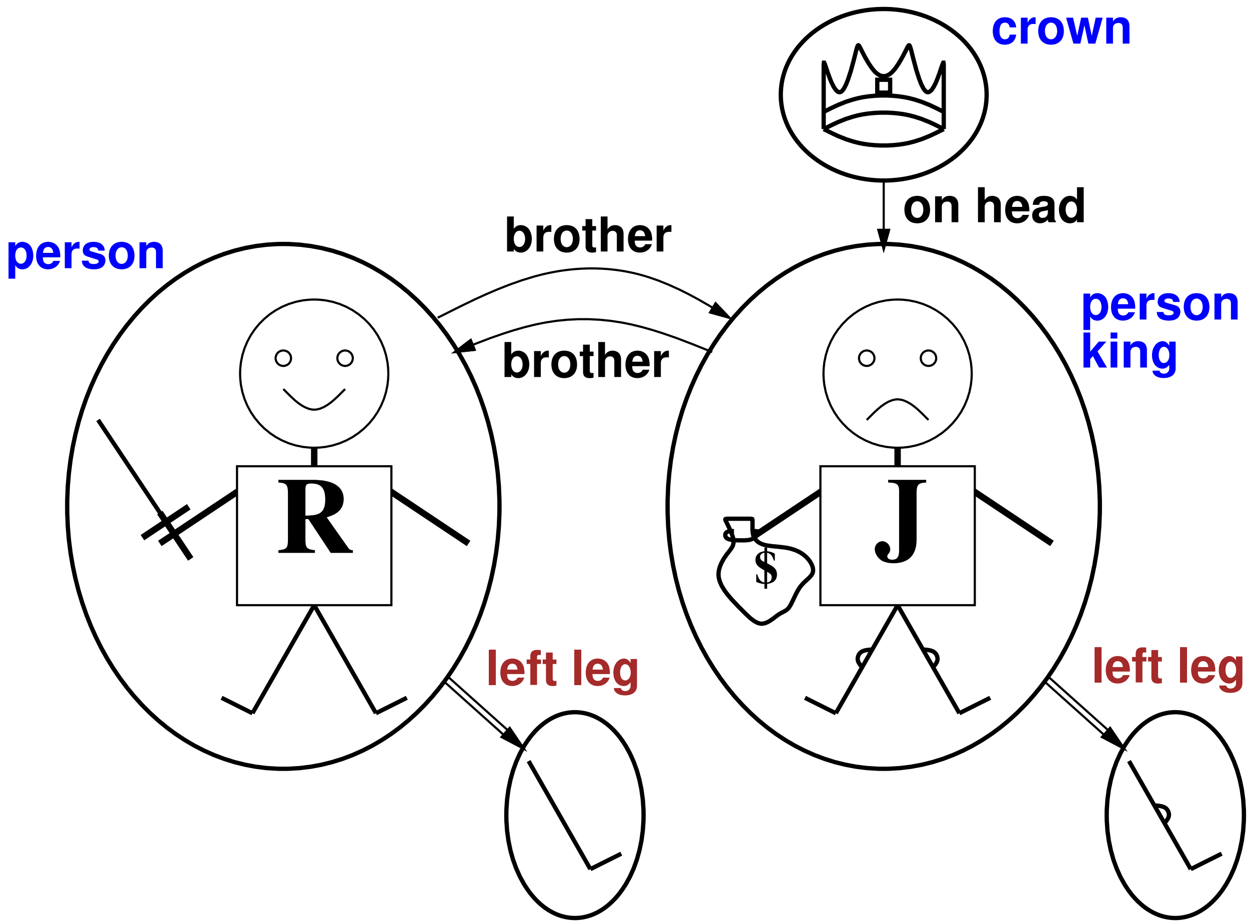
Sentences are true w.r.t. a model and an interpretation

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 $predicate$

Models: Example



Models: Truth

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: Entailment

Entailment in FOL can be computed by enumerating FOL models for a given KB vocabulary

Model checking

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 function on n objects

      For each constant symbol C in the vocabulary

        For each choice of referent for C from n objects . . .

Computing entailment by checking FOL models is not easy
(the domain is infinite or very large)

Interpretation*

Interpretation $I =< I, |I| >$
the domain $|I|$ given
can be any non-empty set
not just formal/mathematical objects
e.g., people, tables, numbers, sentences, the universe, etc.

Interpretation*

I is an (interpretation) mapping

1. If σ is a constant (symbol),
   then $I(\sigma) \in |I|$
2. If π is an n-ary function (symbol),
   then $I(\pi) : |I|^n \rightarrow |I|$;
   for constant c, $I(c) \in |I|$
3. If ρ is an n-ary predicate (symbol),
   then $I(\rho) \subseteq |I|^n$;
   for propositional symbol p, $I(p) = \{\,\}$

In propositional level (PL), it is convenient to assume
$I = I \in [$prop.symbols$\rightarrow \{true, false\}]$

Assignment*

Variable assignment U: given I

a mapping from the variables of L to objects of $|I|$

$U \in [\text{Variables} \to |I|]$

Term assignment T_{IU}: given I and U

1. If τ is an constant,
   then $T_{IU}(\tau) = I(\tau)$
2. If τ is a variable,
   then $T_{IU}(\tau) = U(\tau)$
3. If τ is a term of the form $\pi(\tau_1, \cdots, \tau_n)$ and $I(\pi) = g$ and $T_{IU}(\tau_i) = x_i$,
   then $T_{IU}(\tau) = g(x_1, \cdots, x_n)$

Satisfaction*

Satisfaction $\models_I \phi[U]$ (simply $\models$)

a sentence ϕ is satisfied by an interpretation I and a variable assignment U

1. $\models (\sigma = \tau)$ iff $T_{IU}(\sigma) = T_{IU}(\tau)$
2. $\models \rho(\tau_1, \cdots, \tau_n)$ iff $< T_{IU}(\tau_1), \cdots, T_{IU}(\tau_n) > \in I(\rho)$
3. $\models \neg\phi$ iff $\not\models \phi$
4. $\models \phi \wedge \psi$ iff $\models \phi$ and $\models \psi$
5. $\models \phi \vee \psi$ iff $\models \phi$ or $\models \psi$
6. $\models \phi \Rightarrow \psi$ iff $\not\models \phi$ or $\models \psi$
7. $\models \forall x \phi(x)$ iff for all $d \in |I|$ it is the case that $\models \phi[V]$, where $V(x) = d$ and $V(y) = U(y)$ for $x \neq y$
8. $\models \exists x \phi(x)$ iff for some $d \in |I|$ it is the case that $\models \phi[V]$, where $V(x) = d$ and $V(y) = U(y)$ for $x \neq y$

Models and entailment

Model I

If an interpretation I satisfies a sentence ϕ for all variable assignments, then I is said to be a *model* of ϕ, written $\models_I \phi$ or $I \models \phi$

Similarly (in PL), a sentence is true if it is satisfied in a models

a sentence is valid if it is true in all models

E.g., $\phi(x) \vee \neg\phi(x)$

A sentence is unsatisfiable (inconsistent, contradiction) if it is true in no models

E.g., $\phi(x) \wedge \neg\phi(x)$

Entailment $\models$

Let KB be a set of sentences and ϕ a sentence,
$KB \models \phi$ iff ϕ is true (satisfied) in all models of KB

Completeness

Soundeness and Completeness Theorem of FOL

$KB \vdash \alpha$ iff $KB \models \alpha$

Procedure i is complete if and only if

$KB \vdash_i \alpha \quad \text{whenever} \quad KB \models \alpha$

Historical hints

- Gödel completeness theorem
- Gödel incompleteness theorem

Incompleteness*

By extending the language of FOL to allow for the **mathematical induction** scheme in arithmetic, Gödel proved that there are true arithmetic sentences that cannot be proved

Incompleteness theorem: If the formal arithmetic system N is (ω-) consistent, then there is a sentence S which is not a theorem of N, nor its negation

Hence, N is not complete

What does Gödel incompleteness theorem mean in AI??

Reduction FOL to PL

Suppose the KB contains just the following

$$\forall x \;\; King(x) \land Greedy(x) \Rightarrow Evil(x)$$
$$King(john)$$
$$Greedy(john)$$
$$Brother(richard, john)$$

Instantiating the universal sentence in **all possible** ways, we have

$$King(john) \land Greedy(john) \Rightarrow Evil(john)$$
$$King(richard) \land Greedy(richard) \Rightarrow Evil(richard)$$
$$King(john)$$
$$Greedy(john)$$
$$Brother(richard, john)$$

The new KB is propositionalized: proposition symbols are

$King(john),\; Greedy(john),\; Evil(john), King(richard)$ etc.

Reduction*

A literal (sentence) is ground if it contains no variables

Herbrand Theorem
a ground sentence is entailed by new KB iff entailed by original KB
i.e., FOL KB can be propositionalized to preserve entailment

Idea: propositionalize KB and query, apply resolution, return result

Problem: with function symbols, there are infinitely many ground terms,

e.g., $father(father(father(john)))$

Reduction*

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

Idea: For $n = 0$ to ∞ do
create a propositional KB by instantiating with depth-n terms
see if α is entailed by this KB

Problem: works if α is entailed, loops if α is not entailed

Theorem (Turing/Church, 1936): Entailment in FOL is semidecidable
can find a proof of α if $KB \models \alpha$
cannot always prove that $KB \not\models \alpha$

Cf. **Halting Problem**: proof procedure may be about to terminate with success or failure, or may go on forever

Problems with propositionalization

Propositionalization seems to generate lots of irrelevant sentences
E.g., from

$$\forall x \;\; King(x) \land Greedy(x) \Rightarrow Evil(x)$$
$$King(john)$$
$$\forall y \;\; Greedy(y)$$
$$Brother(richard, john)$$

it seems obvious that $Evil(john)$, but propositionalization produces lots of facts such as $Greedy(richard)$ that are irrelevant

With p k-ary predicates and n constants, there are $p \cdot n^k$ instantiations

With function symbols, it gets much worse

Logical foundation of AI

- Knowledge representation
- Knowledge engineering
- Agents in logic

Knowledge representation

KR (Knowledge Representation) is first and foremost about

knowledge

– meaning and entailment

– find individuals and properties, then encode facts sufficient for entailments

Knowledge representation

Brothers are siblings

Knowledge representation

Brothers are siblings

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

"Sibling" is symmetric

Knowledge representation

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

Knowledge representation

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

Knowledge representation

Brothers are siblings

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

"Sibling" is symmetric

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

One's mother is one's female parent

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

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

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

Knowledge representation

$\lim s_n = c$, c is a real number

$$(\forall \varepsilon)\,(\varepsilon > 0 \Rightarrow (\exists n)\,(n \in \omega \land (\forall k)\,(k \in \omega \land k \geq n \Rightarrow |s_k - c| < \varepsilon)))$$

Two sets are equal iff they have the same elements

$$(x_1 = x_2 \Leftrightarrow (\forall x_3)(x_3 \in x_1 \leftrightarrow x_3 \in x_2))$$

Equality

$term_1 = term_2$ is true under a given interpretation
if and only if $term_1$ and $term_2$ refer to the same object

E.g., $1 = 2$ and $\forall\, x \;\; \times(Sqrt(x), Sqrt(x)) = x$ are satisfiable
$2 = 2$ is valid

E.g., definition of (full) $Sibling$ in terms of $Parent$

$$\forall\, x, y \;\; Sibling(x, y) \;\Leftrightarrow\; [\neg(x = y) \land \exists\, m, f \;\; \neg(m = f) \land Parent(m, x) \land Parent(f, x) \land Parent(m, y) \land Parent(f, y)]$$

Knowledge engineering

Before implementation, needs to understand clearly
- What is to be computed?
- Why and where inference is necessary?

Task: KB with appropriate entailments
- What vocabulary?
- What facts to represent?

Knowledge base

KB is set of sentences

explicit statement of sentences believed (including any assumed connections among non-logical symbols)

$KB \models \phi$

ϕ is a further consequence of what is believed

– explicit knowledge: KB

– implicit knowledge: $\{\phi \mid KB \models \phi\}$

Knowledge-based systems

Building (larger) KB to represent what is explicitly known

e.g. what the system has been told or has learned

Want to influence behavior based on what is implicit in the KB

Requires reasoning

– deductive inference

– – process of calculating entailments of KB

i.e., $KB \models \phi$

Example: a small KB

(KB) The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.

(α) Prove that Col. West is a criminal

Example: a small KB

. . . it is a crime for an American to sell weapons to hostile nations

Example: a small KB

. . . it is a crime for an American to sell weapons to hostile nations
$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$
Nono . . . has some missiles

Example: a small KB

. . . it is a crime for an American to sell weapons to hostile nations

$American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \Rightarrow Criminal(x)$

Nono . . . has some missiles, i.e., $\exists\, x\ Owns(Nono, x) \land Missile(x)$

$Owns(Nono, M_1)$ and $Missile(M_1)$

. . . all of its missiles were sold to it by Colonel West

Example: a small KB

... it is a crime for an American to sell weapons to hostile nations
$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$

Nono ... has some missiles, i.e., $\exists\, x\ Owns(Nono, x) \wedge Missile(x)$
$Owns(Nono, M_1)$ and $Missile(M_1)$

... all of its missiles were sold to it by Colonel West
$\forall\, x\ \ Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$

Missiles are weapons

Example: a small KB

... it is a crime for an American to sell weapons to hostile nations:

$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$

Nono ... has some missiles, i.e., $\exists\, x\; Owns(Nono, x) \wedge Missile(x)$

$Owns(Nono, M_1)$ and $Missile(M_1)$

... all of its missiles were sold to it by Colonel West

$\forall\, x\;\; Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$

Missiles are weapons:

$Missile(x) \Rightarrow Weapon(x)$

An enemy of America counts as "hostile"

Example: a small KB

... it is a crime for an American to sell weapons to hostile nations:
$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$

Nono ... has some missiles, i.e., $\exists\, x\ Owns(Nono, x) \wedge Missile(x)$
$Owns(Nono, M_1)$ and $Missile(M_1)$

... all of its missiles were sold to it by Colonel West
$\forall\, x\ \ Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$

Missiles are weapons
$Missile(x) \Rightarrow Weapon(x)$

An enemy of America counts as “hostile”
$Enemy(x, America) \Rightarrow Hostile(x)$

West, who is American ...
$American(West)$

The country Nono, an enemy of America ...
$Enemy(Nono, America)$

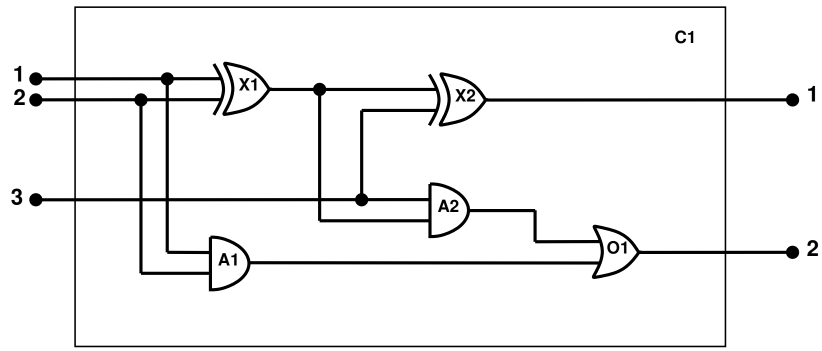
Agents in logic

- Logical agents in propositional level
- Logical agents in first-order case

Logical agents in propositional level

- Wumpus agent
  - The wumpus world KB
  - Finding pits and wumpus using logical inference
  - Translating knowledge into action

- Circuit-based agent



digital (logical) circuit $\Rightarrow$ components $\Rightarrow$ CPU/GPU $\Rightarrow$ VLSI
$= \mathsf{PL}$

- Database agent

Agents in first-order case

Suppose a wumpus-world agent is using an FOL KB
and perceives a smell and a breeze (but no glitter) at $t = 5$

$Tell(KB, Percept([Smell, Breeze, None], 5))$
$Ask(KB, \exists a \;\; Action(a, 5))$

I.e., does KB entail any particular actions at $t = 5$?

Answer: $Yes, \;\; \{a/Shoot\}$ $\leftarrow$ substitution

In general, $Ask(KB, S)$: $KB \models S$

Example: the wumpus world

Perception

$\forall\, b, g, t \;\; Percept([Smell, b, g], t) \Rightarrow Smelt(t)$

$\forall\, s, b, t \;\; Percept([s, b, Glitter], t) \Rightarrow AtGold(t)$

Reflex: $\forall\, t \;\; AtGold(t) \Rightarrow Action(Grab, t)$

Reflex with internal state: do we have the gold already?

$\forall\, t \;\; AtGold(t) \wedge \neg Holding(Gold, t) \Rightarrow Action(Grab, t)$

$Holding(Gold, t)$ cannot be observed

$\Rightarrow$ keeping track of change is essential

Reasoning*

Properties of locations

$\forall\, x, t \;\; At(Agent, x, t) \wedge Smelt(t) \Rightarrow Smelly(x)$

$\forall\, x, t \;\; At(Agent, x, t) \wedge Breeze(t) \Rightarrow Breezy(x)$

Squares are breezy near a pit

Diagnostic rule—infer cause from effect

$$\forall\, y \;\; Breezy(y) \Rightarrow \exists\, x \;\; Pit(x) \wedge Adjacent(x, y)$$

Causal rule—infer effect from cause

$$\forall\, x, y \;\; Pit(x) \wedge Adjacent(x, y) \Rightarrow Breezy(y)$$

Neither of these is complete—e.g., the causal rule doesn't say whether squares far away from pits can be breezy

Definition for the $Breezy$ predicate

$$\forall\, y \;\; Breezy(y) \;\Leftrightarrow\; [\exists\, x \;\; Pit(x) \wedge Adjacent(x, y)]$$

Applications of FOL*

FOL is general enough to applications of intelligence with reason

Facts hold in situations, rather than eternally

E.g., $Holding(Gold, Now)$ rather than just $Holding(Gold)$

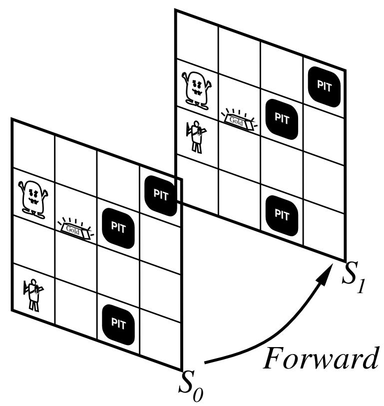
Situation calculus is one way to represent **change** in FOL

Adds a situation argument to each non-eternal predicate

E.g., Now in $Holding(Gold, Now)$ denotes a situation

Situations are connected by the $Result$ function

$Result(a, s)$ is the situation that results from doing a in s



Example: EMI*

EMI (Experiments in Musical Intelligence, subsequent Emily Howell) is a computer program created by and collaborated with David Cope

– can analyze existing music and create new compositions in the style of the original input music

– rule-based (logic) "expert system" for music language understanding

Ref. Cope D, Experiments in Musical Intelligence, Madison, , 1996

Example: ANTON*

ANTON (2011): an automatic system for the composition of renaissance-style music

– about 500 rules for musical knowledge

– in the form of logic programming: **answer set programming** (see the next lecture)