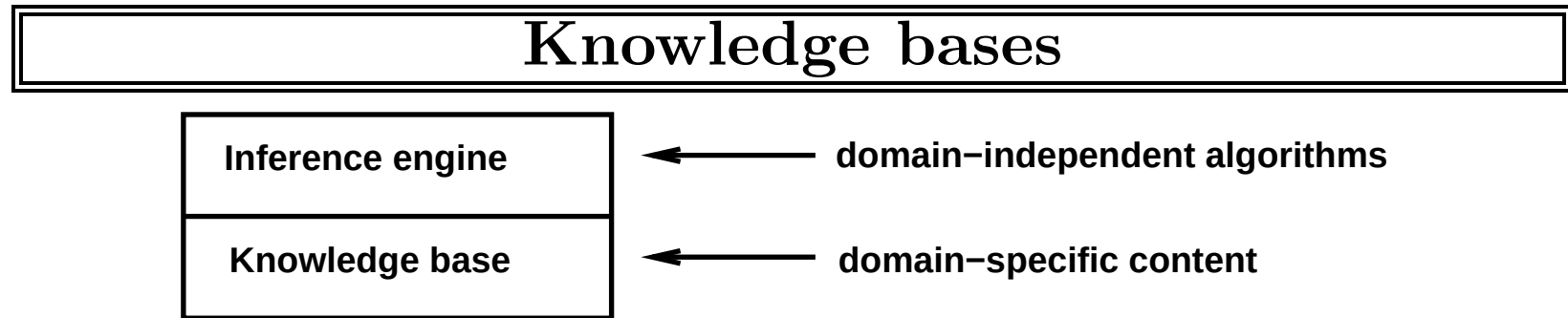




Knowledge base = set of sentences in a **formal** language

Declarative approach to building an agent (or other system):

TELL it what it needs to know

Then it can **ASK** itself what to do—answers should follow from the 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 structures in KB and algorithms that manipulate them

A simple knowledge-based agent

```
function KB-AGENT(percept) returns an action
  static: 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

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; $x^2 + y >$ 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 $x = 0$, $y = 6$

Truth tables for connectives

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>
<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>
<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>
<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>

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**

Note: brains process **syntax** (of some sort)

Models

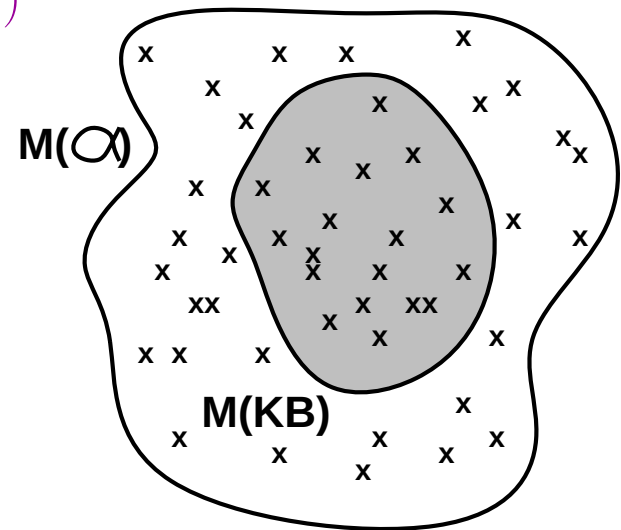
Logicians typically think in terms of **models**, which are formally structured worlds with respect to which truth can be evaluated

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 $KB \models \alpha$ if and only if $M(KB) \subseteq M(\alpha)$

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



Validity and satisfiability

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

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

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*

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

First-order logic

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
. . .

Syntax of FOL: Basic elements

Constants *KingJohn, 2, UCB, ...*

Predicates *Brother, >, ...*

Functions *Sqrt, LeftLegOf, ...*

Variables *x, y, a, b, ...*

Connectives $\wedge \vee \neg \Rightarrow \Leftrightarrow$

Equality $=$

Quantifiers $\forall \exists$

Atomic sentences

Atomic sentence = *predicate*(*term*₁, ..., *term*_{*n*})
or *term*₁ = *term*₂

Term = *function*(*term*₁, ..., *term*_{*n*})
or *constant* or *variable*

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

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. $Sibling(KingJohn, Richard) \Rightarrow Sibling(Richard, KingJohn)$

$$>(1, 2) \vee \leq(1, 2)$$

$$>(1, 2) \wedge \neg >(1, 2)$$

Universal quantification

$\forall \langle \text{variables} \rangle \langle \text{sentence} \rangle$

Everyone at Berkeley is smart:

$\forall x \text{ At}(x, \text{Berkeley}) \Rightarrow \text{Smart}(x)$

$\forall x \ P$ 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

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

A common mistake to avoid

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

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

$$\forall x \text{ } At(x, Berkeley) \wedge Smart(x)$$

means “Everyone is at Berkeley and everyone is smart”

Existential quantification

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

Someone at Stanford is smart:

$\exists x \text{ } At(x, Stanford) \wedge Smart(x)$

$\exists x \text{ } P$ 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

$$\begin{aligned} & (At(KingJohn, Stanford) \wedge Smart(KingJohn)) \\ \vee & (At(Richard, Stanford) \wedge Smart(Richard)) \\ \vee & (At(Stanford, Stanford) \wedge Smart(Stanford)) \\ \vee & \dots \end{aligned}$$

Another common mistake to avoid

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

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

$$\exists x \text{ } At(x, Stanford) \Rightarrow Smart(x)$$

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 $\forall y \exists x$

$\exists x \forall y \text{ Loves}(x, y)$

“There is a person who loves everyone in the world”

$\forall y \exists x \text{ Loves}(x, y)$

“Everyone in the world is loved by at least one person”

Quantifier duality: each can be expressed using the other

$\forall x \text{ Likes}(x, \text{IceCream}) \quad \neg \exists x \neg \text{Likes}(x, \text{IceCream})$

$\exists x \text{ Likes}(x, \text{Broccoli}) \quad \neg \forall x \neg \text{Likes}(x, \text{Broccoli})$

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 $\forall x \neg (Sqrt(x), Sqrt(x)) = x$ are satisfiable
 $2 = 2$ is valid

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

$$\forall x, y \text{ Sibling}(x, y) \Leftrightarrow [\neg(x = y) \wedge \exists m, f \neg(m = f) \wedge \\ \text{Parent}(m, x) \wedge \text{Parent}(f, x) \wedge \text{Parent}(m, y) \wedge \text{Parent}(f, y)]$$

Wumpus World PEAS description

Performance measure

gold +1000, death -1000

-1 per step, -10 for using the arrow

Environment

Squares adjacent to wumpus are smelly

Squares adjacent to pit are breezy

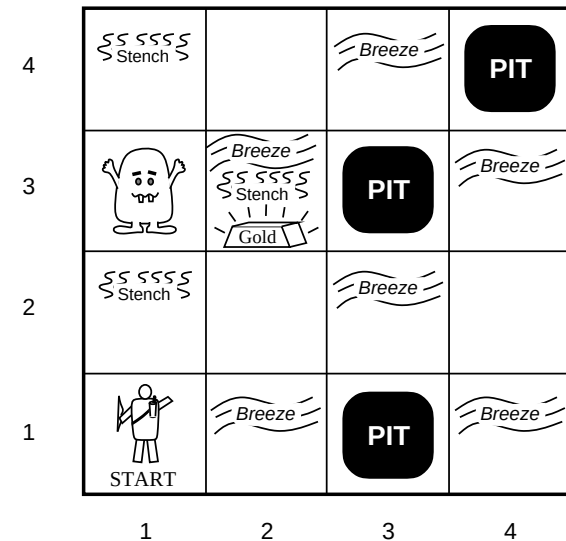
Glitter iff gold is in the same square

Shooting kills wumpus if you are facing it

Shooting uses up the only arrow

Grabbing picks up gold if in same square

Releasing drops the gold in same square



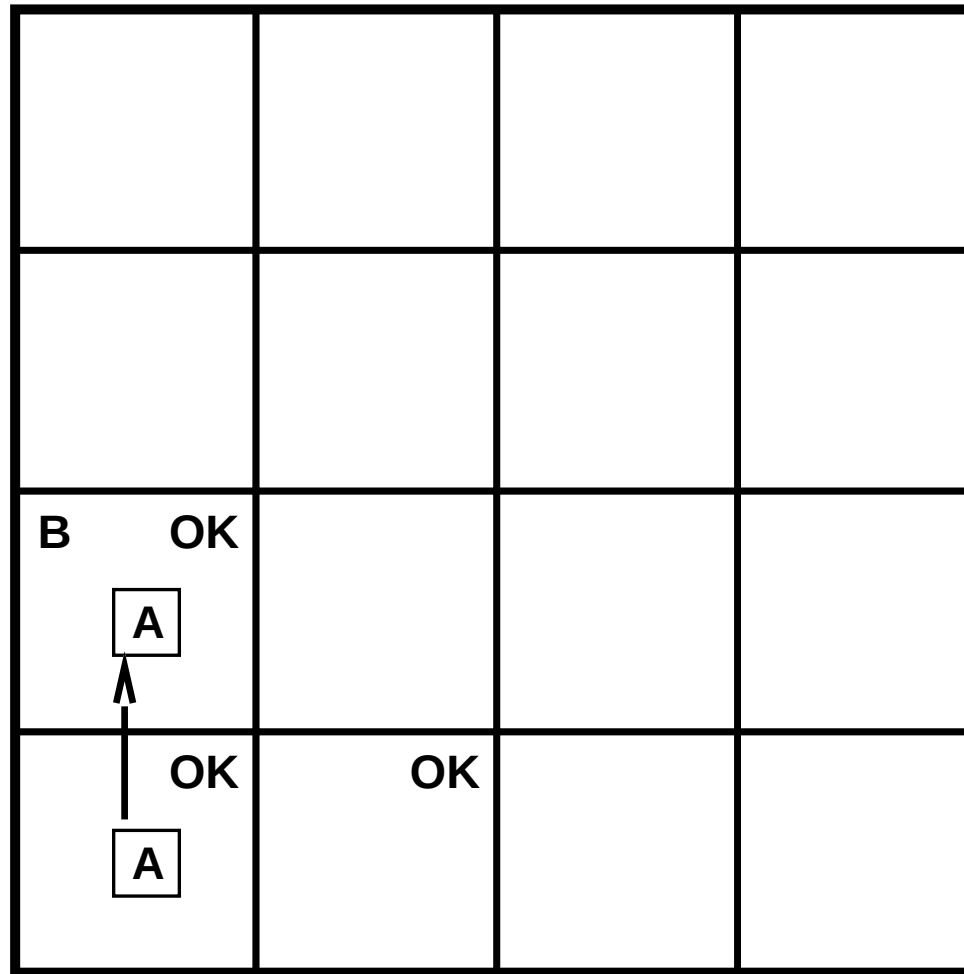
Actuators Left turn, Right turn,
Forward, Grab, Release, Shoot

Sensors Breeze, Glitter, Smell

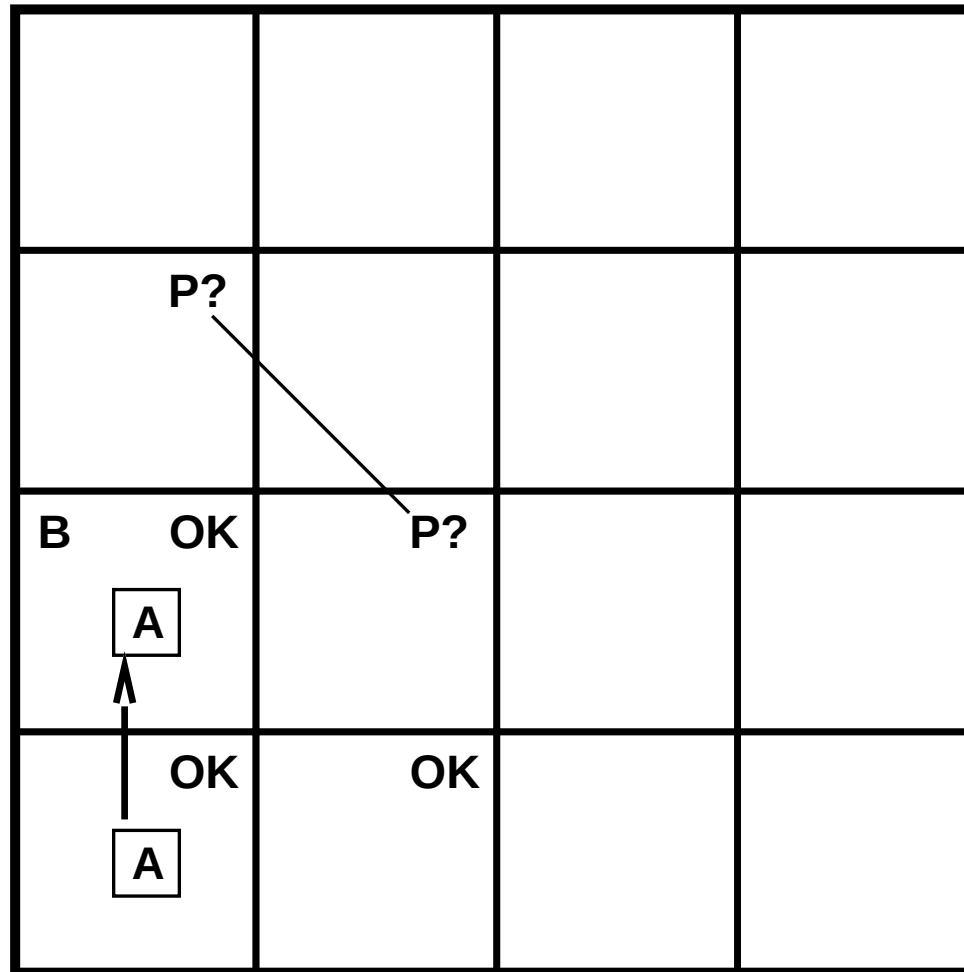
Exploring a wumpus world

OK			
OK A	OK		

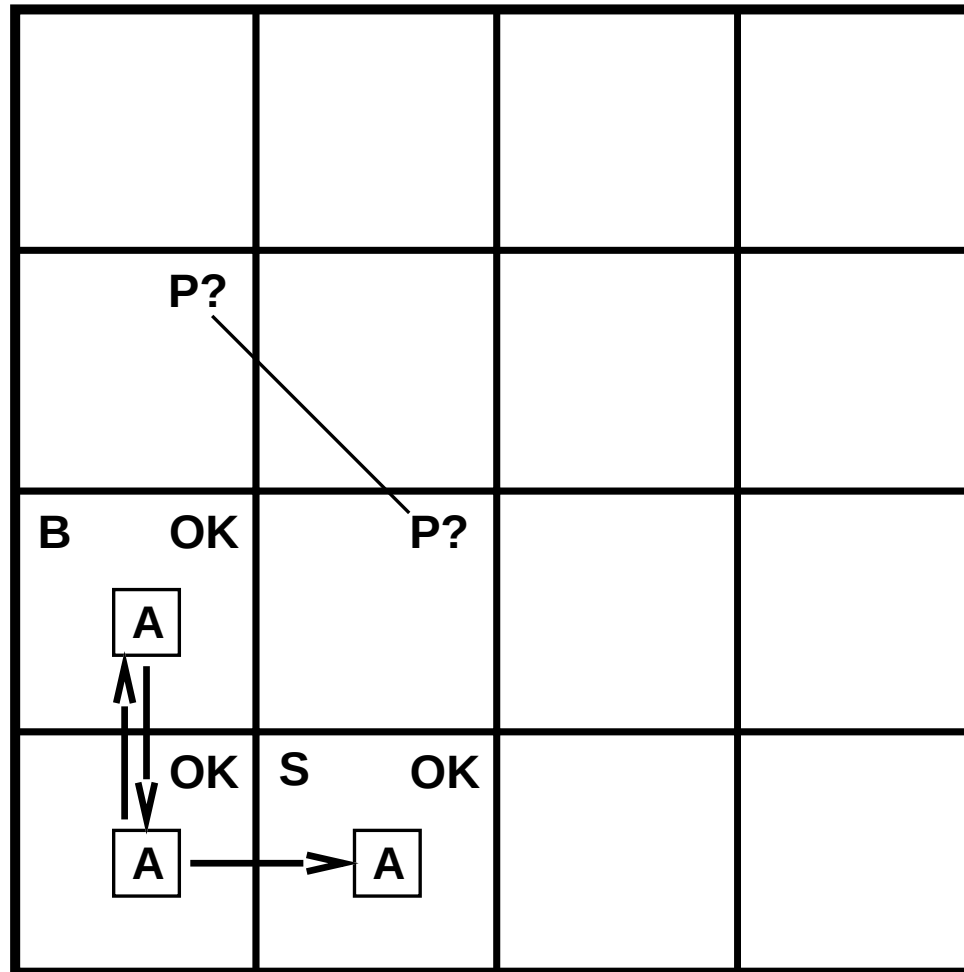
Exploring a wumpus world



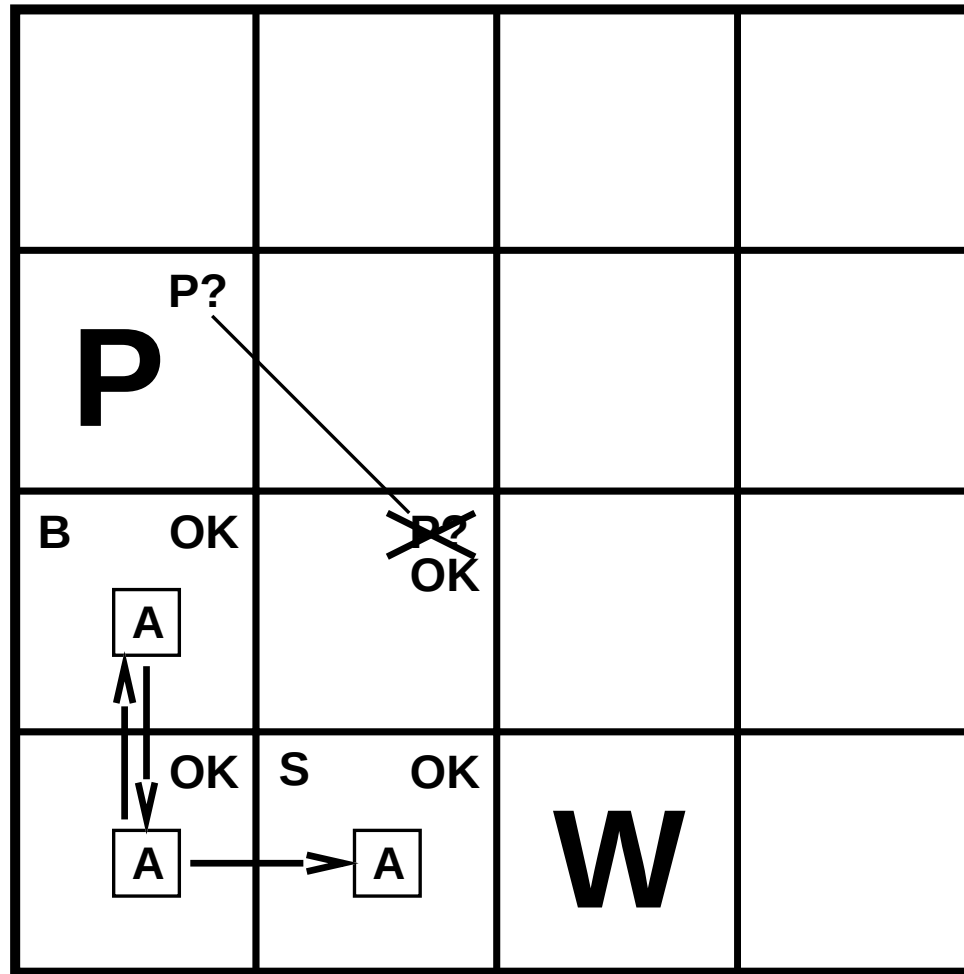
Exploring a wumpus world



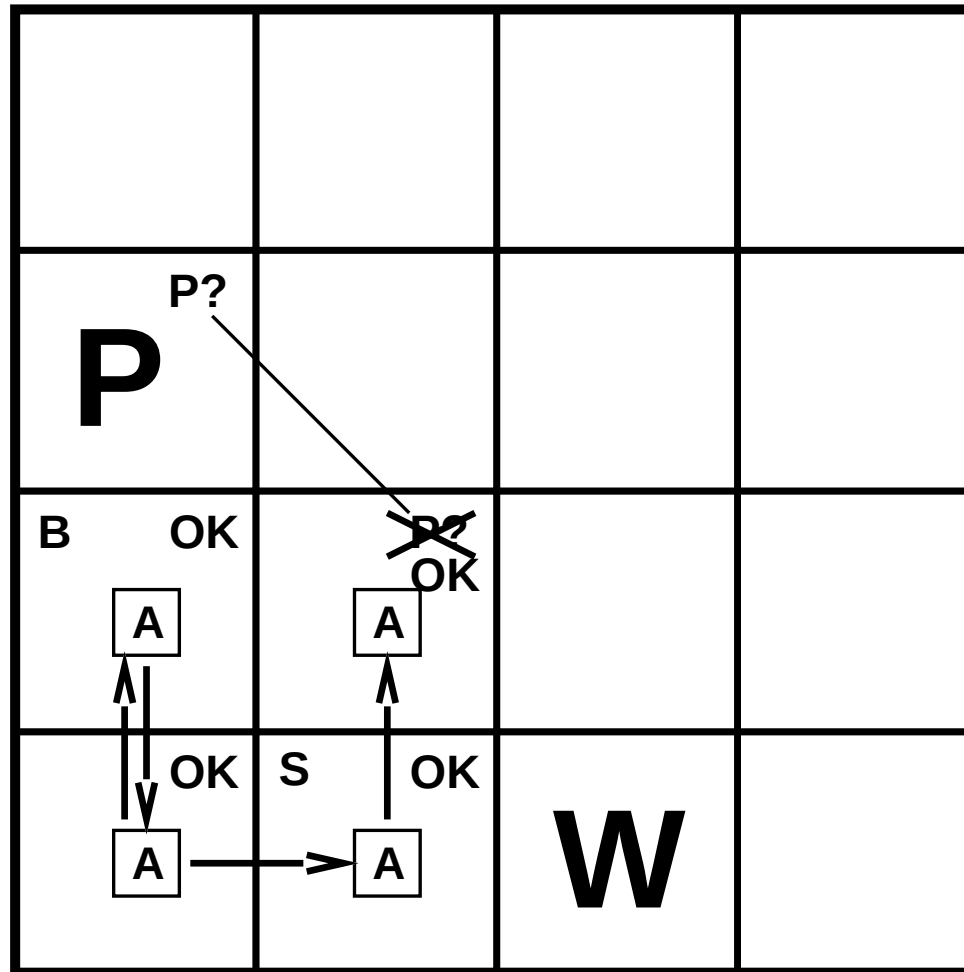
Exploring a wumpus world



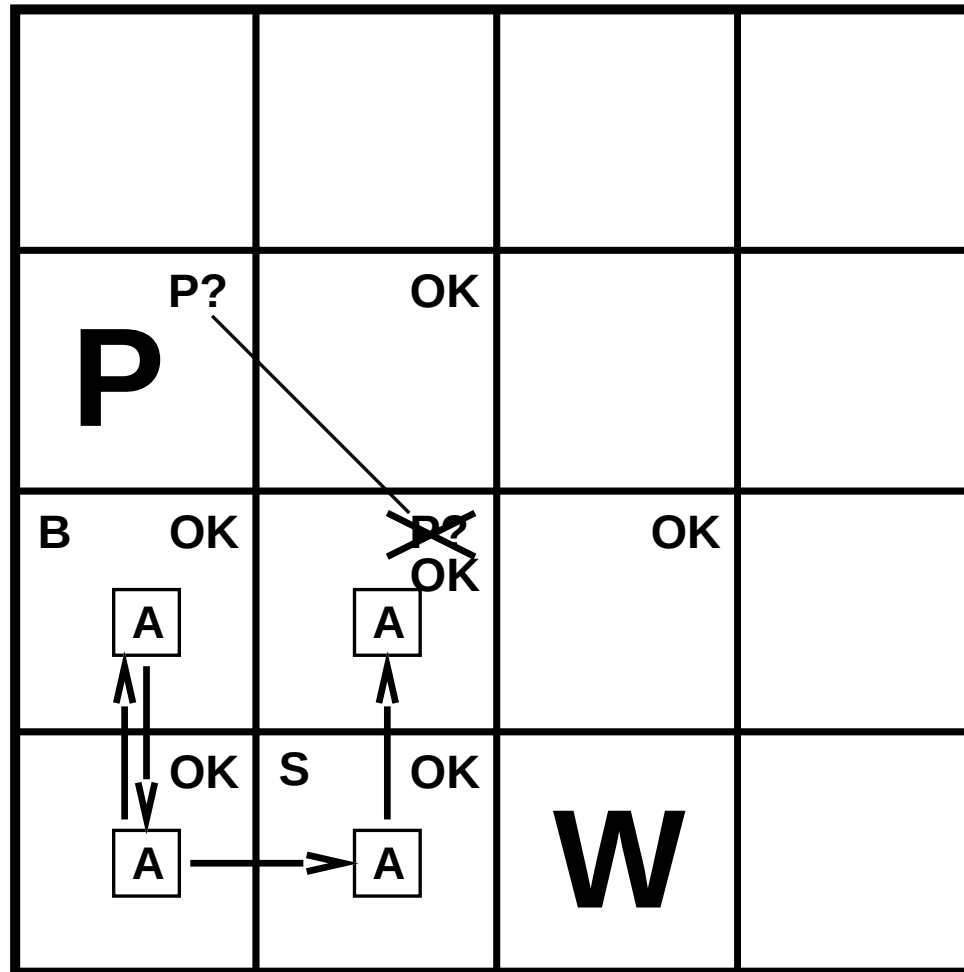
Exploring a wumpus world



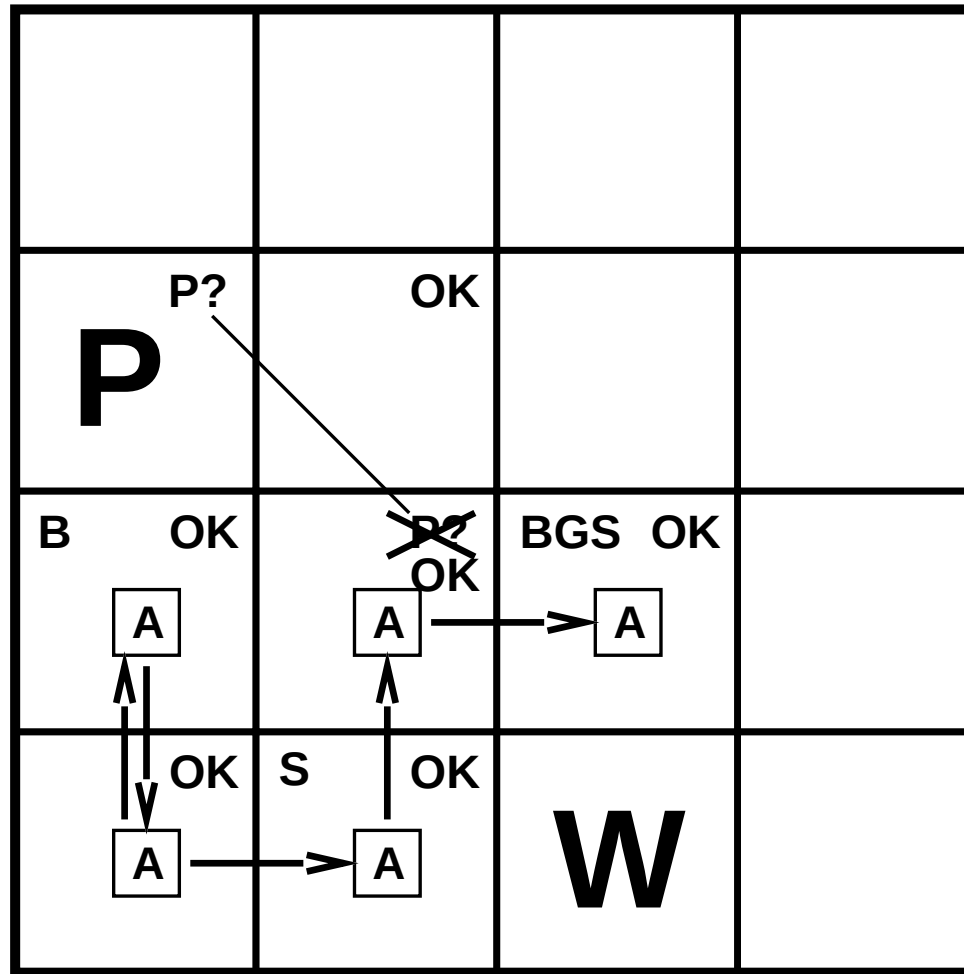
Exploring a wumpus world



Exploring a wumpus world



Exploring a wumpus world



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”

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

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

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

Knowledge base for the wumpus world

“Perception”

$\forall b, g, t \text{ Percept}([Smell, b, g], t) \Rightarrow Smelt(t)$

$\forall s, b, t \text{ Percept}([s, b, Glitter], t) \Rightarrow AtGold(t)$

Reflex: $\forall t \text{ AtGold}(t) \Rightarrow \text{Action}(Grab, t)$

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

$\forall t \text{ AtGold}(t) \wedge \neg Holding(Gold, t) \Rightarrow \text{Action}(Grab, t)$

$Holding(Gold, t)$ cannot be observed

$\Rightarrow$ keeping track of change is essential

Deducing hidden properties

Properties of locations:

$$\forall x, t \text{ } At(Agent, x, t) \wedge Smelt(t) \Rightarrow Smelly(x)$$

$$\forall x, t \text{ } At(Agent, x, t) \wedge Breeze(t) \Rightarrow Breezy(x)$$

Squares are breezy near a pit:

Diagnostic rule—infer cause from effect

$$\forall y \text{ } Breezy(y) \Rightarrow \exists x \text{ } Pit(x) \wedge Adjacent(x, y)$$

Causal rule—infer effect from cause

$$\forall x, y \text{ } Pit(x) \wedge Adjacent(x, y) \Rightarrow Breezy(y)$$

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:

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

Interacting with FOL KBs

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

$Tell(KB, Percept([Smell, Breeze, None], 5))$

$Ask(KB, \exists a \text{ Action}(a, 5))$

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

Answer: $Yes, \{a/Shoot\} \leftarrow \text{substitution (binding list)}$

Given a sentence S and a substitution σ ,

$S\sigma$ denotes the result of plugging σ into S ; e.g.,

$S = Smarter(x, y)$

$\sigma = \{x/Hillary, y/Bill\}$

$S\sigma = Smarter(Hillary, Bill)$

$Ask(KB, S)$ returns some/all σ such that $KB \models S\sigma$

Describing actions I

“Effect” axiom—describe changes due to action

$$\forall s \text{ } AtGold(s) \Rightarrow Holding(Gold, Result(Grab, s))$$

“Frame” axiom—describe **non-changes** due to action

$$\forall s \text{ } HaveArrow(s) \Rightarrow HaveArrow(Result(Grab, s))$$

Frame problem: find an elegant way to handle non-change

- (a) representation—avoid frame axioms
- (b) inference—avoid repeated “copy-overs” to keep track of state

Qualification problem: true descriptions of real actions require endless caveats—what if gold is slippery or nailed down or . . .

Ramification problem: real actions have many secondary consequences—what about the dust on the gold, wear and tear on gloves, . . .

Describing actions II

Successor-state axioms solve the representational frame problem

Each axiom is “about” a **predicate** (not an action per se):

$$\begin{aligned} P \text{ true afterwards} \quad \Leftrightarrow \quad & [\text{an action made } P \text{ true} \\ & \vee \quad P \text{ true already and no action made } P \text{ false}] \end{aligned}$$

For holding the gold:

$$\begin{aligned} \forall a, s \quad & Holding(Gold, Result(a, s)) \Leftrightarrow \\ & [(a = Grab \wedge AtGold(s)) \\ & \vee (Holding(Gold, s) \wedge a \neq Release)] \end{aligned}$$