ARTIFICIAL INTELLIGENCE

COMPLETE ARCHITECTURES FOR INTELLIGENCE?

Search? (Problem Solving Agents)

- Solve the problem of what to do.

Logic and inference? (Knowledge Based Agents)

- Reason about what to do.
- Encoded knowledge/“expert” systems?
  - Know what to do.

Learning? (Learning Agents)

- Learn what to do.

Modern view: It’s complex & multi-faceted.

WHY DO WE NEED LOGIC?

Problem-solving agents were very inflexible: hard code every possible state.

Search is almost always exponential in the number of states.

Problem solving agents cannot infer unobserved information.

We want an algorithm that ***reasons*** in a way that resembles reasoning in humans

REASONING

Making an inference about something that was previously not seen/not known based on some previously stored knowledge.

INFERENCE IN FORMAL SYMBOL SYSTEMS:
ONTOLOGY, REPRESENTATION, INFERENCE

Formal Symbol Systems

- **Symbols** correspond to things/ideas in the world
- **Pattern matching** & rewrite corresponds to inference

Ontology: What exists in the world?

- What must be represented?

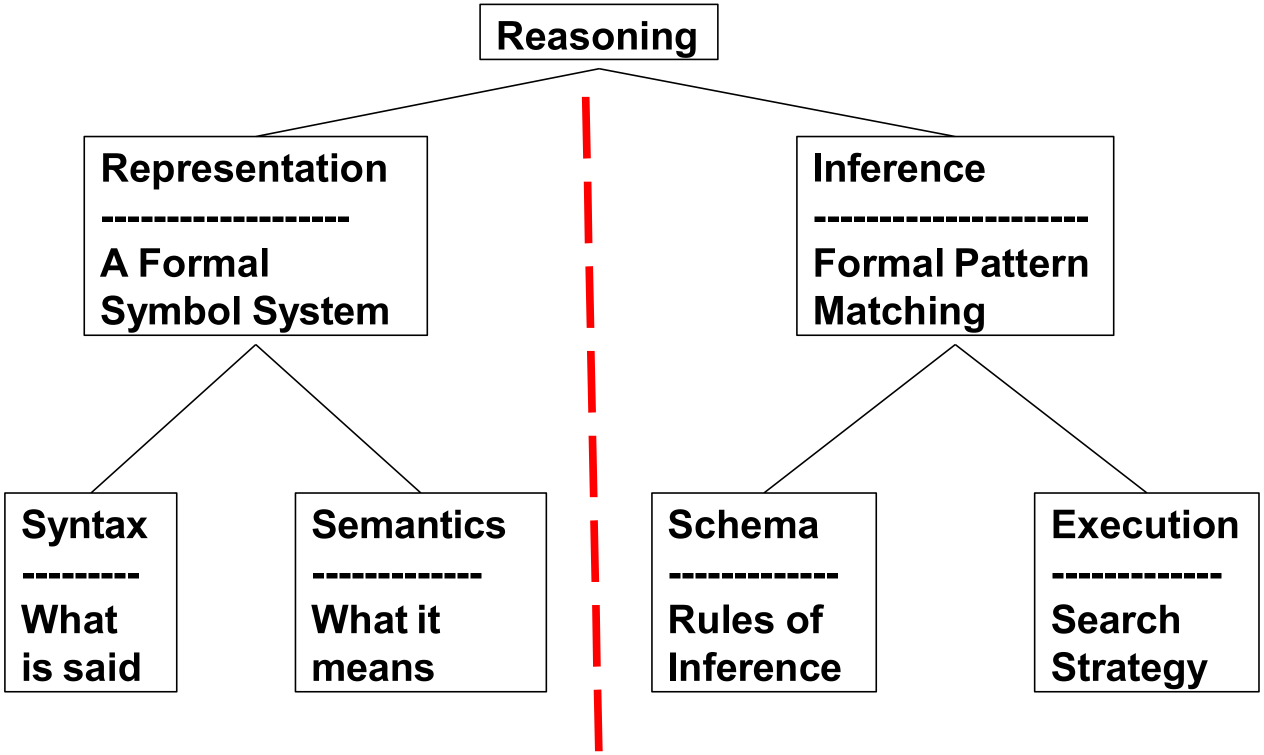
Representation: Syntax vs. Semantics

- What's Said vs. What's Meant

Inference: Schema vs. Mechanism

- Proof Steps vs. Search Strategy

Ontology:
What kind of things exist in the world?
What do we need to describe and reason about?



Reasoning

Representation

A Formal
Symbol System

Inference

Formal Pattern
Matching

Syntax

What
is said

Semantics

What it
means

Schema

Rules of
Inference

Execution

Search
Strategy

LOGICAL OR KNOWLEDGE BASED AGENTS COMPONENTS

Knowledge base / KB (facts)

Knowledge Representation Language (In what language would you tell agents the facts?)

Inference

Background Knowledge of the world

LOGICAL OR KNOWLEDGE BASED AGENTS

Basic Actions: Tell and Ask

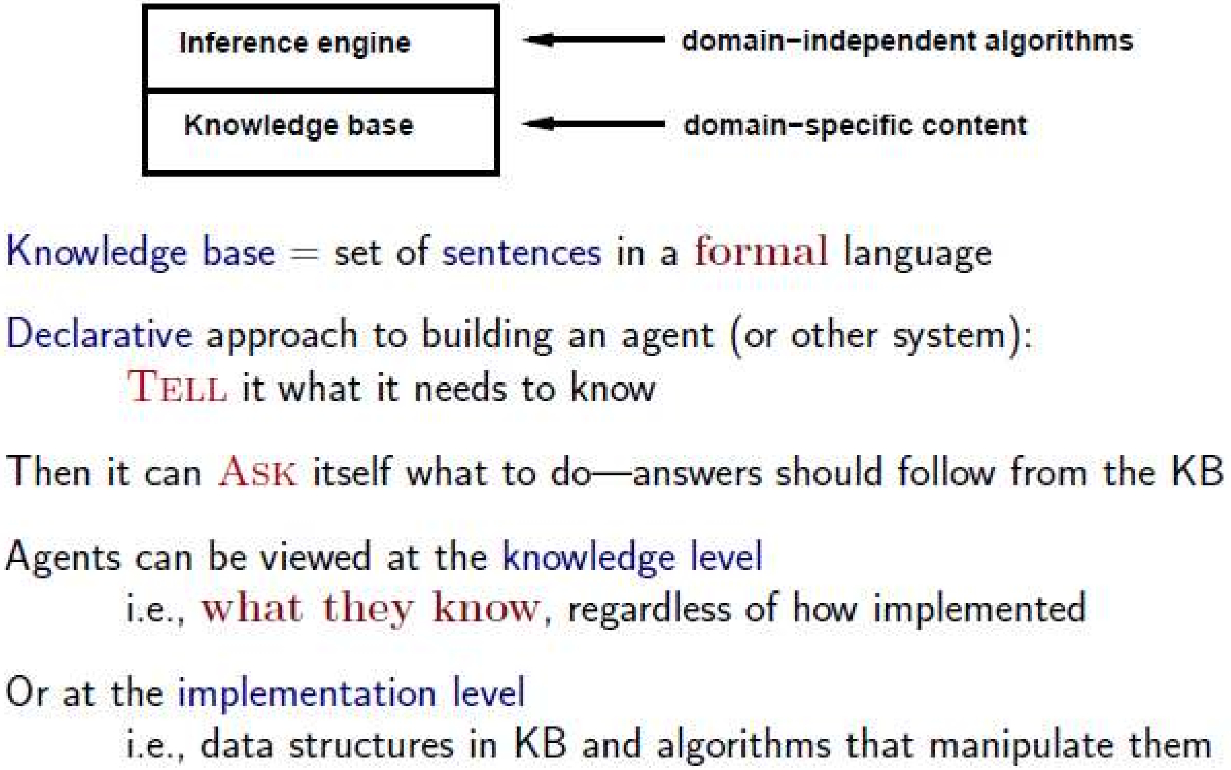
A Knowledge base keeps track of things

We can tell an agent facts and ask for inference

Example:

- Tell: Father of John is Bob
- Tell: Jane is John’s Sister
- Tell: John’s father is the same as John’s sister father
- Ask: Who is Jane’s father? (The answer requires inference on facts)

KNOWLEDGE BASES



Inference engine ← domain−independent algorithms

Knowledge base ← domain−specific content

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

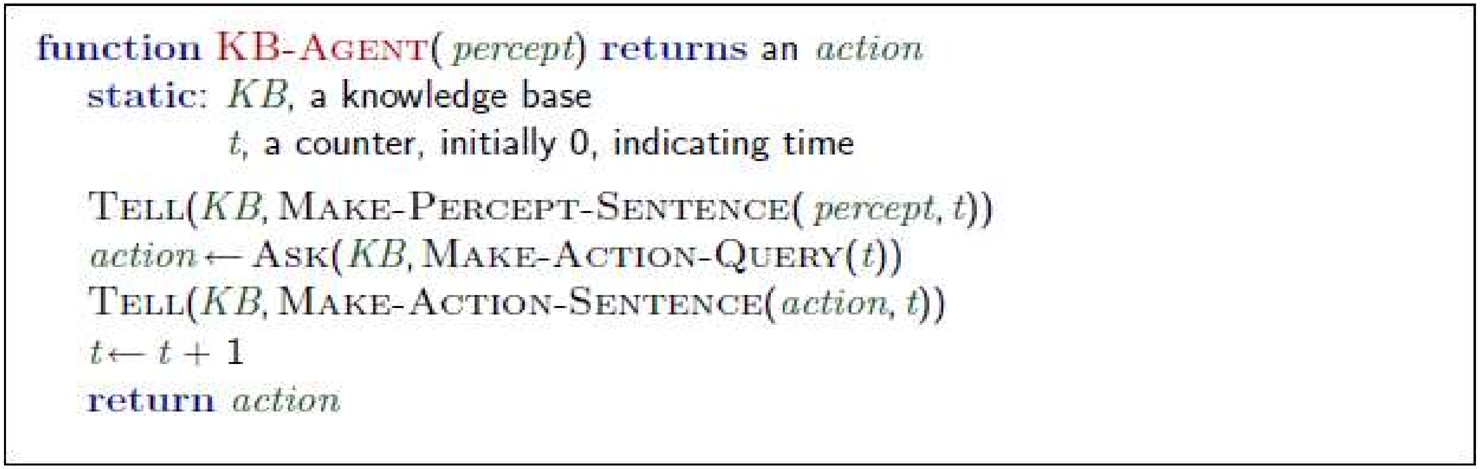
Agents can be viewed at the knowledge level

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



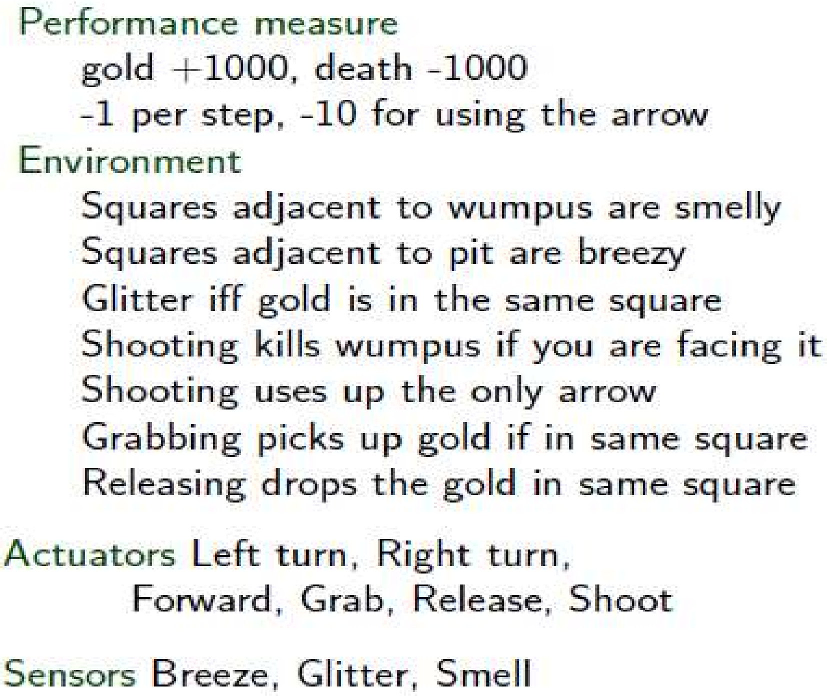
```
function KB-AGENT(percept) returns an action
    static: KB, a knowledge base
            t, a counter, initially 0, indicating time

    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

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

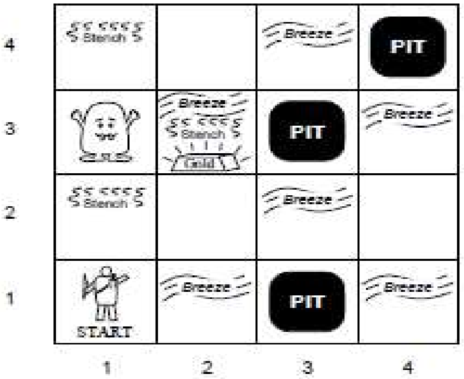
Grabbing picks up gold if in same square

Releasing drops the gold in same square

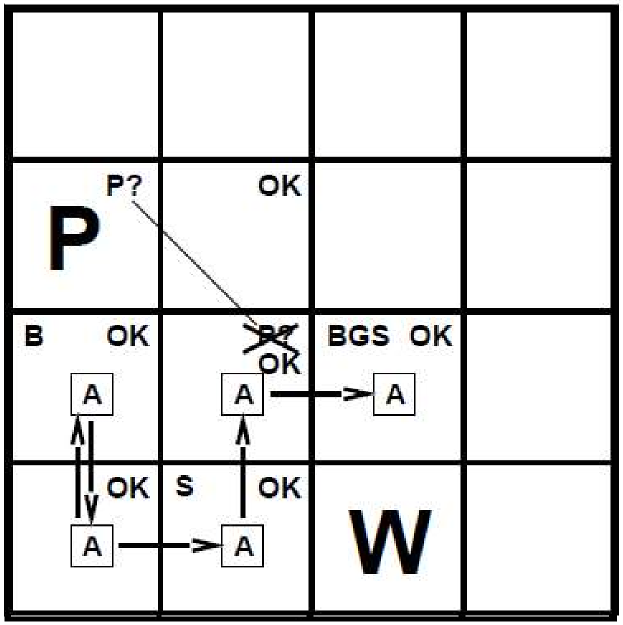
Actuators Left turn, Right turn,

Forward, Grab, Release, Shoot

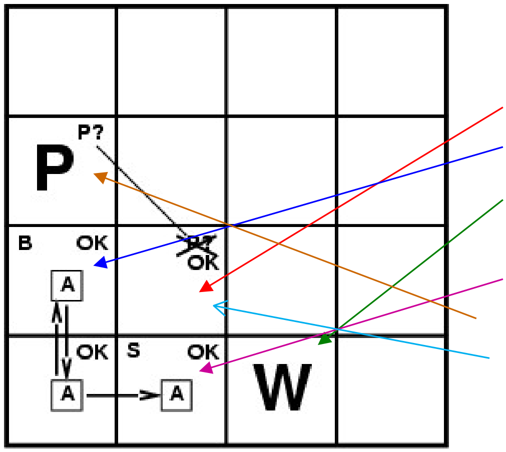
Sensors Breeze, Glitter, Smell



EXPLORING A WUMPUS WORLD



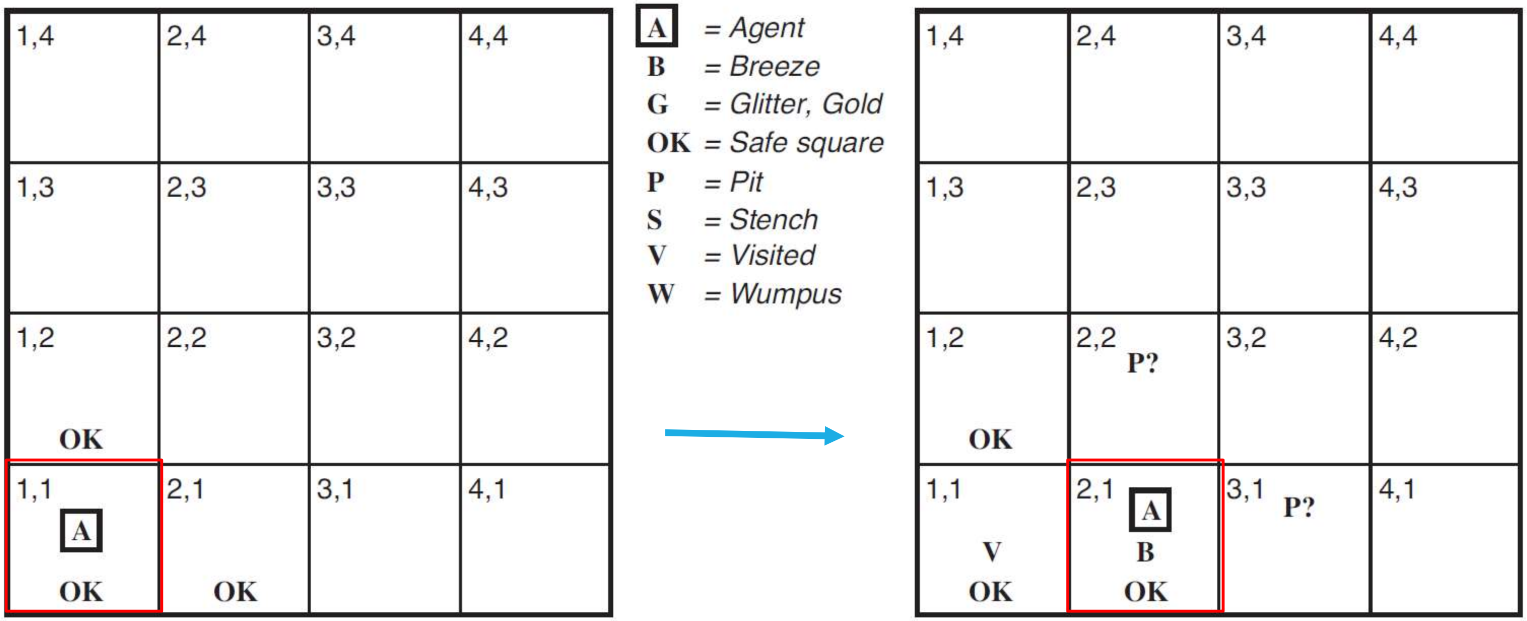
EXPLORING A WUMPUS WORLD



If the Wumpus were here, stench should be here. Therefore it is here.
Since, there is no breeze here, the pit must be there, and it must be OK here

We need rather sophisticated reasoning here!

WUMPUS WORLD



1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 OK	2,2	3,2	4,2
1,1 A OK	2,1 OK	3,1	4,1

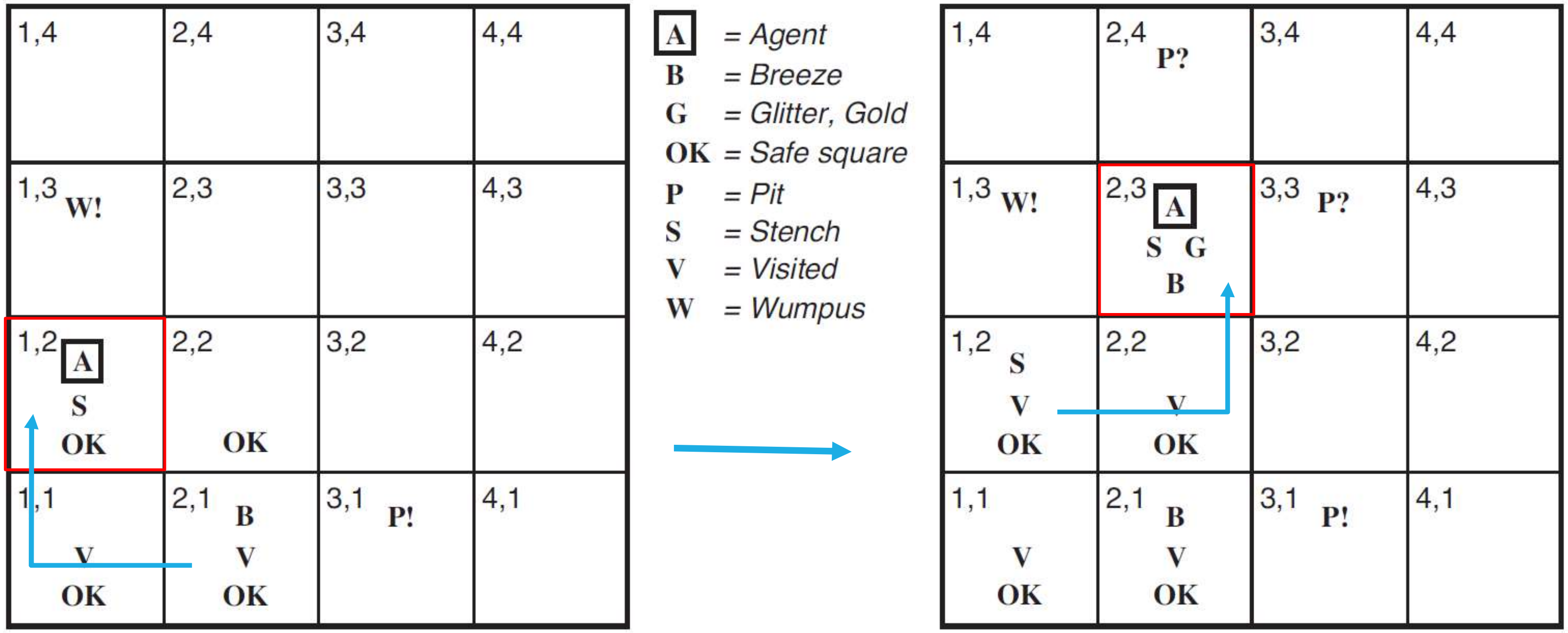
[None, None, None, None, None]

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

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 OK	2,2 P?	3,2	4,2
1,1 V OK	2,1 A B OK	3,1 P?	4,1

[None, Breeze, None, None, None]

WUMPUS WORLD



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

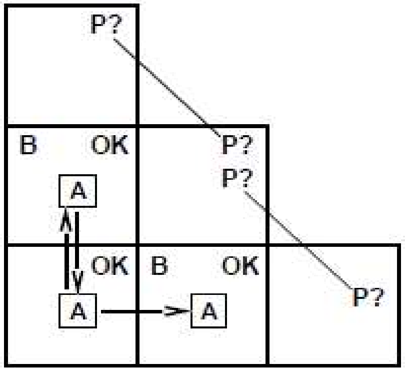
A	= *Agent*
B	= *Breeze*
G	= *Glitter, Gold*
OK	= *Safe square*
P	= *Pit*
S	= *Stench*
V	= *Visited*
W	= *Wumpus*

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

[Stench, None, None, None, None]

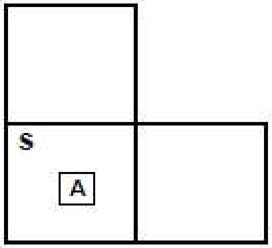
[Stench, Breeze, Glitter, None, None]

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