Agents that Reason Logically

The Story So Far

- **Reflex Agents**.

  Keep no state information.

  Hard to write.

  Can't solve most problems.

- **State Tracking Agents**.

  Maintain state as a single data structure. `((1 1) (1 0) T ((2 2) (3 3)))`.

  In ambiguous situations, we must use a *set* of all consistent states.

  Example: Search-based agent in original Vacuum World.

Neither method can scale up to large problems with ambiguity.

Knowledge-Based Approach

Key Problem: We can't express *partial knowledge* about the current state.

Solution: Use an *intensional representation* based on formal logic.

$$\textsc{At}(Agent, [1,1], s1) \qquad \textsc{Heading}(Agent, [1,0], s1) \qquad \textsc{Power}(On, s1)$$
$$\textsc{Dirt}([2,2], s1) \qquad \textsc{Dirt}([3,3], s1)$$

where, $s1$ is the starting state, $Agent$ is the agent.

Logical Agents

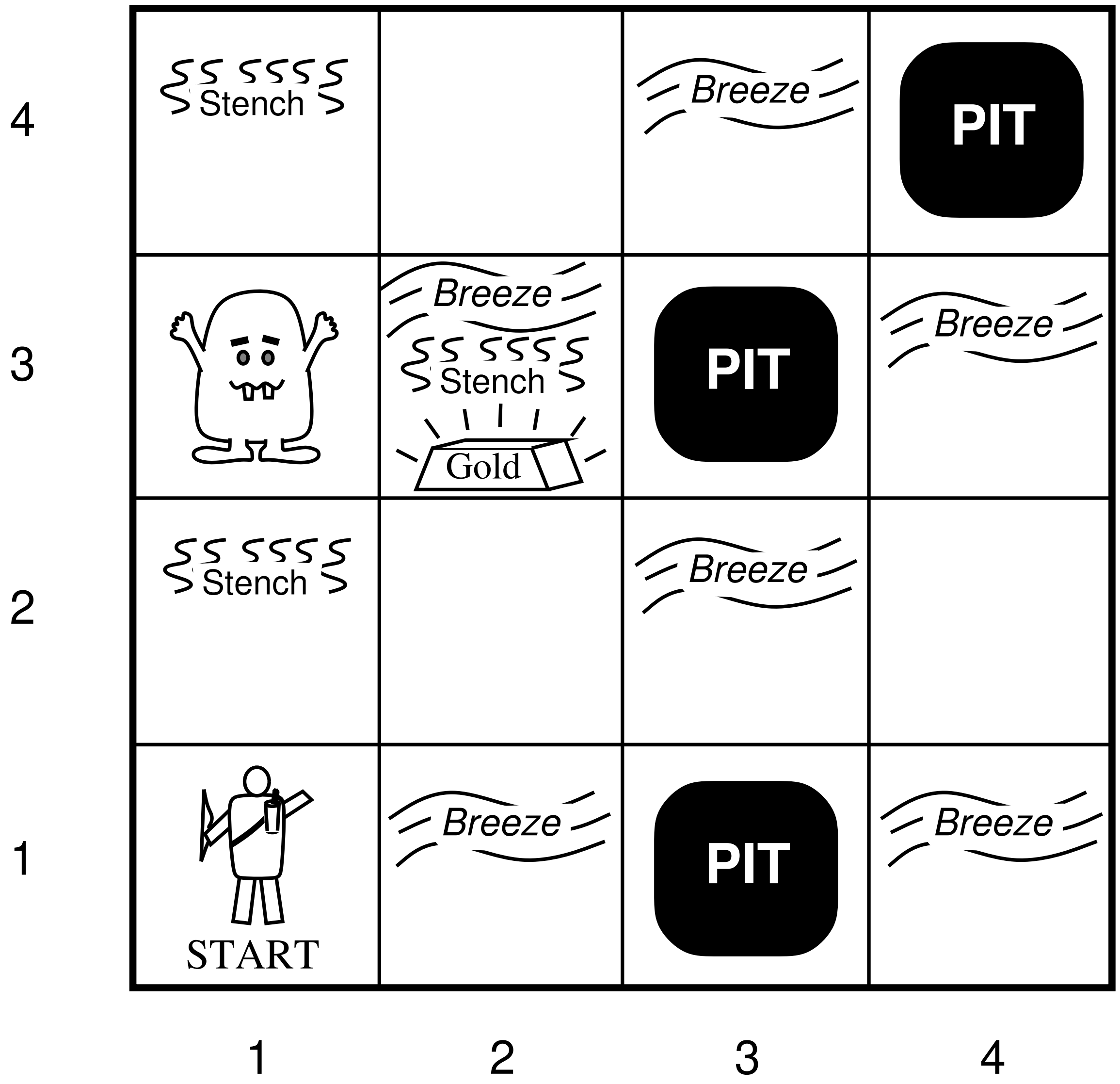
Knowledge Base (KB) abstract data type

- TELL($KB, Fact$) records the $Fact$ in the KB.
- ASK($KB, Query$) asks the KB whether the $Fact$ is true. Returns Yes, No, or Unknown, depending on whether the query can be logically proved from the facts in the knowledge base.

```
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
```

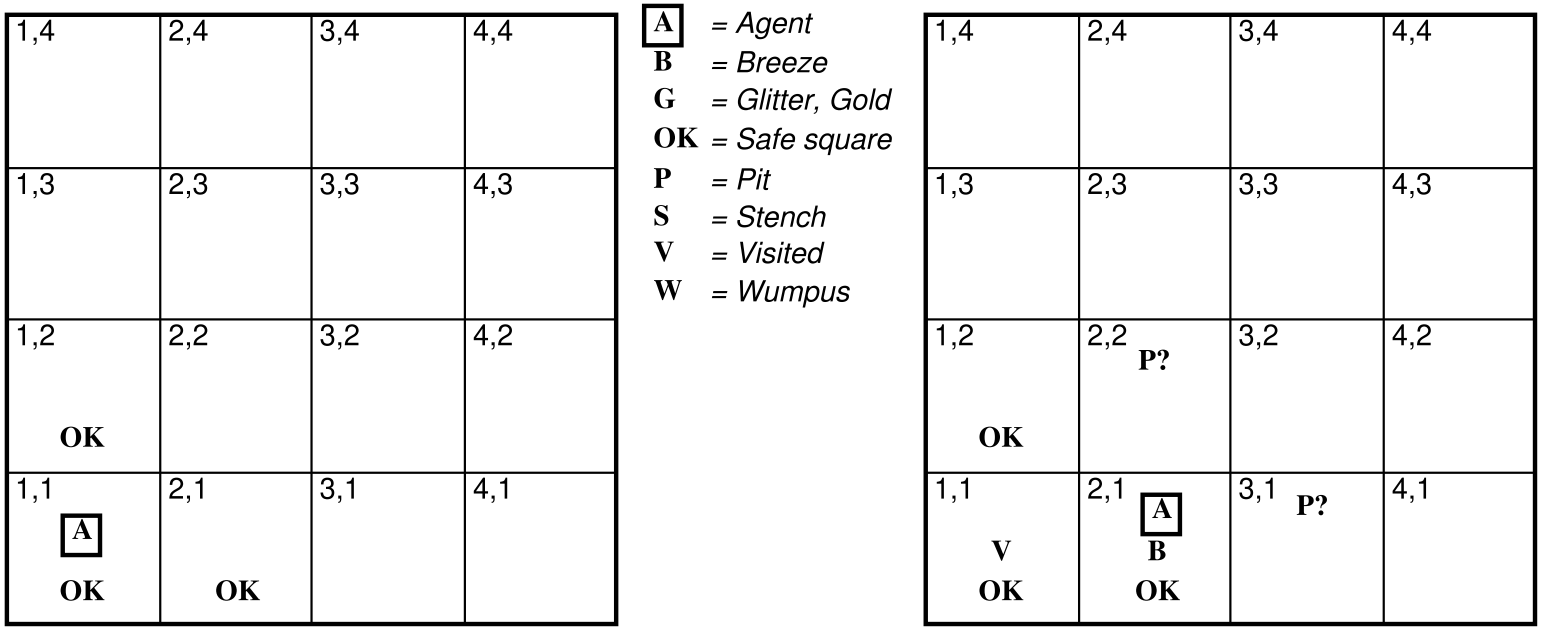
The Wumpus World



The Wumpus World Part 2

- Sensors: Stench, Breeze, Glitter, Bump, Scream
- Actions: Forward, Turn Right, Turn Left, Shoot, Grab, Climb
- There is a stench in every room adjacent to a Wumpus.
- There is a breeze in every room adjacent to a pit.
- There is glitter in the room with the gold.
- If the agent Shoots while facing a wumpus (one or more squares away), the arrow will kill the wumpus and the agent will perceive a scream.
- If the agent Grabs while in the room with the glitter, it will pick up the gold.
- If the agent Climbs while in the start square, it will leave the maze (and end the game).

Acting and Reasoning in the Wumpus World

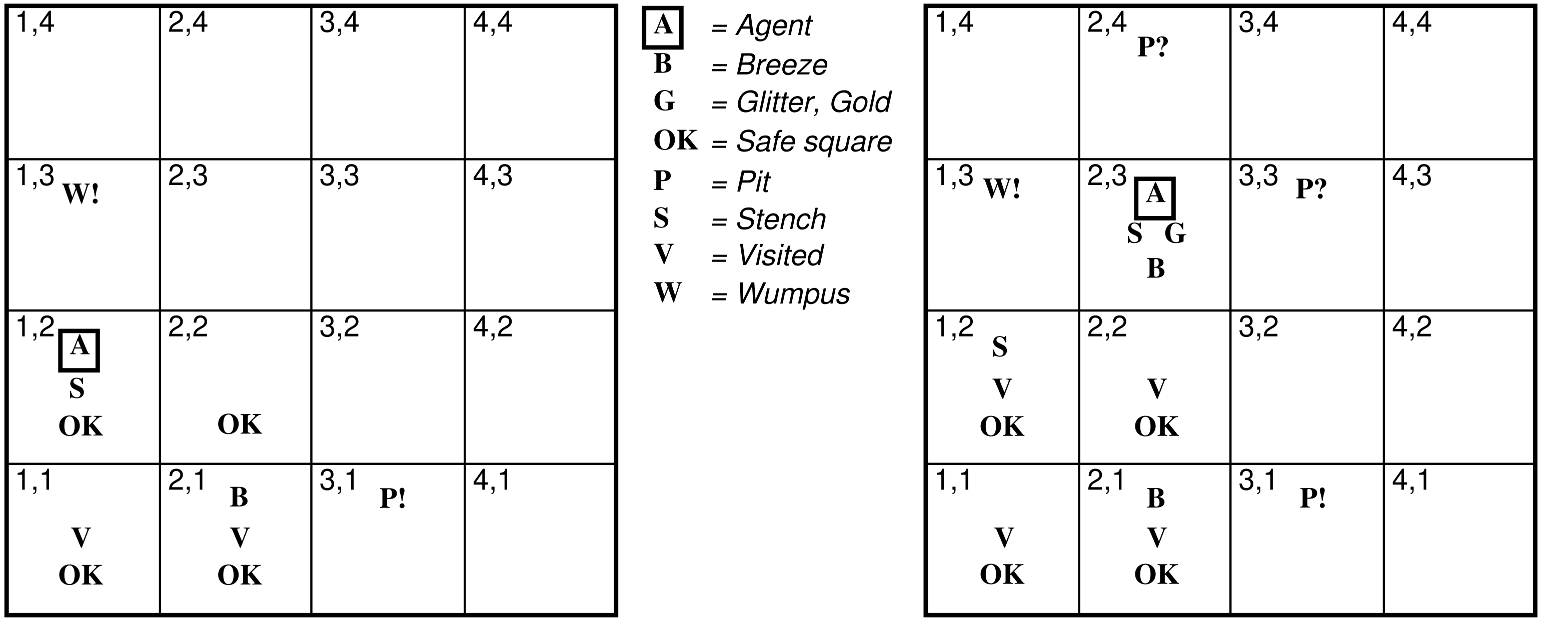


(a) (b)

First action: Forward

First percept: Breeze, no stench, no bump, no scream, no glitter.

Acting and Reasoning in the Wumpus World 2



(a) (b)

Actions: (turn right) (turn right) forward (turn right) forward → (a)

Percept at (a): stench, no breeze, no bump, no scream, no glitter

Actions: (turn right) forward (turn left) forward → (b)

Percept at (b): stench, breeze, glitter, no scream, no bump

First-Order Logic

- **Predicate Symbols:** $\textsc{Dirt}$, $\textsc{At}$, $\textsc{Adjacent}$, $=$
- **Constant Symbols:** $Agent$, $Wumpus$, Pit, S_1, S_2, 1, 2, nil
- **Function Symbols:** $Cons$, $Turn$, $Result$
- **Connectives:** $\land$, $\lor$, $\neg$, $\Rightarrow$, $\Leftrightarrow$
- **Quantifiers:** $\forall$, $\exists$
- **Variables:** s, a, l

Example Logical Formulas:

$$\forall\, l_1, l_2, s\ \textsc{At}(Wumpus, l_1, s) \land \textsc{Adjacent}(l_1, l_2) \Rightarrow \textsc{Smelly}(l_2)$$

$$\forall\, l_1, l_2, s\ \textsc{Pit}(l_1, s) \land \textsc{Adjacent}(l_1, l_2) \Rightarrow \textsc{Breezy}(l_2)$$

Logical Rules of Inference: Affirm Antecedent (Modus Ponens)

$$\frac{\begin{array}{c}\forall\, x\ P(x)\ \Rightarrow\ Q(x)\\ P(a)\end{array}}{Q(a)}$$

Example:

$$\frac{\begin{array}{c}\forall\, l_1, l_2, s\ \textsc{At}(Wumpus, l_1, s)\ \wedge\ \textsc{Adjacent}(l_1, l_2)\ \Rightarrow\ \textsc{Smelly}(l_2)\\ \textsc{At}(Wumpus, [1,2], S_0)\ \wedge\ \textsc{Adjacent}([1,2],[1,3])\end{array}}{Smelly([1,3])}$$

Another Rule of Inference: Deny Consequent

$$\frac{\begin{array}{c}\forall\, x\ P(x)\ \Rightarrow\ Q(x)\\ \neg Q(a)\end{array}}{\neg P(a)}$$

Example:

$$\frac{\begin{array}{c}\forall\, l_1, l_2, s\ \textsc{At}(Wumpus, l_1, s)\ \wedge\ \textsc{Adjacent}(l_1, l_2)\ \Rightarrow\ \textsc{Smelly}(l_2)\\ \neg\textsc{Smelly}([1,3])\end{array}}{\forall\, l_1, s\ \neg(\textsc{At}(Wumpus, l_1, s)\ \wedge\ \textsc{Adjacent}(l_1, [1,3]))}$$

This last line is equivalent (by DeMorgan's law) to

$$\forall\, l_1, s\ \neg\textsc{At}(Wumpus, l_1, s)\ \vee\ \neg\textsc{Adjacent}(l_1, [1,3]))$$

Another Rule of Inference: Deny Disjunct (Resolution)

$$\frac{\begin{array}{c}\forall\, x\ P(x)\ \vee\ Q(x)\\ \neg Q(a)\end{array}}{P(a)}$$

The signs of the predicates can be flipped. Example:

$$\frac{\begin{array}{c}\forall\, l_1, s\ \neg\textsc{At}(Wumpus, l_1, s)\ \vee\ \neg\textsc{Adjacent}(l_1, [1,3])\\ \textsc{Adjacent}([1,2],[1,3])\end{array}}{\forall s \neg\textsc{At}(Wumpus, [1,2], s)}$$

ASK must implement these rules of inference to answer queries.

Many systems convert all logical formulas into disjunctions (also known as *clauses*) and then apply general versions of the resolution rule.

Interpreting Percepts

Extract Individual Percepts

$$\forall\, b, g, u, c, t\ \textsc{Percept}([Stench, b, g, u, c], t) \ \Rightarrow\ \textsc{Stench}(t)$$

$$\forall\, b, u, c, s, t\ \textsc{Percept}([s, Breeze, g, u, c], t) \ \Rightarrow\ \textsc{Breeze}(t)$$

Combine with State to make Inferences about Locations

$$\forall\, l, s\ \textsc{At}(Agent, l, s) \wedge \textsc{Stench}(s) \ \Rightarrow\ \textsc{Smelly}(l)$$

$$\forall\, l, s\ \textsc{At}(Agent, l, s) \wedge \textsc{Breeze}(s) \ \Rightarrow\ \textsc{Breezy}(l)$$

Combine with Causal Rules to Infer Locations of Wumpus, Pits

$$\forall\, l_1, l_2, s\ \textsc{At}(Wumpus, l_1, s) \wedge \textsc{Adjacent}(l_1, l_2) \ \Rightarrow\ \textsc{Smelly}(l_2)$$

$$\forall\, l_1, l_2, s\ \textsc{Pit}(l_1, s) \wedge \textsc{Adjacent}(l_1, l_2) \ \Rightarrow\ \textsc{Breezy}(l_2)$$

$$\forall\, x, t\ (\neg\textsc{At}(Wumpus, x, t) \wedge \neg\textsc{Pit}(x)) \ \Rightarrow\ \mathrm{OK}(x)$$

Connecting Inferences to Actions

Rate Each Action

$$\forall\, l_1, s\ \textsc{At}(Wumpus, l_1, s)\ \wedge\ LocationAhead(Agent, s) = l_1 \quad \Rightarrow \quad \textsc{Deadly}(Forward, s)$$

$$\forall\, l_1, s\ \mathrm{OK}(l_1, s)\ \wedge\ LocationAhead(Agent, s) = l_1\ \wedge\ \neg\textsc{Visited}(l_1, s) \quad \Rightarrow \quad \textsc{Good}(Forward, s)$$

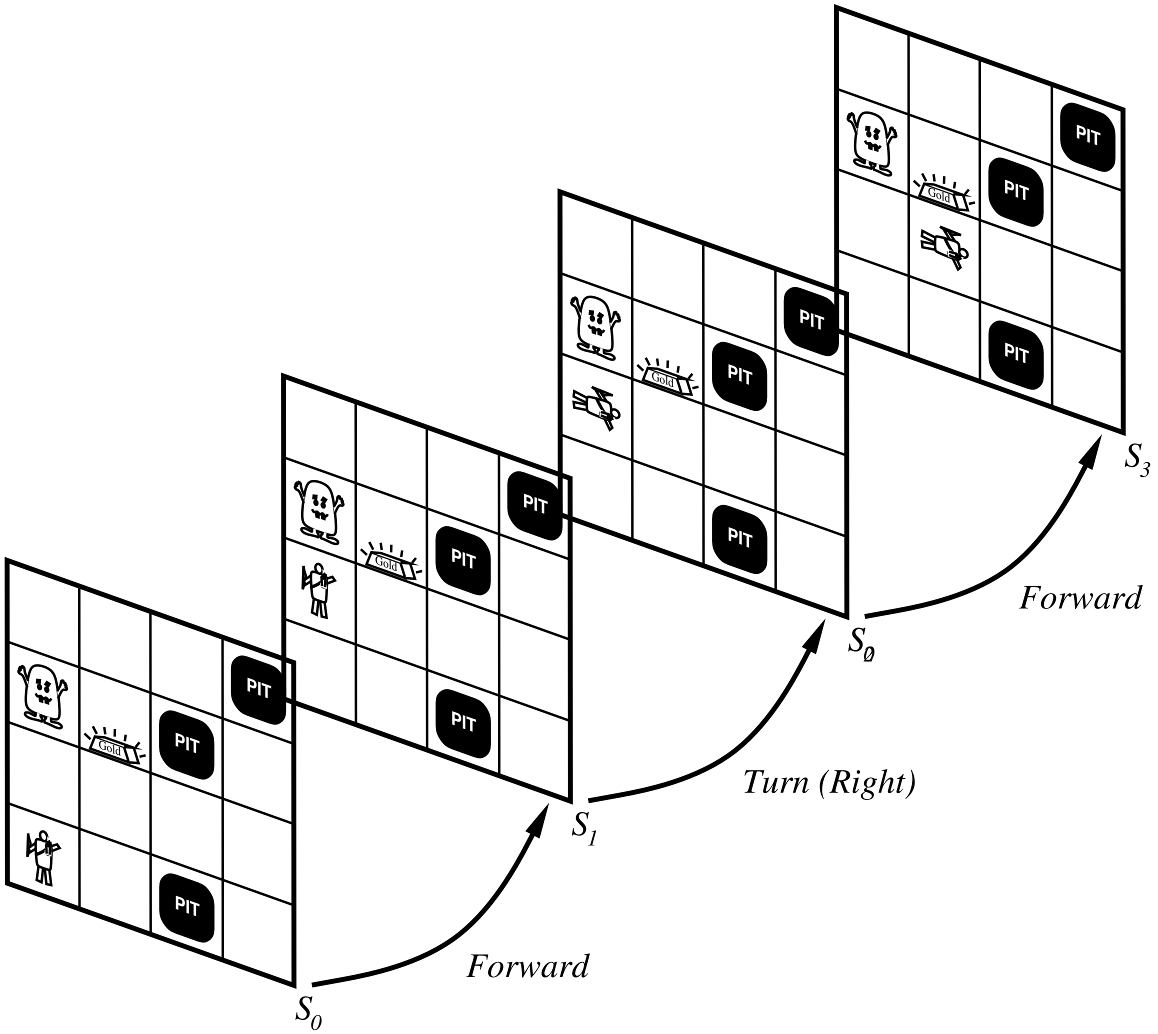
$$\forall\, l_1, s\ \textsc{Gold}(l_1, s)\ \Rightarrow\ \textsc{Great}(Grab, s)$$

Choose the Best Action

$$\forall\, a, s\ \textsc{Great}(a, s) \quad \Rightarrow \quad \textsc{Action}(a, s)$$

$$\forall\, a, s\ \textsc{Good}(a, s)\ \wedge\ (\neg\exists\, b\ \textsc{Great}(b, s)) \quad \Rightarrow \quad \textsc{Action}(a, s)$$

Updating the State Based on Chosen Action



$$Result(Forward, S_0) = S_1 \quad Result(Turn(Right), S_1) = S_2 \quad Result(Forward, S_2) = S_3$$

Updating the State Based on Chosen Action (2)

We need one rule like the following for each action:

$$\begin{aligned}\forall\, a, d, p, s\ \textsc{At}(p, l, Result(a, s)) \ \ \Leftrightarrow \ \ &[(a = Forward \ \wedge \ l = LocationAhead(p, s) \ \wedge \ \neg\textsc{Wall}(l)) \\ &\vee (\textsc{At}(p, l, s) \ \wedge \ a \neq Forward)]\end{aligned}$$

Logical Agents

- Decompose the state representation into individual logical formulas
- Use logical formulas to represent "causal" connections between hidden environment state and percepts.
- Use logical formulas to rank possible actions.
- Use logical formulas to represent the effects of actions.

The resulting agent can be completely generic. All domain-specific knowledge is represented as logical formulas.

Planning Agents

The logical agent we have described is essentially a logical version of our rule-based agent.

It tracks the world (using logical inference) and then chooses an action based on the current state.

We can combine logical agents with search algorithms to produce **Planning Agents** that consider future *sequences* of actions to find a plan that achieves the goals of the agent.

We can also force the logical reasoning system to perform the search internally by issuing queries such as:

$$
\begin{aligned}
&\textsc{Ask}(\mathrm{KB},\\
&\qquad \exists\, a1, a2, a3, a4, a5, t\;\; t = Result(a5, Result(a4, Result(a3, Result(a2, Result(a1, S0)))))\\
&\qquad\qquad\qquad\qquad\qquad \wedge\ \textsc{Holding}(Agents, Gold, t)\ \wedge\ \textsc{At}(Agent, OutsideCave, t))
\end{aligned}
$$

This usually is a mistake, because there is no easy way to provide heuristic guidance to the reasoning system.

Logic, Uncertainty, and Utility

Advantages of Logic-Based Agents

- High-level language for tracking environments.
- Permits modular decomposition of state representation.

Limitations of Logic-Based Agents

- Cannot track stochastic environments.
- Cannot represent and reason with utilities—can't make tradeoffs.