Logical Agents

Logical agents for Wumpus World

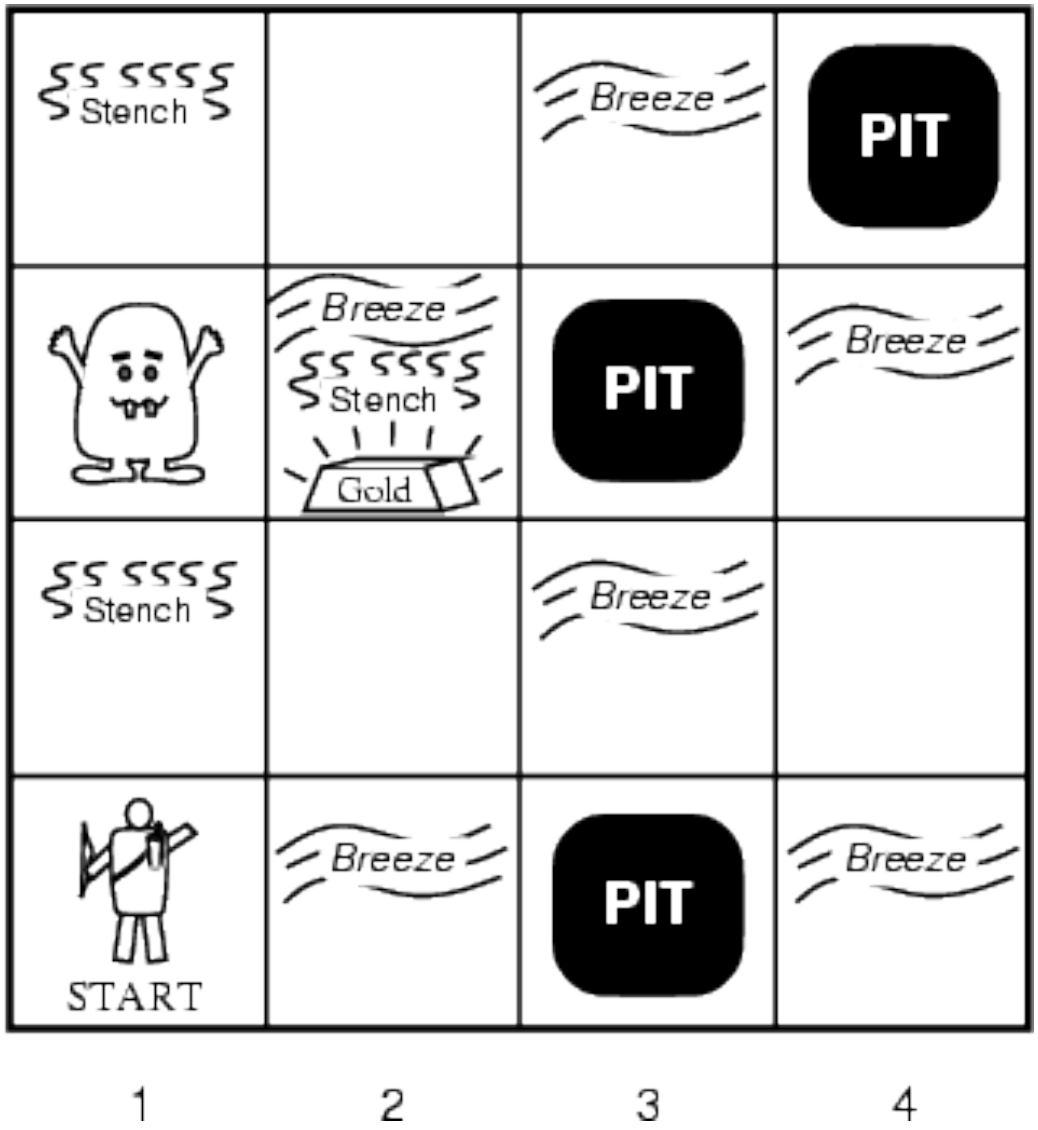
We'll use the Wumpus World domain to explore three (non-exclusive) agent architectures:

- Reflex agents
  - Have rules that classify situations based on percepts and specify how to react to each possible situation
- Model-based agents
  - Construct an internal model of their world
- Goal-based agents
  - Form goals and try to achieve them

AIMA's Wumpus World

The agent always starts in the field [1,1]

Agent's task is to find the gold, return to the field [1,1] and climb out of the cave



A simple reflex agent: if-then rules

- Rules to map percepts into observations:

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

  $\forall s,g,u,c,t\ \text{Percept}([s, \text{Breeze}, g, u, c], t) \rightarrow \text{Breeze}(t)$

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

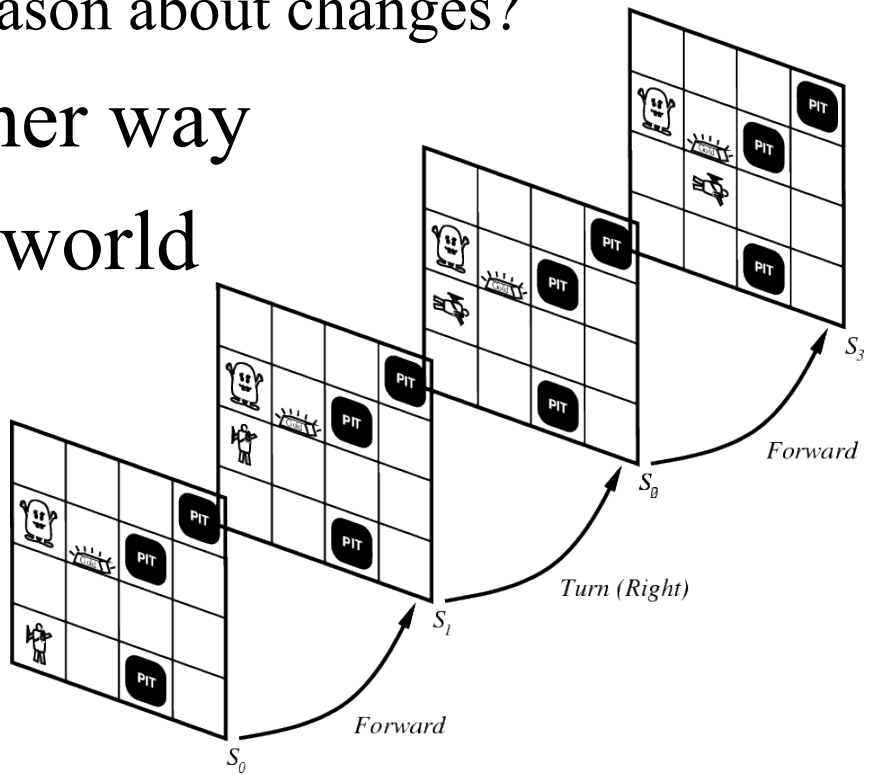
- Rules to select action given observations:

  $\forall t\ \text{AtGold}(t) \rightarrow \text{Action}(\text{Grab}, t);$

- Difficulties:
  - Consider Climb: No percept indicates agent should climb out; position & holding gold not part of percept sequence
  - Loops: percepts repeated when you return to a square, causing same response (unless we maintain some internal model of the world)

Representing change

- Representing changing world in logic can be tricky
- One way is just to change the KB
  - Add and delete sentences from KB to reflect changes
  - How do we remember past, or reason about changes?
- **Situation calculus** is another way
- **Situation**: snapshot of the world at some instant in time
- When the agent performs action A in situation S1, result is new situation S2



Situation calculus

A **situation** is a snapshot of the world at an interval of time during which nothing changes w.r.t a particular situation

- Add **situation variables** to every predicate.
- at(Agent, L) becomes at(Agent, L, s0): at(Agent, L) true in situation (i.e., state) s0
- Or, add a special second order predicate, **holds(f, s),** meaning "f is true in situation s", e.g., holds(at(Agent, L), s0)

Situation calculus

- Add new function, ***result(a, s)***, mapping situation ***s*** to new situation as result of performing action ***a***
  - i.e., *result(forward, s)* is a function returning next situation
- Example: The action agent-walks-to-location-y could be represented by

$$(\forall x)(\forall y)(\forall s)\ (at(Agent, L1, S) \wedge \neg onbox(S)) \rightarrow at(Agent, L2, result(walk(L2), S))$$

Deducing hidden properties

- From the perceptual information we obtain in situations, we can **infer properties of locations**

  $\forall l,s\ at(Agent, L, s) \wedge Breeze(s) \rightarrow Breezy(L)$

  $\forall l,s\ at(Agent, L\ s) \wedge Stench(s) \rightarrow Smelly(L)$

- Neither *Breezy* nor *Smelly* need situation arguments because pits and the Wumpus do not move around

Deducing hidden properties II

- We need rules relating aspects of a single world state (as opposed to between states)
- Two main kinds of such rules:
  - **Causal rules** reflect assumed direction of causality

    $(\forall L1,L2,S)\ at(Wumpus,L1,S) \wedge adjacent(L1,L2) \rightarrow Smelly(L2)$

    $(\forall\ L1,L2,S)\ at(Pit,L1,S) \wedge adjacent(L1,L2) \rightarrow Breezy(L2)$

    - Systems that reason with causal rules are model-based reasoning systems
  - **Diagnostic rules** infer presence of hidden properties directly from percept-derived information, e.g.

    $(\forall\ L,S)\ at(Agent,L,S) \wedge Breeze(S) \rightarrow Breezy(L)$

    $(\forall\ L,S)\ at(Agent,L,S) \wedge Stench(S) \rightarrow Smelly(L)$

Blocks world

The **blocks world** is a micro-world consisting of a table, a set of blocks and a robot hand.

Some domain constraints:

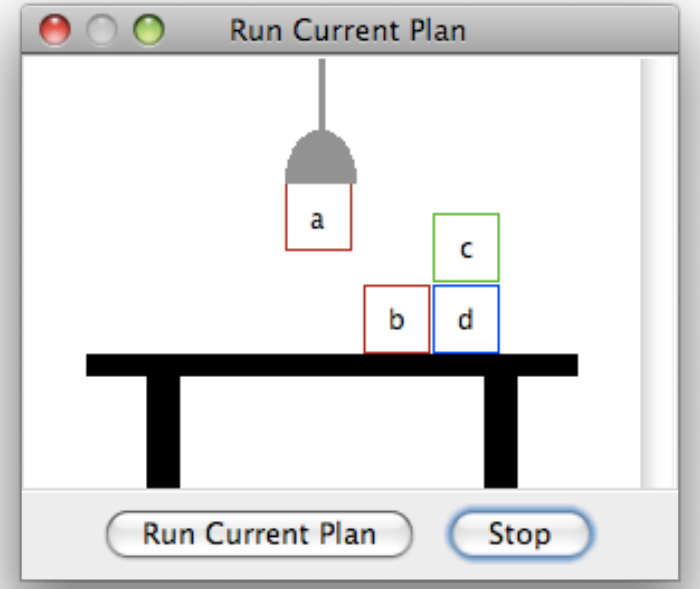
- Only one block can be on another block
- Any number of blocks can be on the table
- The hand can only hold one block

Typical representation:

ontable(b) ontable(d)

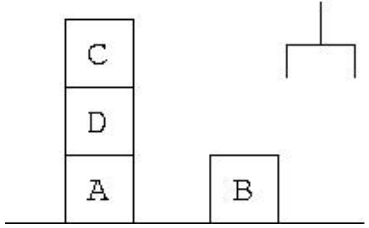
on(c,d) holding(a)

clear(b) clear(c)



Meant to be a simple model!

Representing change



blocks world

- **Frame axioms** encode what's **not** changed by an action
- E.g., moving a clear block to the table doesn't change the location of any other blocks

  On (x, z, s) ∧ Clear (x, s) →
  On (x, table, Result(Move(x, table), s)) ∧
  ¬On(x, z, Result (Move (x, table), s))

  On (y, z, s) ∧ y≠ x → On (y, z, Result (Move (x, table), s))
- Proliferation of frame axioms becomes very cumbersome in complex domains
  - What about color, size, shape, ownership, etc.

The frame problem II

- **Successor-state axiom** characterizes every way in which a particular predicate can become true:
  - Either it can be **made true**, or it can **already be true and not be changed**:
  - On (x, table, Result(a,s)) ↔
    [On (x, z, s) ∧ Clear (x, s) ∧ a = Move(x, table)] v
    [On (x, table, s) ∧ a ≠ Move (x, z)]
- Complex worlds require reasoning about long action chains; even these types of axioms are too cumbersome

  Planning systems use custom inference methods to reason about the expected state of the world during multi-step plans

Qualification problem

- How can you characterize every effect of an action, or every exception that might occur?
- Putting my bread into the toaster, & pushing the button, it will become toasted after two minutes, unless…
  - The toaster is broken, or…
  - The power is out, or…
  - I blow a fuse, or…
  - A neutron bomb explodes nearby and fries all electrical components, or…
  - A meteor strikes the earth, and the world we know it ceases to exist, or…

Ramification problem

Nearly impossible to characterize every side effect of every action, at every level of detail

When I put my bread into the toaster, and push the button, the bread will become toasted after two minutes, and…

- The crumbs that fall off the bread onto the bottom of the toaster over tray will also become toasted, and…
- Some of the those crumbs will become burnt, and…
- The outside molecules of the bread will become "toasted," and…
- The inside molecules of the bread will remain more "breadlike," and…
- The toasting process will release a small amount of humidity into the air because of evaporation, and…
- The heating elements will become a tiny fraction more likely to burn out the next time I use the toaster, and…
- The electricity meter in the house will move up slightly, and…

Knowledge engineering!

- Modeling the *right* conditions and the *right* effects at the *right* level of abstraction is difficult
- Knowledge engineering (creating & maintaining KBs for intelligent reasoning) is field unto itself
- We hope automated knowledge acquisition and machine learning tools can fill the gap:
  - Intelligent systems should **learn** about conditions and effects, just like we do!
  - Intelligent systems should learn when to pay attention to, or reason about, certain aspects of processes, depending on context. (metacognition?)

Preferences among actions

- A problem with the WWKB described so far is how to decide which of several actions is best
- E.g., how to decide between forward and grab, axioms describing when it is OK to move to a square would have to mention glitter
- This is not modular!
- We can solve this problem by separating **facts about actions** from **facts about goals**
- This way our agent can be reprogrammed just by asking it to achieve different goals

Preferences among actions

- First step: describe the desirability of actions independent of each other
- We can use a simple scale: actions can be *Great, Good, Medium, Risky*, or *Deadly*
- Obviously, the agent should always do the best action it can find:

$(\forall a,s)\ \text{Great}(a,s) \rightarrow \text{Action}(a,s)$

$(\forall a,s)\ \text{Good}(a,s) \wedge \neg(\exists b)\ \text{Great}(b,s) \rightarrow \text{Action}(a,s)$

$(\forall a,s)\ \text{Medium}(a,s) \wedge (\neg(\exists b)\ \text{Great}(b,s) \vee \text{Good}(b,s)) \rightarrow \text{Action}(a,s)$

Preferences among actions

Until it finds gold, basic agent strategy can be:

- **Great actions:** picking up the gold when found, climbing out of the cave with the gold
- **Good actions:** moving to a square that's OK and hasn't been visited yet
- **Medium actions:** moving to a square that is OK and has already been visited
- **Risky actions:** moving to a square that's not known to be deadly or OK
- **Deadly actions:** moving into a square that is known to have a pit or a Wumpus

Goal-based agents

- Once gold is found, we must change strategies, requiring a new set of action values.
- We could encode this as a rule:
  - $(\forall s)$ Holding(Gold,s) $\rightarrow$ GoalLocation([1,1]),s)
- We must decide how the agent will work out a sequence of actions to accomplish the goal
- Three possible approaches:
  - **Inference**: good versus wasteful solutions
  - **Search**: a problem with operators and set of states
  - **Planning**: to be discussed later

Coming up next

- Logical inference
- Knowledge representation
- Planning