

Logical Agents

Logical agents for the Wumpus World

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

A simple reflex agent

- Rules to **map percepts into observations**:

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

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

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

- Rules to **select an action given observations**:

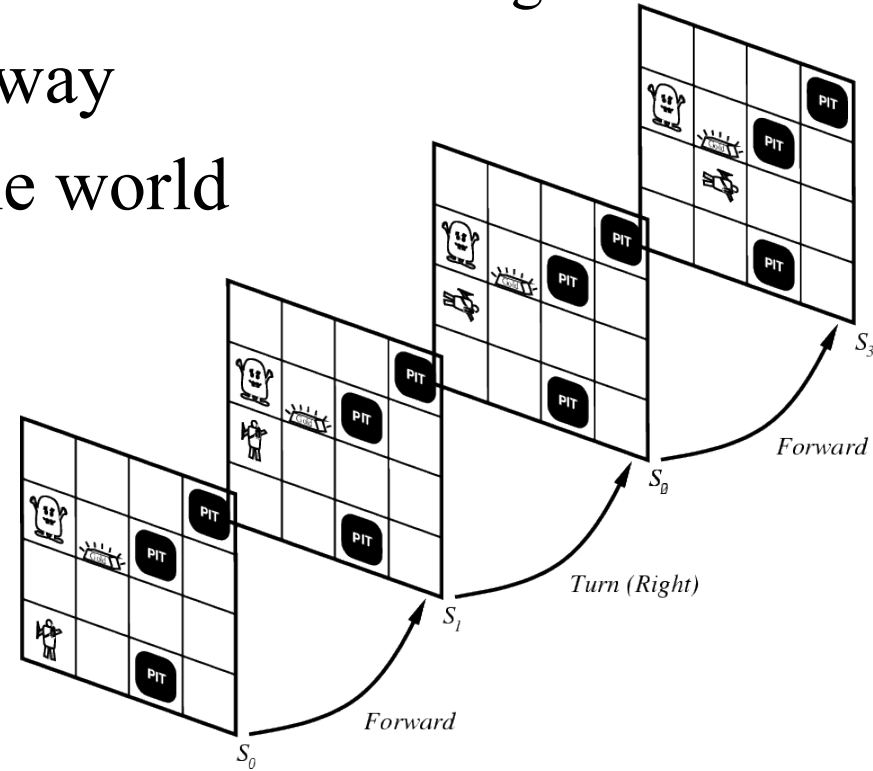
$\forall t \text{ AtGold}(t) \rightarrow \text{Action}(\text{Grab}, t);$

- Some difficulties:

- Consider Climb: There's no percept that indicates the agent should climb out – **position and holding gold are not part of the percept sequence**
- Loops – the percept will be repeated when you return to a square, which should cause the same response (unless we maintain some **internal model of the world**)

Representing change

- Representing a changing world in logic can be tricky
- One way is just to change the KB
 - Add and delete sentences from the KB to reflect changes
 - How do we remember the past, or reason about changes?
- **Situation calculus** is another way
- A **situation** is a 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 (1)

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.
- $\text{at}(\text{Agent}, L)$ becomes $\text{at}(\text{Agent}, L, s_0)$:
 $\text{at}(\text{Agent}, L)$ true in situation (i.e., state) s_0
- Or, add a special second order predicate, **holds(f, s)**, meaning “f is true in situation s”,
e.g., $\text{holds}(\text{at}(\text{Agent}, L), s_0)$

Situation calculus (2)

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

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

Deducing hidden properties

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

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

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

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

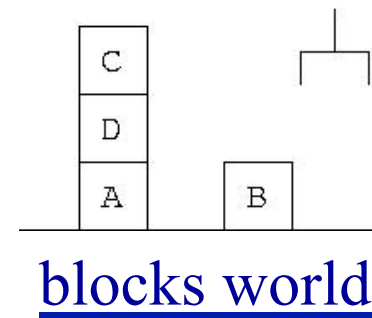
Deducing hidden properties II

- We need rules relating aspects of a single world state (as opposed to between states)
- There are two main kinds of such rules:
 - **Causal rules** reflect assumed direction of causality:
 - $(\forall L1, L2, S) \text{ at}(\text{Wumpus}, L1, S) \wedge \text{adjacent}(L1, L2) \rightarrow \text{Smelly}(L2)$
 - $(\forall L1, L2, S) \text{ at}(\text{Pit}, L1, S) \wedge \text{adjacent}(L1, L2) \rightarrow \text{Breezy}(L2)$

Systems that reason with causal rules are **model-based reasoning systems**

- **Diagnostic rules** infer presence of **hidden properties** directly from the percept-derived information, e.g.
 - $(\forall L, S) \text{ at}(\text{Agent}, L, S) \wedge \text{Breeze}(S) \rightarrow \text{Breezy}(L)$
 - $(\forall L, S) \text{ at}(\text{Agent}, L, S) \wedge \text{Stench}(S) \rightarrow \text{Smelly}(L)$

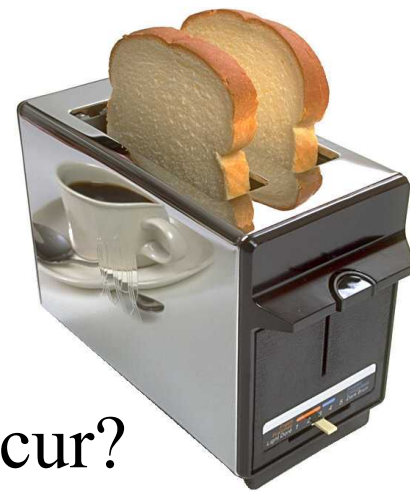
Representing change



- **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
 - $\text{On}(x, z, s) \wedge \text{Clear}(x, s) \rightarrow$
 $\text{On}(x, \text{table}, \text{Result}(\text{Move}(x, \text{table}), s)) \wedge$
 $\neg \text{On}(x, z, \text{Result}(\text{Move}(x, \text{table}), s))$
 - $\text{On}(y, z, s) \wedge y \neq x \rightarrow \text{On}(y, z, \text{Result}(\text{Move}(x, \text{table}), s))$
- The proliferation of frame axioms becomes very cumbersome in complex domains

The frame problem II

- **Successor-state axiom:** General statement that 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**:
 - $\text{On}(x, \text{table}, \text{Result}(a,s)) \leftrightarrow$
 $[\text{On}(x, z, s) \wedge \text{Clear}(x, s) \wedge a = \text{Move}(x, \text{table})] \vee$
 $[\text{On}(x, \text{table}, s) \wedge a \neq \text{Move}(x, z)]$
- In complex worlds, where you want to reason about longer chains of action, even these types of axioms are too cumbersome
 - Planning systems use special-purpose inference methods to reason about the expected state of the world at any point in time during a multi-step plan



Qualification problem

- How can you characterize every effect of an action, or every exception that might occur?
- When I put my bread into the toaster, and push 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

It's nearly impossible to characterize every side effect of every action, at every possible 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 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 and maintaining KBs for intelligent reasoning) is a field unto itself
- Many hope that automated knowledge acquisition and machine learning tools can fill the gap:
 - Our intelligent systems should be able to **learn** about the conditions and effects, just like we do!
 - Our intelligent systems should be able to learn when to pay attention to, or reason about, certain aspects of processes, depending on the context!

Preferences among actions

- A problem with the Wumpus world KB described so far is how to decide which of several possible actions is best
- For example, to decide between a forward and a 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 is to 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 realized this way:

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

Coming up next

- Logical inference
- Knowledge representation
- Planning