CMSC 471
Fall 2012

Class #12

Tuesday, October 9th
First-Order Logic

Kevin Winner, winnerk1@umbc.edu

Today's Class

- Project Groups
- First Order Logic
- Challenges with Designing Logical Agents
- HW2 (maybe)

First-Order Logic

Chapter 8

First-Order Logic

First-order logic (FOL) models the world in terms of

- **Objects,** which are things with individual identities
- **Properties** of objects that distinguish them from other objects
- **Relations** that hold among sets of objects
  - Relations can mostly be classed into **predicates** and **functions**.

Examples:

- Objects: Students, lectures, companies, cars ...
- Properties: blue, oval, even, large, ...
- Predicates: Brother-of, bigger-than, outside, part-of, has-color, occurs-after, owns, visits, precedes, ...
- Functions: father-of, best-friend, second-half, one-more-than ...

User Provides

Constant symbols, which represent individuals in the world

Mary

3

Green

Function symbols, which map individuals to individuals

father-of(Mary) = John

color-of(Sky) = Blue

Predicate symbols, which map individuals to truth values

greater(5,3)

green(Grass)

color(Grass, Green)

FOL Provides

Variable symbols

E.g., x, y, foo

Connectives

Same as in PL: not (¬), and (∧), or (∨), implies (→), if and only if (biconditional ↔)

Quantifiers

Universal ∀**x** or **(Ax)**

Existential ∃**x** or **(Ex)**

Quantifiers

Universal quantification

- $(\forall x)P(x)$ means that P holds for **all** values of x in the domain associated with that variable
- E.g., $(\forall x)$ dolphin(x) $\rightarrow$ mammal(x)

Existential quantification

- $(\exists\, x)P(x)$ means that P holds for **some** value of x in the domain associated with that variable
- E.g., $(\exists\, x)$ mammal(x) $\wedge$ lays-eggs(x)
- Permits one to make a statement about some object without naming it

Quantifiers

Universal quantifiers are often used with "implies" to form "rules":

$(\forall x)$ student(x) $\rightarrow$ smart(x) means "All students are smart"

Universal quantification is *rarely* used to make blanket statements about every individual in the world:

$(\forall x)$student(x)$\wedge$smart(x) means "Everyone in the world is a student and is smart"

Existential quantifiers are usually used with "and" to specify a list of properties about an individual:

$(\exists x)$ student(x) $\wedge$ smart(x) means "There is a student who is smart"

We can combine several quantifiers (one for each variable) in a single sentence

$(\forall x\ \exists y)$ parent(x, y)

It is also possible to negate quantifiers

Quantifier Scope

Switching the order of universal quantifiers *does not* change the meaning:

$$(\forall x)(\forall y)P(x,y) \leftrightarrow (\forall y)(\forall x)\ P(x,y)$$

Similarly, you can switch the order of existential quantifiers:

$$(\exists x)(\exists y)P(x,y) \leftrightarrow (\exists y)(\exists x)\ P(x,y)$$

Switching the order of universals and existentials *does* change meaning:

Everyone likes someone: $(\forall x)(\exists y)$ likes(x,y)

Someone is liked by everyone: $(\exists y)(\forall x)$ likes(x,y)

Connections between All and Exists

We can relate sentences involving $\forall$ and $\exists$ using De Morgan's laws:

$$(\forall x)\ \neg P(x) \leftrightarrow \neg(\exists x)\ P(x)$$

$$\neg(\forall x)\ P \leftrightarrow (\exists x)\ \neg P(x)$$

$$(\forall x)\ P(x) \leftrightarrow \neg\ (\exists x)\ \neg P(x)$$

$$(\exists x)\ P(x) \leftrightarrow \neg(\forall x)\ \neg P(x)$$

Quantified Inference Rules

Universal instantiation

$\forall x \, P(x) \therefore P(A)$

Universal generalization

$P(A) \wedge P(B) \ldots \therefore \forall x \, P(x)$

Existential instantiation

$\exists x \, P(x) \therefore P(F)$ ← **skolem constant F**

Existential generalization

$P(A) \therefore \exists x \, P(x)$

Universal Instantiation (a.k.a. Universal Elimination)

If $(\forall x)\ P(x)$ is true, then $P(C)$ is true, where C is *any* constant in the domain of x

Example:

$(\forall x)$ eats(Ziggy, x) $\Rightarrow$ eats(Ziggy, IceCream)

The variable symbol can be replaced by any ground term, i.e., any constant symbol or function symbol applied to ground terms only

Universal/Existential Generalization (a.k.a. Introduction)

If P(c) is true, then $(\exists x)$ P(x) is inferred.

If P(x) is true for all x in the domain of x, then $(\forall x)$ P(x) is inferred.

Example

eats(Ziggy, IceCream) $\Rightarrow$ $(\exists x)$ eats(Ziggy, x)

All instances of the given constant symbol are replaced by the new variable symbol

Note that the variable symbol cannot already exist anywhere in the expression

Existential Instantiation (a.k.a. existential Elimination)

From $(\exists x)\ P(x)$ infer $P(c)$

Example:

$(\exists x)$ eats(Ziggy, x) $\rightarrow$ eats(Ziggy, Stuff)

Note that the variable is replaced by a **brand-new constant** not occurring in this or any other sentence in the KB

Also known as skolemization; constant is a **skolem constant**

In other words, we don't want to accidentally draw other inferences about it by introducing the constant

Convenient to use this to reason about the unknown object, rather than constantly manipulating the existential quantifier

Translating English to FOL

Every gardener likes the sun.

$\forall x$ gardener(x) $\rightarrow$ likes(x,Sun)

You can fool some of the people all of the time.

$\exists x\ \forall t$ person(x) $\wedge$time(t) $\rightarrow$ can-fool(x,t)

You can fool all of the people some of the time.

$\forall x\ \exists t$ (person(x) $\rightarrow$ time(t) $\wedge$can-fool(x,t))

$\forall x$ (person(x) $\rightarrow \exists t$ (time(t) $\wedge$can-fool(x,t))

All purple mushrooms are poisonous.

$\forall x$ (mushroom(x) $\wedge$ purple(x)) $\rightarrow$ poisonous(x)

No purple mushroom is poisonous.

$\neg\exists x$ purple(x) $\wedge$ mushroom(x) $\wedge$ poisonous(x)

$\forall x$ (mushroom(x) $\wedge$ purple(x)) $\rightarrow \neg$poisonous(x)

There are exactly two purple mushrooms.

$\exists x\ \exists y$ mushroom(x) $\wedge$ purple(x) $\wedge$ mushroom(y) $\wedge$ purple(y) ^ $\neg$(x=y) $\wedge$ $\forall z$ (mushroom(z) $\wedge$ purple(z)) $\rightarrow$ ((x=z) $\vee$ (y=z))

Clinton is not tall.

$\neg$tall(Clinton)

X is above Y iff X is on directly on top of Y or there is a pile of one or more other objects directly on top of one another starting with X and ending with Y.

$\forall x\ \forall y$ above(x,y) $\leftrightarrow$ (on(x,y) $\vee$ $\exists z$ (on(x,z) $\wedge$ above(z,y)))

And Now, for Something Completely Different...

[She's a witch! Burn her!]

➔

?

Monty Python Logic

FIRST VILLAGER: We have found a witch. May we burn her?

ALL: A witch! Burn her!

BEDEVERE: Why do you think she is a witch?

SECOND VILLAGER: She turned *me* into a newt.

B: A newt?

V2 *(after looking at himself for some time)*: I got better.

ALL: Burn her anyway.

B: Quiet! Quiet! There are ways of telling whether she is a witch.

Monty Python cont.

B: Tell me… what do you do with witches?

ALL: Burn them!

B: And what do you burn, apart from witches?

V4: …wood?

B: So **why do witches burn**?

V2 *(pianissimo)*: **because they're made of wood?**

B: Good.

ALL: I see. Yes, of course.

Monty Python cont.

B: So how can we tell if she is made of wood?

V1: Make a bridge out of her.

B: Ah… but can you not also make bridges out of stone?

ALL: Yes, of course… um… er…

B: Does wood sink in water?

ALL: No, no, it floats. Throw her in the pond.

B: Wait. Wait… tell me, what also floats on water?

ALL: Bread? No, no no. Apples… gravy… very small rocks…

B: No, no, no,

Monty Python cont.

KING ARTHUR: A duck!

(They all turn and look at Arthur. Bedevere looks up, very impressed.)

B: Exactly. So… logically…

V1 *(beginning to pick up the thread)*: **If she… weighs the same as a duck… she's made of wood**.

B: And therefore?

ALL: **A witch!**

Monty Python Fallacy #1

$\forall x \; witch(x) \rightarrow burns(x)$

$\forall x \; wood(x) \rightarrow burns(x)$

$\therefore \; \forall x \; witch(x) \rightarrow wood(x)$

$p \rightarrow q$

$r \rightarrow q$

$p \rightarrow r$

Fallacy: Affirming the conclusion

Monty Python Near-Fallacy #2

wood(x) → can-build-bridge(x)

∴ can-build-bridge(x) → wood(x)

B: Ah… but can you not also make bridges out of stone?

Monty Python Fallacy #3

$\forall x$ wood(x) → floats(x)

$\forall x$ duck-weight (x) → floats(x)

$\therefore$ $\forall x$ duck-weight(x) → wood(x)

$p \rightarrow q$

$r \rightarrow q$

$\therefore r \rightarrow p$

Monty Python Fallacy #4

$\forall z\ light(z) \rightarrow wood(z)$
light(W)

$\therefore$ wood(W) ok…………..

witch(W) $\rightarrow$ wood(W) applying universal instan.
to fallacious conclusion #1

wood(W)

$\therefore$ witch(z)

Example: A Simple Genealogy KB in FOL

Build a small genealogy knowledge base using FOL that

- contains facts of immediate family relations (spouses, parents, etc.)
- contains definitions of more complex relations (ancestors, relatives)
- is able to answer queries about relationships between people

Predicates:

- parent(x, y), child(x, y), father(x, y), daughter(x, y), etc.
- spouse(x, y), husband(x, y), wife(x,y)
- ancestor(x, y), descendant(x, y)
- male(x), female(y)
- relative(x, y)

Facts:

- husband(Joe, Mary), son(Fred, Joe)
- spouse(John, Nancy), male(John), son(Mark, Nancy)
- father(Jack, Nancy), daughter(Linda, Jack)
- daughter(Liz, Linda)
- etc.

Rules for genealogical relations

$(\forall x,y)$ parent(x, y) $\leftrightarrow$ child (y, x)

$(\forall x,y)$ father(x, y) $\leftrightarrow$ parent(x, y) $\wedge$ male(x) (similarly for mother(x, y))

$(\forall x,y)$ daughter(x, y) $\leftrightarrow$ child(x, y) $\wedge$ female(x) (similarly for son(x, y))

$(\forall x,y)$ husband(x, y) $\leftrightarrow$ spouse(x, y) $\wedge$ male(x) (similarly for wife(x, y))

$(\forall x,y)$ spouse(x, y) $\leftrightarrow$ spouse(y, x) (**spouse relation is symmetric**)

$(\forall x,y)$ parent(x, y) $\rightarrow$ ancestor(x, y)

$(\forall x,y)(\exists z)$ parent(x, z) $\wedge$ ancestor(z, y) $\rightarrow$ ancestor(x, y)

$(\forall x,y)$ descendant(x, y) $\leftrightarrow$ ancestor(y, x)

$(\forall x,y)(\exists z)$ ancestor(z, x) $\wedge$ ancestor(z, y) $\rightarrow$ relative(x, y)

(related by common ancestry)

$(\forall x,y)$ spouse(x, y) $\rightarrow$ relative(x, y) (related by marriage)

$(\forall x,y)(\exists z)$ relative(z, x) $\wedge$ relative(z, y) $\rightarrow$ relative(x, y) (**transitive**)

$(\forall x,y)$ relative(x, y) $\leftrightarrow$ relative(y, x) (**symmetric**)

Queries

ancestor(Jack, Fred) /* the answer is yes */

relative(Liz, Joe) /* the answer is yes */

relative(Nancy, Matthew)

/* no answer in general, no if under closed world assumption */

$(\exists z)$ ancestor(z, Fred) $\wedge$ ancestor(z, Liz)

Expressing Uniqueness

Sometimes we want to say that there is a single, unique object that satisfies a certain condition

"There exists a unique x such that king(x) is true"

$\exists x\ king(x) \wedge \forall y\ (king(y) \rightarrow x=y)$

$\exists x\ king(x) \wedge \neg\exists y\ (king(y) \wedge x \neq y)$

$\exists!\ x\ king(x)$

"Every country has exactly one ruler"

$\forall c\ country(c) \rightarrow \exists!\ r\ ruler(c,r)$

Logical Agents

A Simple Reflex Agent

Rules to map percepts into observations:

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

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

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

Rules to select an action given observations:

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

Some difficulties:

- Consider Climb. There is 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 change in the world in logic can be tricky.

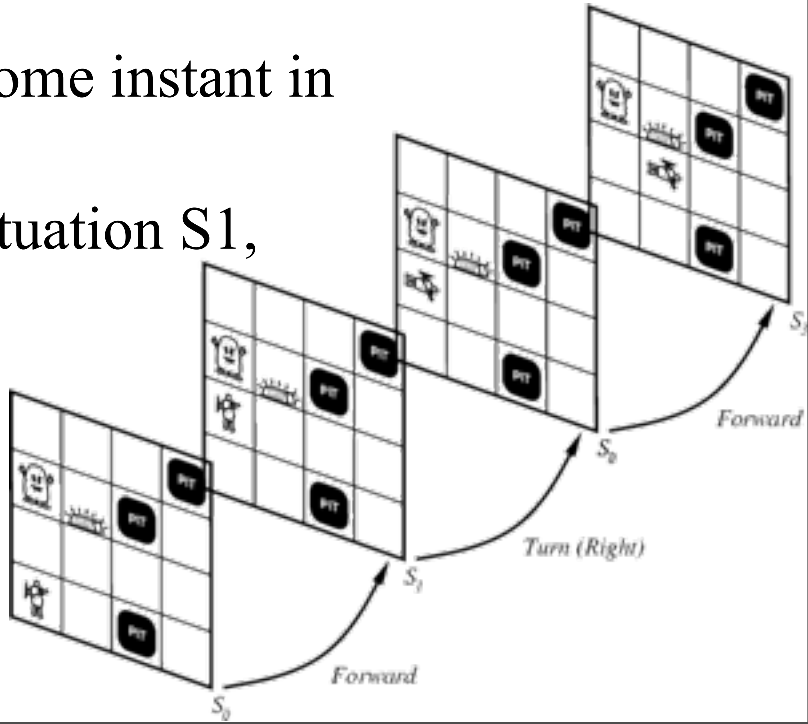
One way is just to change the KB

- Add and delete sentences from the KB to reflect changes
- How do we remember the past, or reason about changes?

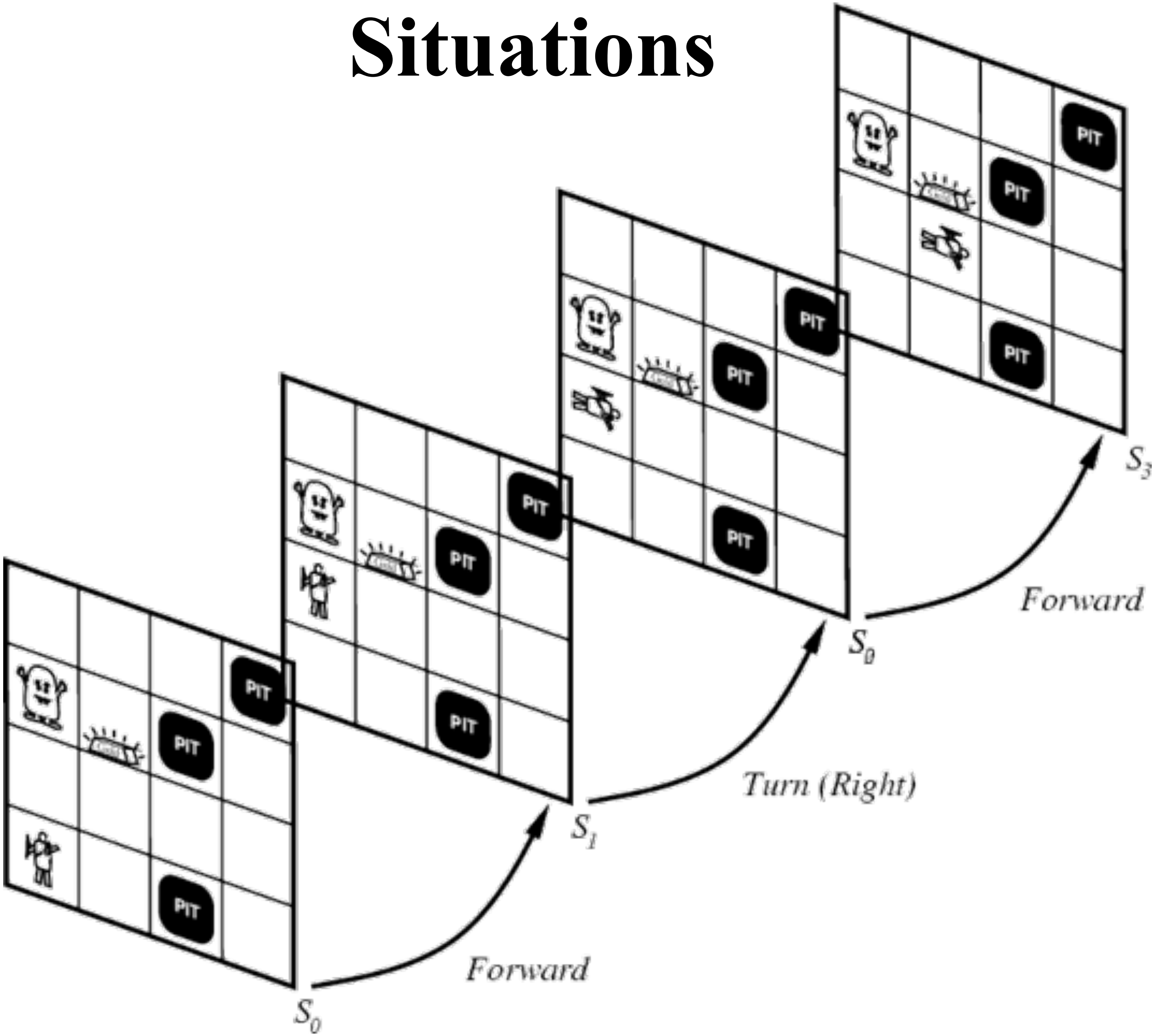
Situation calculus is another way

A **situation** is a snapshot of the world at some instant in time

When the agent performs an action A in situation S1, the result is a new situation S2.



Situations


PIT
PIT
PIT
PIT
PIT
PIT
PIT
PIT
PIT
PIT
PIT
PIT
Forward
Turn (Right)
Forward
S_0
S_1
S_0
S_3

Situation Calculus

A **situation** is a snapshot of the world at an interval of time during which nothing changes

Every true or false statement is made with respect to a particular situation.

- Add **situation variables** to every predicate.
- at(Agent,1,1) becomes at(Agent,1,1,s0): at(Agent,1,1) is true in situation (i.e., state) s0.
- Alernatively, add a special 2nd-order predicate, **holds(f,s),** that means "f is true in situation s." E.g., holds(at(Agent,1,1),s0)

Add a new function, **result(a,s),** that maps a situation s into a new situation as a result of performing action a. For example, result(forward, s) is a function that returns the successor state (situation) to s

Example: The action agent-walks-to-location-y could be represented by

$$(\forall x)(\forall y)(\forall s)\ (at(Agent,x,s) \land \neg onbox(s)) \rightarrow at(Agent,y,result(walk(y),s))$$

Qualification Problem

Qualification problem:

- How can you possibly characterize every single effect of an action, or every single 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

Similarly, it's just about 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 the aforementioned 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 very difficult
- Knowledge engineering (creating and maintaining knowledge bases for intelligent reasoning) is an entire field of investigation
- Many researchers 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 knowledge base that we have built so far is that it is difficult to decide which action is best among a number of possibilities.

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

The first step is to describe the desirability of actions independent of each other.

In doing this we will 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:

$(\forall a,s)\ Great(a,s) \rightarrow Action(a,s)$

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

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

...

Preferences Among Actions

We use this action quality scale in the following way.

Until it finds the gold, the basic strategy for our agent is:

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

Goal-Based Agents

Once the gold is found, it is necessary to change strategies. So now we need 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 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