# Midterm format

- Date: **10/09/2003 from 5:00pm – 6:30pm**
- Location: **THH 208**
- Credits: 35% of overall grade
- Approx. 4 problems, several questions in each.
- Material: everything so far.
- **Not** a multiple choice exam
- **No books** (or other material) are allowed.
- Duration will be 1:30 hours.
- Academic Integrity code: see class main page.

## Last time: Logic and Reasoning

- Knowledge Base (KB): contains a set of <u>sentences</u> expressed using a **knowledge representation language**
  - TELL: operator to add a sentence to the KB
  - ASK: to query the KB
- Logics are KRLs where conclusions can be drawn
  - Syntax
  - Semantics
- Entailment: KB |= a iff a is true in all worlds where KB is true
- Inference: KB $|-_i$ a = sentence a can be derived from KB using procedure $i$
  - Sound: whenever KB $|-_i$ a then KB |= a is true
  - Complete: whenever KB |= a then KB $|-_i$ a

## Last Time: Syntax of propositional logic

Propositional logic is the simplest logic—illustrates ba

The proposition symbols $P_1$, $P_2$ etc are sentences

If $S$ is a sentence, $\neg S$ is a sentence

If $S_1$ and $S_2$ is a sentence, $S_1 \wedge S_2$ is a sentence

If $S_1$ and $S_2$ is a sentence, $S_1 \vee S_2$ is a sentence

If $S_1$ and $S_2$ is a sentence, $S_1 \Rightarrow S_2$ is a sentence

If $S_1$ and $S_2$ is a sentence, $S_1 \Leftrightarrow S_2$ is a sentence

## Last Time: Semantics of Propositional logic

Each model specifies true/false for each proposition symbol

E.g. $A$ $B$ $C$

$True$ $True$ $False$

Rules for evaluating truth with respect to a model $m$:

| | | | | | |
|---|---|---|---|---|---|
| $\neg S$ | is true iff | $S$ | is false | | |
| $S_1 \wedge S_2$ | is true iff | $S_1$ | is true and | $S_2$ | is true |
| $S_1 \vee S_2$ | is true iff | $S_1$ | is true or | $S_2$ | is true |
| $S_1 \Rightarrow S_2$ | is true iff | $S_1$ | is false or | $S_2$ | is true |
| i.e., | is false iff | $S_1$ | is true and | $S_2$ | is false |
| $S_1 \Leftrightarrow S_2$ | is true iff | $S_1 \Rightarrow S_2$ | is true and | $S_2 \Rightarrow S_1$ | is true |

# Last Time: Inference rules for propositional logic

- **Modus Ponens** or **Implication-Elimination**: (From an implication and the premise of the implication, you can infer the conclusion.)

$$\frac{\alpha \Rightarrow \beta, \qquad \alpha}{\beta}$$

- **And-Elimination**: (From a conjunction, you can infer any of the conjuncts.)

$$\frac{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}{\alpha_i}$$

- **And-Introduction**: (From a list of sentences, you can infer their conjunction.)

$$\frac{\alpha_1, \; \alpha_2, \quad \ldots, \quad \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}$$

- **Or-Introduction**: (From a sentence, you can infer its disjunction with anything else at all.)

$$\frac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n}$$

- **Double-Negation Elimination**: (From a doubly negated sentence, you can infer a positive sentence.)

$$\frac{\neg\neg\alpha}{\alpha}$$

- **Unit Resolution**: (From a disjunction, if one of the disjuncts is false, then you can infer the other one is true.)

$$\frac{\alpha \vee \beta, \qquad \neg\beta}{\alpha}$$

- **Resolution**: (This is the most difficult. Because $\beta$ cannot be both true and false, one of the other disjuncts must be true in one of the premises. Or equivalently, implication is transitive.)

$$\frac{\alpha \vee \beta, \qquad \neg\beta \vee \gamma}{\alpha \vee \gamma} \qquad \text{or equivalently} \qquad \frac{\neg\alpha \Rightarrow \beta, \qquad \beta \Rightarrow \gamma}{\neg\alpha \Rightarrow \gamma}$$

## This time

- **First-order logic**
  - Syntax
  - Semantics
  - Wumpus world example

- **Ontology** (ont = ‘to be’;  logica = ‘word’): kinds of things one can talk about in the language

## Why first-order logic?

- We saw that propositional logic is limited because it only makes the ontological commitment that the world consists of **facts**.

- Difficult to represent even simple worlds like the Wumpus world;

  e.g.,

  “don’t go forward if the Wumpus is in front of you” takes 64 rules

# First-order logic (FOL)

- Ontological commitments:
  - **Objects**: wheel, door, body, engine, seat, car, passenger, driver
  - **Relations**: Inside(car, passenger), Beside(driver, passenger)
  - **Functions**: ColorOf(car)
  - **Properties**: Color(car), IsOpen(door), IsOn(engine)
- Functions are relations with single value for each object

## Semantics

there is a correspondence between

- functions, which return values
- predicates, which are true or false

Function: father_of(Mary) = Bill

Predicate: father_of(Mary, Bill)

## Examples:

- “One plus two equals three”

  Objects:

  Relations:

  Properties:

  Functions:

- “Squares neighboring the Wumpus are smelly”

  Objects:

  Relations:

  Properties:

  Functions:

# Examples:

- "One plus two equals three"

  Objects: one, two, three, one plus two

  Relations: equals

  Properties: --

  Functions: plus ("one plus two" is the name of the object obtained by applying function plus to one and two; three is another name for this object)

- "Squares neighboring the Wumpus are smelly"

  Objects: Wumpus, square

  Relations: neighboring

  Properties: smelly

  Functions: --

## FOL: Syntax of basic elements

- **Constant symbols:** 1, 5, A, B, USC, JPL, Alex, Manos, ...
- **Predicate symbols:** >, Friend, Student, Colleague, ...
- **Function symbols:** +, sqrt, SchoolOf, TeacherOf, ClassOf, ...
- **Variables:** *x, y, z, next, first, last, ...*
- **Connectives:** $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$
- **Quantifiers:** $\forall$, $\exists$
- **Equality:** =

## Syntax of Predicate Logic

- Symbol set
  - **constants**
  - **Boolean connectives**
  - variables
  - functions
  - predicates (relations)
  - quantifiers

## Syntax of Predicate Logic

- Terms: a reference to an object
  - variables,
  - constants,
  - functional expressions (can be arguments to predicates)

- Examples:
  - first([a,b,c]), sq_root(9), sq_root(n), tail([a,b,c])

## Syntax of Predicate Logic

- Sentences: make claims about objects
  - (Well-formed formulas, (wffs))
- Atomic Sentences (predicate expressions):
  - loves(John,Mary), brother_of(John,Ted)
- Complex Sentences (Atomic Sentences connected by booleans):
  - loves(John,Mary)
  - brother_of(John,Ted)
  - teases(Ted, John)

# Examples of Terms: Constants, Variables and Functions

- Constants: object constants refer to individuals
  - Alan, Sam, R225, R216
- Variables
  - PersonX, PersonY, RoomS, RoomT
- Functions
  - father_of(PersonX)
  - product_of(Number1,Number2)

## Examples of Predicates and Quantifiers

- Predicates
  - in(Alan,R225)
  - partOf(R225,Pender)
  - fatherOf(PersonX,PersonY)

- Quantifiers
  - All dogs are mammals.
  - Some birds can’t fly.
  - 3 birds can’t fly.

## Semantics

- Referring to individuals
  - Jackie
  - son-of(Jackie), Sam

- Referring to states of the world
  - person(Jackie), female(Jackie)
  - mother(Sam, Jackie)

## FOL: Atomic sentences

AtomicSentence $\rightarrow$ Predicate(Term, ...) | Term = Term

Term $\rightarrow$ Function(Term, ...) | Constant | Variable

- Examples:
  - SchoolOf(Manos)
  - Colleague(TeacherOf(Alex), TeacherOf(Manos))
  - >((+ x y), x)

## FOL: Complex sentences

Sentence $\rightarrow$ AtomicSentence
| Sentence Connective Sentence
| Quantifier Variable, ... Sentence
| $\neg$ Sentence
| (Sentence)

- Examples:
  - $S1 \wedge S2$, $S1 \vee S2$, $(S1 \wedge S2) \vee S3$, $S1 \Rightarrow S2$, $S1 \Leftrightarrow S3$
  - Colleague(Paolo, Maja) $\Rightarrow$ Colleague(Maja, Paolo)
    Student(Alex, Paolo) $\Rightarrow$ Teacher(Paolo, Alex)

## Semantics of atomic sentences

- Sentences in FOL are interpreted with respect to a model
- Model contains objects and relations among them
- Terms: refer to objects (e.g., Door, Alex, StudentOf(Paolo))
  - Constant symbols: refer to objects
  - Predicate symbols: refer to relations
  - Function symbols: refer to functional Relations

- An atomic sentence $predicate(term_1, \ldots, term_n)$ is **true** iff the relation referred to by *predicate* holds between the objects referred to by $term_1, \ldots, term_n$

# Example model

- Objects: John, James, Marry, Alex, Dan, Joe, Anne, Rich

- Relation: sets of tuples of objects
  {<John, James>, <Marry, Alex>, <Marry, James>, ...}
  {<Dan, Joe>, <Anne, Marry>, <Marry, Joe>, ...}

- E.g.:
  Parent relation -- {<John, James>, <Marry, Alex>, <Marry, James>}

  then Parent(John, James) is true
  Parent(John, Marry) is false

## Quantifiers

- Expressing sentences about **collections** of objects without enumeration (naming individuals)

- E.g., All Trojans are clever

  Someone in the class is sleeping

- Universal quantification (for all): $\forall$

- Existential quantification (three exists): $\exists$

## Universal quantification (for all): $\forall$

$\forall$ *<variables> <sentence>*

- *"Every one in the cs561 class is smart"*:
  $\forall\ x \quad In(cs561, x) \Rightarrow Smart(x)$

- **$\forall$ P corresponds to the conjunction of instantiations of P**
  $In(cs561, Manos) \Rightarrow Smart(Manos) \wedge$
  $In(cs561, Dan) \Rightarrow Smart(Dan) \wedge$
  ...
  $In(cs561, Clinton) \Rightarrow Smart(Clinton)$

## Universal quantification (for all): $\forall$

- $\Rightarrow$ is a natural connective to use with $\forall$
- Common mistake: to use $\wedge$ in conjunction with $\forall$
  e.g: $\forall\, x \quad \text{In}(cs561, x) \wedge \text{Smart}(x)$
  means *“every one is in cs561 and everyone is smart”*

## Existential quantification (there exists): $\exists$

$\exists$ *<variables> <sentence>*

- *"Someone in the cs561 class is smart"*:
  $\exists\, x \quad In(cs561, x) \wedge Smart(x)$

- **$\exists$ P corresponds to the disjunction of instantiations of P**
  $In(cs561, Manos) \wedge Smart(Manos) \vee$
  $In(cs561, Dan) \wedge Smart(Dan) \vee$
  ...
  $In(cs561, Clinton) \wedge Smart(Clinton)$

## Existential quantification (there exists): $\exists$

- $\wedge$ is a natural connective to use with $\exists$
- Common mistake: to use $\Rightarrow$ in conjunction with $\exists$
  e.g: $\exists\, x \quad In(cs561, x) \Rightarrow Smart(x)$
  is true if there is anyone that is not in cs561!
  (remember, false $\Rightarrow$ true is valid).

## Properties of quantifiers

$\forall x \ \forall y$ is the same as $\forall y \ \forall x$ (why??)

$\exists x \ \exists y$ is the same as $\exists y \ \exists x$ (why??)

$\exists x \ \forall y$ is not the same as $\forall y \ \exists x$

$\exists x \ \forall y \ \ Loves(x, y)$
"There is a person who loves everyone in the world"

$\forall y \ \exists x \ \ Loves(x, y)$
"Everyone in the world is loved by at least one person"

Not all by one person but each one at least by one

Quantifier duality: each can be expressed using the other

$\forall x \ \ Likes(x, IceCream) \qquad \neg\exists x \ \neg Likes(x, IceCream)$ Proof?

$\exists x \ \ Likes(x, Broccoli) \qquad \neg\forall x \ \neg Likes(x, Broccoli)$

# Proof

- In general we want to prove:

$\forall$ x  P(x) <=> $\neg$ $\exists$ x $\neg$ P(x)

- ❑ $\forall$ x P(x) = $\neg$($\neg$($\forall$ x P(x))) = $\neg$($\neg$(P(x1) ^ P(x2) ^ ... ^ P(xn)) ) = $\neg$($\neg$P(x1) v $\neg$P(x2) v ... v $\neg$P(xn)) )
- ❑ $\exists$ x $\neg$P(x) = $\neg$P(x1) v $\neg$P(x2) v ... v $\neg$P(xn)
- ❑ **$\neg$**$\exists$ x $\neg$P(x) = $\neg$($\neg$P(x1) v $\neg$P(x2) v ... v $\neg$P(xn))

# Example sentences

- Brothers are siblings
  - .
- Sibling is transitive
  - .
- One’s mother is one’s sibling’s mother
  - .
- A first cousin is a child of a parent’s sibling
  - .

## Example sentences

- Brothers are siblings

  $\forall$ x, y   Brother(x, y) $\Rightarrow$ Sibling(x, y)

- Sibling is transitive

  $\forall$ x, y, z   Sibling(x, y) $\wedge$ Sibling(y, z) $\Rightarrow$ Sibling(x, z)

- One's mother is one's sibling's mother

  $\forall$ m, c    Mother(m, c) $\wedge$ Sibling(c, d) $\Rightarrow$ Mother(m, d)

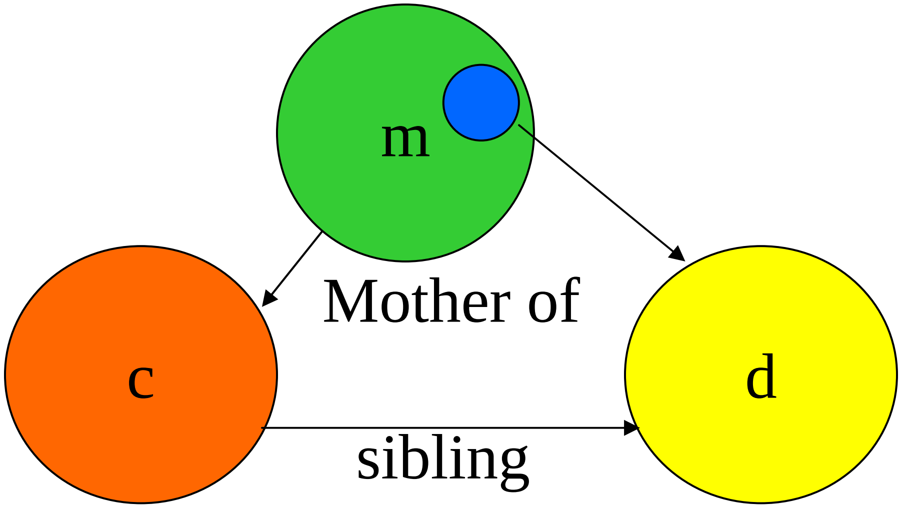
- A first cousin is a child of a parent's sibling

  $\forall$ c, d   FirstCousin(c, d) $\Leftrightarrow$
  $\exists$ p, ps  Parent(p, d) $\wedge$ Sibling(p, ps) $\wedge$ Parent(ps, c)

## Example sentences

- One's mother is one's sibling's mother
  $\forall$ m, c,d   Mother(m, c) $\wedge$ Sibling(c, d) $\Rightarrow$ Mother(m, d)

- $\forall$ c,d $\exists$**m** Mother(m, c) $\wedge$ Sibling(c, d) $\Rightarrow$ Mother(m, d)



## Translating English to FOL

- Every gardener likes the sun.
  $\forall$ **x gardener(x) => likes(x,Sun)**

- You can fool some of the people all of the time.
  $\exists$ **x** $\forall$ **t (person(x) ^ time(t)) => can-fool(x,t)**

## Translating English to FOL

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

  $\forall$ **x** $\exists$ **t (person(x) ^ time(t) => can-fool(x,t)**

- All purple mushrooms are poisonous.

  $\forall$ **x (mushroom(x) ^ purple(x)) => poisonous(x)**

## Translating English to FOL…

- No purple mushroom is poisonous.

**¬(∃ x) purple(x) ^ mushroom(x) ^ poisonous(x)**

or, equivalently,

**(∀ x) (mushroom(x) ^ purple(x)) =>
¬poisonous(x)**

## Translating English to FOL…

- There are exactly two purple mushrooms.

**(∃ x)(∃ y) mushroom(x) ^ purple(x) ^ mushroom(y) ^ purple(y) ^ ¬(x=y) ^ (∀ z) (mushroom(z) ^ purple(z)) => ((x=z) v (y=z))**

- Deb is not tall.

**¬tall(Deb)**

## Translating English to FOL…

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

**(∀ x)(∀ y) above(x,y) <=> (on(x,y) v (∃ z) (on(x,z) ^ above(z,y)))**

# Equality

$term_1 = term_2$ is true under a given interpretation
if and only if $term_1$ and $term_2$ refer to the same object

E.g., $1 = 2$ and $\forall\, x \;\; \times(Sqrt(x), Sqrt(x)) = x$ are satisfiable
$2 = 2$ is valid

E.g., definition of (full) $Sibling$ in terms of $Parent$:
$$\forall\, x, y \;\; Sibling(x, y) \;\Leftrightarrow\; [\neg(x = y) \wedge \exists\, m, f \;\; \neg(m = f) \wedge Parent(m, x) \wedge Parent(f, x) \wedge Parent(m, y) \wedge Parent(f, y)]$$

## Higher-order logic?

- First-order logic allows us to quantify over objects (= the first-order entities that exist in the world).

- Higher-order logic also allows quantification over relations and functions.

  e.g., "two objects are equal iff all properties applied to them are equivalent":

  $$\forall\, x,y \quad (x=y) \Leftrightarrow (\forall\, p,\, p(x) \Leftrightarrow p(y))$$

- Higher-order logics are more expressive than first-order; however, so far we have little understanding on how to effectively reason with sentences in higher-order logic.

# Logical agents for the Wumpus world

## Remember: generic 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
```

1. TELL KB what was perceived
   Uses a KRL to insert new sentences, representations of facts, into KB

2. ASK KB what to do.
   Uses logical reasoning to examine actions and select best.

# Using the FOL Knowledge Base

Suppose a wumpus-world agent is using an FOL KB
and perceives a smell and a breeze (but no glitter) at $t = 5$:

$\text{TELL}(KB, Percept([Smell, Breeze, None], 5))$
$\text{ASK}(KB, \exists a \;\; Action(a, 5))$

I.e., does the KB entail any particular actions at $t = 5$?

Answer: $Yes$, $\{a/Shoot\}$ $\leftarrow$ <u>substitution</u> (binding list)

Set of solutions

Given a sentence $S$ and a substitution $\sigma$,
$S\sigma$ denotes the result of plugging $\sigma$ into $S$; e.g.,
$S = Smarter(x, y)$
$\sigma = \{x/Hillary, y/Bill\}$
$S\sigma = Smarter(Hillary, Bill)$

$\text{ASK}(\text{KB}, \text{S})$ returns some/all $\sigma$ such that $KB \models S\sigma$

# Wumpus world, FOL Knowledge Base

"Perception"

$\forall\, b, g, t \;\; Percept([Smell, b, g], t) \;\Rightarrow\; Smelt(t)$

$\forall\, s, b, t \;\; Percept([s, b, Glitter], t) \;\Rightarrow\; AtGold(t)$

Reflex: $\forall\, t \;\; AtGold(t) \;\Rightarrow\; Action(Grab, t)$

Reflex with internal state: do we have the gold already?

$\forall\, t \;\; AtGold(t) \wedge \neg Holding(Gold, t) \;\Rightarrow\; Action(Grab, t)$

$Holding(Gold, t)$ cannot be observed

$\Rightarrow$ keeping track of change is essential

## Deducing hidden properties

Properties of locations:

$\forall\, l,t \;\; At(Agent,l,t) \wedge Smelt(t) \;\Rightarrow\; Smelly(l)$

$\forall\, l,t \;\; At(Agent,l,t) \wedge Breeze(t) \;\Rightarrow\; Breezy(l)$

Squares are breezy near a pit:

Diagnostic rule—infer cause from effect

$$\forall\, y \;\; Breezy(y) \;\Rightarrow\; \exists\, x \;\; Pit(x) \wedge Adjacent(x,y)$$

Causal rule—infer effect from cause

$$\forall\, x,y \;\; Pit(x) \wedge Adjacent(x,y) \;\Rightarrow\; Breezy(y)$$

Neither of these is complete—e.g., the causal rule doesn't say whether squares far away from pits can be breezy

Definition for the $Breezy$ predicate:

$$\forall\, y \;\; Breezy(y) \;\Leftrightarrow\; [\exists\, x \;\; Pit(x) \wedge Adjacent(x,y)]$$

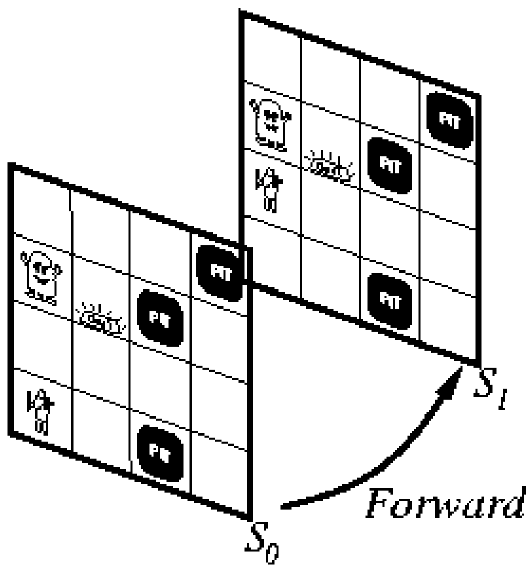
# Situation calculus

Facts hold in situations, rather than eternally
E.g., $Holding(Gold, Now)$ rather than just $Holding(Gold)$

Situation calculus is one way to represent change in FOL:

Adds a situation argument to each non-eternal predicate
E.g., $Now$ in $Holding(Gold, Now)$ denotes a situation

Situations are connected by the $Result$ function
$Result(a, s)$ is the situation that results from doing $a$ is $s$



## Describing actions

"Effect" axiom—describe changes due to action
$\forall s \;\; AtGold(s) \;\Rightarrow\; Holding(Gold, Result(Grab, s))$

"Frame" axiom—describe non-changes due to action
$\forall s \;\; HaveArrow(s) \;\Rightarrow\; HaveArrow(Result(Grab, s))$

May result in too many frame axioms

Frame problem: find an elegant way to handle non-change

(a) representation—avoid frame axioms

(b) inference—avoid repeated "copy-overs" to keep track of state

Qualification problem: true descriptions of real actions require endless caveats—what if gold is slippery or nailed down or . . .

Ramification problem: real actions have many secondary consequences—what about the dust on the gold, wear and tear on gloves, . . .

## Describing actions (cont'd)

Successor-state axioms solve the representational frame problem

Each axiom is "about" a predicate (not an action per se):

P true afterwards $\Leftrightarrow$ [an action made P true

$\vee$ P true already and no action made P false]

For holding the gold:

$$\forall a, s \;\; Holding(Gold, Result(a, s)) \;\Leftrightarrow$$
$$[(a = Grab \wedge AtGold(s))$$
$$\vee\; (Holding(Gold, s) \wedge a \neq Release)]$$

## Planning

Initial condition in KB:

$At(Agent, [1,1], S_0)$
$At(Gold, [1,2], S_0)$

Query: $\text{ASK}(KB, \exists s \;\; Holding(Gold, s))$
i.e., in what situation will I be holding the gold?

Answer: $\{s/Result(Grab, Result(Forward, S_0))\}$
i.e., go forward and then grab the gold

This assumes that the agent is interested in plans starting at $S_0$ and that $S_0$ is the only situation described in the KB

## Generating action sequences

Represent plans as action sequences $[a_1, a_2, \ldots, a_n]$

$PlanResult(p, s)$ is the result of executing $p$ in $s$

Then the query $\text{Ask}(KB, \exists p \;\; Holding(Gold, PlanResult(p, S_0)))$ has the solution $\{p/[Forward, Grab]\}$

Definition of $PlanResult$ in terms of $Result$:

$\forall s \;\; PlanResult([], s) = s$ [ ] = empty plan

$\forall a, p, s \;\; PlanResult([a|p], s) = PlanResult(p, Result(a, s))$

Recursively continue until it gets to empty plan [ ]

Planning systems are special-purpose reasoners designed to do this type of inference more efficiently than a general-purpose reasoner

## Summary

First-order logic:

- objects and relations are semantic primitives
- syntax: constants, functions, predicates, equality, quantifiers

Increased expressive power: sufficient to define wumpus world

Situation calculus:

- conventions for describing actions and change in FOL
- can formulate planning as inference on a situation calculus KB