

CS 561: Artificial Intelligence

Instructor: Sofus A. Macskassy, macskass@usc.edu

TAs: Nadeesha Ranashinghe (nadeeshr@usc.edu)

William Yeoh (wyeoh@usc.edu)

Harris Chiu (chiciu@usc.edu)

Lectures: MW 5:00-6:20pm, OHE 122 / DEN

Office hours: By appointment

Class page: <http://www-rcf.usc.edu/~macskass/CS561-Spring2010/>

This class will use <http://www.uscden.net/> and class webpage

- Up to date information
- Lecture notes
- Relevant dates, links, etc.

Course material:

[AIMA] Artificial Intelligence: A Modern Approach,
by Stuart Russell and Peter Norvig. (2nd ed)

Logistics – HW2

- Homework #2 was due today
- You should have submitted it before class on turnitin

Logistics - MIDTERM

- Midterm 1 is next week
- Date: March 1
- Location: SGM 124
- Time: 5pm – 6:20pm
DEN Students: should have received an email to set up their exam
- Covers: All lectures through this week (Ch. 1-8)
 - It is open book and open notes
 - You can use the book, lecture slides and your notes
- Example midterms available from AIMA site
 - <http://aima.cs.berkeley.edu/instructors.html>

First-order Logic [AIMA Ch. 8]

- Why FOL?
- Syntax and semantics of FOL
- Fun with sentences
- Wumpus world in FOL

Review: Propositional logic - syntax

- Propositional logic is the simplest logic—illustrates basic ideas
- The proposition symbols P_1, P_2 etc are sentences
- If S is a sentence, $\neg S$ is a sentence (negation)
- If S_1 and S_2 are sentences, $S_1 \wedge S_2$ is a sentence (conjunction)
- If S_1 and S_2 are sentences, $S_1 \vee S_2$ is a sentence (disjunction)
- If S_1 and S_2 are sentences, $S_1 \Rightarrow S_2$ is a sentence (implication)
- If S_1 and S_2 are sentences, $S_1 \Leftrightarrow S_2$ is a sentence (biconditional)

Review: Propositional logic - Semantics

Each model specifies true/false for each proposition symbol

E.g. $P_{1,2}$ $P_{2,2}$ $P_{3,1}$
true true false

(With these symbols, 8 possible models, can be enumerated automatically.)

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

Simple recursive process evaluates an arbitrary sentence, e.g.,

$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = \textit{true} \wedge (\textit{false} \vee \textit{true}) = \textit{true} \wedge \textit{true} = \textit{true}$

Review propositional logic [cont'd]

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

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

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

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

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

$$\frac{\alpha_1, \alpha_2, \dots, \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \dots \wedge \alpha_n}$$

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

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

- ◇ **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.)

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

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

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

Why first-order logic?

Pros and cons of propositional logic

- 😊 Propositional logic is **declarative**: pieces of syntax correspond to facts
- 😊 Propositional logic allows partial/disjunctive/negated information (unlike most data structures and databases)
- 😊 Propositional logic is **compositional**:
meaning of $B_{1,1} \wedge P_{1,2}$ is derived from meaning of $B_{1,1}$ and of $P_{1,2}$
- 😊 Meaning in propositional logic is **context-independent** (unlike natural language, where meaning depends on context)
- 😞 Propositional logic has very limited expressive power (unlike natural language)
E.g., cannot say “pits cause breezes in adjacent squares” except by writing one sentence for each square

First-order logic (FOL)

- Whereas propositional logic assumes world contains facts, first-order logic (like natural language) assumes the world contains:
 - **Objects:** wheel, door, body, engine, seat, car, passenger, driver, people, houses, numbers, theories, Ronald McDonald, colors, baseball games, wars, centuries, ...
 - **Relations:** Inside(car, passenger), Beside(driver, passenger), BrotherOf(person, person), BiggerThan(object, object), Inside(), part of(), HasColor(), OccurredAfter(), Owns(), ComesBetween(), ...
 - **Functions:** ColorOf(car), FatherOf(person), BestFriend(person), ThirdInningOf(), OneMoreThan(), EndOf(), ...
 - **Properties:** Color(car), IsOpen(door), IsOn(engine)
- Functions are relations with single value for each object

Logics in general

- Logics are characterized by what they commit to as “primitives”
- Ontological commitment: what exists—facts? objects? time? beliefs?
- Epistemological commitment: what states of knowledge?

Language	Ontological Commitment	Epistemological Commitment
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability logic	facts	degree of belief 0...1
Fuzzy logic	facts, degree of truth	known interval value

Semantics/Interpretation

there is a correspondence between

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

Function: $\text{father_of}(\text{Mary}) = \text{Bill}$

Predicate: $\text{father_of}(\text{Mary}, \text{Bill})$

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:** $x, y, z, next, first, last, \dots$
- **Connectives:** $\wedge, \vee, \Rightarrow, \Leftrightarrow$
- **Quantifiers:** \forall, \exists
- **Equality:** =

FOL: Atomic sentences

AtomicSentence = *predicate*($term_1, \dots, term_n$)
or $term_1 = term_2$

Term = *function*($term_1, \dots, term_n$)
or *constant* or *variable*

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

FOL: Complex sentences

Complex sentences are made from atomic sentences using connectives

$$\neg S, S_1 \wedge S_2, S_1 \vee S_2, S_1 \Rightarrow S_2, S_1 \Leftrightarrow S_2$$

Sentence \rightarrow *AtomicSentence*

| *Sentence* *Connective* *Sentence*
| *Quantifier* *Variable*, ... *Sentence*
| \neg *Sentence*
| (*Sentence*)

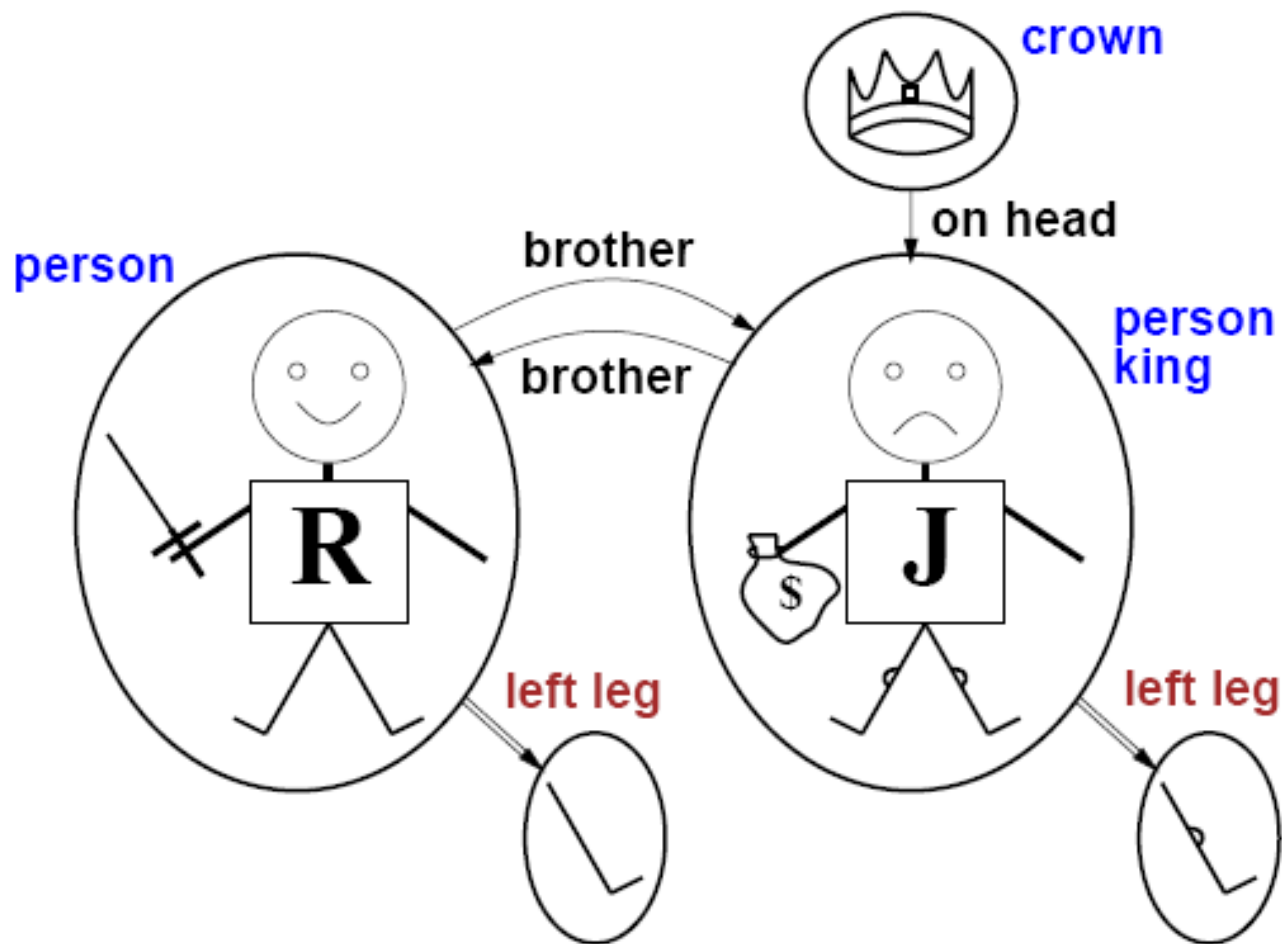
- Examples:

- *Colleague(Paolo, Maja) \Rightarrow Colleague(Maja, Paolo)*
Student(Alex, Paolo) \Rightarrow Teacher(Paolo, Alex)

Truth in first-order logic

- Sentences in FOL are interpreted with respect to a **model** and an **interpretation**
- Model contains ≥ 1 objects (domain elements) and relations among them
- Interpretation specifies referents for
 - Constant symbols: refer to objects
 - Predicate symbols: refer to relations
 - Function symbols: refer to functional Relations
- An atomic sentence $\textit{predicate}(\textit{term}_1, \dots, \textit{term}_n)$ is **true** iff the objects referred to by $\textit{term}_1, \dots, \textit{term}_n$ are in the relation referred to by $\textit{predicate}$

Models for FOL: Example



Truth example

- Consider the interpretation in which
 - *Richard* → Richard the Lionheart
 - *John* → the evil King John
 - *Brother* → the brother relation
-
- Under this interpretation, *Brother*(*Richard*, *John*) is true just in case Richard the Lionheart and the evil King John are in the brother relation in the model

Models for FOL: Lots!

- Entailment in propositional logic can be computed by enumerating models
- We can enumerate the FOL models for a given KB vocabulary:
- For each number of domain elements n from 1 to ∞
 - For each k -ary predicate P_k in the vocabulary
 - For each possible k -ary relation on n objects
 - For each constant symbol C in the vocabulary
 - For each choice of referent for C from n objects ...

Computing entailment by enumerating FOL models is not easy!

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 (there exists): \exists

Universal quantification (for all): \forall

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

- "Every one in the cs561 class is smart":

$$\forall x \quad \text{In}(cs561, x) \Rightarrow \text{Smart}(x)$$

- \forall P corresponds to the conjunction of instantiations of P

$$\text{In}(cs561, \text{Manos}) \Rightarrow \text{Smart}(\text{Manos}) \wedge$$

$$\text{In}(cs561, \text{Dan}) \Rightarrow \text{Smart}(\text{Dan}) \wedge$$

...

$$\text{In}(cs561, \text{Bush}) \Rightarrow \text{Smart}(\text{Bush})$$

Universal quantification (for all): \forall

\Rightarrow is a natural connective to use with \forall

Common mistake: to use \wedge in conjunction with \forall

$$\forall x \text{ In}(cs561, x) \wedge \text{Smart}(x)$$

means "every one is in cs561 and everyone is smart"

Existential quantification (there exists): \exists

\exists $\langle variables \rangle$ $\langle sentence \rangle$

- "Someone in the cs561 class is smart":

$$\exists x \text{ In}(cs561, x) \wedge \text{Smart}(x)$$

- $\exists x P$ is true in a model m iff P is true with x being **some** possible object in the model

- **$\exists P$ corresponds to the disjunction of instantiations of P**

$$(\text{In}(cs561, \text{Manos}) \wedge \text{Smart}(\text{Manos})) \vee$$

$$(\text{In}(cs561, \text{Dan}) \wedge \text{Smart}(\text{Dan})) \vee$$

...

$$(\text{In}(cs561, \text{Bush}) \wedge \text{Smart}(\text{Bush}))$$

Existential quantification (there exists): \exists

- \wedge is a natural connective to use with \exists
- **Common mistake:** to use \Rightarrow in conjunction with \exists

$$\exists x \quad In(cs561, x) \Rightarrow Smart(x)$$

- is true if there is anyone that is not in cs561!
(remember, false \Rightarrow true is valid).

Examples...

$$\forall x \text{ In}(cs561, x) \Rightarrow \text{Smart}(x)$$

$$\forall x \text{ In}(cs561, x) \wedge \text{Smart}(x)$$

$$\exists x \text{ In}(cs561, x) \wedge \text{Smart}(x)$$

$$\exists x \text{ In}(cs561, x) \Rightarrow \text{Smart}(x)$$

$$\text{In}(cs561, x) \wedge \text{Smart}(x)$$

$$\neg \text{In}(cs561, x) \vee \text{Smart}(x)$$



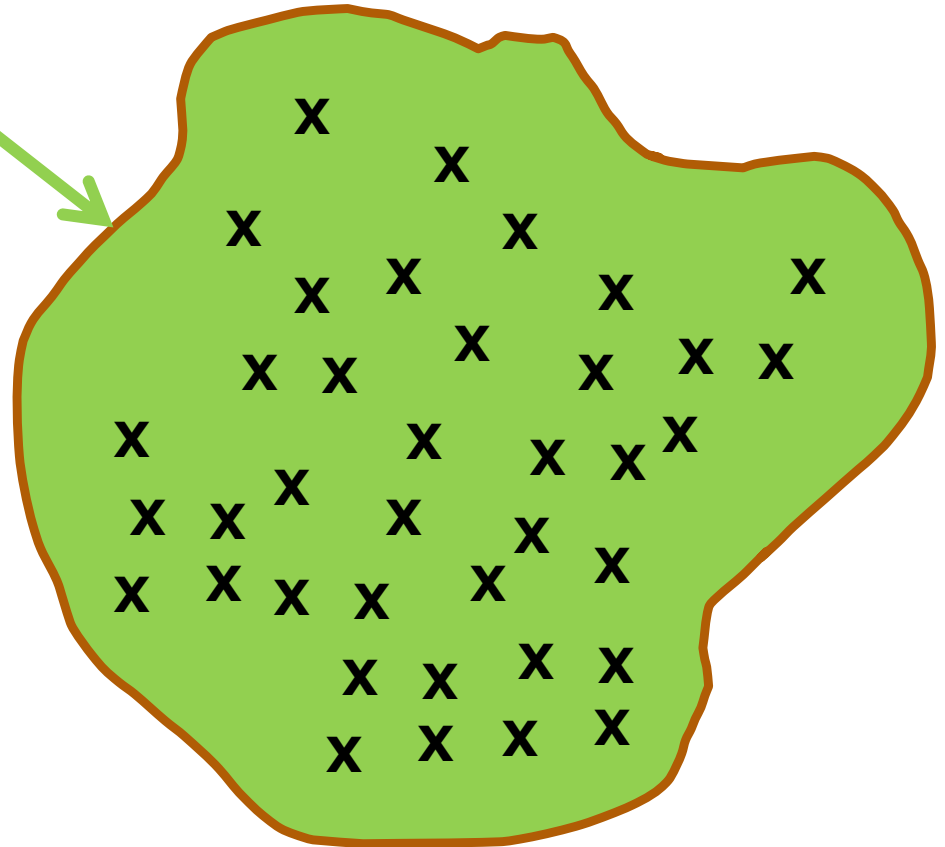
Examples...

$$\forall x \text{ In}(cs561, x) \Rightarrow \text{Smart}(x)$$

$$\forall x \text{ In}(cs561, x) \wedge \text{Smart}(x)$$

$$\exists x \text{ In}(cs561, x) \wedge \text{Smart}(x)$$

$$\exists x \text{ In}(cs561, x) \Rightarrow \text{Smart}(x)$$



Examples...

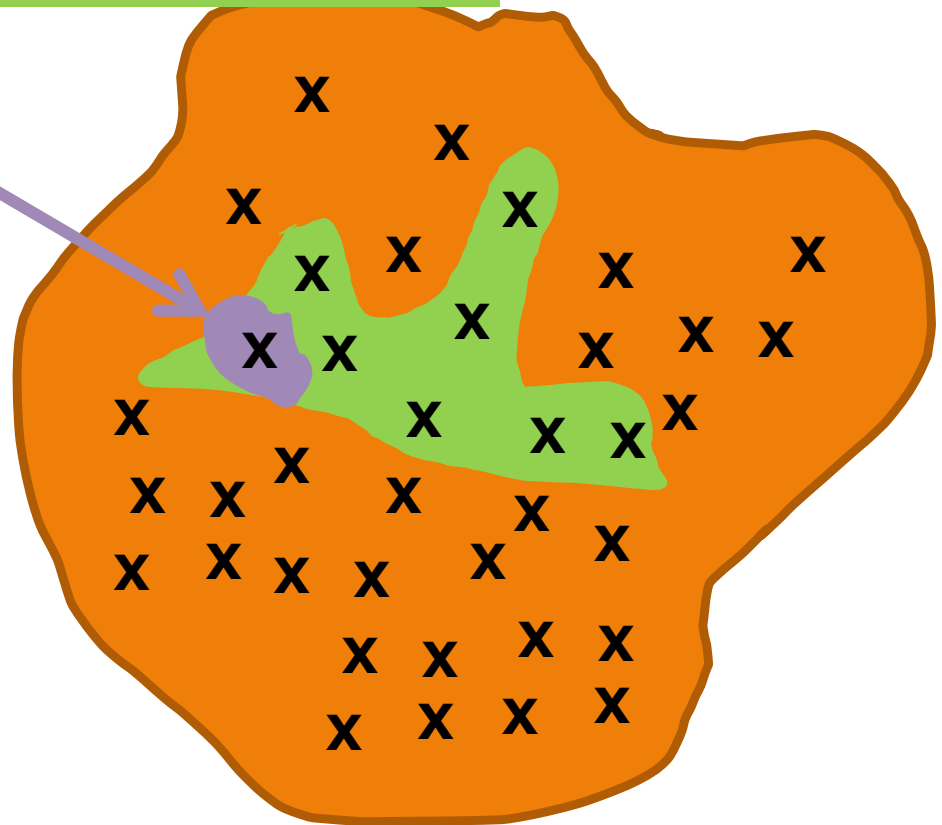
$$\forall x \text{ In}(cs561, x) \Rightarrow \text{Smart}(x)$$

$$\forall x (\text{In}(cs561, x) \wedge \text{Smart}(x))$$

$$\exists x \text{ In}(cs561, x) \wedge \text{Smart}(x)$$

$$\exists x \text{ In}(cs561, x) \Rightarrow \text{Smart}(x)$$

$$\text{In}(cs561, x) \wedge \text{Smart}(x)$$



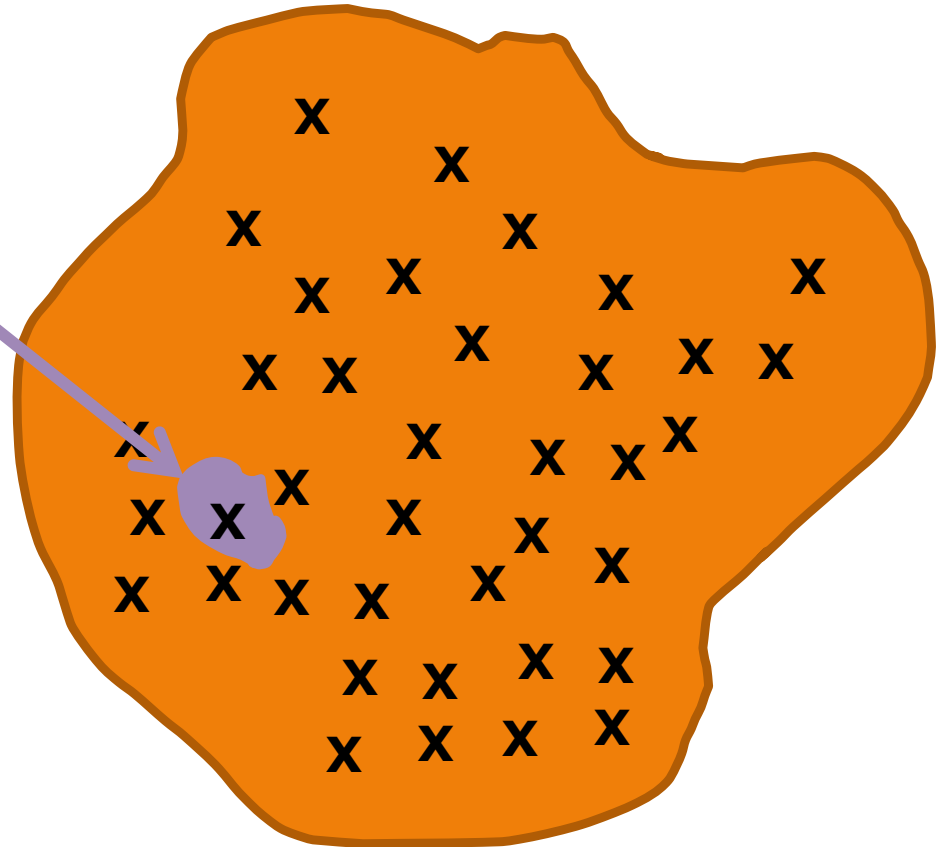
Examples...

$$\forall x \text{ In}(cs561, x) \Rightarrow \text{Smart}(x)$$

$$\forall x \text{ In}(cs561, x) \wedge \text{Smart}(x)$$

$$\exists x \text{ In}(cs561, x) \wedge \text{Smart}(x)$$

$$\exists x \text{ In}(cs561, x) \Rightarrow \text{Smart}(x)$$



$$\neg \text{In}(cs561, x)$$

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 $\forall y \exists x$

$\exists x \forall y \text{ Loves}(x, y)$

“There is a person who loves everyone in the world”

$\forall y \exists x \text{ Loves}(x, y)$

“Everyone in the world is loved by at least one person”

Quantifier duality: each can be expressed using the other

$\forall x \text{ Likes}(x, \text{IceCream}) \quad \neg \exists x \neg \text{Likes}(x, \text{IceCream})$

$\exists x \text{ Likes}(x, \text{Broccoli}) \quad \neg \forall x \neg \text{Likes}(x, \text{Broccoli})$

Fun with sentences

- Brothers are siblings

$$\forall x,y \text{ Brother}(x,y) \Rightarrow \text{Sibling}(x,y)$$

- "Sibling" is symmetric

$$\forall x,y \text{ Sibling}(x,y) \Rightarrow \text{Sibling}(y,x)$$

- One's mother is one's female parent

$$\forall x,y \text{ Mother}(x,y) \Leftrightarrow (\text{Female}(x) \wedge \text{Parent}(x,y))$$

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

$$\forall x,y \text{ FirstCousin}(x,y) \Leftrightarrow \exists p,ps \text{ Parent}(p,x) \wedge \text{Sibling}(ps,p) \wedge \text{Parent}(ps,y)$$

Translating English to FOL

- Every gardener likes the sun.
 $\forall x \text{ gardener}(x) \Rightarrow \text{likes}(x, \text{Sun})$
- You can fool some of the people all of the time.
 $\exists x \forall t (\text{person}(x) \wedge \text{time}(t)) \Rightarrow \text{can-fool}(x, t)$

Translating English to FOL

- You can fool all of the people some of the time.
 $\forall x \exists t (\text{person}(x) \wedge \text{time}(t)) \Rightarrow \text{can-fool}(x, t)$
- All purple mushrooms are poisonous.
 $\forall x (\text{mushroom}(x) \wedge \text{purple}(x)) \Rightarrow \text{poisonous}(x)$

Translating English to FOL...

- No purple mushroom is poisonous.

$\neg (\exists \mathbf{x}) \text{ purple}(\mathbf{x}) \wedge \text{mushroom}(\mathbf{x}) \wedge \text{poisonous}(\mathbf{x})$

or, equivalently,

$(\forall \mathbf{x}) (\text{mushroom}(\mathbf{x}) \wedge \text{purple}(\mathbf{x})) \Rightarrow \neg \text{poisonous}(\mathbf{x})$

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.,
 - $\forall x (Sqrt(x), Sqrt(x)) = x$ is satisfiable
 - $2 = 2$ is valid
- E.g., definition of (full) *Sibling* in terms of *Parent*:
 - $\forall x, y \text{ Sibling}(x, y) \Leftrightarrow [\neg(x=y) \wedge \exists m, f \neg(m=f) \wedge \text{Parent}(m, x) \wedge \text{Parent}(f, x) \wedge \text{Parent}(m, y) \wedge \text{Parent}(f, y)]$

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.

Interacting with FOL KBs

- Suppose a wumpus-world agent is using an FOL KB and perceives a smell and a breeze (but no glitter) at $t = 5$:
 $Tell(KB, Percept([Smell, Breeze, None], 5))$
 $Ask(KB, \exists a Action(a, 5))$
- I.e., does KB entail any particular actions at $t = 5$?
Answer: *Yes*, $\{a/Shoot\} \leftarrow substitution$ (binding list)
Set of solutions
- Given a sentence S and a substitution α ,
 $S\alpha$ denotes the result of plugging α into S ; e.g.,
 $S = Smarter(x, y)$
 $\alpha = \{x/Hillary, y/Bill\}$
 $S\alpha = Smarter(Hillary, Bill)$
- $Ask(KB, S)$ returns some/all α such that $KB \models S\alpha$

Knowledge base for the wumpus world

- "Perception"

$$\forall b, g, t \text{ Percept}([Smell, b, g], t) \Rightarrow Smelt(t)$$

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

- Reflex: $\forall t \text{ AtGold}(t) \Rightarrow \text{Action}(Grab, t)$

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

$$\forall t \text{ AtGold}(t) \wedge \neg \text{Holding}(Gold, t) \Rightarrow \text{Action}(Grab, t)$$

- $\text{Holding}(Gold, t)$ cannot be observed
 \Rightarrow keeping track of change is essential

Deducing hidden properties

- Properties of locations:

$$\forall x, t \text{ At}(\text{Agent}, x, t) \wedge \text{Smelt}(t) \Rightarrow \text{Smelly}(x)$$

$$\forall x, t \text{ At}(\text{Agent}, x, t) \wedge \text{Breeze}(t) \Rightarrow \text{Breezy}(x)$$

- Squares are breezy near a pit:
- Diagnostic rule—infer cause from effect

$$\forall y \text{ Breezy}(y) \Rightarrow \exists x \text{ Pit}(x) \wedge \text{Adjacent}(x, y)$$

- Causal rule—infer effect from cause

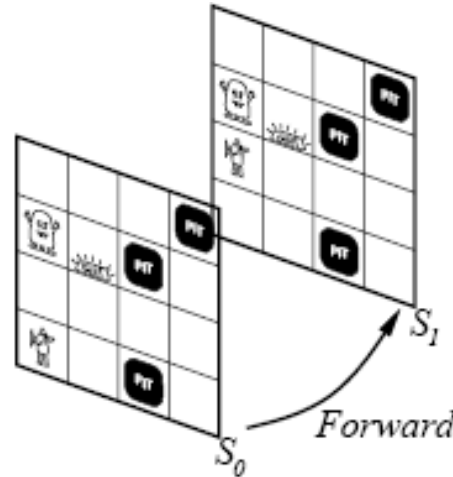
$$\forall x, y \text{ Pit}(x) \wedge \text{Adjacent}(x, y) \Rightarrow \text{Breezy}(y)$$

- 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:

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

Situation calculus (Keeping track of change)

- 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 in s



Describing actions

- “Effect” axiom—describe changes due to action
$$\forall s \text{ AtGold}(s) \Rightarrow \text{Holding}(\text{Gold}, \text{Result}(\text{Grab}, s))$$
- “Frame” axiom—describe non-changes due to action
$$\forall s \text{ HaveArrow}(s) \Rightarrow \text{HaveArrow}(\text{Result}(\text{Grab}, s))$$
- **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:

$$\begin{aligned} \forall a, s \text{ Holding}(\text{Gold}, \text{Result}(a, s)) &\Leftrightarrow \\ &[(a = \text{Grab} \wedge \text{AtGold}(s)) \\ &\vee (\text{Holding}(\text{Gold}, s) \wedge a \neq \text{Release})] \end{aligned}$$

Making plans

- Initial condition in KB :
 $At(Agent, [1, 1], So)$
 $At(Gold, [1, 2], So)$
- Query: $Ask(KB, \exists s Holding(Gold, s))$
i.e., in what situation will I be holding the gold?
- Answer: $\{s/Result(Grab, Result(Forward, So))\}$
i.e., go forward and then grab the gold
- This assumes that the agent is interested in plans starting at So and that So is the only situation described in the KB

Making plans: A better way

- Represent plans as action sequences $[a_1, a_2, \dots, a_n]$
- $PlanResult(p, s)$ is the result of executing p in s
- Then the query $Ask(KB, \exists p Holding(Gold, PlanResult(p, So)))$ has the solution $\{p/[Forward, Grab]\}$
- Definition of $PlanResult$ in terms of $Result$:
 - $\forall s PlanResult([], s) = s$
 - $\forall a, p, s PlanResult([a|p], s) = PlanResult(p, Result(a, s))$
- **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