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 sentences 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, $S_1 \land S_2$ is a sentence

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

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

If $S_1$ and $S_2$ is a sentence, $S_1 \iff S_2$ is a sentence
Last Time: Semantics of Propositional logic

Each model specifies true/false for each proposition symbol

E.g. $A \quad B \quad C$

True True False

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

$\neg S$ is true iff $S$ is false

$S_1 \land S_2$ is true iff $S_1$ is true and $S_2$ is true

$S_1 \lor S_2$ 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.)

\[
\begin{align*}
\alpha & \Rightarrow \beta, \quad \alpha \\
\hline
\beta \\
\end{align*}
\]

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

\[
\begin{align*}
\alpha_1 \land \alpha_2 \land \ldots \land \alpha_n \\
\hline
\alpha_i \\
\end{align*}
\]

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

\[
\begin{align*}
\alpha_1, \alpha_2, \ldots, \alpha_n \\
\hline
\alpha_1 \land \alpha_2 \land \ldots \land \alpha_n \\
\end{align*}
\]

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

\[
\begin{align*}
\alpha_i \\
\hline
\alpha_1 \lor \alpha_2 \lor \ldots \lor \alpha_n \\
\end{align*}
\]

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

\[
\begin{align*}
\neg \neg \alpha \\
\hline
\alpha \\
\end{align*}
\]

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

\[
\begin{align*}
\alpha \lor \beta, \quad \neg \beta \\
\hline
\alpha \\
\end{align*}
\]

◊ **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.)

\[
\begin{align*}
\alpha \lor \beta, \quad \neg \beta \lor \gamma \\
\hline
\alpha \lor \gamma \\
\end{align*}
\]

or equivalently

\[
\begin{align*}
\neg \alpha & \Rightarrow \beta, \quad \beta \Rightarrow \gamma \\
\hline
\neg \alpha \Rightarrow \gamma \\
\end{align*}
\]
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
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:** ∧, ∨, ⇒, ⇔
- **Quantifiers:** ∀, ∃
- **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 → Predicate(Term, …) | Term = Term

Term → Function(Term, …) | Constant | Variable

• Examples:
  • SchoolOf(Manos)
  • Colleague(TeacherOf(Alex), TeacherOf(Manos))
  • >(((+ x y), x)
FOL: Complex sentences

Sentence → AtomicSentence
    | Sentence Connective Sentence
    | Quantifier Variable, … Sentence
    | ¬ Sentence
    | (Sentence)

• Examples:
  • $S_1 \land S_2$, $S_1 \lor S_2$, $(S_1 \land S_2) \lor S_3$, $S_1 \Rightarrow S_2$, $S_1 \Leftrightarrow S_3$
  • $\text{Colleague}(Paolo, Maja) \Rightarrow \text{Colleague}(Maja, Paolo)$
  • $\text{Student}(Alex, Paolo) \Rightarrow \text{Teacher}(Paolo, Alex)$
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 $\text{predicate}(\text{term}_1, \ldots, \text{term}_n)$ is **true** iff
  the relation referred to by $\text{predicate}$ holds between the
  objects referred to by $\text{term}_1, \ldots, \text{term}_n$
Example model

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

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

- 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): \( \forall \)

\[ \forall \ <\text{variables}> \ <\text{sentence}> \]

- "Every one in the cs561 class is smart":
  \[ \forall \ x \ \text{In}(\text{cs561}, \ x) \Rightarrow \text{Smart}(\ x) \]

- \( \forall \ P \) corresponds to the conjunction of instantiations of \( P \)
  \[ \text{In}(\text{cs561}, \ \text{Manos}) \Rightarrow \text{Smart}(	ext{Manos}) \wedge \text{In}(\text{cs561}, \ \text{Dan}) \Rightarrow \text{Smart}(	ext{Dan}) \wedge \ldots \]
  \[ \text{In}(\text{cs561}, \ \text{Clinton}) \Rightarrow \text{Smart}(	ext{Clinton}) \]
Universal quantification (for all): $\forall$

- $\Rightarrow$ is a natural connective to use with $\forall$

- **Common mistake:** to use $\land$ in conjunction with $\forall$
  e.g: $\forall x \ \text{In(cs561, } x) \land \text{Smart}(x)$
  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”:
  $\exists x \; \text{In}(\text{cs561}, x) \land \text{Smart}(x)$

- $\exists P$ corresponds to the disjunction of instantiations of $P$
  \text{In}(\text{cs561}, \text{Manos}) \land \text{Smart}(\text{Manos}) \lor
  \text{In}(\text{cs561}, \text{Dan}) \land \text{Smart}(\text{Dan}) \lor
  \ldots
  \text{In}(\text{cs561}, \text{Clinton}) \land \text{Smart}(\text{Clinton})$
Existential quantification (there exists): \( \exists \)

- \( \land \) is a natural connective to use with \( \exists \)

- **Common mistake:** to use \( \Rightarrow \) in conjunction with \( \exists \)
  e.g. \( \exists x \ln(cs561, x) \Rightarrow \text{Smart}(x) \)
  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"

Quantifier duality: each can be expressed using the other

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

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

• In general we want to prove:

\[ \forall x \ P(x) \iff \neg \exists x \ \neg P(x) \]

\[ \neg \forall x \ P(x) = \neg(\neg(\forall x \ P(x))) = \neg(\neg(P(x_1) \land P(x_2) \land \ldots \land P(x_n))) = \neg(P(x_1) \lor P(x_2) \lor \ldots \lor P(x_n)) \]

\[ \exists x \ \neg P(x) = \neg P(x_1) \lor \neg P(x_2) \lor \ldots \lor \neg P(x_n) \]

\[ \neg \exists x \ \neg P(x) = \neg(\neg P(x_1) \lor \neg P(x_2) \lor \ldots \lor \neg P(x_n)) \]
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
  \[ \forall x, y \quad \text{Brother}(x, y) \Rightarrow \text{Sibling}(x, y) \]

- Sibling is transitive
  \[ \forall x, y, z \quad \text{Sibling}(x, y) \land \text{Sibling}(y, z) \Rightarrow \text{Sibling}(x, z) \]

- One’s mother is one’s sibling’s mother
  \[ \forall m, c \quad \text{Mother}(m, c) \land \text{Sibling}(c, d) \Rightarrow \text{Mother}(m, d) \]

- A first cousin is a child of a parent’s sibling
  \[ \forall c, d \quad \text{FirstCousin}(c, d) \iff \exists p, ps \quad \text{Parent}(p, d) \land \text{Sibling}(p, ps) \land \text{Parent}(ps, c) \]
Example sentences

• One’s mother is one’s sibling’s mother
  \[ \forall m, c, d \; \text{Mother}(m, c) \land \text{Sibling}(c, d) \implies \text{Mother}(m, d) \]

• \[ \forall c, d \exists m \; \text{Mother}(m, c) \land \text{Sibling}(c, d) \implies \text{Mother}(m, d) \]
Translating English to FOL

• Every gardener likes the sun.
  \( \forall \, x \, \text{gardener}(x) \implies \text{likes}(x,\text{Sun}) \)

• You can fool some of the people all of the time.
  \( \exists \, x \, \forall \, t \, (\text{person}(x) \land \text{time}(t)) \implies \text{can-fool}(x,t) \)
Translating English to FOL

- You can fool all of the people some of the time.
  \[ \forall x \exists t \ (\text{person}(x) \land \text{time}(t) \Rightarrow \text{can-fool}(x,t)) \]

- All purple mushrooms are poisonous.
  \[ \forall x \ (\text{mushroom}(x) \land \text{purple}(x) \Rightarrow \text{poisonous}(x)) \]
Translating English to FOL...

• No purple mushroom is poisonous.

\neg (\exists x) \text{ purple}(x) \land \text{ mushroom}(x) \land \text{ poisonous}(x)

or, equivalently,

(\forall x) (\text{ mushroom}(x) \land \text{ purple}(x)) \Rightarrow \neg \text{ poisonous}(x)
Translating English to FOL...

• There are exactly two purple mushrooms.

\[(\exists \ x)(\exists \ y) \mbox{mushroom}(x) \^\ \mbox{purple}(x) \^\ \mbox{mushroom}(y) \^\ \mbox{purple}(y) \^\ \neg(x=y) \^\ (\forall \ z)\]
\[(\mbox{mushroom}(z) \^\ \mbox{purple}(z)) \Rightarrow ((x=z) \^\ (y=z))\]

• Deb is not tall.

\(\neg\ \mbox{tall}(\mbox{Deb})\)
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.

\[(\forall x)(\forall y) \text{above}(x,y) \iff (\text{on}(x,y) \lor (\exists z) (\text{on}(x,z) \land \text{above}(z,y)))\]
Equality

\( \text{term}_1 = \text{term}_2 \) is true under a given interpretation if and only if \( \text{term}_1 \) and \( \text{term}_2 \) refer to the same object.

E.g., \( 1 = 2 \) and \( \forall x \times (\text{Sqrt}(x), \text{Sqrt}(x)) = x \) are satisfiable, \( 2 = 2 \) is valid.

E.g., definition of (full) \textit{Sibling} in terms of \textit{Parent}:

\[
\forall x, y \ Sibling(x, y) \iff \neg (x = y) \land \exists m, f \ - (m = f) \land \\
Parent(m, x) \land Parent(f, x) \land Parent(m, y) \land Parent(f, y)
\]
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”:

\[ \forall x, y \ (x=y) \iff (\forall p, p(x) \iff p(y)) \]

• 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:

\[
\text{function } \text{KB-AGENT}(\text{percept}) \text{ returns an action} \\
\text{static: } KB, \text{ a knowledge base} \\
\quad t, \text{ a counter, initially 0, indicating time} \\
\text{TELL}(KB, \text{MAKE-PERCEPT-SENTENCE}(\text{percept}, t)) \\
\text{action} \leftarrow \text{ASK}(KB, \text{MAKE-ACTION-QUERY}(t)) \\
\text{TELL}(KB, \text{MAKE-ACTION-SENTENCE}(\text{action}, t)) \\
\quad t \leftarrow t + 1 \\
\text{return } \text{action}
\]

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 $t = 5$:

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

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

Answer: Yes, \( \{a/\text{Shoot}\} \) ← substitution (binding list)

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

\[ \text{ASK}(KB, S) \] returns some/all $\sigma$ such that $KB \models S \sigma$
“Perception”
\[ \forall b, g, t \ Percept([\text{Smell}, b, g], t) \Rightarrow \text{Smelt}(t) \]
\[ \forall s, b, t \ Percept([s, b, \text{Glitter}], t) \Rightarrow \text{AtGold}(t) \]

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

\textbf{Reflex with internal state:} do we have the gold already?
\[ \forall t \ \text{AtGold}(t) \land \neg \text{Holding(Gold, t)} \Rightarrow \text{Action(Grab, t)} \]

\underline{\text{Holding(Gold, t) cannot be observed}}
\[ \Rightarrow \text{keeping track of change is essential} \]
Deducing hidden properties

Properties of locations:
\[ \forall l, t \ At(Agent, l, t) \land Smelt(t) \Rightarrow Smelly(l) \]
\[ \forall l, t \ At(Agent, l, t) \land Breeze(t) \Rightarrow Breezy(l) \]

Squares are breezy near a pit:

Diagnostic rule—infer cause from effect
\[ \forall y \ Breezy(y) \Rightarrow \exists x \ Pit(x) \land Adjacent(x, y) \]

Causal rule—infer effect from cause
\[ \forall x, y \ Pit(x) \land Adjacent(x, y) \Rightarrow Breezy(y) \]

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:
\[ \forall y \ Breezy(y) \Leftrightarrow [\exists x \ Pit(x) \land Adjacent(x, y)] \]
**Situation calculus**

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

**Situation calculus** is one way to represent change in FOL:
- Adds a situation argument to each non-eternal predicate
  E.g., $\text{Now}$ in $\text{Holding}(\text{Gold}, \text{Now})$ denotes a situation

Situations are connected by the **Result** function
$\text{Result}(a, s)$ 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
\[ \forall s \ HaveArrow(s) \Rightarrow HaveArrow(Result(Grab, s)) \]

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):

\[ P \text{ true afterwards} \iff [\text{an action made } P \text{ true} \land \neg P \text{ true already and no action made } P \text{ false}] \]

For holding the gold:

\[
\forall a, s \quad Holding(Gold, Result(a, s)) \iff \\
[(a = \text{Grab} \land AtGold(s)) \land \\
\neg (Holding(Gold, s) \land a \neq \text{Release})]
\]
Planning

Initial condition in KB:

\[ \text{At}(\text{Agent}, [1, 1], S_0) \]
\[ \text{At}(\text{Gold}, [1, 2], S_0) \]

Query: \( \text{Ask}(KB, \exists s \ \text{Holding}(\text{Gold}, s)) \)

i.e., in what situation will I be holding the gold?

Answer: \( \{ s/\text{Result}(\text{Grab}, \text{Result}(\text{Forward}, S_0)) \}\)

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 \(\text{Ask}(KB, \exists p \text{ Holding}(Gold, PlanResult(p, S_0)))\) has the solution \(\{p/\{\text{Forward, Grab}\}\}\)

Definition of \(PlanResult\) in terms of \(Result\):

\[\forall s \ PlanResult([], s) = s \quad [\ ] = \text{empty plan}\]

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

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