First-Order Logic & Inference
AI Class 20 (Ch. 8.1–8.3, 9)

Today’s Class
• The last little bit of PL and FOL
  - Axioms and Theorems
  - Sufficient and Necessary
• Logical Agents
  - Reflex
  - Model-Based
  - Goal-Based
• Inference!
  - How do we use any of this?

Axioms, Definitions and Theorems

• **Axioms**: facts and rules that attempt to capture all of the (important) facts and concepts about a domain
• Axioms can be used to prove **theorems**
  - Mathematicians don’t want any unnecessary (dependent) axioms -- ones that can be derived from other axioms
  - Dependent axioms can make reasoning faster, however
  - Choosing a good set of axioms for a domain is a design problem!
• A **definition** of a predicate is of the form “p(X) ↔ …” and can be decomposed into two parts
  - **Necessary** description: “p(x) → …”
  - **Sufficient** description “p(x) ← …”
  - Some concepts don’t have complete definitions (e.g., person(x))

More on Definitions

• **Examples**: define father(x, y) by parent(x, y) and male(x)
  - parent(x, y) is a necessary (but not sufficient) description of father(x, y)
    - father(x, y) → parent(x, y)
  - parent(x, y) ^ male(x) ^ age(x, 35) is a sufficient (but not necessary) description of father(x, y):
    - father(x, y) ← parent(x, y) ^ male(x) ^ age(x, 35)
  - parent(x, y) ^ male(x) is a necessary and sufficient description of father(x, y)
    - parent(x, y) ^ male(x) ↔ father(x, y)

Higher-Order Logics

• FOL only allows to quantify over variables, and variables can only range over objects.
• HOL allows us to quantify over relations
• Example (quantify over functions)
  - “two functions are equal iff they produce the same value for all arguments”
    - ∀f ∀g (f = g) ↔ (∀x f(x) = g(x))
• Example (quantify over predicates)
  - ∀transitive(r) ↔ (∀x∀y∀z (r(x,y) ∧ r(y,z) → r(x,z)))
• More expressive, but undecidable.

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”
    - ∃! x king(x)
  - “Every country has exactly one ruler”
    - ∀c country(c) → ∃! r ruler(c, r)
• Iota operator: “ι x P(x)” means “the unique x such that p(x) is true”
  - “The unique ruler of Freedonia is dead”
    - dead(ι x ruler(freedonia, x))
Logical Agents

Three (non-exclusive) agent architectures:

- **Reflex** agents
  - Have rules that classify situations, specifying 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 Typical Wumpus World

- The agent always starts in the field [1,1].
- The task of the agent is to find the gold, return to the field [1,1] and climb out of the cave.

A Simple Reflex Agent

- Rules to map percepts into observations:
  - $\forall b, g, u, c, t \ Percept([\text{Stench}, b, g, u, c], t) \rightarrow \text{Stench}(t)$
  - $\forall s, g, u, c, t \ Percept([s, \text{Breeze}, g, u, c], t) \rightarrow \text{Breeze}(t)$
  - $\forall s, b, u, c, t \ Percept([s, \text{Glitter}, b, u, c], t) \rightarrow \text{AtGold}(t)$

- Rules to select an action given observations:
  - $\forall t \ AtGold(t) \rightarrow \text{Action(Grab, t)}$

A Simple Reflex Agent

- Some difficulties:
  - 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)

KB-Agents Summary

- Logical agents
  - Reflex: rules map directly from percepts $\rightarrow$ beliefs or percepts $\rightarrow$ actions
    - $\forall b, g, u, c, t \ Percept([\text{Stench}, b, g, u, c], t) \rightarrow \text{Stench(t)}$
    - $\forall t \ AtGold(t) \rightarrow \text{Action(Grab, t)}$
  - Model-based: construct a model (set of t/f beliefs about sentences) as they learn; map from models $\rightarrow$ actions
    - $\text{Action(Grab, t)} \rightarrow \text{HaveGold(t)}$
    - $\text{HaveGold(t)} \rightarrow \text{Action(RetraceSteps, t)}$
  - Goal-based: form goals, then try to accomplish them
    - Encoded as a rule:
      - $\exists s \text{Holding(Gold, s)} \rightarrow \text{GoalLocation([1,1], s)}$
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.

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 wrt. a 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.

Situation Calculus

- Alternatively, 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)
- Or: 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
  \[ \forall y \forall s \forall s' \left( \text{at}(\text{Agent}, y, s) \land \text{atbox}(s) \rightarrow \text{at}(\text{Agent}, y, \text{result}(\text{walk}(y), s)) \right) \]

Situations Summary

- Representing a dynamic world
  - Situations (s0, ..., sn): the world in situation 0-n
    - Teaching(DrM,s0) → today,10:10,whenNotSick, ...
  - Add ‘situation’ argument to statements
  - AtGold(s0)
  - Or, add a ‘holds’ predicate that says ‘sentence is true in this situation’
    - holds(At[2,1], s)
  - Or, add a result(action, situation) function that takes an action and situation, and returns a new situation
    - result(Action(goNorth), s0) ⇒ s,

Deducing Hidden Properties

- From the perceptual information we obtain in situations, we can infer properties of locations
  - l = location, s = situation
  - ∀l, s at(Agent,l,s) ∨ Breeze(s) ⇒ Breezy(l)
  - ∀l, s at(Agent,l,s) ∨ Stench(s) ⇒ Smelly(l)
- Neither Breezy nor Smelly need situation arguments because pits and Wumpuses do not move around
Deducing Hidden Properties II

- We need to write some rules that relate various aspects of a single world state (as opposed to across states).
- There are two main kinds of such rules:
  - **Causal rules** reflect assumed direction of causality:
    
    \[
    \forall l_1, l_2, s \, \text{\text{At(Wumpus,}l_1,l_2)} \land \text{Adjacent(l}_1,l_2) \rightarrow \text{Smelly(l}_2)
    \]
    
    \[
    \forall l_1, l_2, s \, \text{\text{At(Pit,}l_1,l_2)} \land \text{Adjacent(l}_1,l_2) \rightarrow \text{Breezy(l}_2)
    \]
- Systems that reason with causal rules are called **model-based reasoning** systems.

Frames: A Data Structure

- **A frame** divides knowledge into substructures by representing "stereotypical situations."
- Situations can be visual scenes, structures of physical objects,
  - Useful for representing commonsense knowledge.

Representing Change: The Frame Problem

- **Frame axioms**: If property \( x \) doesn’t change as a result of applying action \( a \) in state \( s \), then it stays the same:
  - \( \text{On}(x, z, s) \land \text{Clear}(x, s) \rightarrow \text{On}(x, \text{table}, \text{Result(Move(x, \text{table}, s))}) \land \text{\neg On(x, z, Result(Move(x, \text{table}, s))} \)
  - \( \text{On}(y, z, s) \land \text{y \neq x} \rightarrow \text{On}(y, \text{z, Result(Move(x, \text{table}, s))}) \)
  - 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**:
    - \( \text{On}(x, \text{table}, \text{Result(x,z)}) \leftrightarrow \text{[On}(x, \text{z, s}) \land \text{Clear}(x, s) \land a = \text{Move(x, table)}) \lor \text{[On}(x, \text{table, s}) \land a \neq \text{Move(x, z)})] \)
  - In complex worlds with longer chains of action, even these are too cumbersome:
    - Planning systems use special-purpose inference to reason about the expected state of the world at any point in time during a multi-step plan.
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…
    • 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

• How do you describe every effect of every action?
  • 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 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 a field.
• 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: It’s hard to decide which action is best!
  • Ex: to decide between a forward and a grab, axioms describing when it is okay to move 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:
  (∀a,s) Great(a,s) → Action(a,s)
  (∀a,s) Good(a,s) ∧ ¬(∃b) Great(b,s) → Action(a,s)
  (∀a,s) Medium(a,s) ∧ (¬(∃b) Great(b,s) v Good(b,s)) → 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:
  - $\forall s \text{ Holding(Gold, } s \text{)} \rightarrow \text{GoalLocation([1,1]), } s\text{)}$
- 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: coming soon!

Logical Inference

Chapter 9

Model Checking

- Given KB, does sentence $S$ hold?
  - Basic quick review: What's a KB? What's a sentence?
- Basically generate and test:
  - Generate all the possible models
  - Consider the models $M$ in which KB is TRUE
  - If $\forall M \text{ S}$, then $S$ is provably true
  - If $\forall M \neg S$, then $S$ is provably false
  - Otherwise ($\exists M1 \text{ S } \land \exists M2 \neg S$): $S$ is satisfiable but neither provably true or provably false

Efficient Model Checking

- Davis-Putnam algorithm (DPLL): Generate-and-test model checking with:
  - Early termination (short-circuiting of disjunction and conjunction)
  - Pure symbol heuristic: Any symbol that only appears negated or unnegated must be FALSE/TRUE respectively
  - Can “conditionalize” based on instantiations already produced
  - Unit clause heuristic: Any symbol that appears in a clause by itself can immediately be set to TRUE or FALSE

- WALKSAT: Local search for satisfiability:
  - Pick a symbol to flip (toggles TRUE/FALSE), either using min-conflicts or choosing randomly
  - …or you can use any local or global search algorithm!

Reminder: Inference Rules for FOL

- Inference rules for propositional logic apply to FOL
  - Modus Ponens, And-Introduction, And-Elimination, …
- New (sound) inference rules for use with quantifiers:
  - Universal elimination
  - Existential introduction
  - Existential elimination
  - Generalized Modus Ponens (GMP)

Automating FOL Inference with Generalized Modus Ponens
Automated Inference for FOL

- Automated inference using FOL is harder than PL
  - Variables can take on an infinite number of possible values
  - From their domains, anyway
  - This is a reason to do careful KR!
- From their domains, anyway
  - This is a reason to do careful KR!
- So, potentially infinite ways to apply Universal Elimination
- Gödel's Completeness Theorem says that FOL entailment is only semidecidable*
  - If a sentence is true given a set of axioms, can prove it
  - If the sentence is false, then there is no guarantee that a procedure will ever determine this
- Inference may never halt

*The "halting problem"

Generalized Modus Ponens (GMP)

- Apply modus ponens reasoning to generalized rules
- Combines And-Introduction, Universal-Elimination, and Modus Ponens
  - From P(c) and Q(c) and (∀x)(P(x) ∧ Q(x)) → R(x) derive R(c)
- General case: Given
  - atomic sentences P_1, P_2, ..., P_n
  - implication sentence (Q_1 ∧ Q_2 ∧ ... ∧ Q_n) → R
  - Q_i, ..., Q_n and R are atomic sentences
  - substitution subst(θ, P_i) = subst(θ, Q_i) for i=1,...,N
  - Derive new sentence: subst(θ, R)

Horn Clauses

- A Horn clause is a sentence of the form:
  (∀x) P_1(x) ∧ P_2(x) ∧ ... ∧ P_n(x) → Q(x)
  where:
  - there are 0 or more P_i's and 0 or 1 Q's
  - the P_i's and Q are positive (non-negated) literals
  - Equivalently: P_1(x) ∨ P_2(x) ∨ ... ∨ P_n(x) where the P_i are all atomic and at most one of them is positive
- Horn clauses represent a subset of the set of sentences representable in FOL

Horn Clauses II

- Special cases
  - P_1 ∧ P_2 ∧ ... ∧ P_n → Q
  - P_1 ∧ P_2 ∧ ... ∧ P_n → false
  - true → Q
- These are not Horn clauses:
  - p(a) ∨ q(a)
  - (P ∧ Q) → (R ∨ S)

Forward Chaining

- Proofs start with the given axioms/premises in KB, deriving new sentences using GMP until the goal/query sentence is derived
- This defines a forward-chaining inference procedure because it moves "forward" from the KB to the goal [eventually]
- Inference using GMP is complete for KBs containing only Horn clauses
Forward Chaining Example

- KB:
  - allergies(X) → sneeze(X)
  - cat(Y) ∧ allergic-to-cats(X) → allergies(X)
  - cat(Felix)
  - allergic-to-cats(Lise)

- Goal:
  - sneeze(Lise)

Inference

sneeze(Lise) ← infer truth of goal

- Forward Chaining: apply rules
  - cat(Y) ∧ allergic-to-cats(X) → allergies(X)
  - cat(Felix)
  - allergic-to-cats(Lise)

Backward Chaining

- Backward-chaining deduction using GMP
  - Complete for KBs containing only Horn clauses.
- Proofs:
  - Start with the goal query
  - Find rules with that conclusion
  - Prove each of the antecedents in the implication
  - Keep going until you reach premises!

Backward Chaining Example

- KB:
  - allergies(X) → sneeze(X)
  - cat(Y) ∧ allergic-to-cats(X) → allergies(X)
  - cat(Felix)
  - allergic-to-cats(Lise)

- Goal:
  - sneeze(Lise)

Inference

sneeze(Lise) ← query

- Backward Chaining: apply rules that end with the goal
  - allergies(X) → sneeze(X)
  - cat(Y) ∧ allergic-to-cats(X) → allergies(X)
  - cat(Felix)
  - allergic-to-cats(Lise)

Backward Chaining Algorithm

function BackwardChaining(g, KB) returns a set of substitutions
BackwardChaining(g, KB)

if KB is empty then return (K)

for each clause in KB
  if clause contains g
    return (g)

for each clause in KB
  if clause contains g
    return (g)

if g is a literal then return (K)

if clause ends with g
  return (g)

for each clause in KB
  if clause contains g
    return (g)

function BackwardChaining(g, KB) returns a set of substitutions
BackwardChaining(g, KB)

if KB is empty then return (K)

for each clause in KB
  if clause contains g
    return (g)

for each clause in KB
  if clause contains g
    return (g)

if g is a literal then return (K)

if clause ends with g
  return (g)

for each clause in KB
  if clause contains g
    return (g)

return thenext of BackwardChaining(K, KB)
**Forward vs. Backward Chaining**

- **FC is data-driven**
  - Automatic, unconscious processing
  - E.g., object recognition, routine decisions
  - May do lots of work that is irrelevant to the goal

- **BC is goal-driven, appropriate for problem-solving**
  - Where are my keys? How do I get to my next class?
  - Complexity of BC can be much less than linear in the size of the KB

---

**Completeness of GMP**

- GMP (using forward or backward chaining) is complete for KBs that contain only Horn clauses
- It is **not complete** for simple KBs that contain non-Horn clauses
- The following entail that $S(A)$ is true:
  - $$(\forall x) P(x) \rightarrow Q(x)$$
  - $$(\forall x) \lnot P(x) \rightarrow R(x)$$
  - $$(\forall x) Q(x) \rightarrow S(x)$$
  - $$(\forall x) R(x) \rightarrow S(x)$$
- If we want to conclude $S(A)$, with GMP we cannot, since the second one is not a Horn clause
- It is equivalent to $P(x) \lor R(x)$