Propositional Logic

- **Advantages**
  - Simple
  - Generic

- **Disadvantage**
  - All we have are propositions (facts)
  - Q: What else would we want?
PL for Vacuum World

- Let $D_i$, $i = 1, \ldots, n$, represent “room $i$ is dirty”

- How to say the following?
  - All the rooms are clean
  - The dog is in some room
  - Any room that the dog has been in is dirty
  - All the upstairs rooms are clean, except the bathrooms
  - Rooms next to dirty rooms have a dusty smell
  - No two adjacent rooms are both dirty
  - The vacuum should visit every dirty room
  - When vacuuming a room, check whether any nearby rooms are dirty

PL can be Cumbersome

- For example, introducing $NextTo$ required $O(n^2)$ new propositional symbols
- Inference exponential in number of propositional symbols
  - Infinite domains... forget about it
- Solution: put more structure on facts
First Order Logic

- Create a richer language by adding additional structure.

<table>
<thead>
<tr>
<th>Propositional Logic</th>
<th>First Order Logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logically reason over propositions</td>
<td>Logically reason about objects, their properties, and their relationships.</td>
</tr>
<tr>
<td>A superset of PL.</td>
<td></td>
</tr>
</tbody>
</table>

First Order Logic

- Objects
  - Rooms, dogs, vacuums, wumpus rooms...
- Predicates
  - Tests a property of one or more objects
    - `IsBreezy(x)`, `AreAdjacent(x,y)`, `isCleanerThan(x,y)`
  - Value (true or false) can be evaluated given a model
  - With no arguments, equivalent to PL propositions.
  - Arity??
- Functions
  - Names an object as it relates to other objects
    - `MotherOf(x)`, `RoomEastOf(x)`, `ChildOf(x,y)`
  - By themselves, do not form legal sentences; used along with predicates.
  - With no arguments, is a constant object.... E.g., “John”.
  - Arity??
Predicates and Functions

- These look like “function calls” from Java or C, but are quite different!

- Defined implicitly, not explicitly.
  - Consider:
    - Male(John) ^ Male(Arnold)
    - AreBrothers(John, Arnold) v AreCousins(John, Arnold)
    - FatherOf(John) = FatherOf(Arnold)

- Relationships are built up from sentences of FOL, in which relationships may appear anywhere.
  - The properties are a matter of inference.

Syntax of First-Order Logic

- Sentence → AtomicSentence | ¬Sentence
  | (Sentence Connective Sentence)
  | Quantifier Variable, ... Sentence

- AtomicSentence → Predicate(Term,...) | Term = Term

- Term → Function(Term,...) | Constant | Variable

- Connective → ⇒ | ∧ | ∨ | ⇔

- Quantifier → ∀ | ∃

- Constant → A | X₁ | Room1500EECS | ...

- Variable → a | x | s | ...

- Predicate → IsDirty | IsNextTo | IsComfy | ...

- Function → RoomOnRight | FurnitureInRoom | ...
Equality

- Special built-in predicate symbol
- $\text{Term} = \text{Term}$
  - An atomic sentence
  - $=$: a binary predicate, equivalence reln
  - Written using infix notation

- Similarly, $\neq$.

FOL Models

- Just like in PL, we can evaluate the *satisfiability* of a sentence *with respect to a model*.

- What was a model in PL?
  - Just an assignment of true/false to every proposition

- What is a FOL Model?
FOL Models

- A model must specify:
  - Which constants represent the same objects
    - \( \text{IsCat}(\text{Felix}) \land \text{IsDog}(\text{Fido}) \)
    - \( \text{Owner}(\text{Felix}) = \text{Robert} \)
    - \( \text{Owner}(\text{Fido}) = \text{Mr.Haroldson} \)
    - \( \text{HomeOf}(\text{Felix}) = \text{HomeOf}(\text{Fido}) \)
    - \( \text{LivesAlone}(\text{Robert}) \)
  - Behavior of predicates
    - A truth table over all possible inputs
  - Behavior of functions
    - A map over all possible inputs

A Review Game
Making FOL even more powerful

- So far:
  - A neater way to write propositions
  - Captures object/relation structure, but still need to enumerate all cases
  - Real FOL power comes from ability to quantify over objects using variables

Universal Quantifier

- “All the rooms are clean”
  - $\forall x. \neg \text{Dirty}(x)$
  - $\forall x. \text{Room}(x) \implies \neg \text{Dirty}(x)$

Equivalent to big conjunction, where $x$ is replaced by every object in universe
Existential Quantifier

- “The dog is in some room”
  - $\exists x. \text{DogIn}(x)$
  - $\exists x. \text{Room}(x) \land \text{DogIn}(x)$

Existential quantifier

Equivalent to big disjunction, where $x$ is replaced by every object in universe

“First-Order”

- First-order logic (FOL)
  - allows quantification over objects
- Second-order logic
  - allows quantification over relations
  - can describe properties of relations (e.g., transitivity)
- Higher-order logic (HOL)
  - Quantify over relations over relations...
- Zeroth-order logic
  - No quantification
  - (aka propositional logic)
Sentences with Variables

- $\forall x. \Phi(x)$
  - True in model iff $\Phi(x)$ true no matter what object $x$ denotes
- $\forall x. \text{Room}(x) \Rightarrow \neg \text{Dirty}(x)$
  - True iff $\text{Room}(x) \Rightarrow \neg \text{Dirty}(x)$ is true for all objects $x$
  - Trivially true for any $x$ not a room
- $\exists x. \text{Room}(x) \land \text{DogIn}(x)$
  - True iff there is some $x$ s.t. $x$ is a room with a dog in it

Compare and contrast:
- $\exists x. \text{Room}(x) \Rightarrow \text{DogIn}(x)$
- $\forall x. \text{Room}(x) \land \neg \text{Dirty}(x)$

Sentences about Rock Stars’ Houses

- Rock stars are rich.
- Everybody has a home.
- Rich people have big homes.
- Bruce is a rock star.
Rock Star House KB

∀p. IsRockStar(p) ⇒ IsRich(p)
∀p. IsPerson(p) ⇒ ∃h. HomeOf(h,p)
∀p. IsRich(p) ⇒ ∃h. HomeOf(h,p) ∧ IsBig(h)
IsRockStar(Bruce)

Rock Star House Model
DeclaraAve Approach

- Program an agent by TELLing it things
- Equivalently, construct and install a KB
- Advantages
  - Flexibility: state knowledge independently of how it would be used
  - Transparency to humans
  - Agent behavior *malleable via language*

Knowledge-Based Agent

```plaintext
function KB-Agent(percept) returns an action

static: a knowledge base, KB
        a counter, t, 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
```

Environment

Reasoning (Inference)

Knowledge Base

Query: “What should I do next?”

Query answer: Best action
Developing a KB Agent

- Endow with KB axiomatizing domain
  - Initial situation
  - Causal laws (how world evolves)
  - Etc.
- Develop techniques for
  - Describing percepts as sentences
    - E.g., observe “facts” that hold
  - Get action as result of ASK

Reflex KB Agent

- Actions determined by rules in reaction to situation
  \[
  \forall x, t. \ \text{VacuumIn}(x, t) \land \text{Dirty}(x, t) \Rightarrow \text{Action}(\text{Suck}, t)
  \]
- Recognizing imperfect perception:
  \[
  \forall x, t. \ \text{VacuumIn}(x, t) \land \text{SeemsDirty}(x, t) \Rightarrow \text{Action}(\text{Suck}, t)
  \]
**Model-Based KB Agent**

- **Reflex limitations**
  - Cannot account for historical percepts
  - Rules easily invalidated by additional knowledge
- **Model-based**
  - KB describes internal model of world
  - Includes logical specification of effects of actions

\[ \forall x,t. \text{VacuumIn}(x,t) \land \text{Action}(\text{Suck},t) \Rightarrow \text{Clean}(x,t+1) \]

---

**Goal-Based KB Agents**

- **Goal-based**: model-based, plus a rep’n of objective in terms of desirable states
- **Utility-based**: goal-based, but with degrees of preference
- In all these, action query implemented through logical inference.