Chapter 8: First-Order Logic

CS-4365 Artificial Intelligence

Instructor: Chris Irwin Davis

- As we saw in Chapter 7, propositional logic lacks the expressive power to *concisely* describe an environment with many objects.
- For example, we were forced to write a separate rule about breezes and pits for each square, such as
  - $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$.
- In English, on the other hand, it seems easy enough in say, once and for all, “Squares adjacent to pits are breezy.”
  - The syntax and semantics of English somehow make it possible to describe the environment concisely.

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 theory	facts	degree of belief
Fuzzy logic	facts + degree of truth	known interval value

- 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

- Whereas **propositional logic** assumes world contains facts, **first-order logic** (like natural language) assumes the world contains
  - **Objects**: people, houses, numbers, theories, Ronald McDonald, colors, baseball games, wars, centuries . . .
  - **Relations**: red, round, bogus, prime, multistoried . . ., brother of, bigger than, inside, part of, has color, occurred after, owns, comes between, . . .
  - **Functions**: father of, best friend, third inning of, one more than, end of ...

FOL or Predicate Calculus

- There is need to have variables.
- We want to be able to refer to a set of objects as having some property and avoid to indicate that each member of the set has that property.
  - **Objects**: cats, student, house
  - **Relation**: brother of, sister of
  - **Properties**: fat, red, cold
  - **Functions**: father of, best friend

Syntax of FOL: Basic Elements

Constants	*KingJohn*, 2, *UCB*, ...
Predicates	*Brother*, >, ...
Functions	*Sqrt*, *LeftLegOf*, ...
Variables	*x*, *y*, *a*, *b*,...
Connectives	$\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\Leftrightarrow$
Quantifiers	$\forall$, $\exists$

Atomic Sentences

- Atomic Sentences
  - *predicate*(*term*$_1$, … , *term*$_n$)
  - *term*$_1$ = *term*$_2$
- Term
  - *function*(*term*$_1$, … , *term*$_n$)
  - *constant*
  - *variable*
- Examples:
  - *Brother*(*KingJohn*, *RichardTheLionheart*)
  - GreaterThan(*Length*(*LeftLegOf*(*Richard*)), *Length*(*LeftLegOf*(*KingJohn*)))

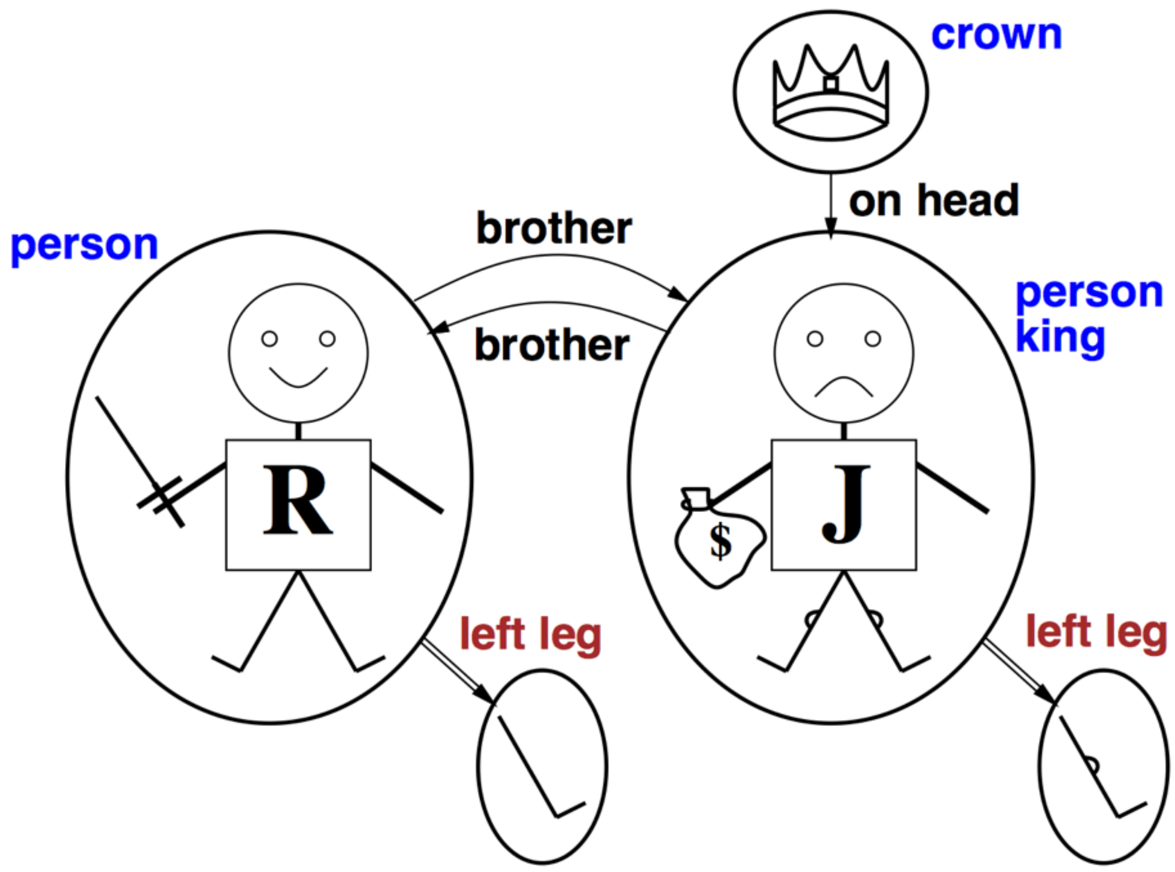
Complex Sentences

- Complex sentences are made from atomic sentences using connectives
  - $\neg S$,
  - $S_1 \wedge S_2$,
  - $S_1 \vee S_2$,
  - $S_1 \Rightarrow S_2$,
  - $S_1 \Leftrightarrow S_2$
- Examples:
  - $Sibling(KingJohn, Richard) \Rightarrow Sibling(Richard, KingJohn)$
  - $>(1, 2) \vee \leq(1, 2)$
  - $>(1, 2) \wedge \neg>(1, 2)$

Truth in FOL

- Sentences are true with respect to a model and an interpretation
- Model contains ≥ 1 objects (domain elements) and relations among them, where…
  - constant symbols → objects
  - predicate symbols → relations
  - function symbols → functional relations
- An atomic sentence *predicate*(*term*$_1$, … , *term*$_n$) is true iff the objects referred to by *term*$_1$, . . . , *term*$_n$ are in the relation referred to by predicate

Models for FOL



Truth Example

- Consider the interpretation in which
  - *Richard* $\longrightarrow$ Richard the Lionheart
  - *John* $\longrightarrow$ the evil King John
  - *Brother* $\longrightarrow$ the brotherhood relation
- Under this interpretation, *Brother*(*Richard*, *John*) is true just in the case where Richard the Lionheart and the evil King John are in the brotherhood relation in the model

Models for FOL

- 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!

Universal Quantification

- $\forall\langle\text{variables}\rangle\ \langle\text{sentence}\rangle$
- Everyone at Berkeley is smart:
  - $\forall x\ At(x,Berkeley) \Rightarrow Smart(x)$
- $\forall x\ P$ is true in a model m iff P is true with x being each possible object in the model
- Roughly speaking, equivalent to the conjunction of instantiations of P
  - $(At(KingJohn, Berkeley) \Rightarrow Smart(KingJohn)) \wedge$
    $(At(Richard, Berkeley) \Rightarrow Smart(Richard)) \wedge$
    $(At(Berkeley, Berkeley) \Rightarrow Smart(Berkeley)) \wedge$
    $\ldots$

Universal Quantification (cont'd)

- There is need to express properties of sets rather than enumerating the same property for ***all*** members of the set.
- Universal quantifier ∀
  - Indicates that a sentence is true for all values of its variables.
  - All cats are mammals. Is logically equivalent to:
    - If x is a cat than x is mammal.
  - $\forall x\ Cat(x) \Rightarrow Mammal(x)$
    - This replaces a large number of relations for all the cats in the world.
    - $\text{Cat(Spot)} \Rightarrow \text{Mammal(Spot)} \land$
    - $\text{Cat(Felix)} \Rightarrow \text{Mammal(Felix)} \land \ldots$
      ∀ is universal quantifier because the logical expression
- $Cat \Rightarrow Mammal$
  - is true for all cats in the universe.

A Common Mistake to Avoid

- Typically, $\Rightarrow$ is the main connective with $\forall$
- **Common mistake**:
  - using $\wedge$ as the main connective with $\forall$:
  - $\forall x\ At(x, Berkeley) \wedge Smart(x)$
  - means "Everyone is at Berkeley *and* everyone is smart"

Existential Quantification

- $\exists\langle variables\rangle\ \langle sentence\rangle$
- Someone at Stanford is smart:
  - $\exists x\ At(x, Stanford) \wedge Smart(x)$
- $\exists x\ P$ is true in a model m iff P is true with x being some possible object in the model
- Roughly speaking, equivalent to the disjunction of instantiations of P
  - $(At(KingJohn, Stanford) \wedge Smart(KingJohn)) \vee$
    $(At(Richard, Stanford) \wedge Smart(Richard)) \vee$
    $(At(Stanford, Stanford) \wedge Smart(Stanford)) \vee$
    $\ldots$

Existential Quantification

- Existential quantifier ∃
- Indicates that a sentence is true for ***some*** values in the domain.
- Spot has a sister who is a cat.
  - $\exists$ x Sister(x, Spot) $\wedge$ Cat(x)
  - Is equivalent to
    (Sister (Spot, Spot) $\wedge$ *Cat(Spot))* $\vee$
    (Sister (Felix, Spot) $\wedge$ *Cat(Felix))* $\vee$
    (Sister (Rebecca, Spot) $\wedge$ *Cat(Rebecca))…*

Another Common Mistake to Avoid

- Typically, $\wedge$ is the main connective with $\exists$
- **Common mistake**:
  - using $\Rightarrow$ as the main connective with $\exists$:
  - $\exists x\, At(x, Stanford) \Rightarrow Smart(x)$
    - …is true if there is anyone who is not at Stanford!

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

Review Caveats / “Gotchas”

- Note that
  - $\Rightarrow$ is normally used with $\forall$
  - $\wedge$ is normally used with $\exists$
- Think why

Nested Quantifiers

- $\forall xy$ is equivalent to $\forall x \forall y$
- $\forall xy\ Parent(x,y) \Rightarrow Child(y,x)$
- Everybody loves somebody
  - $\forall x \exists y\ Loves(x,y)$
- There is someone who is loved by everyone
  - $\exists y \forall x\ Loves(x,y)$

Relationship Between ∀ and ∃

- ∀ and ∃ are related through negation
  - $\forall x\, \neg P(x) \equiv \neg \exists x\, P(x)$
  - $\exists x\, \neg P(x) \equiv \neg \forall x\, P(x)$
- Nobody likes taxes
  - $\forall x\, \neg Likes(x, Taxes) \equiv \neg \exists x\, Likes(x, Taxes)$
- Everybody likes Ice cream
  - $\forall x\, Likes\, (x, IceCream) \equiv \neg \exists x\, \neg Likes\, (x, IceCream)$

Equivalent Quantification

- Some Equivalent Sentences
  - $\neg\exists x\, p(x) \equiv \forall x\, \neg p(x)$
  - $\neg\forall x\, p(x) \equiv \exists \mathrm{x}\, \neg p(x)$
  - $\exists x\, p(x) \equiv \exists y\, p(y)$
  - $\forall x\, q(x) \equiv \forall y\, q(y)$
  - $\forall x\, (p(x) \wedge q(x)) \equiv \forall x\, p(x) \wedge \forall y\, q(y)$
  - $\exists x\, (p(x) \vee q(x)) \equiv \exists x\, p(x) \vee \exists y\, q(y)$

Fun With Sentences

- Brothers are siblings
  - $\forall x,y\ Brother(x,y) \Rightarrow Sibling(x,y).$
- "Sibling" is symmetric
  - $\forall x,y\ Sibling(x,y) \Leftrightarrow Sibling(y,x).$
- One's mother is one's female parent
  - $\forall x,y\ Mother(x,y) \Leftrightarrow (Female(x) \wedge Parent(x,y)).$
- A first cousin is a child of a parent's sibling
  - $\forall x,y\ FirstCousin(x, y) \Leftrightarrow$
    $\exists p,ps\ Parent(p, x) \wedge Sibling(ps, p) \wedge Parent(ps, y)$

Equality

- $term_1 = term_2$ is true under a given interpretation iff $term_1$ 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*:
  - $\forall x,y\ Sibling(x,y) \Leftrightarrow [\neg(x=y) \wedge \exists m,f\ \neg(m=f) \wedge Parent(m, x) \wedge Parent(f, x) \wedge Parent(m, y) \wedge Parent(f, y)]$

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*, ∃*a* *Action*(*a*,5))
    - i.e., does *KB* entail any particular actions at $t = 5$?
  - Answer: *Yes*, {*a*/*Shoot*} ← substitution (binding list)

Interacting with FOL KBs

- Given a sentence S and a substitution σ,
- $S\sigma$ denotes the result of plugging σ into S;
- e.g.,
  - $S = Smarter(x, y)$
    $\sigma = \{x/Hillary, y/Bill\}$
    $S\sigma = Smarter(Hillary, Bill)$
- $Ask(KB, S)$ returns some/all σ such that $KB \models S\sigma$

Interacting with FOL KBs

- A typical percept sentence would be
  - $Percept([Stench, Breeze, Glitter\ , None, None], 5)$.
- To determine which is best, the agent program executes the query
  - $\textsc{AskVars}(\exists a\ BestAction(a, 5))$,
  - which returns a **binding list** such as $\{a/Grab\}$.

Interacting with FOL KBs

- The agent program can then return Grab as the action to take. The raw percept data implies certain facts about the current state. For example:
  - $\forall t, s, g, m, c\ Percept([s, Breeze, g, m, c], t) \Rightarrow Breeze(t)$,
  - $\forall t, s, b, m, c\ Percept([s, b, Glitter , m, c], t) \Rightarrow Glitter(t)$,
  - etc.
- Simple “reflex” behavior can also be implemented by **quantified implication sentences**. For example, we have
  - $\forall t\ Glitter (t) \Rightarrow BestAction(Grab, t)$.

Interacting with FOL KBs

- We have represented the agent's inputs and outputs; now it is time to represent the environment itself. Let us begin with objects.
- Obvious candidates are squares, pits, and the wumpus. We could name each square—$Square_{1,2}$ and so on—but then the fact that $Square_{1,2}$ and $Square_{1,3}$ are adjacent would have to be an "extra" fact, and we would need one such fact for each pair of squares.
- It is better to use a complex term in which the row and column appear as integers; for example, we can simply use the list term [1, 2].
  - Because lists can be parameters to predicates

Interacting with FOL KBs

- Now we have to expressive power to define general concepts, such as…
- Adjacency of any two squares
  - $\forall x, y, a, b\ Adjacent([x, y], [a, b]) \Leftrightarrow$
    $(x=a \wedge ((y=b-1) \vee (y=b+1)) \vee (y=b \wedge ((x=a-1) \vee (x=a+1))$.

Interacting with FOL KBs

- We could name each pit, but this would be inappropriate for a different reason: there is no reason to distinguish among pits.
- It is simpler to use a unary predicate *Pit* that is true of squares containing pits.
- Finally, since there is exactly one wumpus, a constant *Wumpus* is just as good as a unary predicate (and perhaps more dignified from the wumpus' viewpoint).

Interacting with FOL KBs

- The agent's location changes over time, so we write $At(Agent, s, t)$ to mean that the agent is at square s at time t.
- We can fix the wumpus's location with
  - $\forall t\ At(Wumpus, [2, 2], t)$.
- We can then say that objects can only be at one location at a time:
  - $\forall x, s_1, s_2, t\ \ At(x, s_1, t) \wedge At(x, s_2, t) \Rightarrow s_1 = s_2$.

Interacting with FOL KBs

- Given its current location, the agent can infer properties of the square from properties of its current percept.
- For example, if the agent is at a square and perceives a breeze, then that square is breezy:
  - $\forall s,t\ At(Agent, s, t) \wedge Breeze(t) \Rightarrow Breezy(s)$.
- It is useful to know that a square is breezy because we know that the pits cannot move about. Notice that Breezy has no time argument.

Interacting with FOL KBs

- Having discovered which places are breezy (or smelly) and, very important, not breezy (or not smelly), the agent can deduce where the pits are (and where the wumpus is).
- Whereas propositional logic necessitates a separate axiom for each square (q.v. R_2 and R_3 on page 247) and would need a different set of axioms for each geographical layout of the world, first-order logic just needs one axiom:
  - $\forall s\ Breezy(s) \Leftrightarrow \exists r\ Adjacent(r, s) \wedge Pit(r)$.
  - Where s is some square and r is some *other* square
  - Is is possible that $s{=}r$?

Interacting with FOL KBs

- Similarly, in first-order logic we can quantify over time, so we need just one successor-state axiom for each predicate, rather than a different copy for each time step.
- For example, the predicate logic axiom for the arrow (Equation (7.2) on page 267) now becomes
  - $\forall t\ HaveArrow(t{+}1) \Leftrightarrow (HaveArrow(t) \wedge \neg Action(Shoot, t))$.
  - Induction!
    - If we have the arrow at t=1, then we can determine at any point in the game if we still have the arrow.

Interacting with FOL KBs

- From these two example sentences, we can see that the first-order logic formulation is no less concise than the original English-language description given in Chapter 7.
- You are invited to construct analogous axioms for the agent's location and orientation; in these cases, the axioms quantify over both space and time.
- As in the case of propositional state estimation, an agent can use logical inference with axioms of this kind to keep track of aspects of the world that are not directly observed.
- Chapter 10 goes into more depth on the subject of first-order successor-state axioms and their uses for constructing plans.

§8.4 Knowledge Engineering in FOL

Knowledge Engineering in FOL

- Knowledge engineering projects vary widely in content, scope, and difficulty, but all such projects include the following steps:
  - Identify the task
  - Assemble the relevant knowledge
  - Decide on a vocabulary of predicates, functions, and constants
  - Encode general knowledge about the domain
  - Encode a description of the specific problem instance
  - Pose queries to the inference procedure and get answers
  - Debug the knowledge base

Knowledge Engineering in FOL

- **Identify the task**
  - The knowledge engineer must delineate the range of questions that the knowledge base will support and the kinds of facts that will be available for each specific problem instance.
  - For example, does the wumpus knowledge base need to be able to choose actions or is it required to answer questions only about the contents of the environment? Will the sensor facts include the current location? The task will determine what knowledge must be represented in order to connect problem instances to answers.
  - This step is analogous to the PEAS process for designing agents in Chapter 2.

Knowledge Engineering in FOL

- **Assemble the relevant knowledge**
  - The knowledge engineer might already be an expert in the domain, or might need to work with real experts to extract what they know—a process called knowledge acquisition.
  - At this stage, the knowledge is not represented formally. The idea is to understand the scope of the knowledge base, as determined by the task, and to understand how the domain actually works.
  - For the wumpus world, which is defined by an artificial set of rules, the relevant knowledge is easy to identify. (Notice, however, that the definition of adjacency was not supplied explicitly in the wumpus-world rules.)

Knowledge Engineering in FOL

- **Decide on a vocabulary of predicates, functions, and constants**
  - i.e. translate the important domain-level concepts into logic-level names. This involves many questions of knowledge-engineering style.
  - Like programming style, this can have a significant impact on the eventual success of the project.
  - For example
    - Should pits be represented by objects or by a unary predicate on squares?
    - Should the agent's orientation be a function or a predicate?
    - Should the wumpus's location depend on time?

Knowledge Engineering in FOL

- **Decide on a vocabulary of predicates, functions, and constants**
  - Once the choices have been made, the result is a vocabulary that is known as the ontology of the domain.
  - The word ontology means a particular theory of the nature of being or existence.
  - The ontology determines what kinds of things exist, but does not determine their specific properties and interrelationships.

Knowledge Engineering in FOL

- **Encode general knowledge about the domain**
  - The knowledge engineer writes down the axioms for all the vocabulary terms.
  - This pins down (to the extent possible) the meaning of the terms, enabling the expert to check the content.
  - Often, this step reveals misconceptions or gaps in the vocabulary that must be fixed by returning to step 3 and iterating through the process.

Knowledge Engineering in FOL

- **Encode a description of the specific problem instance**
    - If the ontology is well thought out, this step will be easy.
    - It will involve writing simple atomic sentences about instances of concepts that are already part of the ontology.
    - For a logical agent, problem instances are supplied by the sensors, whereas a “disembodied” knowledge base is supplied with additional sentences in the same way that traditional programs are supplied with input data.

Knowledge Engineering in FOL

- **Pose queries to the inference procedure and get answers**
  - This is where the "reward" is:
    - we can let the inference procedure operate on the axioms and problem-specific facts to derive the facts we are interested in knowing.
  - Thus, we avoid the need for writing an application-specific solution algorithm.

Knowledge Engineering in FOL

UTD

- **Debug the knowledge base**
  - Unfortunately, the answers to queries will seldom be correct on the first try. More precisely, the answers will be correct for the knowledge base as written, assuming that the inference procedure is sound, but they will not be the ones that the user is expecting.
  - For example, if an axiom is missing, some queries will not be answerable from the knowledge base. A considerable debugging process could result.
  - Missing axioms or axioms that are too weak can be easily identified by noticing places where the chain of reasoning stops unexpectedly.

Example: Electronic Circuits Domain