# First Order Logic (FOL)

"Artificial Intelligence: A Modern Approach", Chapter 8

# Outline

- Why FOL?
- Syntax and semantics of FOL
- Using FOL
- Wumpus world in FOL

# Pros and cons of propositional logic

- ☺ Propositional logic is declarative
- ☺ Propositional logic allows partial/disjunctive/negated information
- ☺ 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

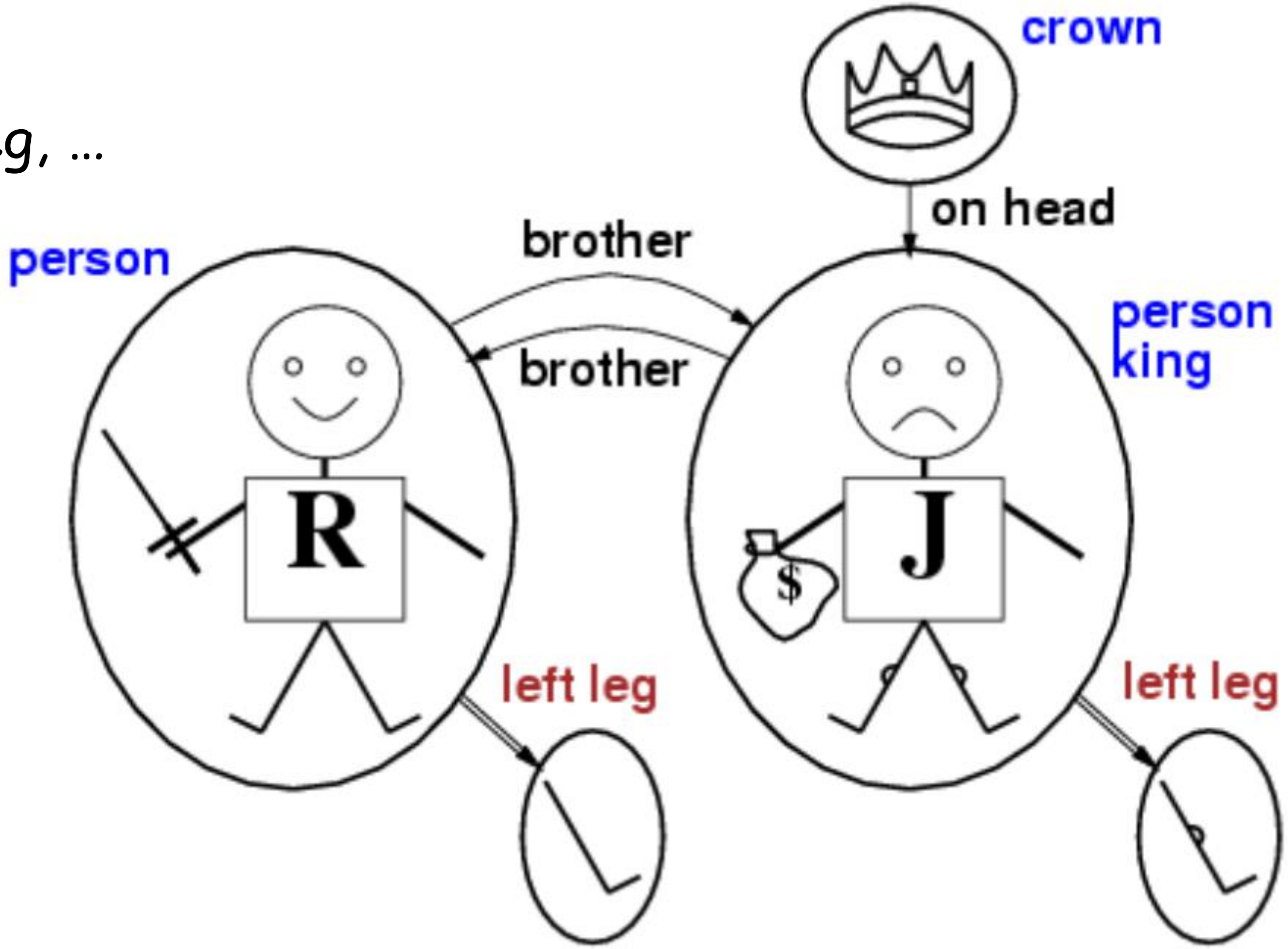
- Whereas propositional logic assumes the world contains facts,
- First-order logic (like natural language) assumes the world contains
  - **Objects:** people, houses, numbers, colors, baseball games, wars, …
  - **Relations:**
    - Unary relation or property: red, round, prime, …
    - n-ary: brother of, bigger than, part of, comes between, …
  - **Functions:** father of, best friend, one more than, plus, …
- Some basic elements of natural language (included also in FOL):
  - Nouns and noun phrases referring to objects
  - Verbs and verb phrases referring to relation among objects
    - Some of them are functions (relations with only one value for each input)

# Symbols & interpretations

- Types of symbols
  - Constant symbols: objects
  - Predicate symbols: relations
  - Function symbols: functional relations

# Example: objects, relations, and functions

- Constant symbols
  - Richard, John, Richard's Left leg, …
- Predicate symbols
  - Brother, OnHead, Person, King, and Crown
- Function symbol
  - LeftLeg



# Syntax of FOL: Basic elements

- Logical elements
  - Connectives $\neg, \Rightarrow, \wedge, \vee, \Leftrightarrow$
  - Quantifiers $\forall, \exists$
- Domain specific elements
  - Constants: KingJohn, 2, NUS,...
  - Predicates Brother, >,...
  - Functions Sqrt, LeftLegOf,...
- Non-logical general elements
  - Variables x, y, a, b,...
  - Equality =

# Syntax of FOL

$$
\begin{array}{rcl}
Sentence & \rightarrow & AtomicSentence \mid ComplexSentence \\
AtomicSentence & \rightarrow & Predicate \mid Predicate(Term, \ldots) \mid Term = Term \\
ComplexSentence & \rightarrow & (\ Sentence\ ) \mid [\ Sentence\ ] \\
& \mid & \neg\ Sentence \\
& \mid & Sentence \land Sentence \\
& \mid & Sentence \lor Sentence \\
& \mid & Sentence \ \Rightarrow\ Sentence \\
& \mid & Sentence \ \Leftrightarrow\ Sentence \\
& \mid & Quantifier\ Variable, \ldots\ Sentence \\
Term & \rightarrow & Function(Term, \ldots) \\
& \mid & Constant \\
& \mid & Variable \\
Quantifier & \rightarrow & \forall \mid \exists \\
Constant & \rightarrow & A \mid X_1 \mid John \mid \cdots \\
Variable & \rightarrow & a \mid x \mid s \mid \cdots \\
Predicate & \rightarrow & True \mid False \mid After \mid Loves \mid Raining \mid \cdots \\
Function & \rightarrow & Mother \mid LeftLeg \mid \cdots \\
\textsc{Operator Precedence} & : & \neg, =, \land, \lor, \Rightarrow, \Leftrightarrow
\end{array}
$$

# Atomic sentences

- Atomic sentence → Predicate
  | Predicate ($Term_1$,...,$Term_n$)
  | $Term_1$ = $Term_2$
- Term → Constant
  | variable
  | Function ($Term_1$,...,$Term_n$)

A **term** is a logical expression that refers to an object

- E.g., Brother(KingJohn, RichardTheLionheart)
  >(Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))

# Complex sentences

- Complex sentences are made from atomic sentences using connectives

$$\neg S,\ S_1 \land S_2,\ S_1 \lor S_2,\ S_1 \Rightarrow S_{2,}\ S_1 \Leftrightarrow S_2$$

- E.g. Sibling(KingJohn, Richard) $\Rightarrow$ Sibling(Richard, KingJohn)

$>(1,2) \lor \leq(1,2)$

$>(1,2) \land \neg >(1,2)$

# Truth in first-order logic

- Sentences are true with respect to a model and an interpretation
- Model contains objects (domain elements) and relations among them
- Interpretation specifies referents for
  - 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

# Quantifiers

- We want to express properties of entire collections of objects, instead of enumerating the objects by name.
- **Quantifiers** let us do this
- First-order logic contains two standard quantifiers
  - Universal
  - Existential

# Universal quantification

- $\forall$<variables> <sentence>
- Example
  - Everyone at AUT is smart:

$$\forall x \; At(x, AUT) \Rightarrow Smart(x)$$

- $\forall x \; P(x)$ is true in a model m iff $P(x)$ is true with x being each possible object in the model
  - Equivalent to the conjunction of instantiations of $P(x)$
    - $At(KingJohn, AUT) \Rightarrow Smart(KingJohn)$

$\wedge \; At(Richard, AUT) \Rightarrow Smart(Richard)$
$\wedge \; At(AUT, AUT) \Rightarrow Smart(AUT)$
$\wedge \ldots$

# A common mistake to avoid

- Typically, $\Rightarrow$ is the main connective with $\forall$
- Common mistake: using $\wedge$ as the main connective with $\forall$:

$$\forall x\ At(x, AUT) \wedge Smart(x)$$

means "Everyone is at AUT and everyone is smart"

# Existential quantification

- $\exists$<variables> <sentence>
- Someone at *AUT* is smart:

$$\exists x\ At(x, AUT) \land Smart(x)$$

- $\exists x\ P(x)$ is true in a model m iff $P(x)$ is true with x being some possible object in the model
  - Equivalent to the disjunction of instantiations of P(x)
    - At(KingJohn, *AUT*) $\land$ Smart(KingJohn)

    $\lor$ At(Richard, *AUT*) $\land$ Smart(Richard)

    $\lor$ At(*AUT*, *AUT*) $\land$ Smart(*AUT*)

    $\lor$...

# Another common mistake to avoid

- Typically, $\wedge$ is the main connective with $\exists$
- Common mistake: using $\Rightarrow$ as the main connective with $\exists$:

$$\exists x\ At(x, AUT) \Rightarrow Smart(x)$$

is true if there is anyone who is not at AUT!

# Properties of quantifiers

- $\forall x\ \forall y \equiv \forall y\ \forall x \equiv \forall x,y$
- $\exists x\ \exists y \equiv \exists y\ \exists x \equiv \exists x,y$
- $\exists x\ \forall y$ 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) | $\neg\exists x\ \neg$Likes(x, IceCream) |
| $\exists x$ Likes(x, Broccoli) | $\neg\forall x\ \neg$Likes(x, Broccoli) |

# Properties of quantifiers

- Because ∀ is really a conjunction over the universe of objects and ∃ is a disjunction, it should not be surprising that they obey De Morgan's rules.
- The De Morgan rules for quantified and unquantified sentences are as follows:

$$
\begin{array}{lcl}
\forall x \;\; \neg P & \equiv & \neg \exists x \;\; P \\
\neg \forall x \;\; P & \equiv & \exists x \;\; \neg P \\
\forall x \;\; P & \equiv & \neg \exists x \;\; \neg P \\
\exists x \;\; P & \equiv & \neg \forall x \;\; \neg P
\end{array}
\qquad\qquad
\begin{array}{lcl}
\neg (P \vee Q) & \equiv & \neg P \wedge \neg Q \\
\neg (P \wedge Q) & \equiv & \neg P \vee \neg Q \\
P \wedge Q & \equiv & \neg (\neg P \vee \neg Q) \\
P \vee Q & \equiv & \neg (\neg P \wedge \neg Q)
\end{array}
$$

# 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., definition of Sibling in terms of Parent

$$\forall x, y \; Sibling(x, y) \Leftrightarrow [\neg(x = y) \land \exists m, f \; \neg(m = f) \land Parent(m, x) \land Parent(f, x) \land Parent(m, y) \land Parent(f, y)]$$

# Using FOL: Kinship domain example

- Objects: people
- Functions: Mother, Father
- Predicates:
  - Unary: Male, Female
  - Binary: Parent, Sibling, Brother , Sister , Child , Daughter, Son, Spouse, Wife, Husband, Grandparent , Grandchild , Cousin, Aunt, and Uncle

# Using FOL: Kinship domain example

- $\forall$ m, c Mother(c) = m $\Leftrightarrow$ (Female(m) $\wedge$ Parent(m, c))
  - One's mother is one's female parent
- $\forall$ x, y Sibling(x, y) $\Leftrightarrow$ Sibling(y, x)
  - "Sibling" is symmetric
- $\forall$ x, y Brother(x, y) $\Leftrightarrow$ Sibling(x, y)
  - Brothers are siblings
- $\forall$ w, h Husband(h,w) $\Leftrightarrow$ Male(h) $\wedge$ Spouse(h,w)
- $\forall$ x Male(x) $\Leftrightarrow$ $\neg$Female(x)
- $\forall$ p, c Parent(p, c) $\Leftrightarrow$ Child (c, p)
- $\forall$ g, c Grandparent(g, c) $\Leftrightarrow$ $\exists$p Parent(g, p) $\wedge$ Parent(p, c)
- $\forall$ x, y Sibling(x, y) $\Leftrightarrow$ x $\neq$ y $\wedge$ $\exists$p Parent(p, x) $\wedge$ Parent(p, y)

# An alternative semantics?

- We believe that Richard has two brothers, John and Geoffrey
- Wrong assertion
  - $Brother(John, Richard) \land Brother(Geoffrey, Richard)$
    - This assertion is true in a model where Richard has only one brother
      - Add $John \neq Geoffrey$
    - The sentence doesn't rule out models in which Richard has many more brothers besides John and Geoffrey
- Correct assertion
  - $Brother(John, Richard) \land Brother(Geoffrey, Richard) \land John \neq Geoffrey$
    $\land \forall x\ Brother(x, Richard) \Rightarrow (x = John \lor x = Geoffrey)$
- Humans may make mistakes in translating their knowledge into first-order logic, resulting in unintuitive behaviors from logical reasoning systems that use the knowledge.

# Database semantics

- Can we devise a semantics that allows a more straightforward logical expression?
  - Unique-names assumption
    - Every constant symbol refer to a distinct object
  - Closed-world assumption
    - Atomic sentences not known to be true are in fact false
  - Domain closure
    - Each model contains no more domain elements than those named by the constant symbols
- We call this database semantics to distinguish it from the standard semantics of first-order logic

# Assertions & queries in FOL KBs

- **Assertions:** sentences added to KB using TELL
  - TELL(KB, King(John))
  - TELL(KB, Person(Richard))
  - TELL(KB, $\forall$ x King(x) $\Rightarrow$ Person(x))
- **Queries** or **goals**: questions asked from KB using ASK
  - ASK(KB, King(John))
  - ASK(KB, $\exists$ x Person(x))
- **Substitution** or **binding list**: AskVars
  - In KB of Horn clauses, every way of making the query true will bind the variables to specific values

    AskVars(KB, Person(x))    results = {{x/Richard} , {x/John}}
  - If KB contains King(John) $\vee$ King(Richard), there is no binding while

    ASK(KB, $\exists$ x king(x)) is true

# Using FOL: Set domain example

- Objects: sets, elements
- Functions: $s_1 \cap s_2$, $s_1 \cup s_2$, $\{x|s\}$
- Predicates:
  - Unary: Set
  - Binary: $x \in s$, $s_1 \subseteq s_2$

# Using FOL: Set domain example

- $\forall$ s Set(s)$\Leftrightarrow$ (s = {})$\vee$ ($\exists$ x,s2 Set(s2) $\wedge$ s = {x|s2})
- $\neg\exists$ x,s {x|s} = {}
- $\forall$ x,s x $\in$ s $\Leftrightarrow$ s = {x|s}
- $\forall$ x,s x $\in$ s $\Leftrightarrow$ [$\exists$ y,s2 (s = {y|s2} $\wedge$ (x = y $\vee$ x $\in$ s2))]
- $\forall$ s1,s2 s1 $\subseteq$ s2 $\Leftrightarrow$ ($\forall$ x x $\in$ s1 $\Rightarrow$ x $\in$ s2)
- $\forall$ s1,s2 s1 = s2 $\Leftrightarrow$ (s1 $\subseteq$ s2 $\wedge$ s2 $\subseteq$ s1 )
- $\forall$ x, s1,s2 x $\in$ (s1 $\cap$ s2) $\Leftrightarrow$ (x $\in$s1 $\wedge$ x $\in$ s2 )
- $\forall$ x, s1,s2 x $\in$ (s1 $\cup$ s2) $\Leftrightarrow$ (x $\in$s1 $\vee$ x $\in$ s2 )

# Using FOL: Wumpus world example

- Environment
  - Objects:
    - Pairs identifying squares [i, j], Agent, Wumpus
  - Relations:
    - Pit, Adjacent, Breezy, Stenchy, Percept, Action, At, HaveArrow
- Perceptions:
  - perceives a smell, a breeze, and glitter at t=5

    Percept([Stench, Breeze, Glitter , None, None], 5)
- Actions:
  - Turn(Right), Turn(Left), Forward , Shoot , Grab, Climb

# 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 BestAction(a,5))
- I.e., does the KB entail some best action at t=5?
- Answer: Yes, {a/Shoot} $\leftarrow$ substitution (binding list)

# Knowledge base for the Wumpus world

- Perceptions implies facts about current state
  - $\forall$ 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)
- Simple reflex behavior
  - $\forall$ t Glitter (t) $\Rightarrow$ BestAction(Grab, t)

# Knowledge base for the Wumpus world

- Samples of rules:
  - $\forall$ 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))
  - At(Agent, s, t)
  - $\exists$x,y $\forall$t At(Wumpus, [x, y], t)
  - $\forall$ x, s1, s2, t At(x, s1, t) $\wedge$ At(x, s2, t) $\Rightarrow$ s1 = s2
  - $\forall$ s, t At(Agent, s, t) $\wedge$ Breeze(t) $\Rightarrow$ Breezy(s)
  - $\forall$ s Breezy(s) $\Leftrightarrow$ $\exists$r Adjacent (r, s) $\wedge$ Pit(r)
- One successor-state axiom for each predicate
  - $\forall$t HaveArrow(t + 1) $\Leftrightarrow$ (HaveArrow(t) $\wedge\neg$Action(Shoot, t))