Chapter 8

First Order Logic

Introduction

- Propositional logic has limited expressive power unlike natural language
- Example: cannot say "pits cause breezes in adjacent squares“, except by writing one sentence for each square
  - $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
- Question: How can we write one sentence only that can be applied to a group of objects?

First order logic (FOL)

Examples of things we can say:

- All men are mortal:
  - $\forall x \; Man(x) \;\Rightarrow\; Mortal(x)$
- Everybody loves somebody
  - $\forall x \; \exists y \; Loves(x, y)$
- The meaning of the word “above”
  - $\forall x \; \forall y \; above(x, y) \Leftrightarrow (on(x, y) \;\vee\; \exists z \; (on(x, z) \wedge above(z, y))$

First Order Logic

- AKA first-order predicate calculus
- First-order logic assumes the world contains:

1. Objects (noun): people, houses, numbers, colors, …
2. Predicates/ Relations (verbs that relate objects) (return T or F):
    - Unary: red, round, prime (properties)
    - N-ary: brother of, bigger than, part of
3. Functions (verbs that relate objects) (return an object):
    - Example: Sqrt, Plus, Father
    - A function is a relation that has one value for one input

Models for first-order logic

- **Models**: the formal structures that constitute the possible worlds under consideration
- **Domain** of a model is the set of objects or **domain elements** it contains

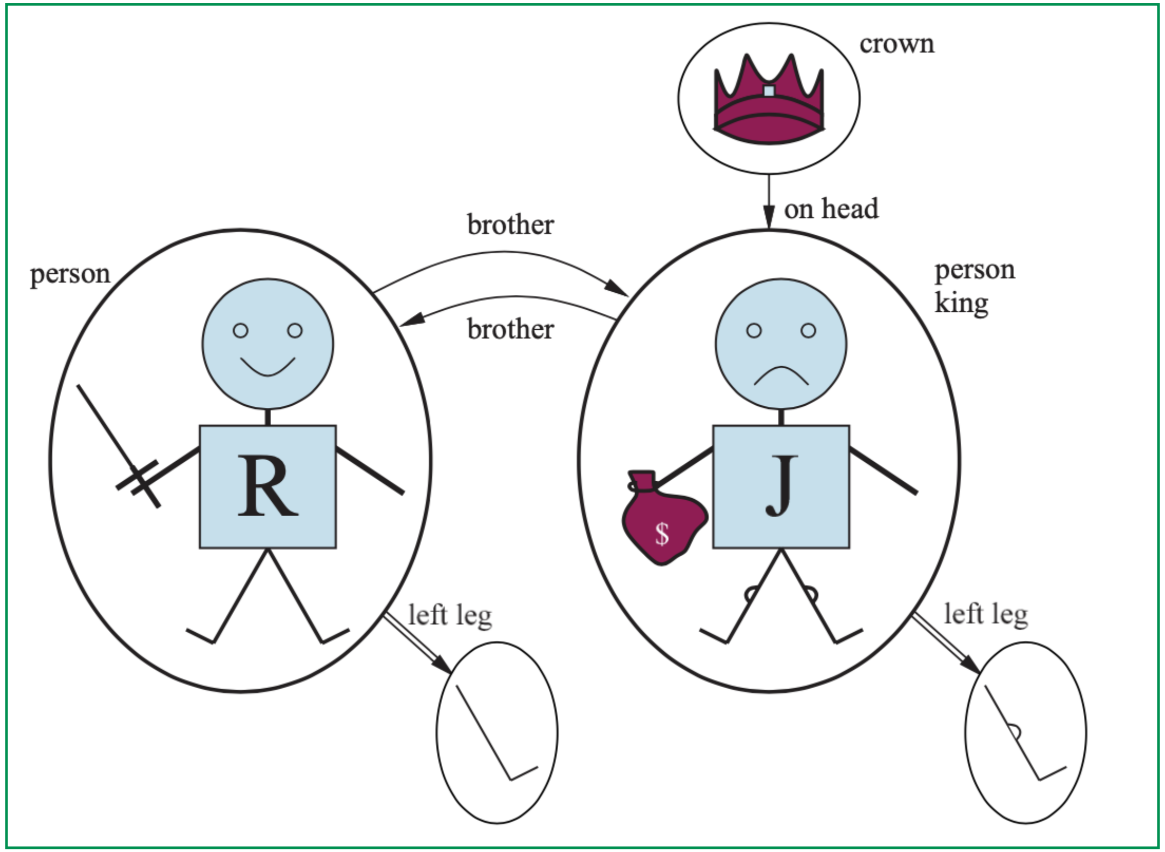


Figure 8.2 A model containing five objects, two binary relations (brother and on-head), three unary relations (person, king, and crown), and one unary function (left-leg).

Syntax of FOL: Basic elements

Element	Example
Constants	John, 2, Black
Predicates	Brother, >
Functions	Sqrt, Father
Variables	x, y, a, b
Connectives	$\neg, \Rightarrow, \wedge, \vee, \Leftrightarrow$
Equality	$:=$
Quantifiers	$\forall, \exists$

Syntax of FOL: Atomic sentences

- **Term**: is a logical expression that refers to an object
  - Is a function $f(term_1, \dots, term_n)$ or constant or variable
  - Example: constant term: John, function: LeftLeg(John)

$$
\begin{array}{lll}
Term & \rightarrow & Function(Term, \dots) \\
 & | & Constant \\
 & | & Variable
\end{array}
$$

- **Atomic sentence**: is a predicate of terms $pred(term_1, \dots, term_n)$ or $term_1 = term_2$
  - $Man(x)$
  - $Brother(John, Richard)$
  - $>(3, 1)$
  - $x = y$
  - $Father(Ali) = Ahmed$

Atomic sentences

$$AtomicSentence \rightarrow Predicate \mid Predicate(Term, \dots) \mid Term = Term$$

Syntax of FOL: Complex sentences

- **Complex sentences:** are made from atomic sentences using
  1. Connectives: $\neg S, S_1 \wedge S_2, S_1 \vee S_2, S_1 \Rightarrow S_2, S_1 \Leftrightarrow S_2$
  2. Quantifiers: Universal quantification (∀), Existential quantification (∃)
     - Example: $\forall x \ \ Man(x) \Rightarrow Mortal(x)$

$$
\begin{aligned}
ComplexSentence \ \rightarrow \ & (\ Sentence \ ) \\
| \ & \neg \ Sentence \\
| \ & Sentence \wedge Sentence \\
| \ & Sentence \vee Sentence \\
| \ & Sentence \ \Rightarrow \ Sentence \\
| \ & Sentence \ \Leftrightarrow \ Sentence \\
| \ & Quantifier \ Variable, \ldots \ Sentence
\end{aligned}
$$

- Examples:
  - $> (1,2) \vee \ \leq \ (1,2)$
  - $> (1,2) \wedge \neg \ > (1,2)$
  - $Sibling(John, Richard) \Rightarrow Sibling(Richard, John)$

Syntax of FOL: Quantifiers

- Quantifiers: Universal (∀) and Existential (∃)
- Allow us to express properties of collections of objects instead of enumerating objects by name
- Universal (∀): “for all”:
  - $\forall < variables > < sentence >$
  - $\forall x \quad At(x, KSU) \Rightarrow Smart(x)$
- Existential (∃) : “there exists”
  - $\exists < variables > < sentence >$
  - $\exists\, x \quad At(x, PNU) \land Smart(x)$

Syntax of FOL: summary

Sentence	$\rightarrow$	*AtomicSentence* | *ComplexSentence*
AtomicSentence	$\rightarrow$	*Predicate* | *Predicate*(*Term*, ...) | *Term* = *Term*
ComplexSentence	$\rightarrow$	(*Sentence*)
	|	$\neg$ *Sentence*
	|	*Sentence* $\land$ *Sentence*
	|	*Sentence* $\lor$ *Sentence*
	|	*Sentence* $\Rightarrow$ *Sentence*
	|	*Sentence* $\Leftrightarrow$ *Sentence*
	|	*Quantifier* *Variable*, ... *Sentence*
Term	$\rightarrow$	*Function*(*Term*, ...)
	|	*Constant*
	|	*Variable*
Quantifier	$\rightarrow$	$\forall$ | $\exists$
Constant	$\rightarrow$	A | X_1 | *John* | $\cdots$
Variable	$\rightarrow$	a | x | s | $\cdots$
Predicate	$\rightarrow$	*True* | *False* | *After* | *Loves* | *Raining* | $\cdots$
Function	$\rightarrow$	*Mother* | *LeftLeg* | $\cdots$
OPERATOR PRECEDENCE	:	$\neg, =, \land, \lor, \Rightarrow, \Leftrightarrow$

Syntax of FOL: Quantifiers

- $\Rightarrow$ is the main connective with ($\forall$)
- Common Mistake:
  - $\forall x \quad At(x, KSU) \land Smart(x)$
  - True when everyone is at KSU and everyone is smart

- $\land$ is the main connective with ($\exists$)
- Common Mistake:
  - $\exists\, x \quad At(x, PNU) \Rightarrow Smart(x)$
  - It is also true for anyone not in PNU!

Syntax of FOL: Properties of quantifiers

- $\forall\, x\, \forall\, y$ is the same as $\forall\, y\, \forall\, x$
- $\exists\, x\, \exists\, y$ is the same as $\exists\, y\, \exists\, x$
- $\exists\, x\, \forall\, y$ is not the same as $\forall\, y\, \exists\, x$:
  - $\exists\, x\, \forall\, y \quad Loves(x, y)$

    "There is a person who loves everyone in the world"
  - $\forall\, y\, \exists\, x \quad Loves(x, y)$

    "Everyone in the world is loved by at least one person"
- Quantifier duality: each can be expressed using the other
  - $\forall\, x \quad Likes(x, IceCream) \equiv \neg\, \exists\, x \quad \neg Likes(x, IceCream)$
  - $\exists\, x \quad Likes(x, Broccoli) \equiv \neg\, \forall\, x \quad \neg Likes(x, Broccoli)$

$$\forall x\ \neg P \equiv \neg\exists x\ P$$
$$\neg\forall x\ P \equiv \exists x\ \neg P$$
$$\forall x\ P \equiv \neg\exists x\ \neg P$$
$$\exists x\ P \equiv \neg\forall x\ \neg P$$

Using FOL: The kinship domain

- Objects in the kinship domain are people
- Two unary predicates: $Male$ and $Female$
- Kinship relations are represented by binary predicates: $Parent$, $Sibling$, $Brother$, $Sister$, $Child$, $Daughter$, Son, $Spouse$, $Wife$, $Husband$, $Grandparent$, $Grandchild$, $Cousin$, $Aunt$, and $Uncle$.
- Use functions for $Mother$ and $Father$, because every person has exactly one of each of these

Using FOL: The kinship domain

- One's husband is one's male spouse:

$$\forall w, h\ Husband(h, w) \ \Leftrightarrow\ Male(h) \land Spouse(h, w)$$

- Male and female are disjoint categories:

$$\forall x\ Male(x) \ \Leftrightarrow\ \neg Female(x)$$

- Parent and child are inverse relations:

$$\forall p, c\ Parent(p, c) \ \Leftrightarrow\ Child(c, p)$$

- A grandparent is a parent of one's parent:

$$\forall g, c\ Grandparent(g, c) \ \Leftrightarrow\ \exists p\ Parent(g, p) \land Parent(p, c)$$

- A sibling is another child of one's parents:

$$\forall x, y\ Sibling(x, y) \ \Leftrightarrow (x \neq y) \land \exists p\ Parent(p, x) \land Parent(p, y)$$

Chapter 9

Inference in First Order Logic

Inference in FOL

- Purpose of inference: $KB \vDash \alpha$?
- First Approach: **Reduce FOL to PL and then apply PL inference**
- Inference by model checking is in general impossible in FOL: the models are generally infinite or at least extremely large.
- The KB propositionalized is not equivalent to the original KB, but entailment is preserved.
- Every FOL KB can be propositionalized so as to preserve entailment
- Inference by reduction to PL: propositionalize KB and query, apply inference rules, return result.

Converting FOL to PL: Substitution

- Substitution: Given a sentence α and binding list σ, the result of applying the substitution σ to α is denoted by $Subst(\sigma, \alpha)$
- Example:

$$\sigma = \{x/Ali, y/Fatima\}$$
$$\alpha = Likes(x, y)$$

$$Subst(\{x/Ali, y/Fatima\}, Likes(x, y)) = Likes(Ali, Fatima)$$

Converting FOL to PL: Converting ∀

- Universal instantiation (UI): given a universal generalization (an ∀ sentence), the rule allows you to infer any instance of that generalization.
- Substitute the variable in a universally quantified sentence by a ground term. A ground term is a term with no variables.
- Example: $\forall x\ King(x) \land Greedy(x) \Rightarrow Evil(x)$ yields:
    - $King(John) \land Greedy(John) \Rightarrow Evil(John)$
    - $King(Richard) \land Greedy(Richard) \Rightarrow Evil(Richard)$
    - $King(Father(John)) \land Greedy(Father(John)) \Rightarrow Evil(Father(John))$
- UI can be applied several times to add new sentences

Converting FOL to PL: Converting ∃

- Existential instantiation (EI): For any sentence α, variable v, and constant symbol k that does not appear elsewhere in the knowledge base (k is called a Skolem): replace v by k
- Example: $\exists x\ Crown(x)\ \wedge\ OnHead(x, John)$ yields:
    - $Crown(C1)\ \wedge\ OnHead(C1, John)$
    - Provided $C1$ is a new constant symbol, called a Skolem constant
- EI can be applied once to replace the existential sentence

Example: Reduction to propositional inference

KB

$\forall x\ King(x)\ \land\ Greedy(x) \Rightarrow Evil(x)$
$King(John)$
$Greedy(John)$
$Brother(Richard, John)$

- Instantiating the universal sentence in all possible ways, we have:

New KB

$King(John)\ \land Greedy(John) \Rightarrow Evil(John)$
$King(Richard)\ \land\ Greedy(Richard) \Rightarrow Evil(Richard)$ → irrelevant substitution
$King(John)$
$Greedy(John)$
$Brother(Richard, John)$

- The new KB is propositionalized.

Propositionalization

- Propositionalization can be made completely general: every FOL KB and query can be propositionalized in such a way that entailment is preserved.
- Problem: when the KB includes a function symbol, the set of possible ground-term substitutions is infinite!
- Example: KB contains $Father$ symbol, then infinitely many nested terms ($Father(Father(Father(John)))$) can be constructed
- Propositional algorithms will have difficulty with an infinitely large set of sentences

Inference in FOL: Inference rules

- Second approach: Instead of translating KB to PL, we can make the inference rules work in FOL.
- For example, given the KB, can we prove $Evil(John)$?

$$\forall x\ King(x) \land Greedy(x) \Rightarrow Evil(x)$$
$$King(John)$$
$$Greedy(John)$$

- The inference that John is evil works like this:
  - Find some x such that x is a $king$ and x is $greedy$,
  - And then infer that x is $evil$.
- It is intuitively clear that we can substitute $\{x/John\}$ and obtain that $Evil(John)$

Inference in FOL: Inference rules

- What if we have:

$\forall\, x\; King(x)\; \land\; Greedy(x) \Rightarrow Evil(x)$
$King(John)$
$\forall\, y\; Greedy(y)$

- It is intuitively clear that we can substitute $\{x/John, y/John\}$ and obtain that $Evil(John)$
  - $King(x)$ is unified with $King(john)$
  - $Greedy(John)$ is unified with $Greedy(y)$

Inference in FOL: Generalized Modus Ponens

- For atomic sentences p_i, p_i', q, and substitution θ, such that $SUBST(\theta, p_i) = SUBST(\theta, p_i')$, for all i:

$$\frac{p_1', p_2', \ldots, p_n', \quad (p_1 \wedge p_2 \wedge \cdots \wedge p_n \Rightarrow q)}{SUBST(\theta, p)}$$

- All variables are assumed universally quantified.

$\forall\, x\ King(x)\ \wedge\ Greedy(x) \Rightarrow Evil(x)$
$King(John)$
$\forall\, y\ Greedy(y)$

p_1' is $King(John)$	p_1 is $King(x)$
p_2' is $Greedy(y)$	p_2 is $Greedy(x)$
θ is $\{x/John, y/John\}$	q is $Evil(x)$
$Subst(\theta, q)$ is $Evil(John)$	

Inference in FOL: Unification

- The UNIFY algorithm takes two sentences and returns a **unifier** for them if one exists:

$$UNIFY(p, q) = \theta \text{ where } SUBST(\theta, p) = SUBST(\theta, q)$$

- We can make the inference if we can find a substitution such that King(x) and Greedy(x) match King(John) and Greedy(y): {x/John,y/John} works

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, OJ)$	$\{x/OJ, y/John\}$
$Knows(John, x)$	$Knows(y, Mother(y))$	$\{y/John, x/Mother(John)\}\}$
$Knows(John, x)$	$Knows(x, OJ)$	$\{fail\}$

Inference in FOL: Resolution

$$\frac{l_1 \vee \cdots \vee l_k, \qquad m_1 \vee \cdots \vee m_n}{Subst(\theta, l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)}$$

where $\theta = Unify(l_i, \neg m_j)$

- Example: $\frac{\neg Rich(x) \vee Unhappy(x)\,,\ Rich(Ken)}{Unhappy(Ken)}$, with $\theta = \{x/Ken\}$

- Apply resolution steps to $CNF(KB \wedge \neg\alpha)$; complete for FOL

Students

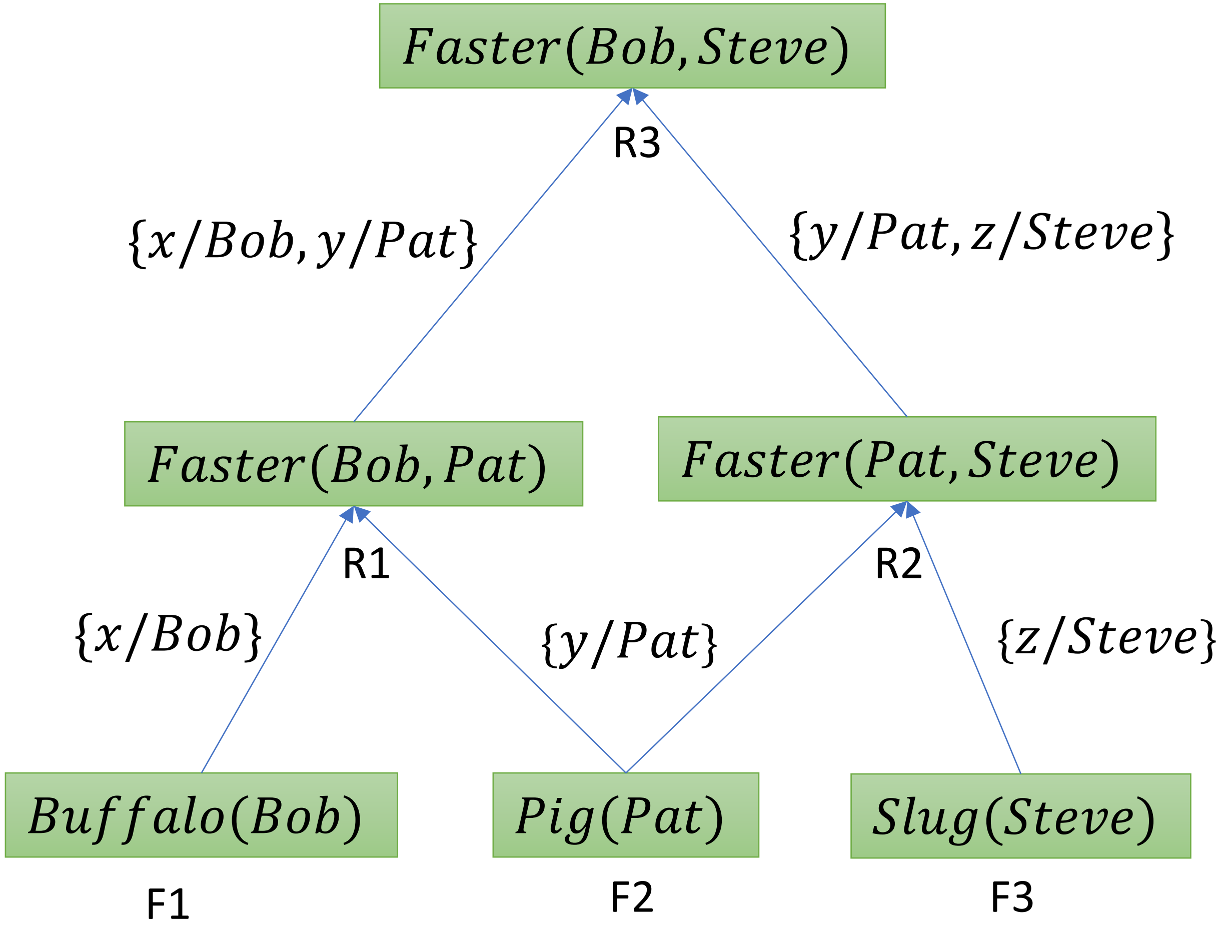
Inference in FOL: Forward chaining

- When a new fact P is added to the KB:

```
For each rule s.t. P unifies with a premise
  if the other premises are known then
    add the conclusion to the KB
  continue chaining
```

- Forward chaining is **data-driven**, for example, inferring conclusions from incoming percepts

Forward chaining example



Rules

1. $Buffalo(x) \land Pig(y) \implies Faster(x, y)$
2. $Pig(y) \land Slug(z) \implies Faster(y, z)$
3. $Faster(x, y) \land Faster(y, z) \implies Faster(x, z)$

Facts

1. $Buffalo(Bob)$
2. $Pig(Pat)$
3. $Slug(Steve)$

New facts

4. $Faster(Bob, Pat)$
5. $Faster(Pat, Steve)$
6. $Faster\ (Bob, Steve)$

Inference in First Order Logic: Backward chaining

- Backward chaining starts with a hypothesis (query) and work backwards, according to the rules in the knowledge base until reaching confirmed findings or facts.

R1. $Pig(y) \land Slug(z) \Rightarrow Faster\ (y,z)$
R2. $Slimy(z) \land Creeps(z) \Rightarrow Slug(z)$
F1. $Pig(Pat)$
F2. $Slimy(Steve)$
F3. $Creeps(Steve)$