First Order Logic

Forest Agostinelli
University of South Carolina

Topics Covered in This Class

- **Part 1: Search**
  - Pathfinding
    - Uninformed search
    - Informed search
  - Adversarial search
  - Optimization
    - Local search
    - Constraint satisfaction
- **Part 2: Knowledge Representation and Reasoning**
  - Propositional logic
  - First-order logic
  - Prolog
- **Part 3: Knowledge Representation and Reasoning Under Uncertainty**
  - Probability
  - Bayesian networks
- **Part 4: Machine Learning**
  - Supervised learning
    - Inductive logic programming
    - Linear models
    - Deep neural networks
    - PyTorch
  - Reinforcement learning
    - Markov decision processes
    - Dynamic programming
    - Model-free RL
  - Unsupervised learning
    - Clustering
    - Autoencoders

Outline

- Motivation
- Objects
- Functions
- Predicates
- Quantifiers
- Examples

Ontological Commitment

- What is being assumed about the nature of reality
- Propositional Logic
    - The world consists of facts
- First-order logic
    - The world consists of objects
    - These objects have certain relations among them that do or do not hold
    - In the simplest case, a relation amongst no objects is the same as a proposition
        - “It is raining”
        - $IsRaining$
    - A relation amongst a single object is a property
        - “The cat is brown”
        - $Brown(Cat)$
    - A relation amongst two or more objects
        - “John and Richard are brothers”
        - $Brother(John, Richard)$

Motivation

- "The cat is brown and the sofa is brown and the cat is on the sofa"
- Propositional Logic
  - $cb \wedge sb \wedge cos$
- First-Order Logic
  - $Brown(Cat) \wedge Brown(Sofa) \wedge On(Cat, Sofa)$
- First order logic has objects and relations (predicates)
- Relations can have no arguments (propositions) be unary (properties) or n-ary (relations between objects)

Motivation

- Imagine a knowledge base that describes sets
- How do we add statements describing intersection?
  - If the world has 3 sets and integers ranging from 0 to 100
- Propositional logic
  - $1 \in (s_1 \cap s_2) \leftrightarrow (1 \in s_1 \wedge 1 \in s_2)$
  - $1 \in (s_1 \cap s_3) \leftrightarrow (1 \in s_1 \wedge 1 \in s_3)$
  - $1 \in (s_2 \cap s_3) \leftrightarrow (1 \in s_2 \wedge 1 \in s_3)$
  - $1 \in (s_2 \cap s_1) \leftrightarrow (1 \in s_2 \wedge 1 \in s_1)$
  - …
- First-order logic
  - $\forall\, x, s_1, s_2\;\; x \in (s_1 \cap s_2) \leftrightarrow (x \in s_1 \wedge x \in s_2)$
- First-order logic uses quantifiers and variables to make statements about entire collections of objects without mentioning a particular object

Motivation

- "All brown cats blend in with brown sofas"
- "There exists a cat that is not brown"
- First-Order Logic
  - $\forall x, y\ Brown(x) \wedge Cat(x) \wedge Brown(y) \wedge Sofa(y) \rightarrow Blends(x, y)$
  - $\exists x\ Cat(x) \wedge \neg Brown(x)$

Motivation

- "All humans are mortal"
- Propositional logic
  - $h_1 \wedge m_1$
  - $h_2 \wedge m_2$
  - $h_3 \wedge m_3$
  - …
- First Order Logic
  - $\forall x\ Human(x) \rightarrow Mortal(x)$

Motivation

- "Every dog wags its tail"
- $\forall x\ Dog(x) \rightarrow Wag(x, Tail(x))$
- First-order logic uses functions that return objects

Overview

- $\forall s \; Smelly(s) \rightarrow Adjacent(Home(Wumpus), s)$

Quantifier	Variable	Predicate	Function	Constant
Universal Existential		Expresses relations. Returns true or false	Maps a tuple of objects to an object	Objects in the world
∀	s	Smelly Adjacent	Home	Wumpus

First-Order Logic (First-Order Predicate Logic)

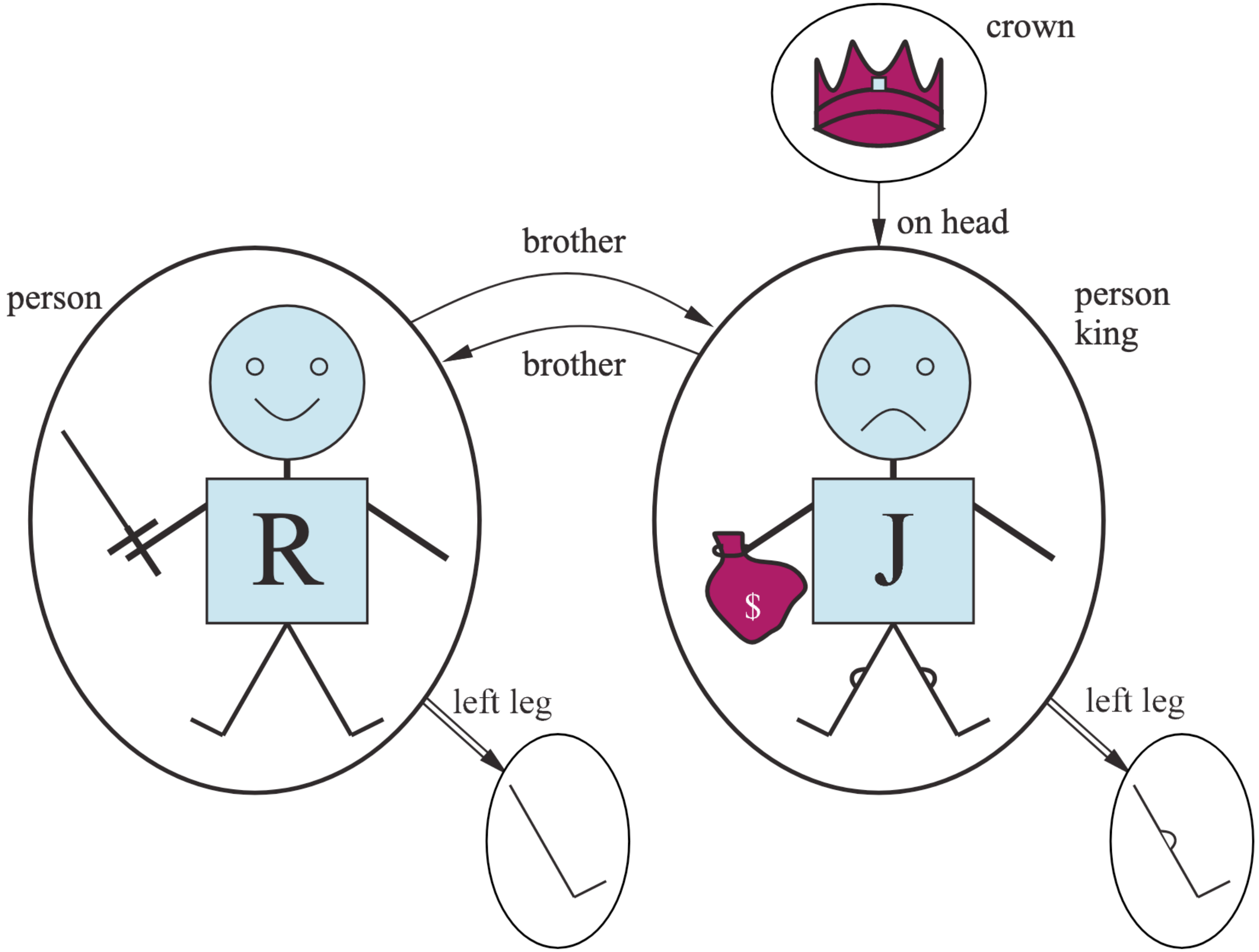
- First-order logic (FOL) allow for objects, relations (predicates) amongst objects, and quantifiers to express properties of many objects without having to explicitly enumerate all objects
  - “First-order” because quantified variables represent objects
  - “Predicate calculus” because it quantifies over predicates on objects

Second-Order Logic

- First-order logic
  - Variables represent objects
  - E.g. we an state that a relationship is transitive
  - $\forall$ x, y, z BrotherOf(x,y) $\wedge$ BrotherOf(y,z) => BrotherOf(x,z)
- Second-order logic
  - Variables represent predicates and functions
  - E.g. we can define transitive
  - $\forall$ P, x, y, z Transitive(P) <=>(P(x,y) $\wedge$ P(y,z) => P(x,z))
  - Second-order logic is beyond the scope of this class

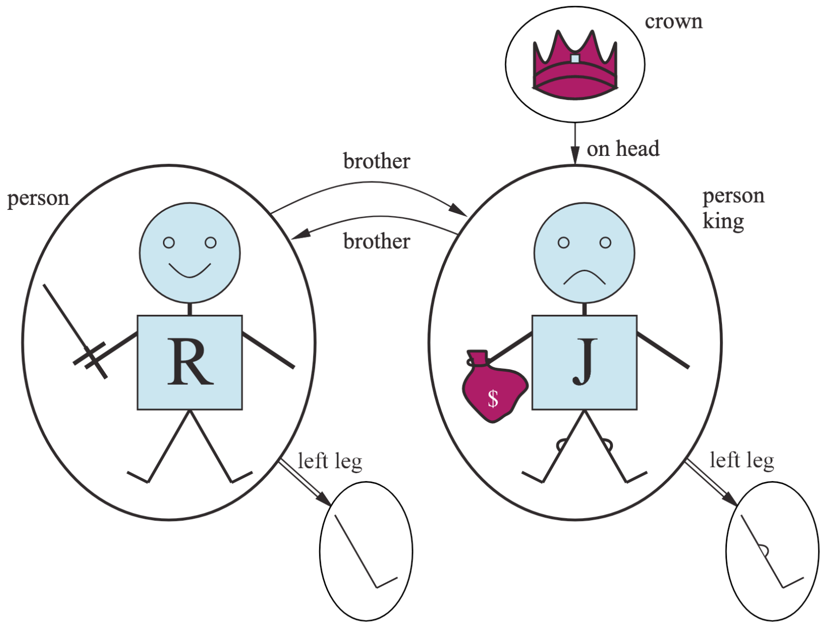
Example

- R=Richard
- J=John



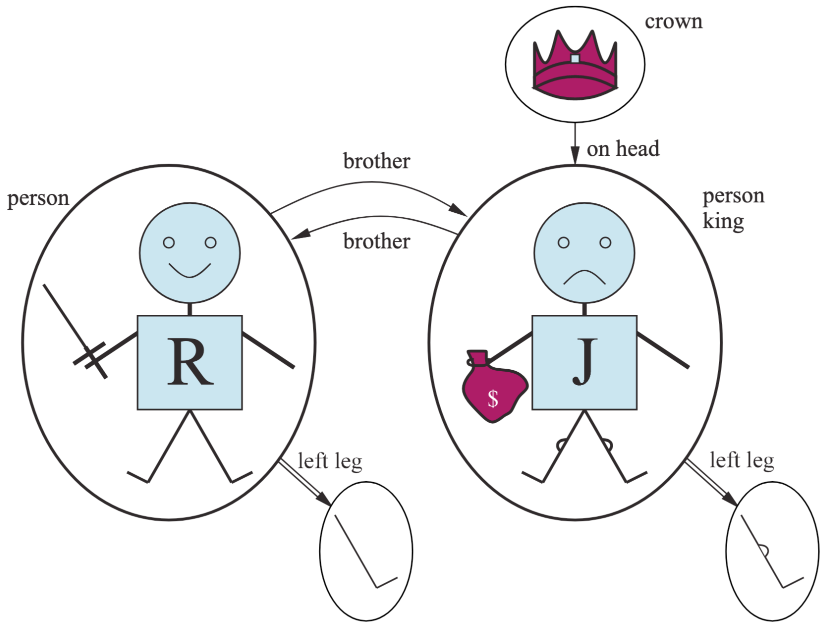
Objects

- Objects are nouns
- Objects: Richard, John, crown, Richard's left leg, John's left leg



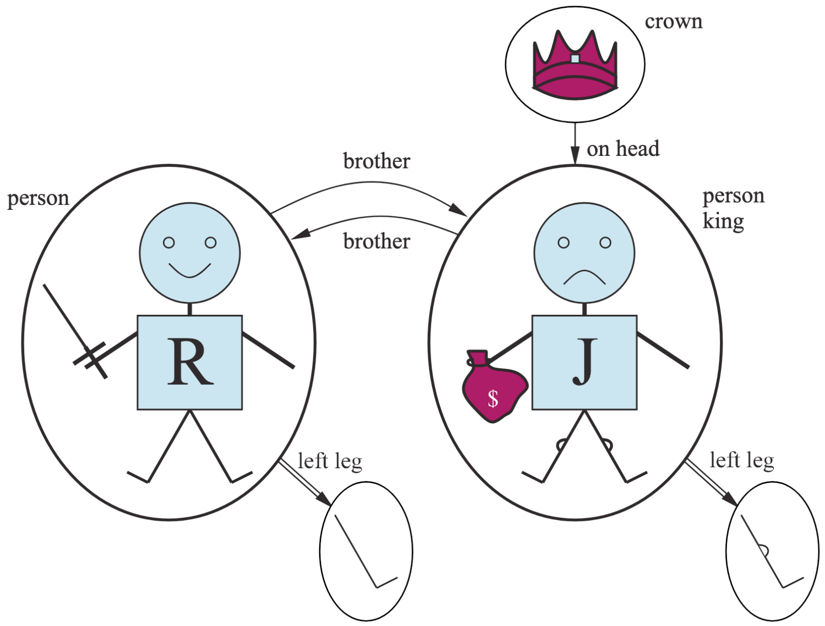
Functions

- Objects do not have to be listed explicitly
- Functions return objects
- Richard's left leg and John's left leg are not given their own name
  - $LeftLeg(John)$
  - $LeftLeg(Crown)$
- What about $LeftLeg(Crown)$?
  - FOL requires total functions: there must be an output for every input tuple
  - To handle this, one can map $LeftLeg(Crown)$ to some "invisible" value (i.e. NULL)



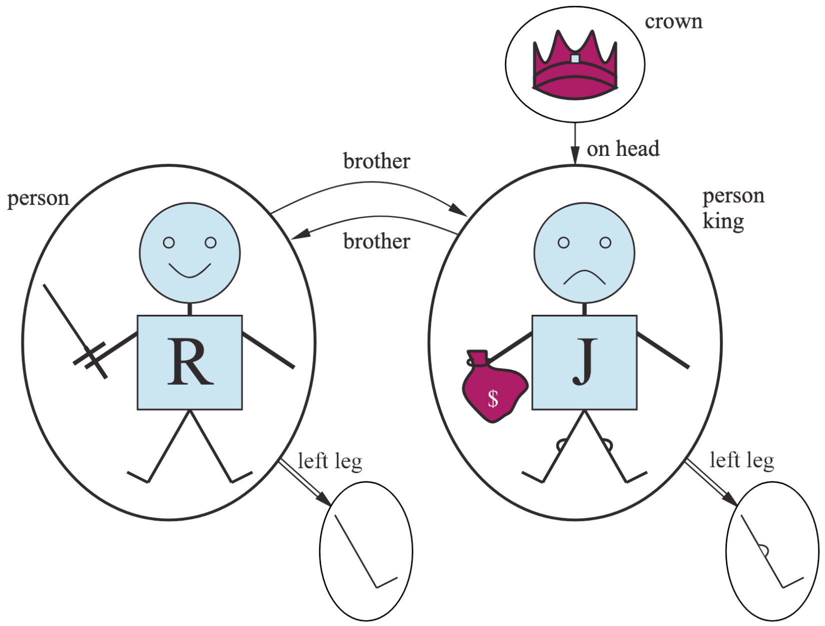
Predicates

- Predicates express relationships among objects
- Returns true or false, depending on its arguments
  - $Brother(John, Richard)$ <- True
  - $Brother(John, LeftLeg(Richard))$ <- False
  - $OnHead(Crown, John)$ <- True
- Predicates with one argument are referred to as properties
  - $Purple(Crown)$
- Predicates with zero arguments are the same as propositions from propositional logic
  - $IsMonarchy$
- FOL can also use equality
  - $Father(John) = Henry$



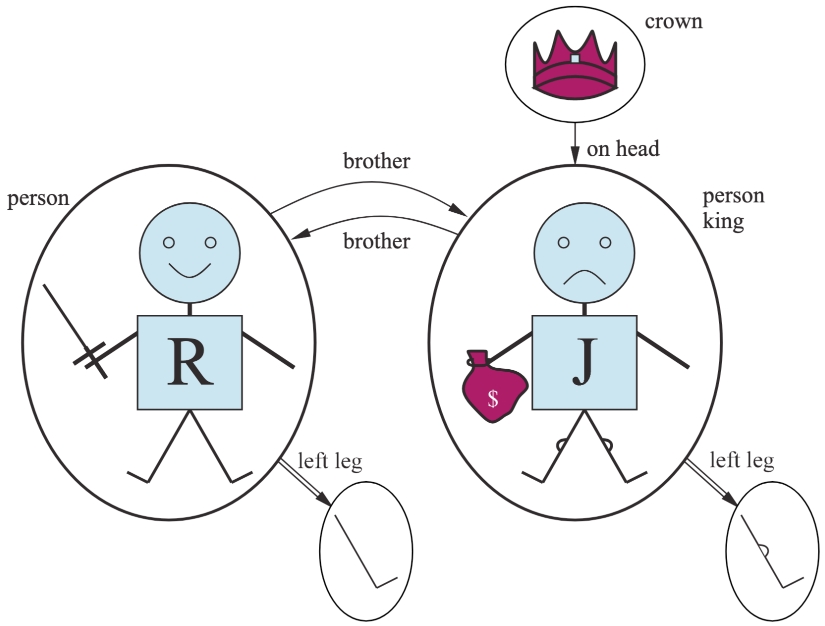
Quantifiers

- Quantifiers express properties across different objects
- Instead of enumerating all possible objects, quantifiers use variables
- Universal quantifiers $\forall$
  - Conjunction (AND) over all objects
- Existential quantifiers $\exists$
  - Disjunction (OR) over all objects
- "All kings are persons"
  - $\forall x\ King(x) \rightarrow Person(x)$
- "John has a crown on his head"
  - $\exists x\ Crown(x) \wedge OnHead(x, John)$



Hints for Quantifiers

- "All kings are persons"
  - $\forall x\ King(x) \rightarrow Person(x)$
  - This is too strong: $\forall x\ King(x) \wedge Person(x)$
- "John has a crown on his head"
  - $\exists x\ Crown(x) \wedge OnHead(x, John)$
  - This is too weak: $\exists x\ Crown(x) \rightarrow OnHead(x, John)$



Nested Quantifiers

The order of "unlike" quantifiers is important.

Like nested variable scopes in a programming language.

Like nested ANDs and ORs in a logical sentence.

$\forall x \exists y$ Loves(x,y)

- For everyone ("all x") there is someone ("exists y") whom they love.
- There might be a different y for each x (y is inside the scope of x)

$\exists y \forall x$ Loves(x,y)

- There is someone ("exists y") whom everyone loves ("all x").
- Every x loves the same y (x is inside the scope of y)

Clearer with parentheses: $\exists y\ (\forall x \quad \text{Loves}(x,y))$

The order of "like" quantifiers does not matter.

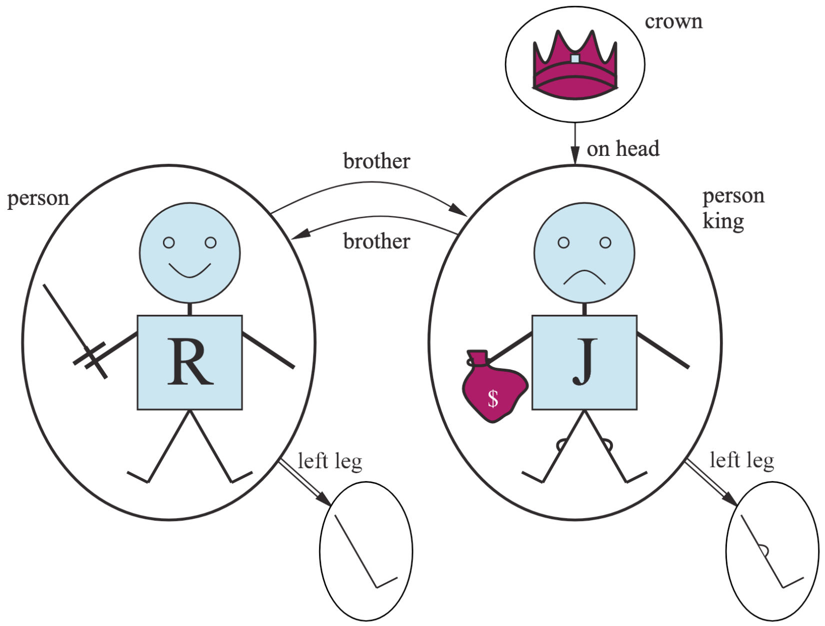
Like nested ANDs and ANDs in a logical sentence

$\forall x\ \forall y\ P(x, y) \equiv \forall y\ \forall x\ P(x, y)$

$\exists x\ \exists y\ P(x, y) \equiv \exists y\ \exists x\ P(x, y)$

Terms

- A term is a logical expression that refers to an object
  - Constant symbols
  - Functions
  - Variables



First-Order Logic Grammar

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

Conventions and Assumptions

- Each quantifier has a unique variable
  - The variable belongs to the innermost quantifier that mentions it
  - This can be confusing
  - $\forall x \left(Crown(x) \vee (\exists x\, Brother(Richard, x)) \right)$ <- confusing
  - $\forall x \left(Crown(x) \vee (\exists z\, Brother(Richard, z)) \right)$ <- better
- Unique–names assumption
  - Every constant symbol refers to a distinct object
  - For example, John and Richard must be two different objects

De Morgan's Rule for Quantifiers

- Universal quantifiers are conjunctions (and) over the universe of objects
- Existential quantifiers are disjunctions (or) over the universe of objects

De Morgan's Law for Quantifiers

De Morgan's Rule	Generalized De Morgan's Rule
$P \wedge Q \equiv \neg(\neg P \vee \neg Q)$	$\forall x\, P \equiv \neg \exists x\, (\neg P)$
$P \vee Q \equiv \neg(\neg P \wedge \neg Q)$	$\exists x\, P \equiv \neg \forall x\, (\neg P)$
$\neg(P \wedge Q) \equiv \neg P \vee \neg Q$	$\neg \forall x\, P \equiv \exists x\, (\neg P)$
$\neg(P \vee Q) \equiv \neg P \wedge \neg Q$	$\neg \exists x\, P \equiv \forall x\, (\neg P)$

- "No one likes parsnips"
  - $\forall x \neg Likes(x, Parsnips) \equiv \neg \exists x\ Likes(x, Parsnips)$
- "Everyone likes ice cream"
  - $\forall x\ Likes(x, IceCream) \equiv \neg \exists x \neg Likes(x, IceCream)$

Examples

- "Brothers are siblings"
  - $\forall x, y\ Brother(x, y) \rightarrow Sibling(x, y)$
- "The function Sibling is symmetric"
  - $\forall x, y\ Sibling(x, y) \leftrightarrow Sibling(y, x)$
- "First cousin is a child of a parent's sibling"
  - $\forall x, y\ FirstCousin(x, y) \leftrightarrow \exists p, ps\ Parent(p, x) \wedge Sibling(ps, p) \wedge Parent(ps, y)$
- "All humans are mortal"
  - $\forall x\ Human(x) \rightarrow Mortal(x)$
- "Fifi has a sister who is a cat"
  - $\exists x\ Sister(Fifi, x) \wedge Cat(x)$

Examples

- “For every food, there is a person who eats that food”
  - Use: $Food(x), Person(y), Eats(y, x)$
- $\forall x \exists y\ Food(x) \rightarrow Person(y) \wedge Eats(y, x)$
- $\forall x\ Food(x) \rightarrow \exists y\ Person(y) \wedge Eats(y, x)$
  - Pushing in the $\exists$
- $\forall x \neg Food(x) \vee (\exists y Person(y) \wedge Eats(y, x))$
  - Implication elimination
- Common mistakes:
  - $\forall x \exists y\ Food(x) \wedge Person(y) \rightarrow Eats(y, x)$
    - For all x, if x is a food and there exists some person y, that person eats food x
  - $\forall x \exists y\ Food(x) \wedge Person(y) \wedge Eats(y, x)$
    - Everything is a food and there exists a person that eats that food

Quick Quiz

- “Every person eats every food”
- “All greedy kings are evil”
- “Everyone has a favorite food”
- “Every person eats some food”
- “Some person eats some food”

Quick Quiz

- "Every person eats every food"
  - $\forall x, y\ Person(x) \land Food(y) \rightarrow Eats(x, y)$
- "All greedy kings are evil"
  - $\forall x\ Greedy(x) \land King(x) \rightarrow Evil(x)$
- "Everyone has a favorite food"
  - $\forall x \exists y\ Person(x) \rightarrow Food(y) \land Favorite(y, x)$
- "Every person eats some food"
  - $\forall x \exists y\ Person(x) \rightarrow Food(y) \land Eats(x, y)$
- "Some person eats some food"
  - $\exists x \exists y\ Person(x) \land Food(y) \land Eats(x, y)$

Quick Quiz

1. (5 pts each, 30 pts total) Fill in each blank below with Y (= Yes) or N (= No) depending on whether or not the first order predicate logic sentence correctly expresses the English sentence.

a. _____ "All cats are mammals." $\forall x \; Cat(x) \;\&\; Mammal(x)$

b. _____ "Spot has a sister who is a cat." $\exists x \; Sister(x, \; Spot) \;\&\; Cat(x)$

c. _____ "For every person, there is someone whom that person likes." $\exists x \forall y \; Likes(x, \; y)$

d. _____ "There is someone who is liked by everyone." $\forall x \exists y \; Likes(x, \; y)$

e. _____ "Everyone likes ice cream." $\neg \exists x \neg \; Likes(x, \; IceCream)$

f. _____ "All men are mortal." $\forall x \; Man(x) \;\Rightarrow\; Mortal(x)$

Quick Quiz

1. (5 pts each, 30 pts total) Fill in each blank below with Y (= Yes) or N (= No) depending on whether or not the first order predicate logic sentence correctly expresses the English sentence.

a. ____N____ "All cats are mammals." $\forall x\ \mathrm{Cat}(x)\ \&\ \mathrm{Mammal}(x)$

$\forall x\ \mathrm{Cat}(x)\ \&\ \mathrm{Mammal}(x)$ means "Everything is a cat and also is a mammal."
$\forall x\ \mathrm{Cat}(x) \Rightarrow \mathrm{Mammal}(x)$ means "All cats are mammals."

Remember that implication is the natural connective to use with $\forall$*.*

b. ____Y____ "Spot has a sister who is a cat." $\exists x\ \mathrm{Sister}(x, \mathrm{Spot})\ \&\ \mathrm{Cat}(x)$

Remember that conjunction is the natural connective to use with $\exists$*.*

c. ____N____ "For every person, there is someone whom that person likes." $\exists x \forall y\ \mathrm{Likes}(x, y)$

$\exists x \forall y\ \mathrm{Likes}(x, y)$ means "There is someone who likes everyone."
$\forall x \exists y\ \mathrm{Likes}(x, y)$ means "For every person, there is someone that that person likes."

Remember that the second quantifier is inside the scope of the first quantifier.

Quick Quiz

d. _____N_____ "There is someone who is liked by everyone." $\forall x \exists y$ Likes(x, y)

$\forall x \exists y$ Likes(x, y) means "For every person, there is someone that that person likes."
$\exists x \forall y$ Likes(y, x) means "There is someone who is liked by everyone."

e. _____Y_____ "Everyone likes ice cream." $\neg \exists x \neg$ Likes(x, IceCream)

f. _____Y_____ "All men are mortal." $\forall$x Man(x) $\Rightarrow$ Mortal(x)

Quick Quiz

2. (5 pts each, 40 pts total) Let $PKF(x, y)$ mean "Person x Knows Fact y". For purposes of this question only, you may assume that the first argument is a person and the second is a fact. For each English sentence below, write the first order predicate logic sentence that best expresses it. Use "¬" to mean "not." The first one is done for you as an example.

a. Every person knows every fact. ___ $\forall x \forall y\, PKF(x, y)$ ___.

b. Every person knows at least one fact. ______________________________.

c. There is a person who knows at least one fact. ______________________________.

d. There is a person who knows every fact. ______________________________.

e. No person knows every fact. ______________________________.

f. There is a person who knows no fact. ______________________________.

g. No person knows any fact. __________________________.

h. There is a fact that is known by every person. ______________________________.

i. There is a fact that no person knows. ______________________________.

Quick Quiz

2. (5 pts each, 40 pts total) Let *PKF*(x, y) mean "Person x Knows Fact y". For purposes of this question only, you may assume that the first argument is a person and the second is a fact. For each English sentence below, write the first order predicate logic sentence that best expresses it. Use "¬" to mean "not." The first one is done for you as an example.

a. Every person knows every fact. $\underline{\quad \forall x \forall y\ PKF(x, y) \qquad\qquad}$.

b. Every person knows at least one fact. $\underline{\quad \forall x \exists y\ PKF(x, y) \qquad\qquad}$.

c. There is a person who knows at least one fact. $\underline{\quad \exists x \exists y\ PKF(x, y) \qquad\qquad}$.

d. There is a person who knows every fact. $\underline{\quad \exists x \forall y\ PKF(x, y) \qquad\qquad}$.

e. No person knows every fact. $\underline{\quad \forall x \exists y\ \neg PKF(x, y)\ .}$

$\underline{\quad \textit{alternatively} \qquad \forall x \neg \forall y\ PKF(x, y) \qquad \textit{or} \qquad \neg \exists x \forall y\ PKF(x, y)}$.

Quick Quiz

f. There is a person who knows no fact. $\exists x \forall y \neg PKF(x, y)$.

alternatively $\exists x \neg \exists y\, PKF(x, y)$ *or* $\neg \forall x \exists y\, PKF(x, y)$.

g. No person knows any fact. $\forall x \forall y \neg PKF(x, y)$.

alternatively $\forall x \neg \exists y\, PKF(x, y)$ *or* $\neg \exists x \exists y\, PKF(x, y)$.

h. There is a fact that is known by every person. $\exists y\, \forall x\, PKF(x, y)$.

i. There is a fact that no person knows. $\exists y\, \forall x \neg PKF(x, y)$.

alternatively $\exists y \neg \exists x\, PKF(x, y)$ *or* $\neg \forall y \exists x\, PKF(x, y)$.

Knowledge Engineering

- In FOL, there are many ways to represent the same thing
- “Ball-5 is red”
- HasColor(Ball-5, Red)
- Red(Ball-5)
- HasProperty(Ball-5, Color, Red)
- ColorOf(Ball-5) = Red
- HasColor(Ball-5(), Red())
  - Where Ball-5 and Red() are functions with no arguments that return an object
- Therefore, it is important to agree upon knowledge representation conventions before encoding knowledge

Knowledge Engineering

- The general process of knowledge base construction
- Steps
  - Identify the questions
  - Assemble the relevant knowledge
  - Decide on a vocabulary of predicates, functions, and constants
  - Encode general knowledge about the domain
  - Encode a description of the problem instance
  - Pose queries to the inference procedure and get answers
  - Debug and evaluate the knowledge base

Knowledge Engineering: Digital Circuit

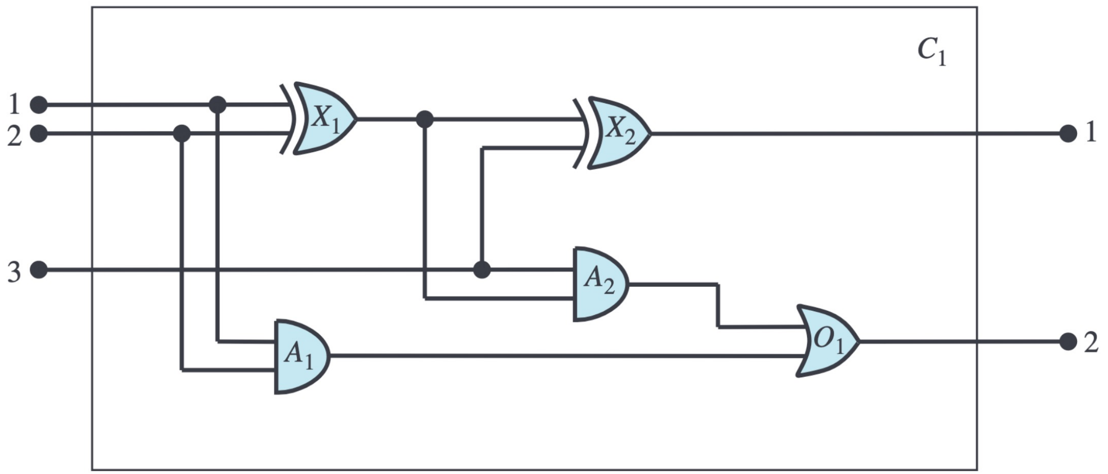


Figure 8.6 A digital circuit C_1, purporting to be a one-bit full adder. The first two inputs are the two bits to be added, and the third input is a carry bit. The first output is the sum, and the second output is a carry bit for the next adder. The circuit contains two XOR gates, two AND gates, and one OR gate.

1-in	2-in	3-in	1-out	2-out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Identify the questions

- There are many aspects of a digital circuit that an engineer may be concerned with
  - Timing, power consumption, resources, etc.
- For this example, we will focus on functionality
  - Does the circuit add properly?

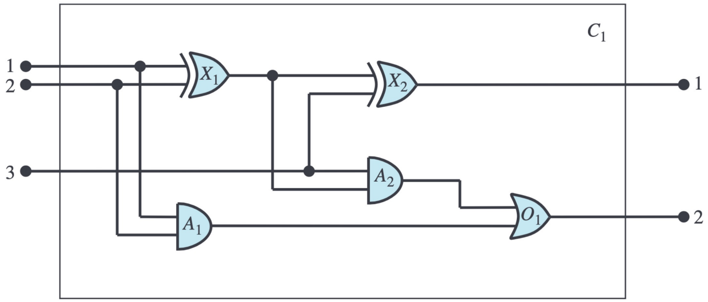


Figure 8.6 A digital circuit C_1, purporting to be a one-bit full adder. The first two inputs are the two bits to be added, and the third input is a carry bit. The first output is the sum, and the second output is a carry bit for the next adder. The circuit contains two XOR gates, two AND gates, and one OR gate.

Assemble the Relevant Knowledge

- Understand the scope of the knowledge
- May have to work with domain experts
- Knowledge relevant to the task
  - Types of gates: AND, OR, XOR
  - How the gates are connected
  - The input and output signal of the gates
- Knowledge irrelevant to the task
  - Size, shape, color, of gates
  - Path the wires take

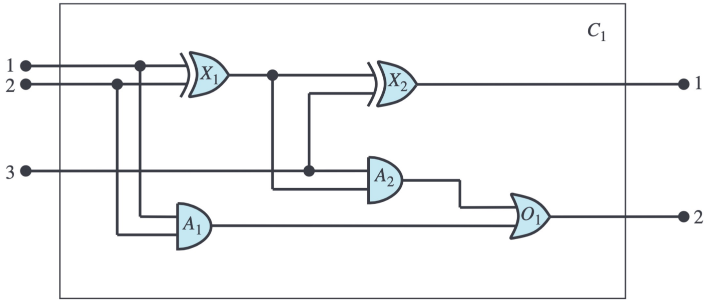


Figure 8.6 A digital circuit C_1, purporting to be a one-bit full adder. The first two inputs are the two bits to be added, and the third input is a carry bit. The first output is the sum, and the second output is a carry bit for the next adder. The circuit contains two XOR gates, two AND gates, and one OR gate.

Decide on a Vocabulary

- This vocabulary is known as the ontology
  - A particular theory of the nature of being or existence
  - Determines what kinds of things exists, but does not determine their specific interrelationships
- To identify a particular terminal, we use the functions In and Out
  - $In(2, C_1)$, $Out(1, X_1)$
    - Second input to C1, first output of X1
- To identify the gate type, we use the function $Type$
  - $Type(X_1) = XOR$
- We use the predicate $Connected$ to represent connectivity
  - $Connected\big(Out(1, X_1), In(1, X_2)\big)$

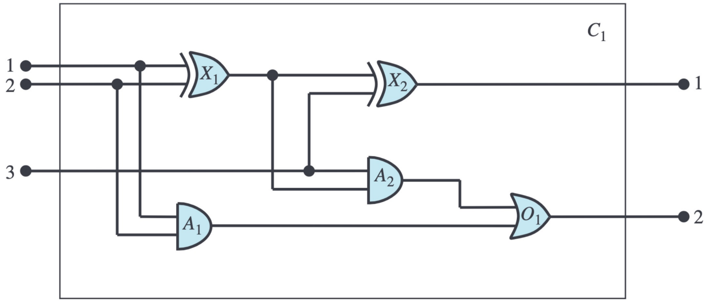


Figure 8.6 A digital circuit C_1, purporting to be a one-bit full adder. The first two inputs are the two bits to be added, and the third input is a carry bit. The first output is the sum, and the second output is a carry bit for the next adder. The circuit contains two XOR gates, two AND gates, and one OR gate.

Encode General Knowledge About the Domain

$\forall t_1, t_2$ Connected$(t_1, t_2) \Rightarrow$ Signal(t_1) = Signal(t_2)

$\forall t$ Signal(t) = 1 $\vee$ Signal(t) = 0

$1 \neq 0$

$\forall t_1, t_2$ Connected$(t_1, t_2) \Rightarrow$ Connected(t_2, t_1)

$\forall g$ Type(g) = OR $\Rightarrow$ Signal(Out(1,g)) = 1 $\Leftrightarrow$ $\exists n$ Signal(In(n,g)) = 1

$\forall g$ Type(g) = AND $\Rightarrow$ Signal(Out(1,g)) = 0 $\Leftrightarrow$ $\exists n$ Signal(In(n,g)) = 0

$\forall g$ Type(g) = XOR $\Rightarrow$ Signal(Out(1,g)) = 1 $\Leftrightarrow$ Signal(In(1,g)) $\neq$ Signal(In(2,g))

$\forall g$ Type(g) = NOT $\Rightarrow$ Signal(Out(1,g)) $\neq$ Signal(In(1,g))

Encode the Specific Problem Instance

$Type(X_1) = XOR$
$Type(A_1) = AND$
$Type(O_1) = OR$

$Type(X_2) = XOR$
$Type(A_2) = AND$

$Connected(Out(1,X_1),In(1,X_2))$
$Connected(Out(1,X_1),In(2,A_2))$
$Connected(Out(1,A_2),In(1,O_1))$
$Connected(Out(1,A_1),In(2,O_1))$
$Connected(Out(1,X_2),Out(1,C_1))$
$Connected(Out(1,O_1),Out(2,C_1))$

$Connected(In(1,C_1),In(1,X_1))$
$Connected(In(1,C_1),In(1,A_1))$
$Connected(In(2,C_1),In(2,X_1))$
$Connected(In(2,C_1),In(2,A_1))$
$Connected(In(3,C_1),In(2,X_2))$
$Connected(In(3,C_1),In(1,A_2))$

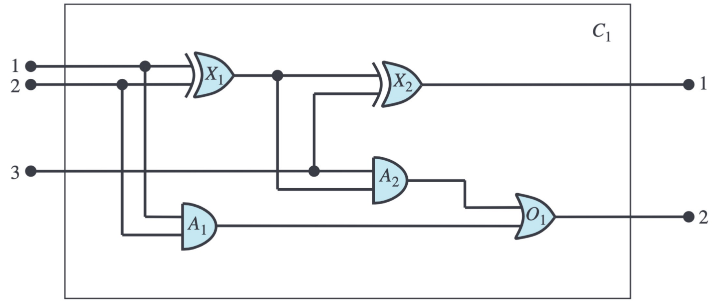


Figure 8.6 A digital circuit C_1, purporting to be a one-bit full adder. The first two inputs are the two bits to be added, and the third input is a carry bit. The first output is the sum, and the second output is a carry bit for the next adder. The circuit contains two XOR gates, two AND gates, and one OR gate.

Pose Queries to the Inference Procedure

$$\exists i_1, i_2, i_3, o_1, o_2\ Signal(In(1,C_1)) = i_1 \land Signal(In(2,C_1)) = i_2 \land Signal(In(3,C_1)) = i_3 \land Signal(Out(1,C_1)) = o_1 \land Signal(Out(2,C_1)) = o_2$$

- The KB should return all possible substitutions
- This should hopefully be the same as the the truth table for a full adder

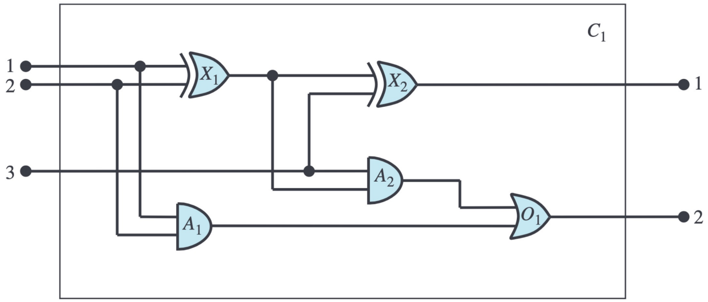


Figure 8.6 A digital circuit C_1, purporting to be a one-bit full adder. The first two inputs are the two bits to be added, and the third input is a carry bit. The first output is the sum, and the second output is a carry bit for the next adder. The circuit contains two XOR gates, two AND gates, and one OR gate.

Debug the Knowledge Base

- For example, if we forget to tell the knowledge base that $0 \neq 1$, we would get unexpected results
- Just like in programming, we will have to get creative when debugging
  - For example, we can look at the output of each gate

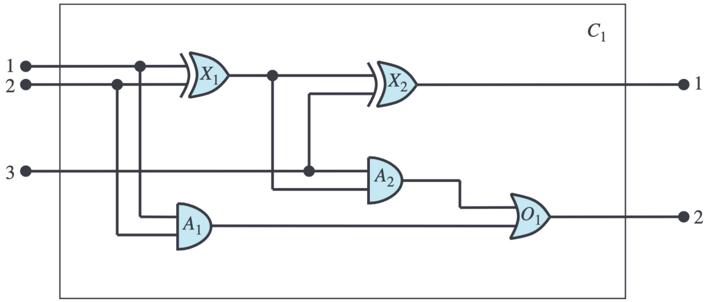


Figure 8.6 A digital circuit C_1, purporting to be a one-bit full adder. The first two inputs are the two bits to be added, and the third input is a carry bit. The first output is the sum, and the second output is a carry bit for the next adder. The circuit contains two XOR gates, two AND gates, and one OR gate.

Summary

- First-Order Logic
  - Quantifiers
  - Variables
  - Constants
  - Functions
  - Predicates
- Order of unlike quantifiers matters
- Knowledge engineering

Next Time

- Inference in first-order logic