Review

- Propositional logic
  - Simplest language
  - Its world only consists of facts (and "explicit rules")
- Very weak way to represent knowledge of complex environments with many objects in a concise way
  - Difficult to represent even the Wumpus world

$$B_{1,1} \Rightarrow P_{1,2} \vee P_{2,1}$$

Would like to say, "squares adjacent to pits are breezy" (not enumerate for all possible squares)

First-order logic

First-Order Logic

- Also called first-order predicate calculus
  - FOL, FOPC
- Makes stronger commitments
  - World consists of objects
    - Things with identities
    - e.g., people, houses, colors, …

First-Order Logic

- Also called first-order predicate calculus
  - FOL, FOPC
- Makes stronger commitments
  - World consists of objects
    - Things with identities
    - e.g., people, houses, colors, …
  - Objects have properties/relations that distinguish them from each other
    - e.g., Properties: red, round, square, …
    - e.g., Relations: brother of, bigger than, inside, …

First-Order Logic

- Also called first-order predicate calculus
  - FOL, FOPC
- Makes stronger commitments
  - World consists of objects
    - Things with identities
    - e.g., people, houses, colors, …
  - Objects have properties/relations that distinguish them from each other
    - e.g., Properties: red, round, square, …
    - e.g., Relations: brother of, bigger than, inside, …
  - Have functional relations
    - Return the object with a certain relation to given “input” object
    - The “inverse” of a (binary) relation
    - e.g., father of, best friend

Examples of Facts as Objects and Properties or Relations

- "Squares neighboring the Wumpus are smelly"
  - Objects
    - Wumpus, squares
  - Property
    - Smelly
  - Relation
    - Neighboring

Syntax of FOL: Basic Elements

- Constant symbols for specific objects

  KingJohn, 2, OSU, …
- Predicate (boolean) properties (unary) / relations (binary+)

  Brother(), Married(), >, …
- Functions (return objects)

  Sqrt() , LeftLegOf(), FatherOf(), …
- Variables

  x, y, a, b, …
- Connectives

  ∧ ∨ ¬ ⇒ ⇔
- Equality

  =
- Quantifiers

  ∀ ∃

Atomic Sentences

- Collection of terms and relation(s) together to state facts
- Atomic sentence
  - $predicate(term_1, \ldots, term_n)$
  - Or $term_1 = term_n$
- Examples

  Brother(Richard, John)

  Married(FatherOf(Richard), MotherOf(John))

Complex Sentences

- Made from atomic sentences using logical connectives

  $\neg S, \quad S_1 \wedge S_2, \quad S_1 \vee S_2, \quad S_1 \Rightarrow S_2, \quad S_1 \Leftrightarrow S_2$

  Examples:

  Older(John, 30) $\Rightarrow \neg$*Younger(John, 30)*

  $x > (1,2) \vee x \leq (1,2)$

Quantifiers

- Currently have logic that allows objects
- Now want to express properties of entire collections of objects
  - Rather than enumerate the objects by name
- Two standard quantifiers
  - Universal ∀
  - Existential ∃

Universal Qualification

- "For all …" (typically use implication ⇒)
  - Allows for "rules" to be constructed
- ∀ <*variables*> <*sentence*>
  - Everyone at OSU is smart

Universal Qualification

- "For all …" (typically use implication ⇒)
  - Allows for "rules" to be constructed
- ∀ <*variables*> <*sentence*>
  - Everyone at OSU is smart

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

Universal Qualification

- "For all …" (typically use implication ⇒)
  - Allows for "rules" to be constructed
- ∀ <*variables*> <*sentence*>
  - Everyone at OSU is smart

    $\forall x \; At(x, OSU) \Rightarrow Smart(x)$
- $\forall x \; P$ is equivalent to conjunction of all instantiations of P

Universal Qualification

- "For all …" (typically use implication ⇒)
  - Allows for "rules" to be constructed
- ∀ <*variables*> <*sentence*>
  - Everyone at OSU is smart

    $\forall x \; At(x, OSU) \Rightarrow Smart(x)$
- $\forall x \; P$ is equivalent to conjunction of all instantiations of P

  $(At(John, OSU) \Rightarrow Smart(John))$

Universal Qualification

- "For all …" (typically use implication ⇒)
  - Allows for "rules" to be constructed
- ∀ <*variables*> <*sentence*>
  - Everyone at OSU is smart

    $\forall x \; At(x, OSU) \Rightarrow Smart(x)$
- $\forall x \; P$ is equivalent to conjunction of all instantiations of P

  $(At(John, OSU) \Rightarrow Smart(John))$

  $\wedge \; (At(Bob, OSU) \Rightarrow Smart(Bob))$

Universal Qualification

- "For all …" (typically use implication ⇒)
  - Allows for "rules" to be constructed
- ∀ <*variables*> <*sentence*>
  - Everyone at OSU is smart

    $\forall x \; At(x, OSU) \Rightarrow Smart(x)$
- $\forall x \; P$ is equivalent to conjunction of all instantiations of P

  $(At(John, OSU) \Rightarrow Smart(John))$

  $\wedge \; (At(Bob, OSU) \Rightarrow Smart(Bob))$

  $\wedge \; (At(Mary, OSU) \Rightarrow Smart(Mary)) \wedge \; ...$

Existential Quantification

- "There exists …" (typically use conjunction ∧)
  - Makes a statement about some object (not all)
- ∃ <*variables*> <*sentences*>

Existential Quantification

- "There exists …" (typically use conjunction ∧)
  - Makes a statement about some object (not all)
- ∃ <*variables*> <*sentences*>
  - Someone at OSU is smart

    $\exists x \; At(x, OSU) \wedge Smart(x)$

Existential Quantification

- "There exists …" (typically use conjunction ∧)
  - Makes a statement about some object (not all)
- ∃ <*variables*> <*sentences*>
  - Someone at OSU is smart

    ∃*x* *At(x, OSU)* ∧ *Smart(x)*
- ∃ *x P* is equivalent to disjunction of all instantiations of *P*

Existential Quantification

- "There exists …" (typically use conjunction ∧)
  - Makes a statement about some object (not all)
- ∃ <*variables*> <*sentences*>
  - Someone at OSU is smart

    $\exists x \; At(x, OSU) \wedge Smart(x)$
- ∃ *x P* is equivalent to disjunction of all instantiations of *P*

  $(At(John, OSU) \wedge Smart(John))$

  $\vee \; (At(Bob, OSU) \wedge Smart(Bob))$

  $\vee \; (At(Mary, OSU) \wedge Smart(Mary)) \; \vee \; \ldots$

Existential Quantification

- "There exists …" (typically use conjunction ∧)
  - Makes a statement about some object (not all)
- ∃ <*variables*> <*sentences*>
  - Someone at OSU is smart

    ∃*x* *At(x, OSU)* ∧ *Smart(x)*
- ∃ *x P* is equivalent to disjunction of all instantiations of *P*

  (At(John, OSU) ∧ *Smart(John))*

  ∨ *(At(Bob, OSU)* ∧ *Smart(Bob))*

  ∨ *(At(Mary, OSU)* ∧ *Smart(Mary))* ∨ ...
- Uniqueness quantifier

  ∃! *x* says a unique object exists (i.e. there is exactly one)

Properties of Quantifiers

- Quantifier duality: Each can be expressed using the other

$\forall x \;\; Person(x) \Rightarrow Likes(x, IceCream)$ "Everybody likes ice cream"

Properties of Quantifiers

- Quantifier duality: Each can be expressed using the other

$\forall x \;\; Person(x) \Rightarrow Likes(x, IceCream)$ "Everybody likes ice cream"

"Not exist anyone who does not like ice cream"

Properties of Quantifiers

- Quantifier duality: Each can be expressed using the other

$\forall x \;\; Person(x) \Rightarrow Likes(x, IceCream)$ "Everybody likes ice cream"

$\neg\exists x \;\; Person(x) \wedge \neg Likes(x, IceCream)$ "Not exist anyone who does not like ice cream"

Properties of Quantifiers

- Quantifier duality: Each can be expressed using the other

$\forall x \ \ Person(x) \Rightarrow Likes(x, IceCream)$	"Everybody likes ice cream"
$\neg\exists x \ \ Person(x) \wedge \neg Likes(x, IceCream)$	"Not exist anyone who does not like ice cream"
$\exists x \ Person(x) \wedge Likes(x, Broccoli)$	"Someone likes broccoli"

Properties of Quantifiers

- Quantifier duality: Each can be expressed using the other

$\forall x \ \ Person(x) \Rightarrow Likes(x, IceCream)$	"Everybody likes ice cream"
$\neg\exists x \ \ Person(x) \wedge \neg Likes(x, IceCream)$	"Not exist anyone who does not like ice cream"
$\exists x \ Person(x) \wedge Likes(x, Broccoli)$	"Someone likes broccoli"
	"Not the case that everyone does not like broccoli"

Properties of Quantifiers

- Quantifier duality: Each can be expressed using the other

$\forall x \;\; Person(x) \Rightarrow Likes(x, IceCream)$ "Everybody likes ice cream"

$\neg\exists x \;\; Person(x) \wedge \neg Likes(x, IceCream)$ "Not exist anyone who does not like ice cream"

$\exists x \; Person(x) \wedge Likes(x, Broccoli)$ "Someone likes broccoli"

$\neg\forall x \; Person(x) \Rightarrow \neg Likes(x, Broccoli)$ "Not the case that everyone does not like broccoli"

Properties of Quantifiers

- Important relations

$\exists x \; P(x) \;=\; \neg\forall x \; \neg P(x)$

$\forall x \; P(x) \;=\; \neg\exists x \; \neg P(x)$

$P(x) \Rightarrow Q(x)$ is same as $\neg P(x) \vee Q(x)$

$\neg\,(P(x) \wedge Q(x))$ is same as $\neg P(x) \vee \neg Q(x)$

Proof

P	Q	$P \Rightarrow Q$
False	False	**TRUE**
False	True	**TRUE**
True	False	**FALSE**
True	True	**TRUE**

$$P(x) \Rightarrow Q(x)$$

is same as

$$\neg P(x) \vee Q(x)$$

P	Q	$\neg P$	$\neg P \vee Q$
False	False	True	**TRUE**
False	True	True	**TRUE**
True	False	False	**FALSE**
True	True	False	**TRUE**

Proof

P	Q	$P \wedge Q$	$\neg(P \wedge Q)$
False	False	False	**TRUE**
False	True	False	**TRUE**
True	False	False	**TRUE**
True	True	True	**FALSE**

$\neg(P(x) \wedge Q(x))$

is same as

$\neg P(x) \vee \neg Q(x)$

P	Q	$\neg P$	$\neg Q$	$\neg P \vee \neg Q$
False	False	True	True	**TRUE**
False	True	True	False	**TRUE**
True	False	False	True	**TRUE**
True	True	False	False	**FALSE**

Proof

P	Q	$P \vee Q$	$\neg(P \vee Q)$
False	False	False	**TRUE**
False	True	True	**FALSE**
True	False	True	**FALSE**
True	True	True	**FALSE**

$\neg(P(x) \vee Q(x))$

is same as

$\neg P(x) \wedge \neg Q(x)$

P	Q	$\neg P$	$\neg Q$	$\neg P \wedge \neg Q$
False	False	True	True	**TRUE**
False	True	True	False	**FALSE**
True	False	False	True	**FALSE**
True	True	False	False	**FALSE**

Conversion Example

1. $\forall x \;\; Person(x) \Rightarrow Likes(x, IceCream)$

2. $\neg \exists x \; \neg (Person(x) \Rightarrow Likes(x, IceCream))$
   [use: $\forall x \; P(x) = \neg \exists x \; \neg P(x)$]

Conversion Example

1. $\forall x \;\; Person(x) \Rightarrow Likes(x, IceCream)$

2. $\neg \exists x \; \neg (Person(x) \Rightarrow Likes(x, IceCream))$
   [use: $\forall x \; P(x) = \neg \exists x \; \neg P(x)$]

3. $\neg \exists x \; \neg (\neg Person(x) \vee Likes(x, IceCream))$
   [use: $P(x) \Rightarrow Q(x)$ is same as $\neg P(x) \vee Q(x)$]

Conversion Example

1. $\forall x \;\; Person(x) \Rightarrow Likes(x, IceCream)$

2. $\neg\exists x \; \neg(Person(x) \Rightarrow Likes(x, IceCream))$
   [use: $\forall x \; P(x) = \neg\exists x \; \neg P(x)$]

3. $\neg\exists x \; \neg(\neg Person(x) \vee Likes(x, IceCream))$
   [use: $P(x) \Rightarrow Q(x)$ is same as $\neg P(x) \vee Q(x)$]

4. $\neg\exists x \; Person(x) \wedge \neg Likes(x, IceCream)$
   [distribute negatives]
   $\neg(P(x) \wedge Q(x))$ is same as $\neg P(x) \vee \neg Q(x)$

Nested Quantifiers

- $\forall x \, \forall y$ is same as $\forall y \, \forall x$ ($\forall x,y$)
- $\exists x \, \exists y$ is same as $\exists y \, \exists x$ ($\exists x,y$)
- $\exists x \, \forall y$ is <u>not same</u> as $\forall y \, \exists x$

Nested Quantifiers

- $\forall x\ \forall y$ is same as $\forall y\ \forall x$ ($\forall x,y$)
- $\exists x\ \exists y$ is same as $\exists y\ \exists x$ ($\exists x,y$)
- $\exists x\ \forall y$ is <u>not same</u> as $\forall y\ \exists x$

$\exists$ y Person(y) $\wedge$ ($\forall$x Person(x) $\Rightarrow$ Loves(x,y))

"There is someone who is loved by everyone"

$\forall x$ *Person(x)* $\Rightarrow \exists y$ *Person(y)* $\wedge$ *Loves(x,y)*

"Everybody loves somebody"

(not guaranteed to be the same person)

Kinds of Rules

- For "Squares are breezy near a pit"
  - Diagnostic rule
    - Lead from observed effects to hidden causes
      - "Infer cause from effect"

$$\forall y \;\; Breezy(y) \Rightarrow \exists x \;\; Pit(x) \wedge Adjacent(x,y)$$

  - Causal "model-based" rule
    - Hidden world properties causes certain percepts
      - "Infer effect from cause"

$$\forall x,y \;\; Pit(x) \wedge Adjacent(x,y) \Rightarrow Breezy(y)$$

General Knowledge and Dealing with Categories

- Can we get further using less knowledge or more general knowledge?

- Classic example in AI is about knowledge and generality in sentences
  birds and flight

Birds and Flight

- Organize facts about birds as listing of facts

(robins fly)	(gannets fly)	(western grebes fly)
(crows fly)	(penguins don't fly)	(ostriches don't fly)
(common loons fly)	(fulmars fly)	(arctic loons fly)

- Approximately 8,600 species of birds in world
  - Big list
  - Small in comparison to world population of ~100 billion birds!

Birds and Flight

- Rather than extending table, simpler to represent most facts with single symbol structure representing that birds of *all* species fly

$$\forall x, s \quad species(s, bird) \wedge inst(x, s) \Rightarrow flys(x)$$

- Reasoning about classes

Categorization

- Categorization is very basic cognitive mechanism
  - Treat different things as equivalent
  - Respond in terms of class membership rather than individuality
  - Fundamental to cognition and knowledge engineering
  - Reasoning by classes reduces complexity
- Overgeneralization
  - Treating all birds as equivalent about questions of flying
  - Need to handle exceptions
    - e.g., penguins (and ostriches) do not fly!

Category Exceptions

- How many circumstances determine whether individual birds can fly?
- Minsky, AAAI-85
  - "Cooked birds can't fly"
    - How about a cooked bird served in airline meal?
  - "Stuffed birds, frozen birds, and drowned birds cannot fly"
  - "Birds wearing concrete overcoats, and wooden bird decoys do not fly"
- Amount of special case knowledge increases
- Whether a particular bird can fly is determined by:
  - Its identity, condition, situation

Qualification Problem

- Abnormalcy predicates
  - Lay out qualifications for abnormal conditions
  - Example: “birds fly unless something abnormal about them”

    $(bird\ x)$ and $(not\ (ab_1\ x)) \rightarrow (flies\ x)$
  - Classes of birds that are abnormal and conditions

    $(disabled\text{-}bird\ x) \rightarrow (ab_1\ x)$

    $(fake\text{-}bird\ x) \rightarrow (ab_1\ x)$

    $(wears\ x\ concrete\text{-}overshoes) \rightarrow (disabled\text{-}bird\ x)$

    $(dead\ x) \rightarrow (disabled\text{-}bird\ x)$

    $(drowned\ x) \rightarrow (dead\ x)$

    $(stuffed\ x) \rightarrow (dead\ x)$

    $(cooked\ x) \rightarrow (dead\ x)$

    $(wooden\text{-}image\ x)$ and $(bird\ x) \rightarrow (fake\text{-}bird\ x)$

Qualification Problem

- Want to separate typical statements from exceptions
- Qualification problem (McCarthy)
  - Proliferation of number of rules
  - Want to organize knowledge in general statements about usual cases
    - Categories are used to exploit regularities of the world
  - Then qualify statements describing their exceptions

Categories and Exceptions

- Benefits of separating knowledge of typical from exceptions
  - Reduces number of sentences
  - Focus on categories of information
    - Guides introduction of new kinds of knowledge into categories to answer questions
- Basic goal to provide framework for assumptions
  - Default statements believed in absence of contradictory information
  - "Unless you know otherwise for a particular bird, assume the bird can fly"
- However, people have much richer model of what's going on for flying

Summary

- First-order logic
  - Increased expressive power over Propositional Logic
  - Objects and relations are semantic primitives
  - Syntax: constants, functions, predicates, equality, quantifiers
    - Two standard quantifiers
      - Universal ∀
      - Existential ∃
- Dealing with categories and exceptions
  - Qualification problem