First-Order Logic

Outline

I. Syntax of FOL

II. Quantifiers

III. Model for FOL

IV. Assertions & queries in FOL

* Figures are from the textbook site unless a source is specifically cited.

I. Propositional Logic: Strength and Weakness

- Programming languages lack a general mechanism for deriving facts from other facts.
- They lack the expressiveness required to handle partial information.

I. Propositional Logic: Strength and Weakness

- ♠ Programming languages lack a general mechanism for deriving facts from other facts.
- ♠ They lack the expressiveness required to handle partial information.
- ♦ Propositional logic addresses the above issues.

I. Propositional Logic: Strength and Weakness

- ♠ Programming languages lack a general mechanism for deriving facts from other facts.
- ♠ They lack the expressiveness required to handle partial information.
- ♦ Propositional logic addresses the above issues.
- ♦ It also has *compositionality* – the meaning of a sentence is a function of the meanings of its parts.

```
¬ rain V ¬ outside V wet
```

I. Propositional Logic: Strength and Weakness

- ♠ Programming languages lack a general mechanism for deriving facts from other facts.
- ♠ They lack the expressiveness required to handle partial information.
- ♦ Propositional logic addresses the above issues.
- ♦ It also has *compositionality* – the meaning of a sentence is a function of the meanings of its parts.

```
¬ rain V ¬ outside V wet
```

- ♠ It lacks the expressive power to describe an environment with *many objects*.

I. Propositional Logic: Strength and Weakness

- ♠ Programming languages lack a general mechanism for deriving facts from other facts.
- ♠ They lack the expressiveness required to handle partial information.
- ♦ Propositional logic addresses the above issues.
- ♦ It also has *compositionality* – the meaning of a sentence is a function of the meanings of its parts.

$$\neg \texttt{rain} \vee \neg \texttt{outside} \vee \texttt{wet}$$

- ♠ It lacks the expressive power to describe an environment with *many objects*.

$$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$$

$$B_{1,2} \Leftrightarrow (P_{1,1} \vee P_{1,3} \vee P_{2,2})$$

$$\vdots$$

// Squares adjacent to pits are breezy.

I. Propositional Logic: Strength and Weakness

- ♠ Programming languages lack a general mechanism for deriving facts from other facts.
- ♠ They lack the expressiveness required to handle partial information.
- ♦ Propositional logic addresses the above issues.
- ♦ It also has *compositionality* – the meaning of a sentence is a function of the meanings of its parts.

$$\neg \texttt{rain} \vee \neg \texttt{outside} \vee \texttt{wet}$$

- ♠ It lacks the expressive power to describe an environment with *many objects*.

$$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$$
$$B_{1,2} \Leftrightarrow (P_{1,1} \vee P_{1,3} \vee P_{2,2})$$
$$\vdots$$

// Squares adjacent to pits are breezy.

- ♠ Propositional logic assumes the world contains *facts only*.

Combining Formal and Natural Languages

First-order logic

- built around objects and relations
  - Objects: people, houses, cars, trees, colors, days, …
  - Relations:
    - unary properties such as `big`, `windy`, …
    - n-ary properties such as `bigger than`, `parent of`, `on`, `owns`, …
  - Functions: `square of`, `best friend`, `age`, …
- capable of expressing facts about some or all objects

Formal Languages

Language	Ontological Commitment (What exists in the world)	Epistemological Commitment (What an agent believes about facts)
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 $\in [0, 1]$
Fuzzy logic	facts with degree of truth $\in [0, 1]$	known interval value

Alphabet of First-Order Logic

- Logical symbols
  - connectives: $\wedge, \vee, \Rightarrow, \Leftrightarrow, \neg$
  - parenthesis: (,) and punctuation ,
  - equality: $=$
  - quantifiers: $\forall$ (universal quantification), $\exists$ (existential quantification)
  - variables: $x, y, z, \ldots$; $x_1, x_2, \ldots$

Alphabet of First-Order Logic

- Logical symbols
  - connectives: $\wedge, \vee, \Rightarrow, \Leftrightarrow, \neg$
  - parenthesis: (,) and punctuation ,
  - equality: $=$
  - quantifiers: $\forall$ (universal quantification), $\exists$ (existential quantification)
  - variables: $x, y, z, \ldots$; $x_1, x_2, \ldots$
- Non-logical symbols
  - constants: Socrates, Turing, 1, earth, …

Alphabet of First-Order Logic

- Logical symbols
  - connectives: $\wedge, \vee, \Rightarrow, \Leftrightarrow, \neg$
  - parenthesis: (,) and punctuation ,
  - equality: $=$
  - quantifiers: $\forall$ (universal quantification), $\exists$ (existential quantification)
  - variables: $x, y, z, \ldots$; $x_1, x_2, \ldots$
- Non-logical symbols
  - constants: Socrates, Turing, 1, earth, …
  - predicate symbols: *true*, *false*

Alphabet of First-Order Logic

- Logical symbols
  - connectives: $\wedge, \vee, \Rightarrow, \Leftrightarrow, \neg$
  - parenthesis: (,) and punctuation ,
  - equality: $=$
  - quantifiers: $\forall$ (universal quantification), $\exists$ (existential quantification)
  - variables: $x, y, z, \ldots$; $x_1, x_2, \ldots$
- Non-logical symbols
  - constants: Socrates, Turing, 1, earth, …
  - predicate symbols: *true*, *false*

    Father(x, y) // x is father of y

    Female(x) // x is female

Alphabet of First-Order Logic

- Logical symbols
  - connectives: $\wedge, \vee, \Rightarrow, \Leftrightarrow, \neg$
  - parenthesis: (,) and punctuation ,
  - equality: $=$
  - quantifiers: $\forall$ (universal quantification), $\exists$ (existential quantification)
  - variables: $x, y, z, \ldots$; $x_1, x_2, \ldots$
- Non-logical symbols
  - constants: Socrates, Turing, 1, earth, …
  - predicate symbols: *true*, *false*
    Father(x, y) // x is father of y
    Female(x) // x is female
  - function symbols:
    gcd(x, y) // greatest common divisor of x and y
    FatherOf(x) // father of x

Terms and Atomic Sentences

- Terms
  - constants: Socrates, Turing, 1, earth, …

Terms and Atomic Sentences

- Terms
  - constants: Socrates, Turing, 1, earth, …
  - variables: $x, y, z, \ldots$; $x_1, x_2, \ldots$

Terms and Atomic Sentences

- Terms
  - constants: Socrates, Turing, 1, earth, …
  - variables: $x, y, z, \ldots$; $x_1, x_2, \ldots$
  - functions: *gcd*(x, y), *FatherOf*(x), …

Terms and Atomic Sentences

- Terms
  - constants: Socrates, Turing, 1, earth, …
  - variables: $x, y, z, \ldots$; $x_1, x_2, \ldots$
  - functions: *gcd*(x, y), *FatherOf*(x), …

$$f(x_1, x_2, \ldots, x_n)$$

function symbol terms

Terms and Atomic Sentences

- Terms
  - constants: Socrates, Turing, 1, earth, …
  - variables: $x, y, z, \ldots$; $x_1, x_2, \ldots$
  - functions: *gcd*(x, y), *FatherOf*(x), …

$$f(x_1, x_2, \ldots, x_n)$$

function symbol terms

- Atomic sentences
  - predicates: *true*, *false*

Terms and Atomic Sentences

- Terms
  - constants: Socrates, Turing, 1, earth, …
  - variables: $x, y, z, \ldots$; $x_1, x_2, \ldots$
  - functions: *gcd*(x, y), *FatherOf*(x), …

$$f(x_1, x_2, \ldots, x_n)$$

function symbol terms

- Atomic sentences
  - predicates: *true*, *false*
    Mother(Aphrodite, Harmonia)
    Male(John)

- Greek mythology related examples are taken/extended from ones used in the book *Logic for Problem Solving* by Robert Kowalski, Elsevier Science Publishing Co., Inc., 1979.

Terms and Atomic Sentences

- Terms
  - constants: Socrates, Turing, 1, earth, …
  - variables: $x, y, z, \ldots$; $x_1, x_2, \ldots$
  - functions: *gcd*(x, y), *FatherOf*(x), …

$$f(x_1, x_2, \ldots, x_n)$$

function symbol terms

- Atomic sentences
  - predicates: *true*, *false*

    Mother(Aphrodite, Harmonia)

    Male(John)
  - term equalities

    FatherOf(Apollo) = Zeus

- Greek mythology related examples are taken/extended from ones used in the book *Logic for Problem Solving* by Robert Kowalski, Elsevier Science Publishing Co., Inc., 1979.

Complex Sentences

- made of atomic sentences using logical connectives

$$\mathit{Father}(x, y) \Rightarrow \mathit{Male}(x)$$

$$\mathit{Female}(x) \vee \neg \mathit{Mother}(x, y)$$

$$\mathit{Likes}(\text{Mary, John}) \Leftrightarrow \mathit{Likes}(\text{John, Mary})$$

$$(\mathit{Parent}(x, y) \wedge \mathit{Parent}(y, z)) \Rightarrow \mathit{GrandParent}(x, z)$$

Complex Sentences

- made of atomic sentences using logical connectives

$$\mathit{Father}(x, y) \Rightarrow \mathit{Male}(x)$$
$$\mathit{Female}(x) \vee \neg \mathit{Mother}(x, y)$$
$$\mathit{Likes}(\text{Mary, John}) \Leftrightarrow \mathit{Likes}(\text{John, Mary})$$
$$(\mathit{Parent}(x, y) \wedge \mathit{Parent}(y, z)) \Rightarrow \mathit{GrandParent}(x, z)$$

- universal quantification

$\forall x\ \mathit{Circle}(x) \Rightarrow \mathit{Ellipse}(x)$ // Every circle is an ellipse.

$\neg \forall x\ \mathit{Likes}(x, \text{sushi})$ // Not everyone likes sushi.

$\forall x\ \mathit{Integer}(x) \Rightarrow (\mathit{Even}(x) \vee \mathit{Odd}(x))$ // Every integer is either even or odd.

Complex Sentences

- made of atomic sentences using logical connectives

$$\mathit{Father}(x, y) \Rightarrow \mathit{Male}(x)$$
$$\mathit{Female}(x) \vee \neg \mathit{Mother}(x, y)$$
$$\mathit{Likes}(\text{Mary, John}) \Leftrightarrow \mathit{Likes}(\text{John, Mary})$$
$$(\mathit{Parent}(x, y) \wedge \mathit{Parent}(y, z)) \Rightarrow \mathit{GrandParent}(x, z)$$

- universal quantification

$\forall x\ \mathit{Circle}(x) \Rightarrow \mathit{Ellipse}(x)$ // Every circle is an ellipse.

$\neg \forall x\ \mathit{Likes}(x, \text{sushi})$ // Not everyone likes sushi.

$\forall x\ \mathit{Integer}(x) \Rightarrow (\mathit{Even}(x) \vee \mathit{Odd}(x))$ // Every integer is either even or odd.

- existential quantification

$\exists x\ \mathit{Star}(x) \wedge \neg\ (x = \text{Sun})$ // There are stars other than the sun.

$\exists x\ \mathit{Whale}(x) \wedge (\mathit{Age}(x) = 200)$ // Some whales live to 200 years.

Syntax of First-Order Logic

$$
\begin{array}{rcl}
\textit{Sentence} & \rightarrow & \textit{AtomicSentence} \mid \textit{ComplexSentence} \\
\textit{AtomicSentence} & \rightarrow & \textit{Predicate} \mid \textit{Predicate}(\textit{Term}, \ldots) \mid \textit{Term} = \textit{Term} \\
\textit{ComplexSentence} & \rightarrow & (\ \textit{Sentence}\ ) \\
& \mid & \neg\ \textit{Sentence} \\
& \mid & \textit{Sentence} \wedge \textit{Sentence} \\
& \mid & \textit{Sentence} \vee \textit{Sentence} \\
& \mid & \textit{Sentence} \Rightarrow \textit{Sentence} \\
& \mid & \textit{Sentence} \Leftrightarrow \textit{Sentence} \\
& \mid & \textit{Quantifier}\ \textit{Variable}, \ldots\ \textit{Sentence} \\
\\
\textit{Term} & \rightarrow & \textit{Function}(\textit{Term}, \ldots) \\
& \mid & \textit{Constant} \\
& \mid & \textit{Variable} \\
\\
\textit{Quantifier} & \rightarrow & \forall \mid \exists \\
\textit{Constant} & \rightarrow & A \mid X_1 \mid \textit{John} \mid \cdots \\
\textit{Variable} & \rightarrow & a \mid x \mid s \mid \cdots \\
\textit{Predicate} & \rightarrow & \textit{True} \mid \textit{False} \mid \textit{After} \mid \textit{Loves} \mid \textit{Raining} \mid \cdots \\
\textit{Function} & \rightarrow & \textit{Mother} \mid \textit{LeftLeg} \mid \cdots \\
\\
\text{OPERATOR PRECEDENCE} & : & \neg, =, \wedge, \vee, \Rightarrow, \Leftrightarrow
\end{array}
$$

II. Scope of a Quantifier

- Quantifiers ∀ and ∃ have the lowest precedence.

* In symbolic logic, ∀ and ∃ have the highest precedence.

II. Scope of a Quantifier

- Quantifiers ∀ and ∃ have the lowest precedence.

$$\forall x\ P(x) \Rightarrow Q(x) \qquad \equiv \qquad \forall x\ (P(x) \Rightarrow Q(x))$$

* In symbolic logic, ∀ and ∃ have the highest precedence.

II. Scope of a Quantifier

- Quantifiers ∀ and ∃ have the lowest precedence.

$$\underbrace{\forall x\ P(x) \Rightarrow Q(x)}_{\text{scope of } \forall} \quad \equiv \quad \forall x\ (P(x) \Rightarrow Q(x))$$

* In symbolic logic, ∀ and ∃ have the highest precedence.

II. Scope of a Quantifier

- Quantifiers ∀ and ∃ have the lowest precedence.

$$\underbrace{\forall x\ P(x) \Rightarrow Q(x)}_{\text{scope of } \forall} \quad \equiv \quad \forall x\ (P(x) \Rightarrow Q(x))$$

$$\forall x\ P(x) \Rightarrow Q(x) \vee \exists y\ R(x, y) \vee S(y) \wedge T(x, y)$$

* In symbolic logic, ∀ and ∃ have the highest precedence.

II. Scope of a Quantifier

- Quantifiers ∀ and ∃ have the lowest precedence.

$$\underbrace{\forall x\ P(x) \Rightarrow Q(x)}_{\text{scope of } \forall} \quad \equiv \quad \forall x\ (P(x) \Rightarrow Q(x))$$

$$\forall x\ P(x) \Rightarrow Q(x) \vee \exists y\ R(x, y) \vee S(y) \wedge T(x, y)$$

$$\equiv \forall x\ (P(x) \Rightarrow (Q(x) \vee \exists y\ (R(x, y) \vee (S(y) \wedge T(x, y)))))$$

* In symbolic logic, ∀ and ∃ have the highest precedence.

II. Scope of a Quantifier

- Quantifiers ∀ and ∃ have the lowest precedence.

$$\underbrace{\forall x\ P(x) \Rightarrow Q(x)}_{\text{scope of } \forall} \quad \equiv \quad \forall x\ (P(x) \Rightarrow Q(x))$$

$$\forall x\ P(x) \Rightarrow Q(x) \vee \exists y\ R(x,y) \vee S(y) \wedge T(x,y)$$

scope of ∀

$$\equiv \forall x\ (P(x) \Rightarrow (Q(x) \vee \exists y\ (R(x,y) \vee (S(y) \wedge T(x,y)))))$$

* In symbolic logic, ∀ and ∃ have the highest precedence.

II. Scope of a Quantifier

- Quantifiers ∀ and ∃ have the lowest precedence.

$$\underbrace{\forall x\ P(x) \Rightarrow Q(x)}_{\text{scope of } \forall} \quad \equiv \quad \forall x\ (P(x) \Rightarrow Q(x))$$

$$\underbrace{\forall x\ P(x) \Rightarrow Q(x) \vee \underbrace{\exists y\ R(x,y) \vee S(y) \wedge T(x,y)}_{\text{scope of } \exists}}_{\text{scope of } \forall}$$

$$\equiv \forall x\ (P(x) \Rightarrow (Q(x) \vee \exists y\ (R(x,y) \vee (S(y) \wedge T(x,y)))))$$

* In symbolic logic, ∀ and ∃ have the highest precedence.

II. Scope of a Quantifier

- Quantifiers ∀ and ∃ have the lowest precedence.

$$\underbrace{\forall x\ P(x) \Rightarrow Q(x)}_{\text{scope of } \forall} \equiv \forall x\ (P(x) \Rightarrow Q(x))$$

$$\underbrace{\forall x\ P(x) \Rightarrow Q(x) \vee \underbrace{\exists y\ R(x,y) \vee S(y) \wedge T(x,y)}_{\text{scope of } \exists}}_{\text{scope of } \forall}$$

$$\equiv \forall x\ (P(x) \Rightarrow (Q(x) \vee \exists y\ (R(x,y) \vee (S(y) \wedge T(x,y)))))$$

- Each of ∀ and ∃ quantifies the remaining scope of the innermost pair of parentheses containing it.

* In symbolic logic, ∀ and ∃ have the highest precedence.

II. Scope of a Quantifier

- Quantifiers ∀ and ∃ have the lowest precedence.

$$\underbrace{\forall x\ P(x) \Rightarrow Q(x)}_{\text{scope of } \forall} \quad \equiv \quad \forall x\ (P(x) \Rightarrow Q(x))$$

$$\forall x\ P(x) \Rightarrow Q(x) \vee \underbrace{\exists y\ R(x,y) \vee S(y) \wedge T(x,y)}_{\text{scope of } \exists}$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad}_{\text{scope of } \forall}$$

$$\equiv \forall x\ (P(x) \Rightarrow (Q(x) \vee \exists y\ (R(x,y) \vee (S(y) \wedge T(x,y)))))$$

- Each of ∀ and ∃ quantifies the remaining scope of the innermost pair of parentheses containing it.

$$\cdots\ (\cdots\ (\cdots\ (\cdots \forall x \quad \cdots \quad)\cdots)\cdots)\cdots$$

$$\underbrace{\qquad\qquad\qquad}_{\text{scope of } \forall}$$

* In symbolic logic, ∀ and ∃ have the highest precedence.

Free and Bound Variables

A variable occurrence is *bound* in a formula if it is quantified.

A variable occurrence is *free* in a formula if it is not quantified.

Free and Bound Variables

A variable occurrence is *bound* in a formula if it is quantified.

A variable occurrence is *free* in a formula if it is not quantified.

$\forall x\ \mathit{Father}(x, y) \Rightarrow \mathit{Male}(x)$

x is bound while y is free.

Free and Bound Variables

A variable occurrence is *bound* in a formula if it is quantified.

A variable occurrence is *free* in a formula if it is not quantified.

$\forall x\ \mathit{Father}(x, y) \Rightarrow \mathit{Male}(x)$ — x is bound while y is free.

$\neg\forall x \exists y \exists z \forall s \forall t\ \ P(x, y, z, s, t)$ — x, y, z, s, t are all bound

Free and Bound Variables

A variable occurrence is *bound* in a formula if it is quantified.

A variable occurrence is *free* in a formula if it is not quantified.

$\forall x\ \mathit{Father}(x, y) \Rightarrow \mathit{Male}(x)$	x is bound while y is free.
$\neg\forall x \exists y \exists z \forall s \forall t\ \ P(x, y, z, s, t)$	x, y, z, s, t are all bound
$\forall x\ \forall y\ (P(x)\ \Rightarrow\ Q(x, f(y), z))$	x, y are bound while z is free.

Free and Bound Variables

A variable occurrence is *bound* in a formula if it is quantified.

A variable occurrence is *free* in a formula if it is not quantified.

$\forall x\ \mathit{Father}(x, y) \Rightarrow \mathit{Male}(x)$ — x is bound while y is free.

$\neg\forall x \exists y \exists z \forall s \forall t\ \ P(x, y, z, s, t)$ — x, y, z, s, t are all bound

$\forall x\ \forall y\ (P(x)\ \Rightarrow\ Q(x, f(y), z))$ — x, y are bound while z is free.

Free and bound variables can have the same name.

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

Free and Bound Variables

A variable occurrence is *bound* in a formula if it is quantified.

A variable occurrence is *free* in a formula if it is not quantified.

$\forall x\ Father(x, y) \Rightarrow Male(x)$ — x is bound while y is free.

$\neg\forall x \exists y \exists z \forall s \forall t\ \ P(x, y, z, s, t)$ — x, y, z, s, t are all bound

$\forall x\ \forall y\ (P(x)\ \Rightarrow\ Q(x, f(y), z))$ — x, y are bound while z is free.

Free and bound variables can have the same name.

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

free

Free and Bound Variables

A variable occurrence is *bound* in a formula if it is quantified.

A variable occurrence is *free* in a formula if it is not quantified.

$\forall x\ Father(x, y) \Rightarrow Male(x)$ — x is bound while y is free.

$\neg\forall x \exists y \exists z \forall s \forall t\ \ P(x, y, z, s, t)$ — x, y, z, s, t are all bound

$\forall x\ \forall y\ (P(x)\ \Rightarrow\ Q(x, f(y), z))$ — x, y are bound while z is free.

Free and bound variables can have the same name.

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

free bound

Free and Bound Variables

A variable occurrence is *bound* in a formula if it is quantified.

A variable occurrence is *free* in a formula if it is not quantified.

$\forall x\ Father(x, y) \Rightarrow Male(x)$ — x is bound while y is free.

$\neg\forall x \exists y \exists z \forall s \forall t\ \ P(x, y, z, s, t)$ — x, y, z, s, t are all bound

$\forall x\ \forall y\ (P(x)\ \Rightarrow\ Q(x, f(y), z))$ — x, y are bound while z is free.

Free and bound variables can have the same name.

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

free bound

$P(x)\ \Rightarrow (\exists x\ Q(x)) \wedge R(x)$

Free and Bound Variables

A variable occurrence is *bound* in a formula if it is quantified.

A variable occurrence is *free* in a formula if it is not quantified.

$\forall x\ Father(x, y) \Rightarrow Male(x)$ — x is bound while y is free.

$\neg\forall x \exists y \exists z \forall s \forall t\ \ P(x, y, z, s, t)$ — x, y, z, s, t are all bound

$\forall x\ \forall y\ (P(x)\ \Rightarrow\ Q(x, f(y), z))$ — x, y are bound while z is free.

Free and bound variables can have the same name.

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

free bound

$P(x)\ \Rightarrow (\exists x\ Q(x)) \wedge R(x)$

same free variable

Free and Bound Variables

A variable occurrence is *bound* in a formula if it is quantified.

A variable occurrence is *free* in a formula if it is not quantified.

$\forall x\ Father(x, y) \Rightarrow Male(x)$ — x is bound while y is free.

$\neg\forall x \exists y \exists z \forall s \forall t\ \ P(x, y, z, s, t)$ — x, y, z, s, t are all bound

$\forall x\ \forall y\ (P(x)\ \Rightarrow\ Q(x, f(y), z))$ — x, y are bound while z is free.

Free and bound variables can have the same name.

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

free bound

different bound variable

$P(x)\ \Rightarrow (\exists x\ Q(x)) \wedge R(x)$

same free variable

Nested Quantifiers

$\forall x \exists y$ *Student*$(x) \wedge$ *Course*(y) $\wedge$ *Enrolled*(x, y)

$\forall x \exists y$ *Brother*$(x, y) \Rightarrow$ *Sibiling*(x, y)

Order matters for quantifiers of different types:

$\forall x \exists y$ *Loves*(x, y) // Everybody (x) loves somebody (y)

$\exists x \forall y$ *Loves*(y, x) // There is someone (x) whom everyone (y) loves.

Nested Quantifiers

$\forall x \exists y\ Student(x) \wedge Course(\text{y}) \wedge Enrolled(x, y)$

$\forall x \exists y\ Brother(x, y) \Rightarrow Sibiling(x, y)$

Order matters for quantifiers of different types:

$\forall x \exists y\ Loves(x, y)$ // Everybody (x) loves somebody (y)

$\exists x \forall y\ Loves(y, x)$ // There is someone (x) whom everyone (y) loves.

but not for those of the same type and appearing next to each other:

$\exists x \exists y\ Loves(x, y) \quad \equiv \quad \exists y \exists x\ Loves(x, y)$

$\forall x \forall y\ (Brother(x, y) \Rightarrow Sibiling(x, y))$

$\equiv \quad \forall y \forall x\ (Brother(x, y) \Rightarrow Sibiling(x, y))$

Connections Between ∀ and ∃ Through ¬

$$\forall x \ \neg Likes(x, Parsnips) \equiv \neg \exists x \ Likes(x, Parsnips)$$

$$\forall x \ Likes(x, Icecream) \equiv \neg \exists x \ \neg Likes(x, Icecream)$$

Connections Between ∀ and ∃ Through ¬

$$\forall x \ \neg \mathit{Likes}(x, \mathit{Parsnips}) \equiv \neg \exists x \ \mathit{Likes}(x, \mathit{Parsnips})$$

$$\forall x \ \mathit{Likes}(x, \mathit{Icecream}) \equiv \neg \exists x \ \neg \mathit{Likes}(x, \mathit{Icecream})$$

De Morgan's rules still apply:

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

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

Connections Between ∀ and ∃ Through ¬

$$\forall x \ \neg \mathit{Likes}(x, \mathit{Parsnips}) \equiv \neg \exists x \ \mathit{Likes}(x, \mathit{Parsnips})$$

$$\forall x \ \mathit{Likes}(x, \mathit{Icecream}) \equiv \neg \exists x \ \neg \mathit{Likes}(x, \mathit{Icecream})$$

De Morgan's rules still apply:

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

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

Move negation inward, flipping the quantifiers:

$$\neg \forall x \exists y \exists z \forall s \forall t \ P(x, y, z, s, t)$$

Connections Between ∀ and ∃ Through ¬

$$\forall x \ \neg \mathit{Likes}(x, \mathit{Parsnips}) \equiv \neg \exists x \ \mathit{Likes}(x, \mathit{Parsnips})$$

$$\forall x \ \mathit{Likes}(x, \mathit{Icecream}) \equiv \neg \exists x \ \neg \mathit{Likes}(x, \mathit{Icecream})$$

De Morgan's rules still apply:

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

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

Move negation inward, flipping the quantifiers:

$$\neg \forall x \exists y \exists z \forall s \forall t \ P(x, y, z, s, t) \equiv \exists x \neg \exists y \exists z \forall s \forall t \ P(x, y, z, s, t)$$

Connections Between ∀ and ∃ Through ¬

$$\forall x \ \neg \mathit{Likes}(x, \mathit{Parsnips}) \equiv \neg \exists x \ \mathit{Likes}(x, \mathit{Parsnips})$$

$$\forall x \ \mathit{Likes}(x, \mathit{Icecream}) \equiv \neg \exists x \ \neg \mathit{Likes}(x, \mathit{Icecream})$$

De Morgan's rules still apply:

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

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

Move negation inward, flipping the quantifiers:

$$\neg \forall x \exists y \exists z \forall s \forall t \ P(x, y, z, s, t) \equiv \exists x \neg \exists y \exists z \forall s \forall t \ P(x, y, z, s, t)$$

$$\equiv \exists x \forall y \neg \exists z \forall s \forall t \ P(x, y, z, s, t)$$

Connections Between ∀ and ∃ Through ¬

$$\forall x \ \neg \mathit{Likes}(x, \mathit{Parsnips}) \equiv \neg \exists x \ \mathit{Likes}(x, \mathit{Parsnips})$$

$$\forall x \ \mathit{Likes}(x, \mathit{Icecream}) \equiv \neg \exists x \ \neg \mathit{Likes}(x, \mathit{Icecream})$$

De Morgan's rules still apply:

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

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

Move negation inward, flipping the quantifiers:

$$\neg \forall x \exists y \exists z \forall s \forall t \ P(x, y, z, s, t) \equiv \exists x \neg \exists y \exists z \forall s \forall t \ P(x, y, z, s, t)$$
$$\equiv \exists x \forall y \neg \exists z \forall s \forall t \ P(x, y, z, s, t)$$
$$\equiv \exists x \forall y \forall z \neg \forall s \forall t \ P(x, y, z, s, t)$$

Connections Between ∀ and ∃ Through ¬

$$\forall x \ \neg \mathit{Likes}(x, \mathit{Parsnips}) \equiv \neg \exists x \ \mathit{Likes}(x, \mathit{Parsnips})$$

$$\forall x \ \mathit{Likes}(x, \mathit{Icecream}) \equiv \neg \exists x \ \neg \mathit{Likes}(x, \mathit{Icecream})$$

De Morgan's rules still apply:

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

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

Move negation inward, flipping the quantifiers:

$$\neg \forall x \exists y \exists z \forall s \forall t \ P(x, y, z, s, t) \equiv \exists x \neg \exists y \exists z \forall s \forall t \ P(x, y, z, s, t)$$
$$\equiv \exists x \forall y \neg \exists z \forall s \forall t \ P(x, y, z, s, t)$$
$$\equiv \exists x \forall y \forall z \neg \forall s \forall t \ P(x, y, z, s, t)$$
$$\equiv \exists x \forall y \forall z \exists s \neg \forall t \ P(x, y, z, s, t)$$

Connections Between ∀ and ∃ Through ¬

$\forall x \ \neg Likes(x, Parsnips) \equiv \neg \exists x \ Likes(x, Parsnips)$

$\forall x \ Likes(x, Icecream) \equiv \neg \exists x \ \neg Likes(x, Icecream)$

De Morgan's rules still apply:

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

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

Move negation inward, flipping the quantifiers:

$$\neg \forall x \exists y \exists z \forall s \forall t \ P(x, y, z, s, t)$$
$$\equiv \exists x \neg \exists y \exists z \forall s \forall t \ P(x, y, z, s, t)$$
$$\equiv \exists x \forall y \neg \exists z \forall s \forall t \ P(x, y, z, s, t)$$
$$\equiv \exists x \forall y \forall z \neg \forall s \forall t \ P(x, y, z, s, t)$$
$$\equiv \exists x \forall y \forall z \exists s \neg \forall t \ P(x, y, z, s, t)$$
$$\equiv \exists x \forall y \forall z \exists s \exists t \ \neg P(x, y, z, s, t)$$

Equality

- It states that two terms refer to the same object.

$$\mathit{Father}(\text{Zeus}) = \text{Cronus}$$

$$\mathit{Father}(\text{Cronus}) = \text{Uranus}$$

Equality

- It states that two terms refer to the same object.

$$\mathit{Father}(\text{Zeus}) = \text{Cronus}$$

$$\mathit{Father}(\text{Cronus}) = \text{Uranus}$$

- The symbol can also be used to state that two terms are not the same object.

Equality

- It states that two terms refer to the same object.

$$\mathit{Father}(\text{Zeus}) = \text{Cronus}$$
$$\mathit{Father}(\text{Cronus}) = \text{Uranus}$$

- The symbol can also be used to state that two terms are not the same object.

// Zeus has exactly two brothers

Equality

- It states that two terms refer to the same object.

$$\mathit{Father}(\text{Zeus}) = \text{Cronus}$$
$$\mathit{Father}(\text{Cronus}) = \text{Uranus}$$

- The symbol can also be used to state that two terms are not the same object.

// Zeus has exactly two brothers

$$\exists x, y\ \mathit{Brother}(x, \text{Zeus}) \wedge \mathit{Brother}(y, \text{Zeus}) \wedge \neg(x = y)$$
$$\wedge\ (\forall z\ \mathit{Brother}(z, \text{Zeus}) \Rightarrow (z = x) \vee (z = y))$$

Equality

- It states that two terms refer to the same object.

$$\textit{Father}(\text{Zeus}) = \text{Cronus}$$
$$\textit{Father}(\text{Cronus}) = \text{Uranus}$$

- The symbol can also be used to state that two terms are not the same object.

// Zeus has exactly two brothers
// ($x \equiv$ Poseidon and $y \equiv$ Hades, or $x \equiv$ Hades and $y \equiv$ Poseidon)

$$\exists x, y\ \textit{Brother}(x, \text{Zeus}) \wedge \textit{Brother}(y, \text{Zeus}) \wedge \neg(x = y)$$
$$\wedge\ (\forall z\ \textit{Brother}(z, \text{Zeus}) \Rightarrow (z = x) \vee (z = y))$$

III. Model for First-Order Logic

- Sentences are true with respect to a model M.

III. Model for First-Order Logic

- Sentences are true with respect to a model M.
- The model M contains objects (called *domain elements*) and interpretations of symbols.

III. Model for First-Order Logic

- Sentences are true with respect to a model M.
- The model M contains objects (called *domain elements*) and interpretations of symbols.
    - constant symbols → objects in domain D

III. Model for First-Order Logic

- Sentences are true with respect to a model M.
- The model M contains objects (called *domain elements*) and interpretations of symbols.
  - constant symbols → objects in domain D
  - predicate symbols → relations

III. Model for First-Order Logic

- Sentences are true with respect to a model M.
- The model M contains objects (called *domain elements*) and interpretations of symbols.
  - constant symbols → objects in domain D
  - predicate symbols → relations
  - function symbols → functional relations

III. Model for First-Order Logic

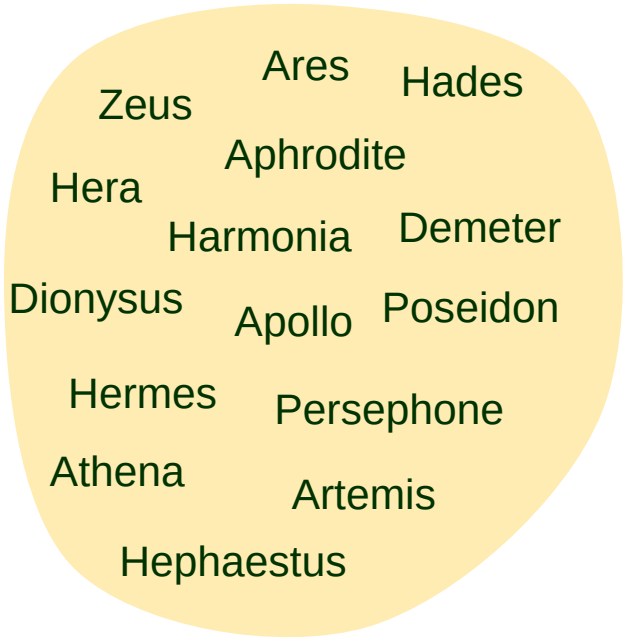
- Sentences are true with respect to a model M.
- The model M contains objects (called *domain elements*) and interpretations of symbols.
  - constant symbols → objects in domain D
  - predicate symbols → relations
  - function symbols → functional relations
- Each predicate $P(x_1, \ldots, x_k)$ is mapped to a relation, which is a set of k-tuples over D.

III. Model for First-Order Logic

- Sentences are true with respect to a model M.
- The model M contains objects (called *domain elements*) and interpretations of symbols.
  - constant symbols → objects in domain D
  - predicate symbols → relations
  - function symbols → functional relations
- Each predicate $P(x_1, \ldots, x_k)$ is mapped to a relation, which is a set of k-tuples over D.
- Each function $f(x_1, \ldots, x_k)$ is mapped to a function from D^k to $D \cup \{o\}$, where o is some invisible object.

Model Example

Model for the family relationships of the Greek gods (incomplete).



domain D

Father(Zeus, Hermes)

Mother(Hera, Ares)

Mother(Aphrodite, Harmonia)

Father(Zeus, Athena)

⋮

predicates

Weapon(Zeus) // ≡ Thunderbolt

Weapon(Apollo) // ≡ BowAndArrows

Carry(Hermes) // ≡ Flute

Carry(Aphrodite) // ≡ Apple

⋮

functions

Truth in First-Order Logic

- A predicate $P(t_1, \ldots, t_k)$ is true if the objects referred to by the terms $t_1, \ldots, t_k$ are in the relation referred to by the predicate.

Truth in First-Order Logic

- A predicate $P(t_1, \ldots, t_k)$ is true if the objects referred to by the terms $t_1, \ldots, t_k$ are in the relation referred to by the predicate.

- $t_1 = t_2$ is true if the two terms t_1 and t_2 refer to the same object.

Truth in First-Order Logic

- A predicate $P(t_1, \ldots, t_k)$ is true if the objects referred to by the terms $t_1, \ldots, t_k$ are in the relation referred to by the predicate.

- $t_1 = t_2$ is true if the two terms t_1 and t_2 refer to the same object.

- The semantics of sentences formed with logical connectives are identical to those in propositional logic.

Truth in First-Order Logic

- A predicate $P(t_1, \ldots, t_k)$ is true if the objects referred to by the terms $t_1, \ldots, t_k$ are in the relation referred to by the predicate.

- $t_1 = t_2$ is true if the two terms t_1 and t_2 refer to the same object.

- The semantics of sentences formed with logical connectives are identical to those in propositional logic.

Quantifiers allow us to express properties of a collection of objects instead of enumerating them by name.

∀ (universal): "for all"

∃ (existential): "there exists"

Truths with Quantifications

- $\forall x\ P(x)$ is true in a model M iff $P(x)$ is true with x assuming every object in the model

$\forall x\ Father(x, y) \Rightarrow Male(x)$ — *true* (in every model)

$\forall x\ Ellipse(x) \Rightarrow Circle(x)$ — *true* (in every model)

Truths with Quantifications

- $\forall x\ P(x)$ is true in a model M iff $P(x)$ is true with x assuming every object in the model

$\forall x\ Father(x, y) \Rightarrow Male(x)$ — *true* (in every model)

$\forall x\ Ellipse(x) \Rightarrow Circle(x)$ — *true* (in every model)

- $\exists x\ P(x)$ is true in a model M iff $P(x)$ is true with x assuming some object in the model.

$\exists x \neg Likes(x, \text{sushi})$ — *true* (in a model that includes all the people in the world)

Truths with Quantifications

- $\forall x\ P(x)$ is true in a model M iff $P(x)$ is true with x assuming every object in the model

$\forall x\ \mathit{Father}(x, y) \Rightarrow \mathit{Male}(x)$ — *true* (in every model)

$\forall x\ \mathit{Ellipse}(x) \Rightarrow \mathit{Circle}(x)$ — *true* (in every model)

- $\exists x\ P(x)$ is true in a model M iff $P(x)$ is true with x assuming some object in the model.

$\exists x \neg \mathit{Likes}(x, \text{sushi})$ — *true* (in a model that includes all the people in the world)

$\exists x\ \mathit{Mother}(x, \text{Ares}) \wedge \mathit{Mother}(x, \text{Harmonia})$ — *false* (in the model of Greek mythology)

IV. Knowledge Engineering

A field of AI dedicated to representing information about the world in a form that can be utilized by a computer to solve complex tasks such as:

- medical diagnosis
- dialog in a natural language
- etc.

♦ Knowledge representation (logical rules, semantic nets, frames, etc.)

♦ Automated reasoning (inference engines, theorem provers, etc.)

IV. Knowledge Engineering

A field of AI dedicated to representing information about the world in a form that can be utilized by a computer to solve complex tasks such as:

- medical diagnosis
- dialog in a natural language
- etc.

♦ Knowledge representation (logical rules, semantic nets, frames, etc.)

♦ Automated reasoning (inference engines, theorem provers, etc.)

A *domain* is some part of the world about which we wish to express some knowledge.

Assertions and Queries in FOL

- Add sentences, called *assertions*, to a KB using TELL.

  TELL(*KB*, *Likes*(John, *Icecream*))

  TELL(*KB*, *Father*(Zeus, Athena))

  TELL(*KB*, $\forall x \exists y$ *Brother*$(x, y) \Rightarrow$ *Sibiling*(x, y))

Assertions and Queries in FOL

- Add sentences, called *assertions*, to a KB using TELL.

  TELL(*KB*, *Likes*(John, *Icecream*))

  TELL(*KB*, *Father*(Zeus, Athena))

  TELL(*KB*, $\forall x \exists y$ *Brother*$(x, y) \Rightarrow$ *Sibiling*(x, y))

- Ask the KB questions using ASK.

  ASK(*KB*, *Likes*(John, *Icecream*))

Assertions and Queries in FOL

- Add sentences, called *assertions*, to a KB using TELL.

  TELL(*KB*, *Likes*(John, *Icecream*))

  TELL(*KB*, *Father*(Zeus, Athena))

  TELL(*KB*, $\forall x \exists y$ *Brother*$(x, y) \Rightarrow$ *Sibiling*(x, y))

- Ask the KB questions using ASK.

  ASK(*KB*, *Likes*(John, *Icecream*))

  Query: question asked

Assertions and Queries in FOL

- Add sentences, called *assertions*, to a KB using TELL.

  TELL(*KB*, *Likes*(John, *Icecream*))

  TELL(*KB*, *Father*(Zeus, Athena))

  TELL(*KB*, $\forall x \exists y$ *Brother*$(x, y) \Rightarrow$ *Sibiling*(x, y))

- Ask the KB questions using ASK.

  ASK(*KB*, *Likes*(John, *Icecream*))

  Query: question asked

Any query is entailed by the KB should be answered affirmatively.

Substitution

Suppose another KB has the following predicates:

$Bird(\text{Swan})$, $Bird(\text{Crane})$, $Bird(\text{Parrot})$,

- Quantified query

  $\text{ASK}(KB, \exists x\ Bird(x))$

Substitution

Suppose another KB has the following predicates:

Bird(Swan), *Bird*(Crane), *Bird*(Parrot),

- Quantified query

  ASK(*KB*, $\exists x$ *Bird*(x)) returns *true*

Substitution

Suppose another KB has the following predicates:

Bird(Swan), *Bird*(Crane), *Bird*(Parrot),

- Quantified query

  ASK(*KB*, $\exists x$ *Bird*(x)) returns *true*

- To know what values of x make the sentence true

  ASKVARS(*KB*, *Bird*(x))

Substitution

Suppose another KB has the following predicates:

Bird(Swan), *Bird*(Crane), *Bird*(Parrot),

- Quantified query

  ASK(*KB*, $\exists x$ *Bird*(x)) returns *true*

- To know what values of x make the sentence true

  ASKVARS(*KB*, *Bird*(x))

  The query returns

  $\{x/$ Swan$\}$, $\{x/$ Crane$\}$, and $\{x/$ Parrot$\}$

Substitution

Suppose another KB has the following predicates:

Bird(Swan), *Bird*(Crane), *Bird*(Parrot),

- Quantified query

  ASK(*KB*, $\exists x$ *Bird*(x)) returns *true*

- To know what values of x make the sentence true

  ASKVARS(*KB*, *Bird*(x))

  The query returns

  $\{x/\text{ Swan}\}$, $\{x/\text{ Crane}\}$, and $\{x/\text{ Parrot}\}$

  substitution or *binding list*

The Kinship Domain

Kinship relations are represented by binary predicates.

// One's mother is the person's female parent.

$\forall m, c \; \textit{Mother}(c) = m \Leftrightarrow \textit{Female}(m) \wedge \textit{Parent}(m, c)$

The Kinship Domain

Kinship relations are represented by binary predicates.

// One's mother is the person's female parent.

$\forall m, c\ \mathit{Mother}(c) = m \Leftrightarrow \mathit{Female}(m) \wedge \mathit{Parent}(m, c)$

// One's husband is the person's male spouse.

$\forall w, h\ \mathit{Husband}(h, w) \Leftrightarrow \mathit{Male}(h) \wedge \mathit{Spouse}(h, w)$

The Kinship Domain

Kinship relations are represented by binary predicates.

// One's mother is the person's female parent.

$\forall m, c\ \mathit{Mother}(c) = m \Leftrightarrow \mathit{Female}(m) \wedge \mathit{Parent}(m, c)$

// One's husband is the person's male spouse.

$\forall w, h\ \mathit{Husband}(h, w) \Leftrightarrow \mathit{Male}(h) \wedge \mathit{Spouse}(h, w)$

// Parent and child are inverse relations.

$\forall p, c\ \mathit{Parent}(p, c) \Leftrightarrow \mathit{Child}(c, p)$

The Kinship Domain

Kinship relations are represented by binary predicates.

// One's mother is the person's female parent.

$\forall m, c \; Mother(c) = m \Leftrightarrow Female(m) \wedge Parent(m, c)$

// One's husband is the person's male spouse.

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

// Parent and child are inverse relations.

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

// A grand parent is a parent of one's parent

$\forall g, c \; GrandParent(g, c) \Leftrightarrow \exists p \; (Parent(g, p) \wedge Parent(p, c))$

The Kinship Domain

Kinship relations are represented by binary predicates.

// One's mother is the person's female parent.

$\forall m, c\ \mathit{Mother}(c) = m \Leftrightarrow \mathit{Female}(m) \wedge \mathit{Parent}(m, c)$

// One's husband is the person's male spouse.

$\forall w, h\ \mathit{Husband}(h, w) \Leftrightarrow \mathit{Male}(h) \wedge \mathit{Spouse}(h, w)$

// Parent and child are inverse relations.

$\forall p, c\ \mathit{Parent}(p, c) \Leftrightarrow \mathit{Child}(c, p)$

// A grand parent is a parent of one's parent

$\forall g, c\ \mathit{GrandParent}(g, c) \Leftrightarrow \exists p\ (\mathit{Parent}(g, p) \wedge \mathit{Parent}(p, c))$

// A sibling is another child of one's parent

$\forall x, s\ \mathit{Sibling}(x, s) \Leftrightarrow x \neq s \wedge \exists p\ (\mathit{Parent}(p, x) \wedge \mathit{Parent}(p, s))$

The Kinship Domain

Kinship relations are represented by binary predicates.

Axioms

// One's mother is the person's female parent.

$\forall m, c\ Mother(c) = m \Leftrightarrow Female(m) \wedge Parent(m, c)$

// One's husband is the person's male spouse.

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

// Parent and child are inverse relations.

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

// A grand parent is a parent of one's parent

$\forall g, c\ GrandParent(g, c) \Leftrightarrow \exists p\ (Parent(g, p) \wedge Parent(p, c))$

// A sibling is another child of one's parent

$\forall x, s\ Sibling(x, s) \Leftrightarrow x \neq s \wedge \exists p\ (Parent(p, x) \wedge Parent(p, s))$

The Kinship Domain

Kinship relations are represented by binary predicates.

Axioms

// One's mother is the person's female parent.

$\forall m, c\ \mathit{Mother}(c) = m \Leftrightarrow \mathit{Female}(m) \wedge \mathit{Parent}(m, c)$

// One's husband is the person's male spouse.

$\forall w, h\ \mathit{Husband}(h, w) \Leftrightarrow \mathit{Male}(h) \wedge \mathit{Spouse}(h, w)$

// Parent and child are inverse relations.

$\forall p, c\ \mathit{Parent}(p, c) \Leftrightarrow \mathit{Child}(c, p)$

// A grand parent is a parent of one's parent

$\forall g, c\ \mathit{GrandParent}(g, c) \Leftrightarrow \exists p\ (\mathit{Parent}(g, p) \wedge \mathit{Parent}(p, c))$

// A sibling is another child of one's parent

$\forall x, s\ \mathit{Sibling}(x, s) \Leftrightarrow x \neq s \wedge \exists p\ (\mathit{Parent}(p, x) \wedge \mathit{Parent}(p, s))$

These are *definitions* in the form of $\forall x, y\ P(x, y) \Leftrightarrow \ldots$ and built upon a basic set of predicates *Child*, *Male*, *Female*, etc.

Axioms and Theorems

- *Axioms* in a domain are logical sentences that are taken to be true without being derived.
- *Theorems* are logical sentences entailed by axioms.

Axioms and Theorems

- *Axioms* in a domain are logical sentences that are taken to be true without being derived.

- *Theorems* are logical sentences entailed by axioms.

$\forall x, y\ Sibling(x, y) \Leftrightarrow Sibling(y, x)$

// entailed by

// $\forall x, s\ Sibling(x, s) \Leftrightarrow x \neq s \wedge \exists p\ (Parent(p, x) \wedge Parent(p, s))$

Axioms and Theorems

♠ *Axioms* in a domain are logical sentences that are taken to be true without being derived.

♠ *Theorems* are logical sentences entailed by axioms.

$\forall x, y\ \mathit{Sibling}(x, y) \Leftrightarrow \mathit{Sibling}(y, x)$

// entailed by

// $\forall x, s\ \mathit{Sibling}(x, s) \Leftrightarrow x \neq s \wedge \exists p\ (\mathit{Parent}(p, x) \wedge \mathit{Parent}(p, s))$

ASK(*KB*, $\forall x, y\ \mathit{Sibling}(x, y) \Leftrightarrow \mathit{Sibling}(y, x)$) should return *true*.

Axioms and Theorems

♠ *Axioms* in a domain are logical sentences that are taken to be true without being derived.

♠ *Theorems* are logical sentences entailed by axioms.

$\forall x, y\ \mathit{Sibling}(x, y) \Leftrightarrow \mathit{Sibling}(y, x)$

// entailed by

// $\forall x, s\ \mathit{Sibling}(x, s) \Leftrightarrow x \neq s \wedge \exists p\ (\mathit{Parent}(p, x) \wedge \mathit{Parent}(p, s))$

ASK(*KB*, $\forall x, y\ \mathit{Sibling}(x, y) \Leftrightarrow \mathit{Sibling}(y, x)$) should return *true*.

♠ Some axioms are not definitions.

$\forall x\ \mathit{Person}(x) \Leftrightarrow \cdots$ // no clear way to define

Axioms and Theorems

♠ *Axioms* in a domain are logical sentences that are taken to be true without being derived.

♠ *Theorems* are logical sentences entailed by axioms.

$\forall x, y\ \mathit{Sibling}(x, y) \Leftrightarrow \mathit{Sibling}(y, x)$

// entailed by

// $\forall x, s\ \mathit{Sibling}(x, s) \Leftrightarrow x \neq s \wedge \exists p\ (\mathit{Parent}(p, x) \wedge \mathit{Parent}(p, s))$

ASK(*KB*, $\forall x, y\ \mathit{Sibling}(x, y) \Leftrightarrow \mathit{Sibling}(y, x)$) should return *true*.

♠ Some axioms are not definitions.

$\forall x\ \mathit{Person}(x) \Leftrightarrow \cdots$ // no clear way to define

♠ Some predicates have no complete definitions.

Axioms and Theorems

♠ *Axioms* in a domain are logical sentences that are taken to be true without being derived.

♠ *Theorems* are logical sentences entailed by axioms.

$\forall x, y\ Sibling(x, y) \Leftrightarrow Sibling(y, x)$

// entailed by

// $\forall x, s\ Sibling(x, s) \Leftrightarrow x \neq s \wedge \exists p\ (Parent(p, x) \wedge Parent(p, s))$

ASK(*KB*, $\forall x, y\ Sibling(x, y) \Leftrightarrow Sibling(y, x)$) should return *true*.

♠ Some axioms are not definitions.

$\forall x\ Person(x) \Leftrightarrow \cdots$ // no clear way to define

♠ Some predicates have no complete definitions.

$\forall x\ Person(x) \Longrightarrow \cdots$ // partial specification of

$\forall x\ \cdots \Longrightarrow Person(x)$ // properties