FOL Models and Usage

Outline

I. Quantifiers

II. Model for FOL

III. Assertions & queries in FOL

IV. Natural numbers & sets

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

I. Scope of a Quantifier

- Quantifiers ∀ and ∃ have the lowest precedence.

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

I. 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.

I. 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.

I. Scope of a Quantifier

- Quantifiers ∀ and ∃ have the lowest precedence.

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

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

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

I. Scope of a Quantifier

- Quantifiers ∀ and ∃ have the lowest precedence.

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

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

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

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

I. Scope of a Quantifier

- Quantifiers ∀ and ∃ have the lowest precedence.

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

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

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

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

I. 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.

I. Scope of a Quantifier

- Quantifiers ∀ and ∃ have the lowest precedence.

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

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

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

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

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

scope of ∀

* 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

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

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

Order matters for quantifiers of different types:

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

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

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

$\exists x \exists y\ \mathit{Loves}(x, y) \quad \equiv \quad \exists y \exists x\ \mathit{Loves}(x, y)$

$\forall x \forall y\ (\mathit{Brother}(x, y) \Rightarrow \mathit{Sibiling}(x, y))$

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

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 ¬

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

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

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:

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

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

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

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.

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

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

// Zeus has exactly two brothers

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

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)

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

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)

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

// Zeus has at least
// two brothers.

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)

$$\exists x \exists y\ \textit{Brother}(x, \text{Zeus}) \wedge \textit{Brother}(y, \text{Zeus}) \wedge \neg(x = y)$$
// Zeus has at least
// two brothers.

$$\wedge\ (\forall z\ \textit{Brother}(z, \text{Zeus}) \Rightarrow (z = x) \vee (z = y))$$
// Zeus has no other brothers
// besides x, y.

II. Model for First-Order Logic

- Sentences are true with respect to a model M.

II. 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.

II. 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

II. 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

II. 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

II. Model for First-Order Logic

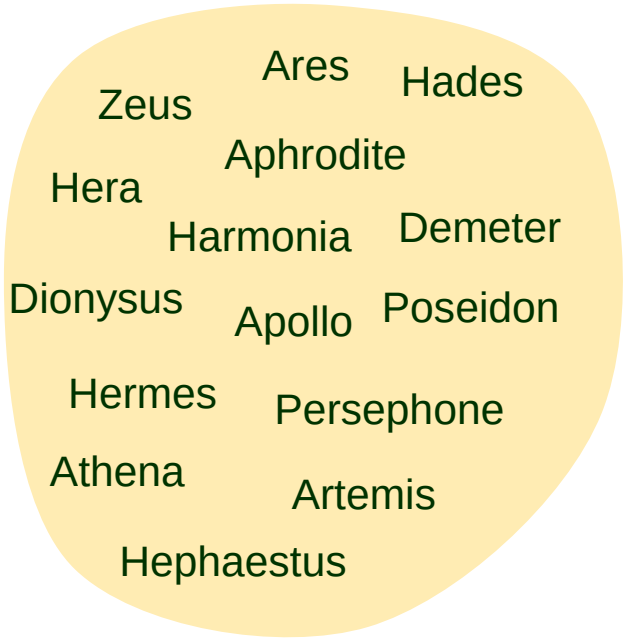
- Sentences are true with respect to a model M.
- The model M contains objects (called *domain elements*) and interpretations of symbols.
  - constant symbols → objects in domain D
  - predicate symbols → relations
  - function symbols → functional relations
- Each predicate $P(x_1, \ldots, x_k)$ is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

II. 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)

$\forall x\ Circle(x) \Rightarrow Ellipse(x)$ — *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)

$\forall x\ Circle(x) \Rightarrow Ellipse(x)$ — *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)

$\forall x\ \mathit{Circle}(x) \Rightarrow \mathit{Ellipse}(x)$ — *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)

III. Using FOL

Knowledge engineering represents 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.)

III. Using FOL

Knowledge engineering represents 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(Swan), *Bird*(Crane), *Bird*(Parrot),

- Quantified query

  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.

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

The Kinship Domain

Kinship relations are represented by binary predicates.

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

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

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

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

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.

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

// 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.

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

// 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.

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

// entailed by

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

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.

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

// 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.

$\forall x\ \mathit{Person}(x) \Rightarrow \cdots$ // partial specification of

$\forall x\ \cdots \Rightarrow \mathit{Person}(x)$ // properties

IV. Domain of Natural Numbers

Describe the theory of natural numbers using merely:

- one constant symbol, 0
- one function symbol, S (successor)
- one predicate, *NatNum*

IV. Domain of Natural Numbers

Describe the theory of natural numbers using merely:

- one constant symbol, 0
- one function symbol, S (successor)
- one predicate, *NatNum*

♦ Recursive definition:

$NatNum(0)$

$\forall n\ NatNum(n) \Rightarrow NatNum(S(n))$

IV. Domain of Natural Numbers

Describe the theory of natural numbers using merely:

- one constant symbol, 0
- one function symbol, S (successor)
- one predicate, *NatNum*

♦ Recursive definition:

$NatNum(0)$

$\forall n\ NatNum(n) \Rightarrow NatNum(S(n))$

♦ Axioms to constrain the successor function:

$\forall n\ \ 0 \neq S(n)$

$\forall m, n\ \ m \neq n \Rightarrow S(m)\ \neq S(n)$

Defining Addition

$\forall m\ \mathit{NatNum}(m) \Rightarrow +(0, m) = m$

Defining Addition

$$\forall m \; \mathit{NatNum}(m) \Rightarrow +(0, m) = m$$

prefix notation

Defining Addition

$\forall m\ \mathit{NatNum}(m) \Rightarrow +(0, m) = m$

prefix notation

Use infix notation
for readability.

$\forall m\ \mathit{NatNum}(m) \Rightarrow 0 + m = m$

Defining Addition

$\forall m\ \mathit{NatNum}(m) \Rightarrow +(0, m) = m$

prefix notation

Use infix notation
for readability.

$\forall m\ \mathit{NatNum}(m) \Rightarrow 0 + m = m$

infix notation

Defining Addition

$\forall m\ NatNum(m) \Rightarrow +(0, m) = m$

prefix notation

Use infix notation
for readability.

$\forall m\ NatNum(m) \Rightarrow 0 + m = m$

infix notation

$\forall m, n\ NatNum(m) \wedge NatNum(n) \Rightarrow +(S(m), n) = S(+(m, n))$

Defining Addition

$\forall m\ \mathit{NatNum}(m) \Rightarrow +(0, m) = m$

prefix notation

Use infix notation for readability.

$\forall m\ \mathit{NatNum}(m) \Rightarrow 0 + m = m$

infix notation

$\forall m, n\ \mathit{NatNum}(m) \wedge \mathit{NatNum}(n) \Rightarrow +(S(m), n) = S(+(m, n))$

Write $S(n)$ as $n + 1$.

$\forall m, n\ \mathit{NatNum}(m) \wedge \mathit{NatNum}(n) \Rightarrow (m + 1) + n = (m + n) + 1$

Defining Addition

$\forall m\ \mathit{NatNum}(m) \Rightarrow +(0, m) = m$

prefix notation

$\forall m\ \mathit{NatNum}(m) \Rightarrow 0 + m = m$

infix notation

Use infix notation for readability.

Syntactic sugar: an extension to the standard syntax that does not change the semantics but improves readability.

$\forall m, n\ \mathit{NatNum}(m) \wedge \mathit{NatNum}(n) \Rightarrow +(S(m), n) = S(+(m, n))$

Write $S(n)$ as $n + 1$.

$\forall m, n\ \mathit{NatNum}(m) \wedge \mathit{NatNum}(n) \Rightarrow (m + 1) + n = (m + n) + 1$

Defining Addition

$\forall m\ NatNum(m) \Rightarrow +(0, m) = m$

prefix notation

Use infix notation for readability.

Syntactic sugar: an extension to the standard syntax that does not change the semantics but improves readability.

$\forall m\ NatNum(m) \Rightarrow 0 + m = m$

infix notation

$\forall m, n\ NatNum(m) \wedge NatNum(n) \Rightarrow +(S(m), n) = S(+(m, n))$

Write $S(n)$ as $n + 1$.

$\forall m, n\ NatNum(m) \wedge NatNum(n) \Rightarrow (m + 1) + n = (m + n) + 1$

Syntactic sugar: $\forall m \forall n$ according to syntax

Domain of Sets

Syntatic sugar:

- {} for the empty set
- One unary predicate, *Set*

Domain of Sets

Syntatic sugar:

- {} for the empty set
- One unary predicate, *Set*
- Binary predicate, $\in$

  e.g., $x \in s$ (x is a member of set s)

Domain of Sets

Syntatic sugar:

- $\{\}$ for the empty set
- One unary predicate, *Set*
- Binary predicate, $\in$

  e.g., $x \in s$ (x is a member of set s)

- Binary predicate, $\subseteq$

  e.g., $s_1 \subseteq s_2$ (set s_1 is a subset of set s_2)

Domain of Sets

Syntatic sugar:

- {} for the empty set
- One unary predicate, *Set*
- Binary predicate, ∈

  e.g., $x \in s$ (x is a member of set s)

- Binary predicate, ⊆

  e.g., $s_1 \subseteq s_2$ (set s_1 is a subset of set s_2)

- Binary functions, ∩ (intersection), ∪ (union), *Add*

  $s_1 \cap s_2$

Domain of Sets

Syntatic sugar:

- {} for the empty set
- One unary predicate, *Set*
- Binary predicate, ∈

  e.g., $x \in s$ (x is a member of set s)

- Binary predicate, ⊆

  e.g., $s_1 \subseteq s_2$ (set s_1 is a subset of set s_2)

- Binary functions, ∩ (intersection), ∪ (union), *Add*

  $s_1 \cap s_2$ $s_1 \cup s_2$

Domain of Sets

Syntatic sugar:

- {} for the empty set
- One unary predicate, *Set*
- Binary predicate, $\in$

  e.g., $x \in s$ (x is a member of set s)

- Binary predicate, $\subseteq$

  e.g., $s_1 \subseteq s_2$ (set s_1 is a subset of set s_2)

- Binary functions, $\cap$ (intersection), $\cup$ (union), *Add*

  $s_1 \cap s_2$ $s_1 \cup s_2$ $Add(x, s)$

  the set resulting from add element x to set s

Eight Axioms for Sets

- A set is either an empty set or made by adding something to a set.

$$\forall s\ \mathit{Set}(s) \Leftrightarrow (s = \{\}) \vee (\exists x, s_0\ \ \mathit{Set}(s_0) \wedge s = \mathit{Add}(x, s_0))$$

Eight Axioms for Sets

- A set is either an empty set or made by adding something to a set.

$$\forall s\ Set(s) \Leftrightarrow (s = \{\}) \vee (\exists x, s_0\ \ Set(s_0) \wedge s = Add(x, s_0))$$

- The empty set has no elements added to it.

$$\neg \exists x, s\ \ Add(x, s) = \{\}$$

// equivalently, no way to decompose {}
// into a smaller set and an element

Eight Axioms for Sets

- A set is either an empty set or made by adding something to a set.

$$\forall s\ Set(s) \Leftrightarrow (s = \{\}) \vee (\exists x, s_0\ \ Set(s_0) \wedge s = Add(x, s_0))$$

- The empty set has no elements added to it.

$$\neg\exists x, s\ \ Add(x, s) = \{\}$$

// equivalently, no way to decompose {}
// into a smaller set and an element

- Adding an element already in the set has no effect.

$$\forall x, s\ \ x \in s \Leftrightarrow s = Add(x, s)$$

Eight Axioms for Sets

- A set is either an empty set or made by adding something to a set.

$$\forall s\ Set(s) \Leftrightarrow (s = \{\}) \vee (\exists x, s_0\ \ Set(s_0) \wedge s = Add(x, s_0))$$

- The empty set has no elements added to it.

$$\neg\exists x, s\ \ Add(x, s) = \{\}$$

// equivalently, no way to decompose {}
// into a smaller set and an element

- Adding an element already in the set has no effect.

$$\forall x, s\ \ \ x \in s \Leftrightarrow s = Add(x, s)$$

- The only members of a set are the added elements.

Eight Axioms for Sets

- A set is either an empty set or made by adding something to a set.

$$\forall s\ Set(s) \Leftrightarrow (s = \{\}) \vee (\exists x, s_0\ \ Set(s_0) \wedge s = Add(x, s_0))$$

- The empty set has no elements added to it.

$$\neg\exists x, s\ \ Add(x, s) = \{\}$$

// equivalently, no way to decompose {}
// into a smaller set and an element

- Adding an element already in the set has no effect.

$$\forall x, s\ \ x \in s \Leftrightarrow s = Add(x, s)$$

- The only members of a set are the added elements.

$$\forall x, s\ \ x \in s \Leftrightarrow \exists y, s_2\ \ (s = Add(y, s_2) \wedge (x = y \vee x \in s_2))$$

// expressed recursively

Eight Axioms for Sets

- A set is either an empty set or made by adding something to a set.

$$\forall s\ Set(s) \Leftrightarrow (s = \{\}) \vee (\exists x, s_0\ \ Set(s_0) \wedge s = Add(x, s_0))$$

- The empty set has no elements added to it.

$$\neg\exists x, s\ \ Add(x, s) = \{\}$$

// equivalently, no way to decompose {}
// into a smaller set and an element

- Adding an element already in the set has no effect.

$$\forall x, s\ \ x \in s \Leftrightarrow s = Add(x, s)$$

- The only members of a set are the added elements.

$$\forall x, s\ \ x \in s \Leftrightarrow \exists y, s_2\ \ (s = Add(y, s_2) \wedge (x = y \vee x \in s_2))$$

// expressed recursively

must be true at some recursion level

Set Axioms (cont'd)

- Subset relationship

$$\forall s_1, s_2 \quad s_1 \subseteq s_2 \Leftrightarrow (\forall x \;\; x \in s_1 \Rightarrow x \in s_2)$$

Set Axioms (cont'd)

- Subset relationship

$$\forall s_1, s_2 \quad s_1 \subseteq s_2 \Leftrightarrow (\forall x \;\; x \in s_1 \Rightarrow x \in s_2)$$

- Set equality

$$\forall s_1, s_2 \quad s_1 = s_2 \Leftrightarrow (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1)$$

Set Axioms (cont'd)

- Subset relationship

$$\forall s_1, s_2 \quad s_1 \subseteq s_2 \Leftrightarrow (\forall x \;\; x \in s_1 \Rightarrow x \in s_2)$$

- Set equality

$$\forall s_1, s_2 \quad s_1 = s_2 \Leftrightarrow (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1)$$

- Intersection

$$\forall x, s_1, s_2 \quad x \in s_1 \cap s_2 \Leftrightarrow (x \in s_1 \wedge x \in s_2)$$

Set Axioms (cont'd)

- Subset relationship

$$\forall s_1, s_2 \quad s_1 \subseteq s_2 \Leftrightarrow (\forall x \;\; x \in s_1 \Rightarrow x \in s_2)$$

- Set equality

$$\forall s_1, s_2 \quad s_1 = s_2 \Leftrightarrow (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1)$$

- Intersection

$$\forall x, s_1, s_2 \quad x \in s_1 \cap s_2 \Leftrightarrow (x \in s_1 \wedge x \in s_2)$$

- Union

$$\forall x, s_1, s_2 \quad x \in s_1 \cup s_2 \Leftrightarrow (x \in s_1 \vee x \in s_2)$$

Equality

Axiomatize equality: write down sentences about equality in the *KB.*

- One for each basic axioms.

$\forall x \;\; x = x$ // reflexive

$\forall x, y \;\; x = y \Leftrightarrow y = x$ // symmetric

$\forall x, y, z \;\; x = y \wedge y = z \Rightarrow x = z$ // transitive

Equality

Axiomatize equality: write down sentences about equality in the *KB.*

- One for each basic axioms.

$\forall x \;\; x = x$ // reflexive

$\forall x, y \;\; x = y \Leftrightarrow y = x$ // symmetric

$\forall x, y, z \;\; x = y \wedge y = z \Rightarrow x = z$ // transitive

- One for each predicate.

$\forall x, y \;\; x = y \Rightarrow (P_1(x) \Leftrightarrow P_1(y))$

$\vdots$

Equality

Axiomatize equality: write down sentences about equality in the *KB.*

- One for each basic axioms.

$\forall x \;\; x = x$ // reflexive

$\forall x, y \;\; x = y \Leftrightarrow y = x$ // symmetric

$\forall x, y, z \;\; x = y \wedge y = z \Rightarrow x = z$ // transitive

- One for each predicate.

$\forall x, y \;\; x = y \Rightarrow (P_1(x) \Leftrightarrow P_1(y))$

$\vdots$

- One for each function.

$\forall x, y \;\; x = y \Rightarrow \;\; (F_1(x) = F_1(y))$

$\vdots$