September 28 Slides

Elizabeth Orrico

Predicates

“A function that evaluates to True or False”

“A proposition missing the noun(s)”

“A proposition template”

Universal Quantifier (∀)

∀ = "for all" or "given any"
It expresses that a **propositional function** can be satisfied by **every member of the domain**

Domain: People   L(x, y) = *x loves y*

∀x L(x, Raymond) means ???

Universal Quantifier (∀)

∀ = "for all" or "given any"

It expresses that a **propositional function** can be satisfied by **every member of the domain.**

Domain: People   L(x, y) = *x loves y*

∀x L(x, Raymond) means "For all people x, each one loves Raymond"

"Given any person x, that person loves Raymond"

"Every person loves Raymond"

Existential Quantifier (∃)

∃ = "there exists", "there is at least one", or "for some"
It expresses that a **propositional function** can be satisfied by **at least one member of the domain.**

Domain: People   L(x, y) = *x loves y*

¬∃x L(x, Chris) means “There does not exist one person who loves Chris”

∃ and ∀

Domain: People $L(x, y)$ = *x loves y*

¬∃x L(x, Chris) means "There does not exist one person who loves Chris"
∀x ¬L(x, Chris) means "For all people, each one does not love Chris"

$$\neg \exists x\, L(x, \text{Chris}) \equiv \forall x\, \neg L(x, \text{Chris})$$

∃ and ∀ -- Your Turn!

Domain: People   L(x, y) = *x loves y*

¬∃x L(x, Chris) means "There does not exist one person who loves Chris"
∀x ¬L(x, Chris) means "For all people, each one does not love Chris"

$$\neg \exists x\, L(x, \text{Chris}) \equiv \forall x\, \neg L(x, \text{Chris})$$

∀x L(x, Raymond) ≡ **[using ∃]**

∃ and ∀ -- Your Turn!

Domain: People   L(x, y) = *x loves y*

$\neg \exists x\ L(x, \text{Chris}) \equiv \forall x\ \neg L(x, \text{Chris})$

$\forall x\ L(x, \text{Raymond}) \equiv \neg \exists x\ \neg L(x, \text{Raymond})$

∃ and ∀ -- Your Turn!

Domain: {Daniel, Melanie, Josh, Jenn}

How could we check in code that everyone hates Chris?

∃ and ∀ -- Your Turn!

Domain: {Daniel, Melanie, Josh, Jenn}

How could we check in code that everyone hates Chris?

(psuedocode)

```
hatesChris = True
for p in {Daniel, Melanie, Josh, Jenn}:
    if L(p, Chris):
        hatesChris = False
    End
Return hatesChris
```

∃ and ∀ -- Your Turn!

Domain: {Daniel, Melanie, Josh, Jenn}

How could we check in code that everyone loves Raymond?

```
lovesRaymond = True
for p in {Daniel, Melanie, Josh, Jenn}:
    if not L(p, Raymond):
        lovesRaymond = False
    End
Return lovesRaymond
```

∃ and ∀ -- Your Turn!

Domain: {Daniel, Melanie, Josh, Jenn}

Everyone hates Chris

How could we write this out as a boolean expression?

$$\neg \exists x\, L(x, \text{Chris}) \equiv \forall x\, \neg L(x, \text{Chris})$$

∃ and ∀ -- Your Turn!

Domain: {Daniel, Melanie, Josh, Jenn}

Everyone hates Chris

How could we write this out as a boolean expression?

$$\forall x\ \neg L(x, \text{Chris})$$

$$\neg L(\text{Daniel, Chris}) \land \neg L(\text{Melanie, Chris}) \land \neg L(\text{Josh, Chris}) \land \neg L(\text{Jenn, Chris})$$

∃ and ∀ -- Your Turn!

Domain: {Daniel, Melanie, Josh, Jenn}

Everyone hates Chris

What about the "There exists version"?

$$\forall x\ \neg L(x, \text{Chris})$$

$$\neg L(\text{Daniel, Chris}) \land \neg L(\text{Melanie, Chris}) \land \neg L(\text{Josh, Chris}) \land \neg L(\text{Jenn, Chris})$$

$$\neg \exists x\ L(x, \text{Chris})$$

$$\neg(L(\text{Daniel, Chris}) \lor L(\text{Melanie, Chris}) \lor L(\text{Josh, Chris}) \lor L(\text{Jenn, Chris}))$$

∃ and ∀ -- Your Turn!

Domain: {Daniel, Melanie, Josh, Jenn}

By DeMorgan's Law, these are equivalent!

$$\neg L(\text{Daniel}, \text{Chris}) \land \neg L(\text{Melanie}, \text{Chris}) \land \neg L(\text{Josh}, \text{Chris}) \land \neg L(\text{Jenn}, \text{Chris})$$

$$\equiv$$

$$\neg(L(\text{Daniel}, \text{Chris}) \lor L(\text{Melanie}, \text{Chris}) \lor L(\text{Josh}, \text{Chris}) \lor L(\text{Jenn}, \text{Chris}))$$

∃ and ∀ -- Your Turn!

Domain: {Daniel, Melanie, Josh, Jenn}

Associate "for all" with AND's since it becomes false if just one truth value is false

∀x ¬L(x, Chris) = ¬L(Daniel, Chris) ∧ ¬L(Melanie, Chris) ∧¬L(Josh, Chris) ∧¬L(Jenn, Chris)

Associate "there exists" with OR's since it becomes true if just one truth value is true

¬∃x L(x, Chris) = ¬(L(Daniel, Chris) ∨ L(Melanie, Chris) ∨ L(Josh, Chris) ∨ L(Jenn, Chris))

Quick Intro to Multiple Quantifiers:

Domain: People   L(x, y) = *x loves y*

Are these equivalent?

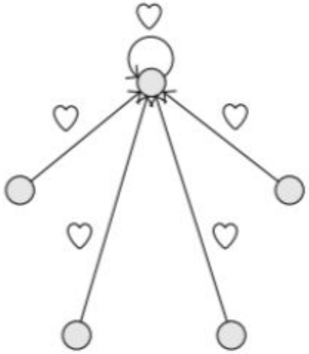
$$\exists y \forall x\, L(x,y) \overset{?}{\equiv} \forall x \exists y\, L(x,y)$$

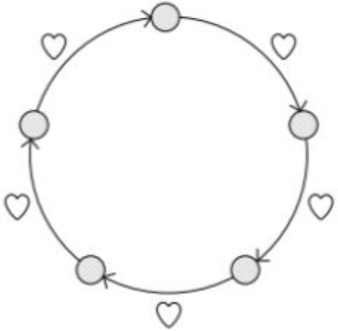
Quick Intro to Multiple Quantifiers:

Domain: People   L(x, y) = *x loves y*

Are these equivalent?

$\exists y \forall x\, L(x,y)$ **is not equivalent to** $\forall x \exists y\, L(x,y)$





Quick Intro to Multiple Quantifiers:

Domain: People   L(x, y) = *x loves y*

Are these equivalent?

$\exists y \forall x\, L(x,y)$ **is not equivalent to** $\forall x \exists y\, L(x,y)$

Remember

When you are dealing with mixed quantifiers, the order is very important. $\forall x \exists y\, R(x, y)$ is not logically equivalent to $\exists y \forall x\, R(x, y)$.

How can we ensure that this y person doesn't love themself?

Domain: People   L(x, y) = *x loves y*

$\exists y \forall x \, L(x,y)$???