Logik für Informatiker
Logic for computer scientists

Multiple Quantifiers

Till Mossakowski

WiSe 2009/10

Multiple quantifiers

$$\forall x \exists y \; Likes(x, y)$$

is very different from

$$\exists y \forall x \; Likes(x, y)$$

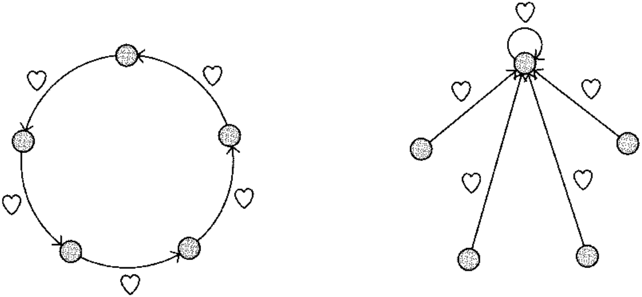


Figure 11.1: A circumstance in which $\forall x\, \exists y\, \mathsf{Likes}(x, y)$ holds versus one in which $\exists y\, \forall x\, \mathsf{Likes}(x, y)$ holds. It makes a big difference to someone!

Prenex Normal Form

Goal: shift all quantifiers to the top-level

Rules for conjunctions and disjunctions

$$(\forall x P) \wedge Q \leadsto \forall x(P \wedge Q) \qquad (\exists x P) \wedge Q \leadsto \exists x(P \wedge Q)$$

$$P \wedge (\forall x Q) \leadsto \forall x(P \wedge Q) \qquad P \wedge (\exists x Q) \leadsto \exists x(P \wedge Q)$$

$$(\forall x P) \vee Q \leadsto \forall x(P \vee Q) \qquad (\exists x P) \vee Q \leadsto \exists x(P \vee Q)$$

$$P \vee (\forall x Q) \leadsto \forall x(P \vee Q) \qquad P \vee (\exists x Q) \leadsto \exists x(P \vee Q)$$

Prenex Normal Form (cont'd)

Rules for negations, implications, equivalences

$\neg\forall x P \rightsquigarrow \exists x(\neg P)$ $\qquad$ $\neg\exists x P \rightsquigarrow \forall x(\neg P)$

$(\forall x P) \rightarrow Q \rightsquigarrow \exists x(P \rightarrow Q)$ $\qquad$ $(\exists x P) \rightarrow Q \rightsquigarrow \forall x(P \rightarrow Q)$

$P \rightarrow (\forall x Q) \rightsquigarrow \forall x(P \rightarrow Q)$ $\qquad$ $P \rightarrow (\exists x Q) \rightsquigarrow \exists x(P \rightarrow Q)$

$P \leftrightarrow Q \rightsquigarrow (P \rightarrow Q) \wedge (Q \rightarrow P)$

Prenex Normal Form: example

What is the prenex normal form of

$$\exists x Cube(x) \rightarrow \forall y Small(y)$$

Proof methods for quantifiers

Universal elimination

Universal statments can be instantiated to any object.

From $\forall x S(x)$, we may infer $S(c)$.

Existential introduction

If we have established a statement for an instance, we can also establish the corresponding existential statement.

From $S(c)$, we may infer $\exists x S(x)$.

Example

$$\forall x[\mathsf{Cube}(x) \rightarrow \mathsf{Large}(x)]$$
$$\forall x[\mathsf{Large}(x) \rightarrow \mathsf{LeftOf}(x, b)]$$
$$\mathsf{Cube}(d)$$

$$\exists x[\mathsf{Large}(x) \wedge \mathsf{LeftOf}(x, b)]$$

Existential instantiation (elimination)

From $\exists x S(x)$, we can infer $S(c)$, if c is a new name not used otherwise.
Example: Scotland Yard searched a serial killer. The did not know who he was, but for their reasoning, they called him *"Jack the ripper"*.
This would have been an unfair procedure if there had been a real person named Jack the ripper.

Example

$$\forall x[\mathrm{Cube}(x) \rightarrow \mathrm{Large}(x)]$$
$$\forall x[\mathrm{Large}(x) \rightarrow \mathrm{LeftOf}(x, b)]$$
$$\exists x \mathrm{Cube}(x)$$

$$\exists x[\mathrm{Large}(x) \land \mathrm{LeftOf}(x, b)]$$

Universal generalization (introduction)

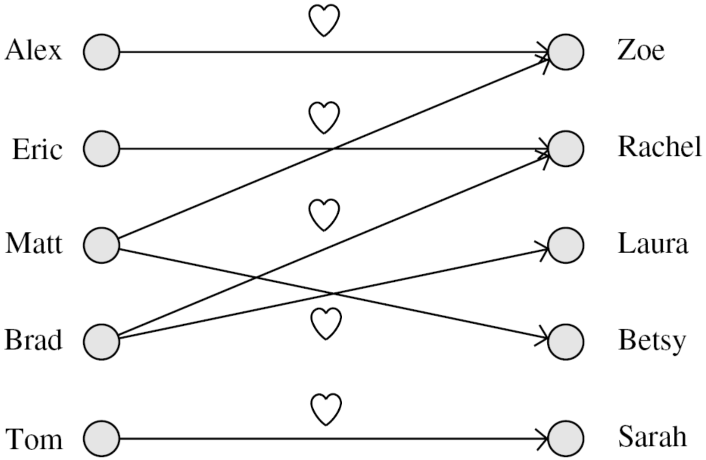
If we introduce a new name c that is not used elsewhere, and can prove $S(c)$, then we can also infer $\forall x S(x)$.
Example:
Theorem Every even number greater zero is the sum of two odd numers.
Proof Let $n > 0$ be even, i.e. $n = 2m$ with $m > 0$. If m is odd, then $m + m = n$ does the job. If m is even, consider $(m - 1) + (m + 1) = n$.

Arguments involving multiple quantifiers

$$
\begin{array}{|l}
\exists y[\mathrm{Girl}(y) \wedge \forall x(\mathrm{Boy}(x) \rightarrow \mathrm{Likes}(x, y))] \\
\hline
\forall x[\mathrm{Boy}(x) \rightarrow \exists y(\mathrm{Girl}(y) \wedge \mathrm{Likes}(x, y))]
\end{array}
$$

$$
\begin{array}{|l}
\forall x[\mathrm{Boy}(x) \rightarrow \exists y(\mathrm{Girl}(y) \wedge \mathrm{Likes}(x, y))] \\
\hline
\exists y[\mathrm{Girl}(y) \wedge \forall x(\mathrm{Boy}(x) \rightarrow \mathrm{Likes}(x, y))]
\end{array}
$$

A (counter)example



Common Algebraic Specification Language

- strongly typed; types are declated using the *sort* keyword
  sort Blocks
- predicates have to be declared with their types
  preds Cube, Dodec, Tet : Blocks
- propositional variables = nullary predicates
  preds A,B,C : ()
- constants have to be declared with their types
  ops a,b,c : Blocks

Example CASL specification: blocks

```
spec Tarski1 =     sort Blocks
  preds Cube, Dodec, Tet, Small, Medium, Large : Blocks
  ops a,b,c : Blocks
  . not a=b . not a=c  . not b=c
  . Small(a) => Cube(a)     %(small_cube_a)%
  . Small(a) <=> Small(b)   %(small_a_b)%
  . Small(b) \/ Medium(b)   %(small_medium_b)%
  . Medium(b) => Medium(c)  %(medium_b_c)%
  . Medium(c) => Tet(c)     %(medium_tet_c)%
  . not Tet(c)              %(not_tet_c)%
  . Cube(a)                 %(cube_a)% %implied
  . Cube(b)                 %(cube_b)% %implied
```

Exercises

- chapter 10: 10.20 to 10.31
- chapters 11 and 12
- additional exercise (grade 1): write a complete axiomatization of Tarski's World in CASL.