Multiple Quantifiers: Syntax and Semantics

Governance

- We have stated that a quantifier governs the shortest full sentence following it.
- Thus far, we have restricted our attention to sentences all of whose components are governed by a single quantifier, if any.
- In the rest of the course, we will consider sentences some of whose components are governed by more than one quantifier.
  - (∀x)(∃y)Lxy.
  - ‘(∃y)’ governs ‘Lxy’.
  - ‘(∀x)’ governs ‘(∃y)Lxy’.
- The use of multiple quantifiers greatly increases the expressive power of Predicate Logic.

Scope and Bound Variables

- We have stated that a quantifer binds all occurrences of its variable in the sentence it governs.
  - ‘(∃y)’ governs ‘Lxy’ and so it binds ‘y’ in ‘(∃y)Lxy’.
- We now state that the **scope** of a quantifier is the sentence governed by the quantifier.
- We state in a preliminary way that a variable **u** is **bound** if and only if it occurs in the scope of a quantifier (∀**u**) or (∃**u**) containing it.
  - ‘y’ is bound by ‘(∃y)’ in ‘(∃y)Lxy’.

Free Variables and Binding

- A variable is **free** if and only if it is not bound.
  - ‘y’ is free in ‘(∃x)Lxy’.
- A problem case arises when a variable occurs within the scope of two quantifiers containing it.
  - (∃x)[(∀x)Lxa ⊃ Lxb]
- Is the ‘x’ in ‘Lxa’ bound by ‘(∃x)’, by ‘(∀x)’, or both?
- We solve the problem by saying that a quantifier binds only the free variables in the shortest full sentence following it.
  - ‘x’ in ‘Lxa’ is bound by ‘(∀x)’ only, since it is not free in ‘(∀x)Lxa ⊃ Lxb’.
- It may be easiest to think of binding as occurring at the initial application of the quantifier as the sentence is being built up.

Substitution Instances with Multiple Quantification

- The original definition of a substitution instance is inadequate when a variable occurs within the scope of two quantifiers containing it.
- Using the name 'c', how do we instantiate '(∃x)[(∀x)Lxa ⊃ Lxb]'?
  - (∀x)Lca ⊃ Lcb?
  - (∀x)Lxa ⊃ Lcb?
- The answer lies in the fact that the initial 'x' in '(∀x)Lxa ⊃ Lxb' is bound to '(∀x)' and thus should not be replaced by a name.
- Thus a substitution instance is formed when all free occurrences of the variable occuring in sentence governed by the quantifier are replaced by names (or more generally, by constant terms).

Semantics for Sentences with Multiple Quantifiers

- The interpretation of sentences with multiple quantifiers is an extension of the semantics for sentences with a single quantifier.
- A variable assignment **d** satisifies '(∀x)(∀y)Lxy' just in case for all **o** in the domain, **d**[**o**/x] satisfies '(∀y)Lxy'.
- The x-variant **d**[**o**/x] satisfies '(∀x)Lxy' just in case for all **o** in the domain, **d**[**o**/x, **o**/y] satisfies 'Lxy'.
- In other words, the y-variants of **d**[**o**/x] for all **o** in the domain satisfy 'Lxy'.
- Thus, we use variants of variants (of variants . . .) until we arrive at atomic sentences.

An Example

- I = {D, v}, D = {Adam, Eve}, v(L) = {⟨Adam, Adam⟩, ⟨Adam, Eve⟩}
- The target sentence is '(∃x)(∀y)Lxy'.
- We can see by inspection that the sentence is true in I, since there is one member of the domain, Adam, who bears the relation indicated by 'L' to every member of the domain.
- That is, d[Adam/x, Adam/y] satisfies 'Lxy', as does d[Adam/x, Eve/y].
- So, d[Adam/x] satsfies '(∀y)Lxy', in which case there is an x-variant of d which satisfies the open sentence, and d satisfies '(∃x)(∀y)Lxy'.
- Because the choice of d was arbitrary, the sentence is satisifed by all variable assignments and hence is true.

Another Example

- I = {D, v}, D = {Adam, Eve}, v(L) = {⟨Adam, Adam⟩, ⟨Adam, Eve⟩}
- The target sentence is '(∀x)(∃y)Lxy'.
- We can see by inspection that the sentence is false in I, since there is one member of the domain, Eve, who does not bear the relation indicated by 'L' to at least one member of the domain.
- That is, d[Eve/x, Adam/y] does not satisfy 'Lxy', nor does d[Eve/x, Eve/y].
- So, d[Eve/x] does not satisfy '(∃y)Lxy', in which case there is an x-variant of d which does not satisfy the open sentence, and d does not satisfy '(∀x)(∃y)Lxy'.
- Because the choice of d was arbitrary, the sentence is satisifed by no variable assignments and hence is false.

Multiple Quantifiers in Substitutional Semantics

- Recall that a universally quantified sentences is true in the substitutional semantics just in case all its substitution instances are true, and an existentially quantified sentence is true just in case at least one of its substitution instances is true.
- For multiple quantification, these truth-definitions are iterated and involve substitution instances of substitution instances.
- For example '(∀x)(∃y)Lxy' is true in an interpretation **I** if and only if all substitution instances '(∃y)Lay', '(∃y)Lby', etc. are true in the interpetation.
- This holds just in case at least one substitution instance of (∃y)Lay', i.e., 'Laa,', 'Lab', etc. is true in the interpretation, and so for the substitution instances of '(∃y)Lby', etc.

Logical Equivalence

- Two closed sentences of Predicate Logic are logically equivalent if and only if they have the same truth value in all interpretations.
- For example, '(∀x)∼Fx' is logically equivalent to '∼(∃x)Fx'.
- An interpretation **I** makes '(∀x)∼Fx' true if and only if for all **d**, **d** satisifies '(∀x)∼Fx'.
- This holds just in case for all **o** in the domain of **I**, **d**[**o**/x] satisfies '∼Fx', and thus **d**[**o**/x] does not satisfy 'Fx' for all **o** in the domain.
- But this holds just in case **d** does not satisfy '(∃x)Fx', and **d** satisfies '∼(∃x)Fx, which holds for all **d**, QED.

Logical Equivalence for Open Sentences

- We should like to be able to say that '(∀x)∼Lxy' is logically equivalent to '∼(∃x)Lxy'.
- However, our definition of logical equivalence covers only closed sentences.
- It is easy to extend the notion of logical equivalence to open sentences.
- Two open sentences are logically equivalent if and only if they are satisfied by exactly the same variable assignments in any interpretation.

An Example

- '(∀x)∼Lxy' is logically equivalent to '∼(∃x)Lxy'.
- A variable assignment **d**[$\mathbf{o}_i$/y] satisfies '(∀x)∼Lxy' if and only if for any **o** in the domain, **d**[$\mathbf{o}_i$/x, **o**/y] satisfies '∼Lxy'.
- This holds if and only if for all **o** in the domain, **d**[$\mathbf{o}_i$/x, **o**/y] does not satisfy 'Lxy'.
- This holds if and only if **d**[$\mathbf{o}_i$/x] does not satisfy '(∃x)Lxy'.
- This holds if and only if **d**[$\mathbf{o}_i$/x] satisfies '∼(∃x)Lxy', QED.

A Law of Substitution

- If **X** is the result of substituting a logically equivalent sentence for a sentence occurring in **Y**, then **X** and **Y** are logically equivalent.
- For example:
  - '(∃x)(∀y)∼Lxy' is equivalent to '(∃x)∼(∃y)Lxy'.
  - '(∀x)(∀y)Fxy' is equivalent to '(∀x)(∀z)Fxz'.
  - '(∀x)(∃y)Fxy' is equivalent to '(∀y)(∃x)Fyx'.
- This results holds because of the fact that the substituting and substituted sentences are satisfied by exactly the same variable assignments, so the truth-values of **X** and **Y** are unaffected by the substitution.

Logical Equivalence and Substitutional Semantics

- Logical equivalence is not straightforwardly defined in substitutional semantics.
- The problem is that open sentences such as '(∃x)∼Lxy' have no truth-value and are not subsitution instances of any sentence.

- One could define a “name-replacement” as of an open sentence as a sentence resulting from replacing the occurrences of a free variable with an name, as ‘$(\exists x)\sim Lxa$’.
- Then two open sentences with one free variable are logically equivalent just in case all name-replacements have the same truth-value.
  - ‘$(\exists x)\sim Lxa$’ is true if and only if ‘$\sim(\forall x)Lxa$’ is true, etc.
- Alternatively, one could hold that two open sentences are equivalent if and only if the universal generalization of their biconditional is logically true.
  - $\vDash (\forall y)[(\exists x)\sim Lxy \equiv \sim(\forall x)Lxy]$.