Faculty of Philosophy

Formal Logic

Lecture 14

Peter Smith

- Our next task
- Basic subject/predicate structure
- How not to add quantifiers

Divide and rule

- The intended content of arguments presented in the vernacular is often obscure; the individual premisses and/or conclusion often unclear or ambiguous.

Divide and rule

- The intended content of arguments presented in the vernacular is often obscure; the individual premisses and/or conclusion often unclear or ambiguous.
- So, presented 'raw', everyday arguments often leave us with a double task; clarifying what is being said, and then assessing the clarified argument.

Divide and rule

- The intended content of arguments presented in the vernacular is often obscure; the individual premisses and/or conclusion often unclear or ambiguous.
- So, presented ‘raw’, everyday arguments often leave us with a double task; clarifying what is being said, and then assessing the clarified argument.
- Our way – the standard way – of handling this double task, already illustrated in the case of basic propositional logic:

Divide and rule

- The intended content of arguments presented in the vernacular is often obscure; the individual premisses and/or conclusion often unclear or ambiguous.
- So, presented 'raw', everyday arguments often leave us with a double task; clarifying what is being said, and then assessing the clarified argument.
- Our way – the standard way – of handling this double task, already illustrated in the case of basic propositional logic:
  1. Clarify at least the relevant logical structure of premisses and conclusion by regimenting the argument into an appropriate formalized language.

Divide and rule

- The intended content of arguments presented in the vernacular is often obscure; the individual premisses and/or conclusion often unclear or ambiguous.
- So, presented 'raw', everyday arguments often leave us with a double task; clarifying what is being said, and then assessing the clarified argument.
- Our way – the standard way – of handling this double task, already illustrated in the case of basic propositional logic:
  1. Clarify at least the relevant logical structure of premisses and conclusion by regimenting the argument into an appropriate formalized language.
  2. Assess the argument as couched in the formalized language.

Formalized languages

- The formalized languages of logic are intended to be contentful languages in which real arguments can be presented. Using a formalized language isn't just playing a game with uninterpreted symbols.

Formalized languages

- The formalized languages of logic are intended to be contentful languages in which real arguments can be presented. Using a formalized language isn't just playing a game with uninterpreted symbols.
- So to specify a formal language, we give its syntax (rules for defining its wffs) *and* its semantics (rules that determine what the wffs mean).

Formalized languages

- The formalized languages of logic are intended to be contentful languages in which real arguments can be presented. Using a formalized language isn't just playing a game with uninterpreted symbols.
- So to specify a formal language, we give its syntax (rules for defining its wffs) *and* its semantics (rules that determine what the wffs mean).
- In a properly designed formal language,

Formalized languages

- The formalized languages of logic are intended to be contentful languages in which real arguments can be presented. Using a formalized language isn't just playing a game with uninterpreted symbols.
- So to specify a formal language, we give its syntax (rules for defining its wffs) *and* its semantics (rules that determine what the wffs mean).
- In a properly designed formal language,
    1. syntactic form determines semantic structure,

Formalized languages

- The formalized languages of logic are intended to be contentful languages in which real arguments can be presented. Using a formalized language isn't just playing a game with uninterpreted symbols.
- So to specify a formal language, we give its syntax (rules for defining its wffs) *and* its semantics (rules that determine what the wffs mean).
- In a properly designed formal language,
  1. syntactic form determines semantic structure,
  2. the rules determine unique interpretation for each wff.

The language QL

- Our next main task is to develop the formal language QL for representing propositions whose logically relevant structure includes not just the familiar connectives but quantifiers (like 'all', 'each', 'some', 'none', 'only', . . .).

The language QL

- Our next main task is to develop the formal language QL for representing propositions whose logically relevant structure includes not just the familiar connectives but quantifiers (like 'all', 'each', 'some', 'none', 'only', . . .).
- We'll need to define the syntax of QL: which strings of symbols are well-formed formulae (wffs)?

The language QL

- Our next main task is to develop the formal language QL for representing propositions whose logically relevant structure includes not just the familiar connectives but quantifiers (like 'all', 'each', 'some', 'none', 'only', . . .).
- We'll need to define the syntax of QL: which strings of symbols are well-formed formulae (wffs)?
- We'll need to define the semantics of QL: how are we to interpret the wffs?

Arguments involving quantifiers

- Consider arguments like

Arguments involving quantifiers

- Consider arguments like

  All good philosophers like logic
  Bertrand is a good philosopher
  Bertrand likes logic

Arguments involving quantifiers

- Consider arguments like

  All good philosophers like logic
  Bertrand is a good philosopher
  Bertrand likes logic

  Every analytic philosopher admires some logician
  No logician is admired by Jean-Paul
  Jean-Paul is not an analytic philosopher

Arguments involving quantifiers

- Consider arguments like

All good philosophers like logic
Bertrand is a good philosopher
Bertrand likes logic

Every analytic philosopher admires some logician
No logician is admired by Jean-Paul
Jean-Paul is not an analytic philosopher

Only Bob and Ted love Mary
Mary kissed someone who loves her
Mary kissed either Bob or Ted

Arguments involving quantifiers

- Consider arguments like

  All good philosophers like logic
  Bertrand is a good philosopher
  Bertrand likes logic

  Every analytic philosopher admires some logician
  No logician is admired by Jean-Paul
  Jean-Paul is not an analytic philosopher

  Only Bob and Ted love Mary
  Mary kissed someone who loves her
  Mary kissed either Bob or Ted

- They depend for their validity on the sub-propositional structure of the premisses and conclusions.

Arguments involving quantifiers

- Consider arguments like

  All good philosophers like logic
  Bertrand is a good philosopher
  Bertrand likes logic

  Every analytic philosopher admires some logician
  No logician is admired by Jean-Paul
  Jean-Paul is not an analytic philosopher

  Only Bob and Ted love Mary
  Mary kissed someone who loves her
  Mary kissed either Bob or Ted

- They depend for their validity on the sub-propositional structure of the premisses and conclusions.
- QL needs to have ways of representing sub-propositional structure.

- Our next task
- Basic subject/predicate structure
- How not to add quantifiers

Names, predicates – 1

- QL will need, for a start, two classes of expressions constants (or names) and predicates.
- Constants/names serve to pick out particular people/things (Bertrand, Jean-Paul, Fido, Mount Everest, the martini glass on the table, a particular water atom, the number three, . . . , any individual thing).

Names, predicates – 1

- QL will need, for a start, two classes of expressions constants (or names) and predicates.
- Constants/names serve to pick out particular people/things (Bertrand, Jean-Paul, Fido, Mount Everest, the martini glass on the table, a particular water atom, the number three, . . . , any individual thing).
- Predicates express properties and relations. Cf. the English
  - '. . . is blue'
  - '. . . is even'
  - '. . . loves . . . '
  - '. . . is shorter than . . . '
  - '. . . is between . . . and . . . '
  - '. . . is to . . . as . . . is to . . . '

Names, predicates – 2

- QL doesn't need to segment propositional clauses any finer than name/predicate structure. So we'll use the simplest possible expressions for names and predicates.

Names, predicates – 2

- QL doesn't need to segment propositional clauses any finer than name/predicate structure. So we'll use the simplest possible expressions for names and predicates.
- Constants/names: $a, b, c, l, m, n, \ldots$.

Names, predicates – 2

- QL doesn't need to segment propositional clauses any finer than name/predicate structure. So we'll use the simplest possible expressions for names and predicates.
- Constants/names: $a, b, c, l, m, n, \ldots$.
- Predicates:

Names, predicates – 2

- QL doesn't need to segment propositional clauses any finer than name/predicate structure. So we'll use the simplest possible expressions for names and predicates.
- Constants/names: $a, b, c, l, m, n, \ldots$.
- Predicates:
  - monadic: $F, G, H, \ldots$

Names, predicates – 2

- QL doesn't need to segment propositional clauses any finer than name/predicate structure. So we'll use the simplest possible expressions for names and predicates.
- Constants/names: $a, b, c, l, m, n, \ldots$.
- Predicates:
  - monadic: $F, G, H, \ldots$
  - dyadic: $L, M, \ldots$

Names, predicates – 2

- QL doesn't need to segment propositional clauses any finer than name/predicate structure. So we'll use the simplest possible expressions for names and predicates.
- Constants/names: $a, b, c, l, m, n, \ldots$.
- Predicates:
  - monadic: $F, G, H, \ldots$
  - dyadic: $L, M, \ldots$
  - triadic: $R, S, \ldots$

Names, predicates – 2

- QL doesn't need to segment propositional clauses any finer than name/predicate structure. So we'll use the simplest possible expressions for names and predicates.
- Constants/names: $a, b, c, l, m, n, \ldots$.
- Predicates:
  - monadic: $F, G, H, \ldots$
  - dyadic: $L, M, \ldots$
  - triadic: $R, S, \ldots$
  - (other polyadic predicates ...)

Names, predicates – 2

- QL doesn't need to segment propositional clauses any finer than name/predicate structure. So we'll use the simplest possible expressions for names and predicates.
- Constants/names: $a, b, c, l, m, n, \ldots$.
- Predicates:
  - monadic: $F, G, H, \ldots$
  - dyadic: $L, M, \ldots$
  - triadic: $R, S, \ldots$
  - (other polyadic predicates ...)
- NB: we shouldn't really use open-ended lists, but we'll be careless for the moment.

Names, predicates – 2

- QL doesn't need to segment propositional clauses any finer than name/predicate structure. So we'll use the simplest possible expressions for names and predicates.
- Constants/names: $a, b, c, l, m, n, \ldots$.
- Predicates:
  - monadic: $F, G, H, \ldots$
  - dyadic: $L, M, \ldots$
  - triadic: $R, S, \ldots$
  - (other polyadic predicates ...)
- NB: we shouldn't really use open-ended lists, but we'll be careless for the moment.
- NB: QL predicates all have a fixed adicity (compare ordinary language multigrade predicates like 'work well together', 'conspired to commit murder').

Atomic sentences – 1

- The most simple kind of sentence in QL – the atomic sentences – are formed by taking an *n*-place predicate and following it by *n* names.

Atomic sentences – 1

- The most simple kind of sentence in QL – the atomic sentences – are formed by taking an *n*-place predicate and following it by *n* names.
- The order here – predicate followed by name(s) – is purely conventional but utterly standard.

Atomic sentences – 1

- The most simple kind of sentence in QL – the atomic sentences – are formed by taking an *n*-place predicate and following it by *n* names.
- The order here – predicate followed by name(s) – is purely conventional but utterly standard.
- Suppose that a, b, c are names, F is a one-place predicate, L is two-place, R is three-place.
  Then the following are atomic sentences:

Atomic sentences – 1

- The most simple kind of sentence in QL – the atomic sentences – are formed by taking an *n*-place predicate and following it by *n* names.
- The order here – predicate followed by name(s) – is purely conventional but utterly standard.
- Suppose that a, b, c are names, F is a one-place predicate, L is two-place, R is three-place.
  Then the following are atomic sentences:
  - $Fa, Fb \ldots$

Atomic sentences – 1

- The most simple kind of sentence in QL – the atomic sentences – are formed by taking an n-place predicate and following it by n names.
- The order here – predicate followed by name(s) – is purely conventional but utterly standard.
- Suppose that a, b, c are names, F is a one-place predicate, L is two-place, R is three-place.
  Then the following are atomic sentences:
  - Fa, Fb ...
  - Lab, Lba, Laa ...

Atomic sentences – 1

- The most simple kind of sentence in QL – the atomic sentences – are formed by taking an *n*-place predicate and following it by *n* names.
- The order here – predicate followed by name(s) – is purely conventional but utterly standard.
- Suppose that *a*, *b*, *c* are names, *F* is a one-place predicate, *L* is two-place, *R* is three-place.
  Then the following are atomic sentences:
  - *Fa*, *Fb* ...
  - *Lab*, *Lba*, *Laa* ...
  - *Rcab*, *Raaa*, *Rbab* ...

Atomic sentences – 2

- The interpretation of an atomic sentence – a predicate followed by name(s) – is as you'd expect. The sentence says that the individuals named have the property/stand in the relation expressed by the predicate (order of names matters!).

Atomic sentences – 2

- The interpretation of an atomic sentence – a predicate followed by name(s) – is as you'd expect. The sentence says that the individuals named have the property/stand in the relation expressed by the predicate (order of names matters!).
- Suppose that:

  a names Romeo, b names Juliet, c names Rosaline.
  F means ① *is a boy*, G means ① *is girl*,
  L means ① *loves* ②, R means ① *prefers* ② *to* ③

Atomic sentences – 2

- The interpretation of an atomic sentence – a predicate followed by name(s) – is as you'd expect. The sentence says that the individuals named have the property/stand in the relation expressed by the predicate (order of names matters!).
- Suppose that:

a names Romeo, *b* names Juliet, *c* names Rosaline. *F* means ① *is a boy*, *G* means ① *is girl*, *L* means ① *loves* ②, *R* means ① *prefers* ② *to* ③

- Then:

Atomic sentences – 2

- The interpretation of an atomic sentence – a predicate followed by name(s) – is as you'd expect. The sentence says that the individuals named have the property/stand in the relation expressed by the predicate (order of names matters!).
- Suppose that:

a names Romeo, *b* names Juliet, *c* names Rosaline.
F means ① *is a boy*, *G* means ① *is girl*,
L means ① *loves* ②, *R* means ① *prefers* ② *to* ③

- Then:
  - *Fa* means Romeo is a boy

Atomic sentences – 2

- The interpretation of an atomic sentence – a predicate followed by name(s) – is as you'd expect. The sentence says that the individuals named have the property/stand in the relation expressed by the predicate (order of names matters!).
- Suppose that:

a names Romeo, *b* names Juliet, *c* names Rosaline. *F* means ① *is a boy*, *G* means ① *is girl*, *L* means ① *loves* ②, *R* means ① *prefers* ② *to* ③

- Then:
  - *Fa* means Romeo is a boy
  - *Lab* means Romeo loves Juliet

Atomic sentences – 2

- The interpretation of an atomic sentence – a predicate followed by name(s) – is as you'd expect. The sentence says that the individuals named have the property/stand in the relation expressed by the predicate (order of names matters!).
- Suppose that:

a names Romeo, *b* names Juliet, *c* names Rosaline.
F means ① *is a boy*, *G* means ① *is girl*,
L means ① *loves* ②, *R* means ① *prefers* ② *to* ③

- Then:
  - *Fa* means Romeo is a boy
  - *Lab* means Romeo loves Juliet
  - *Lba* means Juliet loves Romeo

Atomic sentences – 2

- The interpretation of an atomic sentence – a predicate followed by name(s) – is as you'd expect. The sentence says that the individuals named have the property/stand in the relation expressed by the predicate (order of names matters!).
- Suppose that:

a names Romeo, *b* names Juliet, *c* names Rosaline. *F* means ① *is a boy*, *G* means ① *is girl*, *L* means ① *loves* ②, *R* means ① *prefers* ② *to* ③

- Then:
  - *Fa* means Romeo is a boy
  - *Lab* means Romeo loves Juliet
  - *Lba* means Juliet loves Romeo
  - *Rabc* means Romeo prefers Juliet to Rosaline

Atomic sentences – 2

- The interpretation of an atomic sentence – a predicate followed by name(s) – is as you'd expect. The sentence says that the individuals named have the property/stand in the relation expressed by the predicate (order of names matters!).
- Suppose that:

a names Romeo, *b* names Juliet, *c* names Rosaline. *F* means ① *is a boy*, *G* means ① *is girl*, *L* means ① *loves* ②, *R* means ① *prefers* ② *to* ③

- Then:
  - *Fa* means Romeo is a boy
  - *Lab* means Romeo loves Juliet
  - *Lba* means Juliet loves Romeo
  - *Rabc* means Romeo prefers Juliet to Rosaline
  - *Raba* means Romeo prefers Juliet to himself

Adding connectives – 1

- a names Romeo, b names Juliet, c names Rosaline.
  F means ① *is a boy*, G means ① *is girl*,
  L means ① *loves* ②, R means ① *prefers* ② *to* ③

Adding connectives – 1

- a names Romeo, b names Juliet, c names Rosaline.
  F means ① *is a boy*, G means ① *is girl*,
  L means ① *loves* ②, R means ① *prefers* ② *to* ③
- We'll now add to QL the now-familiar truth-functional propositional connectives

Adding connectives – 1

- *a* names Romeo, *b* names Juliet, *c* names Rosaline.
  F means ① *is a boy*, *G* means ① *is girl*,
  L means ① *loves* ②, *R* means ① *prefers* ② *to* ③
- We'll now add to QL the now-familiar truth-functional propositional connectives
- Then how would we translate the following?

Adding connectives – 1

- *a* names Romeo, *b* names Juliet, *c* names Rosaline.
  F means ① *is a boy*, *G* means ① *is girl*,
  L means ① *loves* ②, *R* means ① *prefers* ② *to* ③
- We'll now add to QL the now-familiar truth-functional propositional connectives
- Then how would we translate the following?
  - Juliet is not a boy

Adding connectives – 1

- *a* names Romeo, *b* names Juliet, *c* names Rosaline.
  F means ① *is a boy*, *G* means ① *is girl*,
  L means ① *loves* ②, *R* means ① *prefers* ② *to* ③
- We'll now add to QL the now-familiar truth-functional propositional connectives
- Then how would we translate the following?
  - Juliet is not a boy
  - Romeo loves Juliet and she loves him

Adding connectives – 1

- *a* names Romeo, *b* names Juliet, *c* names Rosaline.
  F means ① *is a boy*, *G* means ① *is girl*,
  L means ① *loves* ②, *R* means ① *prefers* ② *to* ③
- We'll now add to QL the now-familiar truth-functional propositional connectives
- Then how would we translate the following?
  - Juliet is not a boy
  - Romeo loves Juliet and she loves him
  - Juliet and Rosaline are both girls

Adding connectives – 1

- a names Romeo, b names Juliet, c names Rosaline.
  F means ① *is a boy*, G means ① *is girl*,
  L means ① *loves* ②, R means ① *prefers* ② *to* ③
- We'll now add to QL the now-familiar truth-functional propositional connectives
- Then how would we translate the following?
  - Juliet is not a boy
  - Romeo loves Juliet and she loves him
  - Juliet and Rosaline are both girls
  - Romeo loves either Rosaline or Juliet

Adding connectives – 1

- *a* names Romeo, *b* names Juliet, *c* names Rosaline.
  F means ① *is a boy*, *G* means ① *is girl*,
  L means ① *loves* ②, *R* means ① *prefers* ② *to* ③
- We'll now add to QL the now-familiar truth-functional propositional connectives
- Then how would we translate the following?
  - Juliet is not a boy
  - Romeo loves Juliet and she loves him
  - Juliet and Rosaline are both girls
  - Romeo loves either Rosaline or Juliet
  - If Romeo prefers himself to Juliet, then she doesn't love him.

Adding connectives – 2

- *a* names Romeo, *b* names Juliet, *c* names Rosaline.
  F means ① *is a boy*, *G* means ① *is girl*,
  L means ① *loves* ②, *R* means ① *prefers* ② *to* ③

Adding connectives – 2

- a names Romeo, b names Juliet, c names Rosaline.
  F means ① *is a boy*, G means ① *is girl*,
  L means ① *loves* ②, R means ① *prefers* ② *to* ③
- Juliet is not a boy $\Rightarrow \neg Fb$

Adding connectives – 2

- a names Romeo, b names Juliet, c names Rosaline.
  F means ① *is a boy*, G means ① *is girl*,
  L means ① *loves* ②, R means ① *prefers* ② *to* ③
- Juliet is not a boy $\Rightarrow \neg Fb$
- Romeo loves Juliet and she loves him $\Rightarrow (Lab \wedge Lba)$

Adding connectives – 2

- a names Romeo, b names Juliet, c names Rosaline.
  F means ① *is a boy*, G means ① *is girl*,
  L means ① *loves* ②, R means ① *prefers* ② *to* ③
- Juliet is not a boy $\Rightarrow \neg Fb$
- Romeo loves Juliet and she loves him $\Rightarrow (Lab \wedge Lba)$
- Juliet and Rosaline are both girls $\Rightarrow (Gb \wedge Gc)$
  NOT $G(b \wedge c)$

Adding connectives – 2

- a names Romeo, b names Juliet, c names Rosaline.
  F means ① *is a boy*, G means ① *is girl*,
  L means ① *loves* ②, R means ① *prefers* ② *to* ③
- Juliet is not a boy $\Rightarrow \neg Fb$
- Romeo loves Juliet and she loves him $\Rightarrow (Lab \wedge Lba)$
- Juliet and Rosaline are both girls $\Rightarrow (Gb \wedge Gc)$
  NOT $G(b \wedge c)$
- Romeo loves either Rosaline or Juliet $\Rightarrow (Lab \vee Lac)$
  NOT $La(b \vee c)$

Adding connectives – 2

- a names Romeo, b names Juliet, c names Rosaline.
  F means ① *is a boy*, G means ① *is girl*,
  L means ① *loves* ②, R means ① *prefers* ② *to* ③
- Juliet is not a boy ⇒ $\neg Fb$
- Romeo loves Juliet and she loves him ⇒ $(Lab \wedge Lba)$
- Juliet and Rosaline are both girls ⇒ $(Gb \wedge Gc)$
  NOT $G(b \wedge c)$
- Romeo loves either Rosaline or Juliet ⇒ $(Lab \vee Lac)$
  NOT $La(b \vee c)$
- If Romeo prefers himself to Juliet, then she doesn't love him
  ⇒ $(Raab \supset \neg Lba)$

- Our next task
- Basic subject/predicate structure
- How not to add quantifiers

Some vagaries of English quantificatiom

- Compare

Some vagaries of English quantificatiom

- Compare

  All students like logic
  Every student likes logic
  Any student likes logic
  Each student likes logic

Some vagaries of English quantificatiom

- Compare

  All students like logic
  Every student likes logic
  Any student likes logic
  Each student likes logic

- These might seem equivalent, but they embed differently, e.g. compare

Some vagaries of English quantificatiom

- Compare

  All students like logic
  Every student likes logic
  Any student likes logic
  Each student likes logic

- These might seem equivalent, but they embed differently, e.g. compare

  If all students like logic, I'll be surprised
  If any student likes logic, I'll be surprised

Some vagaries of English quantificatiom

- Compare

  All students like logic
  Every student likes logic
  Any student likes logic
  Each student likes logic

- These might seem equivalent, but they embed differently, e.g. compare

  If all students like logic, I'll be surprised
  If any student likes logic, I'll be surprised

- And compare

Some vagaries of English quantificatiom

- Compare

  All students like logic
  Every student likes logic
  Any student likes logic
  Each student likes logic

- These might seem equivalent, but they embed differently, e.g. compare

  If all students like logic, I'll be surprised
  If any student likes logic, I'll be surprised

- And compare

  Not all students turned up to the class.
  Not any students turned up to the class [?]

Some vagaries of English quantificatiom

- Compare

  All students like logic
  Every student likes logic
  Any student likes logic
  Each student likes logic

- These might seem equivalent, but they embed differently, e.g. compare

  If all students like logic, I'll be surprised
  If any student likes logic, I'll be surprised

- And compare

  Not all students turned up to the class.
  Not any students turned up to the class [?]

- In QL we'll have just one style of universal quantifier.

Quantifiers modelled on English?

- To a fair extent, "everyone" can grammatically appear in English sentences in slots where (personal) names can appear. Thus compare

Quantifiers modelled on English?

- To a fair extent, "everyone" can grammatically appear in English sentences in slots where (personal) names can appear. Thus compare
- Juliet loves Romeo / Everyone loves Romeo

Quantifiers modelled on English?

- To a fair extent, “everyone” can grammatically appear in English sentences in slots where (personal) names can appear. Thus compare
- Juliet loves Romeo / Everyone loves Romeo
- If Juliet does, Romeo does / If everyone does, Romeo does

Quantifiers modelled on English?

- To a fair extent, "everyone" can grammatically appear in English sentences in slots where (personal) names can appear. Thus compare
- Juliet loves Romeo / Everyone loves Romeo
- If Juliet does, Romeo does / If everyone does, Romeo does
- Juliet loves Romeo / Everyone loves everyone

Quantifiers modelled on English?

- To a fair extent, "everyone" can grammatically appear in English sentences in slots where (personal) names can appear. Thus compare
- Juliet loves Romeo / Everyone loves Romeo
- If Juliet does, Romeo does / If everyone does, Romeo does
- Juliet loves Romeo / Everyone loves everyone
- (But cf. "Oh Juliet, my beloved ...": we can't have "Oh everyone, my beloved". Or cf. "Someone other than Juliet loves Romeo": we can't have "Someone other than everyone loves Romeo".)

Quantifiers modelled on English?

- To a fair extent, "everyone" can grammatically appear in English sentences in slots where (personal) names can appear. Thus compare
- Juliet loves Romeo / Everyone loves Romeo
- If Juliet does, Romeo does / If everyone does, Romeo does
- Juliet loves Romeo / Everyone loves everyone
- (But cf. "Oh Juliet, my beloved ...": we can't have "Oh everyone, my beloved". Or cf. "Someone other than Juliet loves Romeo": we can't have "Someone other than everyone loves Romeo".)
- Can we dismiss exceptions as linguistic quirks?

Quantifiers modelled on English?

- Suppose we try to introduce a quantifier $\mathcal{E}$ as QL's way of talking about everyone with the following two rules that English seems (partially) to obey:

Quantifiers modelled on English?

- Suppose we try to introduce a quantifier $\mathcal{E}$ as QL's way of talking about everyone with the following two rules that English seems (partially) to obey:
    1. syntax: if $\varphi(a)$ is grammatical, so is $\varphi(\mathcal{E})$. (So we get the syntactic interchangeability of name and quantifier which we often have in English.)

Quantifiers modelled on English?

- Suppose we try to introduce a quantifier $\mathcal{E}$ as QL's way of talking about everyone with the following two rules that English seems (partially) to obey:
  1. syntax: if $\varphi(a)$ is grammatical, so is $\varphi(\mathcal{E})$. (So we get the syntactic interchangeability of name and quantifier which we often have in English.)
  2. semantics: $\varphi(\mathcal{E})$ says of everyone what $\varphi(a)$ says of what a names.

Quantifiers modelled on English?

- Suppose we try to introduce a quantifier $\mathcal{E}$ as QL's way of talking about everyone with the following two rules that English seems (partially) to obey:
  1. syntax: if $\varphi(a)$ is grammatical, so is $\varphi(\mathcal{E})$. (So we get the syntactic interchangeability of name and quantifier which we often have in English.)
  2. semantics: $\varphi(\mathcal{E})$ says of everyone what $\varphi(a)$ says of what a names.
- Will this work?

Hopeless for logic! –1

- To repeat, the suggestion is:

Hopeless for logic! –1

- To repeat, the suggestion is:
  1. syntax: if $\varphi(a)$ is grammatical, so is $\varphi(\mathcal{E})$.

Hopeless for logic! –1

- To repeat, the suggestion is:
  1. syntax: if $\varphi(a)$ is grammatical, so is $\varphi(\mathcal{E})$.
  2. semantics: $\varphi(\mathcal{E})$ says of everyone what $\varphi(n)$ says of what n names.

Hopeless for logic! –1

- To repeat, the suggestion is:
    1. syntax: if $\varphi(a)$ is grammatical, so is $\varphi(\mathcal{E})$.
    2. semantics: $\varphi(\mathcal{E})$ says of everyone what $\varphi(n)$ says of what n names.
- Suppose again a names Romeo, b names Juliet, L means ① *loves* ②,

Hopeless for logic! –1

- To repeat, the suggestion is:
  1. syntax: if $\varphi(a)$ is grammatical, so is $\varphi(\mathcal{E})$.
  2. semantics: $\varphi(\mathcal{E})$ says of everyone what $\varphi(n)$ says of what n names.
- Suppose again a names Romeo, b names Juliet, L means ① *loves* ②,
- Then the syntactic rule makes e.g. $\neg L\mathcal{E}b$ ambiguous in terms of its constructional history.

Hopeless for logic! –1

- To repeat, the suggestion is:
  1. syntax: if $\varphi(a)$ is grammatical, so is $\varphi(\mathcal{E})$.
  2. semantics: $\varphi(\mathcal{E})$ says of everyone what $\varphi(n)$ says of what n names.
- Suppose again a names Romeo, b names Juliet, L means ① *loves* ②,
- Then the syntactic rule makes e.g. $\neg L\mathcal{E}b$ ambiguous in terms of its constructional history.
  1. Do we first "quantify into" Lab to get $L\mathcal{E}b$, and then negate the result to get $\neg L\mathcal{E}b$?

Hopeless for logic! –1

- To repeat, the suggestion is:
  1. syntax: if $\varphi(a)$ is grammatical, so is $\varphi(\mathcal{E})$.
  2. semantics: $\varphi(\mathcal{E})$ says of everyone what $\varphi(n)$ says of what n names.
- Suppose again a names Romeo, b names Juliet, L means ① *loves* ②,
- Then the syntactic rule makes e.g. $\neg L\mathcal{E}b$ ambiguous in terms of its constructional history.
  1. Do we first "quantify into" Lab to get $L\mathcal{E}b$, and then negate the result to get $\neg L\mathcal{E}b$?
  2. Or do we first negate Lab to get $\neg Lab$, and then "quantify in" to get $\neg L\mathcal{E}b$?

Hopeless for logic! –1

- To repeat, the suggestion is:
  1. syntax: if $\varphi(a)$ is grammatical, so is $\varphi(\mathcal{E})$.
  2. semantics: $\varphi(\mathcal{E})$ says of everyone what $\varphi(n)$ says of what n names.
- Suppose again a names Romeo, b names Juliet, L means ① *loves* ②,
- Then the syntactic rule makes e.g. $\neg L\mathcal{E}b$ ambiguous in terms of its constructional history.
  1. Do we first "quantify into" Lab to get $L\mathcal{E}b$, and then negate the result to get $\neg L\mathcal{E}b$?
  2. Or do we first negate Lab to get $\neg Lab$, and then "quantify in" to get $\neg L\mathcal{E}b$?
- And the semantic rule generates a corresponding semantic ambiguity.

Hopeless for logic! –2

- a names Romeo, b names Juliet, L means ① *loves* ②,

Hopeless for logic! –2

- *a* names Romeo, *b* names Juliet, *L* means ① *loves* ②,
- semantics: $\varphi(\mathcal{E})$ says of everyone what $\varphi(n)$ says of what *n* names.

Hopeless for logic! –2

- *a* names Romeo, *b* names Juliet, *L* means ① *loves* ②,
- semantics: $\varphi(\mathcal{E})$ says of everyone what $\varphi(n)$ says of what n names.
- So, "quantifying into" Lab to get $L\mathcal{E}b$ gives us a proposition which means everyone loves Juliet. Negating that to get $\neg L\mathcal{E}b$ gives us a proposition which means that not everyone loves Juliet.

Hopeless for logic! –2

- *a* names Romeo, *b* names Juliet, *L* means ① *loves* ②,
- semantics: $\varphi(\mathcal{E})$ says of everyone what $\varphi(n)$ says of what *n* names.
- So, "quantifying into" Lab to get $L\mathcal{E}b$ gives us a proposition which means everyone loves Juliet. Negating that to get $\neg L\mathcal{E}b$ gives us a proposition which means that not everyone loves Juliet.
- But negating Lab to get $\neg Lab$ gives us a proposition which says Romeo doesn't love Juliet. And then our semantic rule tells us that $\neg L\mathcal{E}b$ says of everyone what $\neg Lab$ says of what *a* names. So $\neg L\mathcal{E}b$ says of everyone that he/she doesn't love Juliet – i.e. no-one loves Juliet.

Hopeless for logic! –2

- a names Romeo, b names Juliet, L means ① *loves* ②,
- semantics: $\varphi(\mathcal{E})$ says of everyone what $\varphi(n)$ says of what n names.
- So, "quantifying into" Lab to get $L\mathcal{E}b$ gives us a proposition which means everyone loves Juliet. Negating that to get $\neg L\mathcal{E}b$ gives us a proposition which means that not everyone loves Juliet.
- But negating Lab to get $\neg Lab$ gives us a proposition which says Romeo doesn't love Juliet. And then our semantic rule tells us that $\neg L\mathcal{E}b$ says of everyone what $\neg Lab$ says of what a names. So $\neg L\mathcal{E}b$ says of everyone that he/she doesn't love Juliet – i.e. no-one loves Juliet.
- So our suggested device for quantifying introduces an ambiguity into the language, exactly what we don't want in a formalized language.

Similar ambiguities in English

- Consider how scope ambiguities can arise when a quantifier is similarly combined with negation in English. E.g.

Similar ambiguities in English

- Consider how scope ambiguities can arise when a quantifier is similarly combined with negation in English. E.g.

  Everyone has not yet arrived.

Similar ambiguities in English

- Consider how scope ambiguities can arise when a quantifier is similarly combined with negation in English. E.g.

  Everyone has not yet arrived.

- Conversation 1:

Similar ambiguities in English

- Consider how scope ambiguities can arise when a quantifier is similarly combined with negation in English. E.g.

  Everyone has not yet arrived.

- Conversation 1:

  'Everyone seems to be here, so we can begin.'
  — 'No! Hold on. Everyone has not yet arrived. Jack is missing.'

Similar ambiguities in English

- Consider how scope ambiguities can arise when a quantifier is similarly combined with negation in English. E.g.

  Everyone has not yet arrived.

- Conversation 1:

  'Everyone seems to be here, so we can begin.'
  — 'No! Hold on. Everyone has not yet arrived. Jack is missing.'

- Conversation 2:

Similar ambiguities in English

- Consider how scope ambiguities can arise when a quantifier is similarly combined with negation in English. E.g.

  Everyone has not yet arrived.

- Conversation 1:

  'Everyone seems to be here, so we can begin.'
  — 'No! Hold on. Everyone has not yet arrived. Jack is missing.'

- Conversation 2:

  'If anyone arrives early, the surprise will be spoilt.'
  — 'Don't worry! It's still dead quiet outside. Everyone has not yet arrived.'