1 Scope

A generalized quantifier is (the characteristic function of) a set of properties. A quantified (declarative) sentence asserts that some property is in the GQ denoted by the widest-scope quantifier in the sentence. For example,

\[
\begin{align*}
(1) & \quad a. \quad [\text{every dog barks}]_{\mathcal{M},g} = 1 \text{ iff } [\text{barks}]_{\mathcal{M},g} \in [\text{every dog}]_{\mathcal{M},g} \\
& \quad b. \quad [[\text{every dog}]_{3} \text{ loves his}_{3} \text{ master}]_{\mathcal{M},g} = 1 \text{ iff } \\
& \quad \quad \{x \mid [\text{loves}]_{\mathcal{M},g}[[\text{master}]_{\mathcal{M},g}(x)](x)\} \in [\text{every dog}]_{\mathcal{M},g} \text{ (bound his)} \\
& \quad c. \quad [[\text{every dog}]_{3} \text{ loves his}_{4} \text{ master}]_{\mathcal{M},g} = 1 \text{ iff } \\
& \quad \quad \{x \mid [\text{loves}]_{\mathcal{M},g}[[\text{master}]_{\mathcal{M},g}(g(4))](x)\} \in [\text{every dog}]_{\mathcal{M},g} \text{ (free his)} \\
& \quad d. \quad [\text{every dog chased some cat}]_{\mathcal{M},g} = 1 \text{ iff } \text{(wide scope every dog)} \\
& \quad \quad \{x \mid \exists y[[\text{cat}]_{\mathcal{M},g}(y) \land [\text{chased}]_{\mathcal{M},g}(y)(x)]\} \in [\text{every dog}]_{\mathcal{M},g} \\
& \quad e. \quad [\text{every dog chased some cat}]_{\mathcal{M},g} = 1 \text{ iff } \text{(wide scope some cat)} \\
& \quad \quad \{y \mid \forall x[[\text{dog}]_{\mathcal{M},g}(x) \rightarrow [\text{chased}]_{\mathcal{M},g}(y)(x)]\} \in [\text{some cat}]_{\mathcal{M},g}
\end{align*}
\]

Scope is a property of expressions, which is defined in terms of the way that the denotation of the expression is semantically combined with the denotations of other expressions in the sentence.

The scope of a quantificational DP, on a given analysis of the sentence, is that part of the sentence which denotes a property that is asserted to be an element of the generalized quantifier denoted by DP on that analysis. (Szabolcsi 2010: 10)

In the examples in (1a)-(1d), the scope of the expression every dog is whatever linguistic material denotes the sets that are claimed to be in the GQ [every dog]_{\mathcal{M},g}.

- In (1a) the scope of every dog is barks.
- In (1b) and (1c) the scope of every dog is loves his master.
- In (1d) the scope of every dog is chased some cat.
- In (1e) the scope of every dog is just chased.

In (1e) some cat takes scope over every dog chased. Note that this holds regardless of your syntactic commitments: the idea that semantic scope is determined by syntactic hierarchy is a hypothesis (due to Reinhart) and is controversial in formal semantics.
Although scope is a simple concept, finding out how to derive a meaning for a sentence that delivers the appropriate scope compositionally is often not simple. Useful terminology (for 2-quantifier sentences):

- ‘Direct scope’: The expression which occurs in a higher position in surface syntax also has wider semantic scope.
- ‘Inverse scope’: The expression which occurs in a lower position in surface syntax has wider semantic scope.

### 2 Montague 1973

Montague’s approach was to build a sentence with free variables and choose the order in which we want to employ \( \lambda \)-abstraction over them, and then apply the generalized quantifiers in the appropriate order. Liberal use of string-rewrite rules ensures that the DPs denoting the relevant quantifiers get pronounced in the appropriate places.

\[
\lambda P \exists_{>1} z [dragon'(z) \land P(z)](\lambda x_2 \forall y [man'(y) \rightarrow spot'(y)(x_2)]) = \exists_{>1} z [dragon'(z) \land \lambda x_2 \forall y [man'(y) \rightarrow spot'(y)(x_2)](z)] = \exists_{>1} z [dragon'(z) \land \forall y [man'(y) \rightarrow spot'(y)(z)]]
\]
Object > Subject reading

\[
\forall y[\text{man}'(y) \rightarrow \exists z[\text{dragon}'(z) \land \text{spot}'(y)(z)]]
\]

\[
\lambda Q \forall y[\text{man}'(y) \rightarrow Q(y)] \\
\lambda x_1 \exists z[\text{dragon}'(z) \land \text{spot}'(x_1)(z)] \\
\lambda x_2 \exists z[\text{dragon}'(z) \land \text{spot}'(x_1)(z)] \\
\lambda x_3 \exists z[\text{dragon}'(z) \land \text{spot}'(x_1)(z)]
\]

\[
\lambda Q \forall y[\text{man}'(y) \rightarrow Q(y)](\lambda x_3 \exists z[\text{dragon}'(z) \land \text{spot}'(x_3)(z)]) = \\
\forall y[\text{man}'(y) \rightarrow \exists z[\text{dragon}'(z) \land \text{spot}'(x_3)(z)](y)] = \\
\forall y[\text{man}'(y) \rightarrow \exists z[\text{dragon}'(z) \land \text{spot}'(y)(z)]]
\]

3 Quantifier lowering (not in Szabolcsi 2010)

Architecture of early Transformational Grammar and Generative Semantics:

Semantic interpretation ⇐ Deep Structure ⇒ Surface Structure

Adopting the assumption that Deep Structure is the level of syntax which feeds interpretation, Lakoff (1971) argued that quantifiers are syntactically attached at DS wherever the interpretation demands. Their surface positions are then produced by transformational lowering, along with a deletion operation that erases the bound pronouns that are needed to get the interpretation right. Anachronistically, the two structures associated with the string Every dog chases some cat would be something like:

(2) Direct:
   a. DS: [every dog] [Op₁ [some cat] [Op₂ [t₁ chased t₂]]]
   b. SS: Op₁ Op₂ [every dog] t₁ / chased [some cat] t₂
   c. Interpretation: [every dog]^{M,s}(\lambda x₁.[some cat]^{M,s}(\lambda x₂.[chased]^{M,s}(x₂)(x₁))

(3) Inverse:
a. DS: [some cat] [Op₂ [ [every dog] [Op₁ [ t₁ chased t₂ ] ] ] ]
b. SS: Op₂ Op₁ [every dog] t₁ [ chased [some cat] t₂ ]
c. Interpretation: [some cat]^{M,θ}(λx₂.[every dog]^{M,θ}(λx₁.[chased]^{M,θ}(x₂)(x₁)))

We generate the truth-conditions from the DSs by supposing that Opᵢ is interpreted as an
instruction to λ-abstract over the variable xᵢ in its syntactic sister, and that a tᵢ denotes the
variable xᵢ.

This approach is no longer used, for lots of reasons: e.g., even theorists who are otherwise
free in their use of transformations generally don’t like to make use of lowering operations.
But it’s not much different from Montague’s more influential proposal, and:

- It was presumably a major source of inspiration for May’s (1977; 1985) influential
  Quantifier Raising proposal — though Lakoff and others who argued for this (McCawley,
  Lewis) are rarely cited in that tradition. (They were in the Bad Guys camp of the
day.)

- Semantically, the general form of the proposal is still what every theory of quantifier
  scope gives you: what varies in the two interpretations is the order in which the two
  variables are abstracted over, i.e., in how the syntax is supposed to create the required
  semantic structure.

4 Quantifier raising

Interpretive semantics architecture (Jackendoff 1972; Chomsky 1976; May 1977; Heim &
Kratzer 1998: etc.). Modifications associated with Minimalism are annotated “MP”.

\[
\begin{array}{c}
\text{DS (MP: numeration)} \rightarrow \text{SS (MP: Spell-out)} \rightarrow \text{LF} \rightarrow \text{Semantic interpretation} \\
\Rightarrow \text{PF}
\end{array}
\]

Assumptions (May 1977, 1985; Heim & Kratzer 1998, etc.):

- Quantifier phrases can be raised and adjoined to any S node/node whose denotation
  has type t
- This process is called “Quantifier raising” or “QR”
- This is a “covert movement” process – one which is “silent”, i.e., which is not associated
  with any overt phonetic realization.
  - creates an extra node — I’ll call it Opᵢ — which is interpreted as triggering λ-
    abstraction of some variable xᵢ
  - leaves behind a trace which is obligatorily interpreted as the same variable xᵢ
    that Opᵢ triggers abstraction of.

\[
[...QP...]_S \Rightarrow [QP_i[Op[...]_S]]_S
\]
Direct:

a. SS: [every dog] [ chased [some cat] ]

b. LF: [every dog] [ Op₁ [ [some cat] [ Op₂ [ t₁ chased t₂ ] ] ] ]

c. Interpretation: \([\text{every dog}]^{\mathcal{M}}\mathcal{g}(\lambda x₁.\,[\text{some cat}]^{\mathcal{M}}\mathcal{g}(\lambda x₂.\,[\text{chased}]^{\mathcal{M}}\mathcal{g}(x₂)(x₁)))\]

Inverse:

a. SS: [every dog] [ chased [some cat] ]

b. LF: [some cat] [ Op₂ [ [every dog] [ Op₁ [ t₁ chased t₂ ] ] ] ]

c. Interpretation: \([\text{some cat}]^{\mathcal{M}}\mathcal{g}(\lambda x₂.\,[\text{every dog}]^{\mathcal{M}}\mathcal{g}(\lambda x₁.\,[\text{chased}]^{\mathcal{M}}\mathcal{g}(x₂)(x₁)))\)

Note how extremely similar this is to (our reinterpretation of) Quantifier Lowering: we’ve mainly traded the labels “DS” and “LF”, in the process reversing assumptions about which level is derived from which.

Sometimes QR is required because the derivation would be ruled out otherwise (unless, of course, we have other semantic mechanisms at our disposal — see below).

- Nothing forced us to move the subject in (5); but we could not have avoided moving the object, because the verb requires an object of type \(e\) rather than \((e,t),t\).

- Another case in which interpretation has been claimed to be impossible without QR (or something with the same effect) is antecedent-contained deletion: sentences like *Mary saw every movie that Bill did* (Sag 1976; Heim & Kratzer 1998; Jacobson 2014).

- Next time we’ll consider in more detail arguments for QR from scope restrictions and their similarity with restrictions on overt *wh*-movement.

Dispensing with traces creates problems. Fox 2002 replaces the trace interpretation rule with a rule that the lower copy is interpreted as a bound definite description: *Mary saw every movie* is interpreted as “For every movie \(x\), Mary saw the movie \(x\).”

5 Hendriks’s (1993) Flexible Types

Many theorists dislike the idea of covert syntactic movement intensely (presumably, a superset of those who dislike the idea of syntactic movement of any kind intensely). Can we deal with quantifier scope ambiguities without covert movement? Absolutely. Hendriks (1993) show how to derive the semantic effect of QR in a directly compositional way, with no syntactic structures that aren’t directly motivated by the surface string.

Theories of quantifier scope ambiguities share the goal of building a pair of semantic representations like these for the surface string *Every dog chased some cat* — and doing so in a syntactically and semantically respectable way.

\begin{align*}
(6) & a. \quad [\text{every dog}]^{\mathcal{M}}\mathcal{g}(\lambda x₁.\,[\text{some cat}]^{\mathcal{M}}\mathcal{g}(\lambda x₂.\,[\text{chased}]^{\mathcal{M}}\mathcal{g}(x₂)(x₁))) \\
& b. \quad [\text{some cat}]^{\mathcal{M}}\mathcal{g}(\lambda x₂.\,[\text{every dog}]^{\mathcal{M}}\mathcal{g}(\lambda x₁.\,[\text{chased}]^{\mathcal{M}}\mathcal{g}(x₂)(x₁)))
\end{align*}

Abstracting away from the precise choice of verb and QPs, this gives the target schemata:
We can then λ which the corresponding arguments appear as arguments to the verb.

that are underdetermined by verb meaning variables over objects of type \( /uni27E8/uni27E8 \) to the surface syntax — no covert mechanisms needed.

Second conceptual shift: instead of thinking of these as schemata for sentence meanings that are underdetermined by the sentence’s phonological form, think of them as schemata for the verb meaning that are underdetermined by the verb’s phonological form.

\( Q_2 \) will be the first syntactic argument, and \( Q_2 \) will be the second; but their scope will differ between (9a) and (9b). What this means is that — as long as we have a way of ensuring that both of these interpretations are available for \( V \) — the order in which the verb takes the arguments in the syntax does not necessarily determine the relative scope of the arguments. In other words, syntactic hierarchy does not necessarily determine semantic scope.

The representations in (7) are of course equivalent to the less brief ones in (10):

\[
\begin{align*}
(7) & \quad \text{a. } Q_P A (\lambda x_1. Q_P B (\lambda x_2. V (x_2) (x_1))) \quad \text{(direct scope)} \\
& \quad \text{b. } Q_P B (\lambda x_2. Q_P B (\lambda x_1. V (x_2) (x_1))) \quad \text{(indirect scope)}
\end{align*}
\]

Note that

- \( Q_P A \) and \( Q_P B \) are both objects of type \( (e, t), t \), and \( V \) is an object of type \( (e, (e, t)) \).
- By the definition of scope given on page 1, \( Q_P A \) takes scope over \( Q_P B \) in (7a), and \( Q_P B \) takes scope over \( Q_P A \) in (7b).

Now we need to make our first conceptual shift: Think of these unnamed generalized quantifiers, not as names for specific GQs, but as variables over GQs. Letting \( Q_1, Q_2, ... \) be variables over objects of type \( (e, t), t \):

\[
\begin{align*}
(8) & \quad \text{a. } Q_1 (\lambda x_1. Q_2 (\lambda x_2. V (x_2) (x_1))) \\
& \quad \text{b. } Q_2 (\lambda x_2. Q_1 (\lambda x_1. V (x_2) (x_1)))
\end{align*}
\]

We can then λ-abstact over these GQ-type variables. Importantly, there is no particular connection between the order in which abstract over the GQ meanings, and the order in which the corresponding arguments appear as arguments to the verb.

\[
\begin{align*}
(9) & \quad \text{a. } \lambda Q_2 \lambda Q_1 [Q_1 (\lambda x_1. Q_2 (\lambda x_2. V (x_2) (x_1)))] \\
& \quad \text{b. } \lambda Q_2 \lambda Q_1 [Q_2 (\lambda x_2. Q_1 (\lambda x_1. V (x_2) (x_1)))]
\end{align*}
\]

Second conceptual shift: instead of thinking of these as schemata for sentence meanings that are underdetermined by the sentence’s phonological form, think of them as schemata for the verb meaning that are underdetermined by the verb’s phonological form.

\( Q_2 \) will be the first syntactic argument, and \( Q_2 \) will be the second; but their scope will differ between (9a) and (9b). What this means is that — as long as we have a way of ensuring that both of these interpretations are available for \( V \) — the order in which the verb takes the arguments in the syntax does not necessarily determine the relative scope of the arguments. In other words, syntactic hierarchy does not necessarily determine semantic scope.

The representations in (7) are of course equivalent to the less brief ones in (10):

\[
\begin{align*}
(10) & \quad \text{a. } [\lambda Q_2 \lambda Q_1 [Q_1 (\lambda x_1. Q_2 (\lambda x_2. V (x_2) (x_1)))](Q_P B)(Q_P A) \\
& \quad \quad = \lambda Q_1 [Q_1 (\lambda x_1. Q_P B (\lambda x_2. V (x_2) (x_1)))](Q_P A) \\
& \quad \quad = Q_P A (\lambda x_1. Q_P B (\lambda x_2. V (x_2) (x_1))) \quad \text{(direct scope, = (7a))}
\end{align*}
\]

\[
\begin{align*}
& \quad \text{b. } [\lambda Q_2 \lambda Q_1 [Q_2 (\lambda x_2. Q_1 (\lambda x_1. V (x_2) (x_1)))](Q_P B)(Q_P A) \\
& \quad \quad = [\lambda Q_1 [Q_P B (\lambda x_2. Q_1 (\lambda x_1. V (x_2) (x_1)))])(Q_P A) \\
& \quad \quad = Q_P B (\lambda x_2. Q_P A (\lambda x_1. V (x_2) (x_1))) \quad \text{(indirect scope, = (7b))}
\end{align*}
\]

but this time we’ve generated both scopes while introducing the generalized quantifiers in the same order. That is, we’ve introduced them into the interpretation in a way that conforms to the surface syntax — no covert mechanisms needed.
OK, this all works if we have some reason to think that a verb like *see* is ambiguous in this way. Is there a way to generate both meanings from the basic meaning (type \(e, (e, t)\)) without simply stipulating it in the lexicon?

Hendriks proposes that all expressions, including verbs like *chase*, should be assigned *flexible types*. Hendriks shows that the full versions of these two rules — together with a third rule (*Value raising*) designed to deal with intensional contexts — can be used to generate meanings that are equivalent to all possible applications of the syntactic rule QR. Let’s turn to Barker (2005) for explication (appendix).

### 6 Constraints on quantifier scope

The question remains — for Hendriks and also (e.g.) Barker & Shan (2014) — whether there is a well-motivated way to restrict its application in cases in which not all scopes are possible:

(11) Most boys like more than 3 movies. (√ direct, ?? inverse)

- **Reinhart (1997)** follows a long tradition which argues that syntactic approaches are motivated by restrictions on the scope of certain quantificational expressions which resemble independently motivated syntactic constraints (roughly, island constraints).

(12) a. Somebody wondered whether I had considered all the facts. (# inverse)

   b. # What did somebody wonder whether I had considered?

- **Defenders of theories inspired by Hendriks** would presumably take heart

  – if Value Raising is not needed, so that variable scope is automatically clause-bounded (Jacobson 2014). Then we need a non-scope-based way of dealing with quantifying into intensional contexts; or

  – if non-syntactic, processing-based theories of islandhood turn out to be right (Hofmeister & Sag 2010). Perhaps a similar account of restrictions on the scope of certain quantifiers is possible ...

### References


