Constraining scope: Glue and licensing*

Mary Dalrymple

September 26, 2026

The Glue approach to the syntax-semantics interface incorporates a treatment of quantification and modification which predicts scope ambiguity for quantifiers and modifiers without relying on syntactic ambiguity. However, the original proposals for the treatment of quantification and modification (Dalrymple et al., 1993, 1995, 1997), provided no treatment of constraints on relative quantifier or modifier scope, or of scope islands. Subsequent work proposed to address this gap (Crouch & van Genabith, 1999; Gotham, 2019, 2021; Findlay & Haug, 2022), but the proposed solutions are problematic or not completely general; see Dalrymple et al. (2025) for detailed discussion. Building on Dalrymple et al. (2025), we propose a general theory of scope licensing as a part of the logical deduction, enabling lexical and configurational specification of constraints governing scope islands and relative scope of modifiers and quantifiers.

1 Restrictions on quantifier and modifier scope

1.1 Restrictions on relative scope of quantifiers

Constraints on relative quantifier scope have been a subject of debate since the early days of transformational grammar, with claims that linear order, grammatical function, lexically specified properties, or a combination of these factors can constrain or determine the relative scope of quantifiers. Coming from very different theoretical perspectives, Lakoff (1971) and Jackendoff (1972) each claim that the scope of clausemate quantifiers is determined by linear order: that an example like *Few books are read by many men* is unambiguous, with *few books* scoping

*Work on this topic began in collaboration with Jamie Findlay and Dag Haug on a visit to Oslo in April–June 2024, and the initial results of the research were reported by Dalrymple et al. (2025). I am grateful to Jamie Findlay and Dag Haug for extensive discussion of these issues, to Vijay Saraswat, Gianluca Giorgolo, Vineet Gupta, and John Lamping for further discussion and vital assistance with the linear logic provers that were essential in the research, and to Jamie Findlay for comments on this draft. The Latex for the proofs in the paper was automatically produced by Dag Haug's Linear Logic Prover (`https://github.com/daghaug/linear-logic-nd`), with post-editing to handle premises with universal quantification (since universally quantified premises are not handled by this prover), to add types, and to simplify the left-hand (meaning) side of proofs involving licenses as described by Dalrymple et al. (2025, 4.2). Claude Sonnet 5 and Claude Fable 5 provided help with post-editing of proofs in Latex, copyediting and spellchecking, and bibliography management.

over *many men*. Indeed, the principle that linear order determines quantifier scope has been called 'Jackendoff's principle' (Janssen & Zimmermann, 2025).

Ioup (1975) disagrees with this simple claim, and proposes that both linear order and grammatical function can constrain quantifier scope. She further notes that different quantifiers have different strengths, and constrain relative scope to different degrees. She provides the following examples to show that a quantifier such as *each* can scope over a quantifier to its left:

(1) Ioup (1975, 73):
   a. I saw a picture of each child. (*each child* > *a picture*)
   b. She knows a solution to every problem. (*every problem* > *a solution*)
   c. Ethel has a dress for every occasion. (*every occasion* > *a dress*)

However, if *all* is substituted for *each/every*, Ioup claims that the preferred reading (though not the only reading) has narrow scope for *all*.

(2) Ioup (1975, 74):
   a. I saw a picture of all the children. (*a picture* > *all the children*)
   b. She knows a solution to all problems. (*a solution* > *all problems*)
   c. Ethel has a dress for all occasions. (*a dress* > *all occasions*)

Larson (1990) (citing David Lebeaux) provides the following examples in which grammatical function can constrain relative quantifier scope; in (3-a) either scope is available, but in (3-b), the primary object scopes over the secondary object:

(3) Larson (1990, 604):
   a. The teacher assigned one problem to every student. (ambiguous)
   b. The teacher assigned one student every problem. (unambiguous, object *one student* has wide scope)

Similarly, Larson (1990) cites Schneider-Zioga (1988) for the observation that although (4-a) allows both scopes, (4-b) is unambiguous, with wide scope for the object:

(4) Larson (1990, 604):
   a. The worker loaded one box on every truck. (ambiguous)
   b. The worker loaded one truck with every box. (unambiguous, object *one truck* has wide scope)

In a subsequent exploration of constraints on relative quantifier scope, Liu (1997) notes the following contrast:

(5) Liu (1997, 1, 8):
   a. Every man loves some woman. (ambiguous)
   b. Every man loves fewer than three women. (unambiguous, subject *every man* has wide scope)

Liu also discusses the double-object examples presented by Larson and Schneider-Zioga, pointing out that the choice of quantifier also affects the possibility of scope ambiguity:

(6) Liu (1997, 177):
   a. John assigned some student every problem. (unambiguous)
   b. John assigned three students most problems. (ambiguous)

Liu's conclusion is that scope possibilities are affected by properties of individual quantifiers. Similar observations are made by Szabolcsi (1997a) on the basis of examples like the following:

(7) Szabolcsi (1997a, 110):
   a. Three referees read every abstract. (ambiguous)
   b. Three referees read few abstracts. (unambiguous, subject *three referees* has wide scope)

Further evidence that lexical and syntactic properties of quantifiers constrain scope possibilities is provided by, inter alia, Beghelli & Stowell (1997) and Dalrymple et al. (2025).

1.2 Scope islands

In the early days of transformational grammar, Rodman (1976) proposed that constraints on quantifier scope mirror constraints on long-distance dependencies in wh-question formation: wide scope for a quantifier is possible only in positions from which wh-fronting is also possible. Reinhart (1997) provides the following examples to illustrate Rodman's claim:

(8) Reinhart (1997, 336): *every new patient* can take wide scope
   a. A doctor will interview every new patient.
   b. A doctor will try to assist every new patient personally.
   c. A doctor will make sure that we give every new patient a tranquilizer.

(9) Reinhart (1997, 336): *which patients* can undergo wh-movement:
   a. Which patients will a doctor interview *e*?
   b. Which patients will a doctor try to assist *e* personally?
   c. Which patients will a doctor make sure that we give *e* a tranquilizer?

(10) Reinhart (1997, 336): *every new patient* cannot take wide scope
   a. A doctor will examine the possibility that we give every new patient a tranquilizer.
   b. A doctor should worry if we sedate every new patient.

(11) Reinhart (1997, 336): *which patients* cannot undergo wh-movement:
   a. *Which patients will a doctor examine the possibility that we give *e* a tranquilizer?

b. *Which patients should a doctor worry if we sedate *e*?

Reinhart (1997) goes on to point out that Rodman's generalization was quickly found to be inadequate; quantifiers vary in their ability to escape wh-islands. In particular, while *each* behaves as Rodman predicts, *every* is more restricted, and indefinites are much more free in their ability to scope outside wh-islands.

In more recent work, Barker (2022) addresses the claim that all finite clauses are scope islands for all quantifiers, criticizing the ensuing need for various escape hatches to restore the attested readings that are predicted to be impossible by this basic assumption. He instead proposes a four-way distinction in strength distinguishing different kinds of islands of different lexically specified strengths, combined with a five-way distinction in different types of quantifiers, as shown in the following table:

(12) Island strength vs. escaper strength (Barker, 2022, 648)

	damn	*someone*	*anyone*	*everyone*	*no one*	Island strength ↓	
only		*	*	*	*	4	
doubt			*	*	*	3	
claimed				*	*	2	
make sure					*	1	
	4	3	2	1	0	←	Escaper strength

We agree with Barker that scope islands are best characterized in terms of lexically specified properties of particular island-creating items, interacting with lexically specified properties of particular scope-taking items.

1.3 Restrictions on relative scope of modifiers

The relative scope of modifiers is constrained by the configuration in which the modifiers appear. Andrews (1982) points out that the scope of an intensional adjective like *alleged* depends on adjective order, with a modifier closer to the head taking narrow scope with respect to modifiers that are farther from the head (see also Andrews & Manning 1999, Andrews 2018, Findlay & Haug 2022):

(13) Andrews (1982, 696):
   a. an alleged English baron (someone who is alleged to be an English baron; *alleged* scopes over *English baron*)
   b. an English alleged baron (someone who is English and is alleged to be a baron; *alleged* scopes over *baron*)

If the modifiers are equally distant from the head (one preceding the head and the other following the head), as in (14), both readings are available. This is expected if relative distance from the head determines scope constraints.

(14) Andrews (1982, 696):
an alleged baron from England (both readings available)

It may appear that the generalization is that semantic composition is governed by phrasal configuration, with elements of the English noun phrase strictly composing according to their phrase structure position. However, Partee & Borschev (1998) (citing Norvin Richards) show that semantic composition in English noun phrases is not in general limited in this way: possessors may scope either inside or outside an intensional adjective (see also Partee & Borschev 2003, Larson & Cho 2003).

(15) Partee & Borschev (1998, 77): *Mary's former mansion*
   a. Mary's possession which was formerly (but is not currently) a mansion; or
   b. something which was formerly Mary's (which may or may not currently be a mansion)

Strict composition following phrasal constituency would not produce the right result here: the scope constraint affects only the relative scope of modifiers, not other constituents.

Similar facts hold for adverbs. Andrews (1982) points out that each of the examples in (16) is unambiguous, with the adverb closest to the head taking narrow scope.

(16) Andrews (1982, 695):
   a. John knocked on the door intentionally twice. (two instances of intentional knocking: *twice* scopes over *knocked on the door intentionally*)
   b. John knocked on the door twice intentionally. (one intentional instance of knocking twice: *intentionally* scopes over *knocked on the door twice*)

Andrews (1982) further points out that both of the examples in (17) are ambiguous, and that each has both of the readings in (16). Again, it is relative distance from the head which determines scope constraints.

(17) Andrews (1982, 696): Both scopes available
   a. John intentionally knocked on the door twice.
   b. John twice knocked on the door intentionally.

1.4 Summary

The scope of quantifiers and modifiers can be constrained by a number of factors, including:

- For quantifier scope:
  - the strength of the quantifier relative to the strength of other quantifiers in the same domain
  - the grammatical function of the quantifier

  - the strength of the quantifier relative to the strength of any island in which the quantifier may appear

- For modifier scope: relative distance to the head

In the following, we present a general means of constraining the scope of quantifiers and modifiers in a Glue setting. We first provide an introduction to Glue, Lexical Functional Grammar, and the standard LFG/Glue treatment of quantifiers, modifiers, and scope ambiguity. We then introduce our proposal for constraining quantifier and modifier scope, based on work by Fry (1997, 1999) on negative polarity items, and building on proposals by Dalrymple et al. (2025).

2 Quantifiers and modifiers in Glue

Glue is a theory of the syntax-semantics interface according to which meaning composition is governed by logical deduction from a set of premises contributed by lexical items or phrasal configurations. The premises in a meaning deduction are pairs consisting of an expression in some meaning language (here, for simplicity, we use the predicate calculus, but other meaning languages are commonly used) paired with a logical expression governing how these meanings can combine; these pairs are called *meaning constructors*. We adopt Lexical Functional Grammar as the background syntactic theory, though Glue (including the approach to constraining scope presented here) can be paired with any syntactic theory (for example, see Asudeh & Crouch 2001 for HPSG, and Gotham 2018 for Minimalism). For overviews of Glue, see Dalrymple et al. (2019), Asudeh (2022, 2023), and Asudeh & Dalrymple (in press).

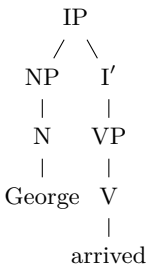
2.1 Lexical Functional Grammar

To facilitate discussion in the following, we provide a very brief overview of LFG. For a more complete introduction, see Dalrymple et al. (2019) and Dalrymple (2023), in particular Belyaev (2023a,b).

As is standard in LFG, we propose the following syntactic structures for the sentence *George arrived*. Constituent structure (c-structure) represents word order and phrasal groupings, while functional structure (f-structure) represents grammatical functions and features. A full analysis would include additional f-structure features encoding tense, aspect, person, number, and other functional features, but we follow standard LFG practice in omitting features that are not relevant for the current discussion.

(18) George arrived.

Constituent structure:



Functional structure:

$$\begin{bmatrix} \text{PRED} & \text{`ARRIVE}\langle\text{SUBJ}\rangle\text{'} \\ \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{`GEORGE'} \end{bmatrix} \end{bmatrix}$$

These structures are licensed by the annotated phrase structure rules and lexical items in (19) and (20). In the annotations on the phrase structure rules, the up arrow $\uparrow$ refers to the f-structure of the mother node, and the down arrow $\downarrow$ refers to the f-structure of the node with the annotations. For example, in the IP rule, the $(\uparrow \text{SUBJ})=\downarrow$ annotation on the NP daughter of IP requires that in the f-structure $\uparrow$ corresponding to the IP node, the f-structure $\downarrow$ of the NP daughter is the value of the SUBJ attribute, and the $\uparrow=\downarrow$ annotation on the I′ daughter requires that the f-structure of the IP is the same as the f-structure of the I′.

(19) Basic phrase structure rules for English:

$$\begin{array}{lcl}
\text{IP} & \longrightarrow & \underset{(\uparrow \text{SUBJ})=\downarrow}{\text{NP}} \quad \underset{\uparrow=\downarrow}{\text{I′}} \\
\text{NP} & \longrightarrow & \underset{\uparrow=\downarrow}{\text{N}} \\
\text{I′} & \longrightarrow & \left(\underset{\uparrow=\downarrow}{\text{I}}\right) \quad \underset{\uparrow=\downarrow}{\text{VP}} \\
\text{VP} & \longrightarrow & \underset{\uparrow=\downarrow}{\text{V}} \quad \left(\underset{(\uparrow \text{OBJ})=\downarrow}{\text{NP}}\right)
\end{array}$$

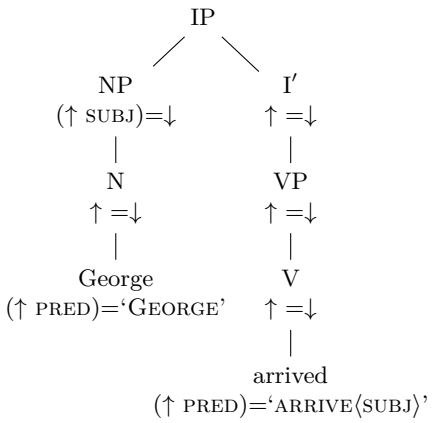
The lexical entries in (20) constrain phrase structure category and f-structure contribution: for example, the word *George* is of category N, and it contributes the attribute PRED with value 'GEORGE' to its f-structure.

(20) Lexical entries (preliminary version, syntactic information only):

George	N	$(\uparrow \text{PRED})=\text{`GEORGE'}$
arrived	V	$(\uparrow \text{PRED})=\text{`ARRIVE}\langle\text{SUBJ}\rangle\text{'}$

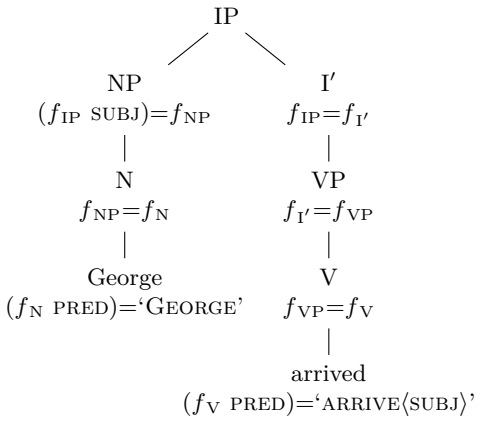
In (21), we have added these annotations to the phrase structure tree:

(21) Annotated constituent structure:



Using f_{IP} to name the f-structure corresponding to the IP node, f_{NP} for the f-structure corresponding to the NP node, and similarly for the other nodes, we can replace the up and down arrows with the names of the nodes they represent.

(22) Annotated constituent structure with instantiated node names:



This produces the f-structure in (23), as desired, where each f-structure is associated with the labels in (22): the outer f-structure bears the labels f_{IP}, $f_{\text{I′}}$, f_{VP}, and f_{V}, and the value of the SUBJ attribute bears the labels f_{NP} and f_{N}.

(23) F-structure solution to the equations in (22):

$$\begin{matrix} f_{\text{IP}} \\ f_{\text{I′}} \\ f_{\text{VP}} \\ f_{\text{V}} \end{matrix} \begin{bmatrix} \text{PRED} & \text{'ARRIVE⟨SUBJ⟩'} \\ \text{SUBJ} & \begin{matrix} f_{\text{NP}} \\ f_{\text{N}} \end{matrix} \big[\text{PRED 'GEORGE'}\big] \end{bmatrix}$$

2.2 Glue premises and deductions

We now augment the lexical entries in (20) with meaning constructors encoding the semantic contribution of each word, paired with instructions expressed in a resource logic which govern semantic composition. In the lexical entry for the word *George*, the meaning constructor is a pair with the meaning *george* on the left and the resource $\uparrow_e$ on the right, where e is the type of the meaning *george*. Similarly, the meaning constructor for the word *arrived* is a pair in which the meaning *arrive* on the left is paired with the logical formula $(\uparrow \textsc{subj})_e \multimap \uparrow_t$ on the right.[1]

(24) Lexical entries including glue premises:

George	N	$(\uparrow \textsc{pred})$='George'
		$george : \uparrow_e$
arrived	V	$(\uparrow \textsc{pred})$='arrive⟨subj⟩'
		$arrive : (\uparrow \textsc{subj})_e \multimap \uparrow_t$

We use g to label the f-structure whose PRED is 'George' and a for the f-structure whose PRED is 'arrive⟨subj⟩':

(25) F-structure for *George arrived*:

$$_a\begin{bmatrix} \textsc{pred} & \text{'arrive⟨subj⟩'} \\ \textsc{subj} & {}_g\big[\textsc{pred}\ \text{'George'}\big] \end{bmatrix}$$

This gives us the instantiated meaning constructors for *George* and *arrived* in (26), where the meaning *george* is paired with the resource g with type e, and the meaning *arrive* is paired with an implication from g of type e to a of type t.

(26) Meaning constructors for *George* and *arrived*:

$$george : g_e$$
$$arrive : g_e \multimap a_t$$

The symbol $\multimap$ represents *linear implication* in linear logic (Girard, 1987), a resource logic in which resources may not be reused or discarded. It is useful to think of linear implication as a special kind of resource-sensitive implication in which the antecedent resource is consumed and the consequent resource is produced. We can think of the meaning constructors in (26) as follows:

[1] We simplify the presentation in two ways in the following; these simplifications do not affect the overall treatment of quantifiers or licenses. First, in most LFG-based Glue analyses, meanings are assumed to be paired with glue expressions over semantic (σ) structures, not functional structures. Since we do not rely on semantic structure attributes or the internal structure of semantic structures, we simplify by assuming that the right-hand (linear logic) side of a meaning constructor is an expression over f-structures. Second, we simplify the examples in various respects that are not relevant to the topic under discussion: for example, in our discussion of sentential adverbials in Section 2.6, we provide the constant d as the meaning for the definite noun phrase *the door*. For introductions to standard Glue theory, see Dalrymple et al. (2019), Asudeh (2022, 2023), and Asudeh & Dalrymple (in press).

(27)

$george : g_e$	The meaning *george* is associated with the resource g of type e.
$arrive : g_e \multimap a_t$	The meaning *arrive* is associated with a linear implication in which the resource g_e (of type e) is consumed and the resource a_t (of type t) is produced.

The right hand (linear logic) side of these expressions can combine according to the rule of modus ponens: given the resources $A \multimap B$ and A, we can produce B. Modus ponens on the glue side of a meaning constructor corresponds to function application on the meaning side, following the Curry-Howard Isomorphism. Thus, we have the deduction in (28) of the meaning $arrive(george)$ (of type t, associated with the f-structure labeled a) from the instantiated meaning constructors contributed by *George* and *arrived*.

(28) Proof for *George arrived*:

$$\frac{arrive : (g_e \multimap a_t) \qquad george : g_e}{arrive(george) : a_t} \to E$$

Glue derivations are required to be semantically complete and coherent: the deduction must produce a single meaning from the premises that are provided, with no meaning constructors left unused. This condition is satisfied by the deduction in (28), since we have produced a meaning associated with the f-structure a of type t, and all of the premises were used in the deduction with none left over.

2.3 Quantifiers in Glue

The standard lexical entry for a quantifier like *everyone* is given in (29) (Dalrymple et al., 1997, 2019).[2,3]

(29) Lexical entry for *everyone*:

everyone N $(\uparrow \text{PRED}) = \text{`EVERYONE'}$
$\lambda P.\, everyone(P) : \forall S.(\uparrow_e \multimap S) \multimap S$

Intuitively, this meaning constructor is looking for a meaning constructor with meaning P whose right-hand side is of the form $\uparrow_e \multimap S$ for some S. When it finds this resource, it consumes it, and produces a resource represented as S, associated with the meaning $everyone(P)$. The universal quantifier over scope structures S means that the scope is freely chosen.

[2]The meaning for *everyone* is often given as a generalized quantifier, a relation between two properties:

(i) $\lambda P.\, every(\lambda x.person(x), P) : \forall S.(\uparrow_e \multimap S) \multimap S$

For simplicity and to save space, we use the shorter form in (29). For a full explanation of the Glue treatment of quantificational determiners like *every*, see Dalrymple et al. (2019, 8.2).

[3]Standardly, the resource represented by S in the meaning constructor for *everyone* has type t, but we have left it without a type; we return to this issue below.

For the sentence *Everyone arrived*, we have the f-structure in (30):

(30) F-structure for *Everyone arrived*:

$$_a\begin{bmatrix}\text{PRED} & \text{`ARRIVE}\langle\text{SUBJ}\rangle\text{'} \\ \text{SUBJ} & _e\big[\text{PRED `EVERYONE'}\big]\end{bmatrix}$$

Given this f-structure, the instantiated meaning constructors for *everyone* and *arrived* are:

(31) Instantiated meaning constructors for *everyone* and *arrived*:

$$\lambda P.\, everyone(P) : \forall S.(e_e \multimap S) \multimap S$$
$$arrive : e_e \multimap a_t$$

These are the premises in the following proof:

(32) Proof for *Everyone arrived*:

$$\dfrac{\dfrac{everyone : \forall S.\,(e_e \multimap S) \multimap S}{everyone : (e_e \multimap a_t) \multimap a_t}\,{}_{\forall E} \qquad arrive : (e_e \multimap a_t)}{everyone(arrive) : a_t}\,{}_{\to E}$$

Again, this proof obeys the conditions of semantic completeness and coherence: we have derived a meaning for a_t, with no meaning constructors left unused.

2.4 Variable scope and scope ambiguity in Glue

Given the universal quantification over scope resources in the meaning constructor for quantifiers, we expect that if there is more than one quantifier in a sentence, they can apply in any order, producing a quantifier scope ambiguity. For the example *Everyone greeted someone*, we need the following additional lexical entries:

(33) Lexical entries including glue premises:

someone	N	$(\uparrow \text{PRED})=\text{`SOMEONE'}$
		$\lambda Q.\, someone(Q) : \forall T.(\uparrow_e \multimap T) \multimap T$
greeted	V	$(\uparrow \text{PRED})=\text{`GREET}\langle\text{SUBJ, OBJ}\rangle\text{'}$
		$\lambda X.\lambda Y.greet(X,Y) : (\uparrow \text{SUBJ})_e \multimap (\uparrow \text{OBJ})_e \multimap \uparrow_t$

We then have the following f-structure and instantiated meaning constructors:

(34) F-structure and instantiated meaning constructors for *Everyone greeted someone*:

$$_g\begin{bmatrix}\text{PRED} & \text{`GREET}\langle\text{SUBJ,OBJ}\rangle\text{'} \\ \text{SUBJ} & _e\big[\text{PRED `EVERYONE'}\big] \\ \text{OBJ} & _s\big[\text{PRED `SOMEONE'}\big]\end{bmatrix}$$

$$\lambda P.\, everyone(P) : \forall S.(e_e \multimap S) \multimap S$$
$$\lambda Q.\, someone(Q) : \forall T.(s_e \multimap T) \multimap T$$
$$\lambda X.\lambda Y.\, greet(X, Y) : e_e \multimap (s_e \multimap g_t)$$

For this example, both scopes are available; in a Glue setting, this means that there are two different proofs from these premises, with each proof assigning a different meaning to g.

(35) Two proofs for *Everyone greeted someone*:

a. Wide scope for *everyone*:

$$\dfrac{\dfrac{everyone : \forall S.\,(e_e \multimap S) \multimap S}{everyone : (e_e \multimap g_t) \multimap g_t}\,{\scriptstyle \forall E} \qquad \dfrac{\dfrac{someone : \forall S.\,(s_e \multimap S) \multimap S}{someone : (s_e \multimap g_t) \multimap g_t}\,{\scriptstyle \forall E} \qquad \dfrac{\dfrac{greet : (e_e \multimap (s_e \multimap g_t)) \qquad [X_2 : e_e]_1}{greet(X_2) : (s_e \multimap g_t)}\,{\scriptstyle \to E}}{someone(greet(X_2)) : g_t}\,{\scriptstyle \to E}}{\lambda X_2.someone(greet(X_2)) : (e_e \multimap g_t)}\,{\scriptstyle \to I_1}}{everyone(\lambda X_2.someone(greet(X_2))) : g_t}\,{\scriptstyle \to E}$$

b. Wide scope for *someone*:

$$\dfrac{\dfrac{someone : \forall S.\,(s_e \multimap S) \multimap S}{someone : (s_e \multimap g_t) \multimap g_t}\,{\scriptstyle \forall E} \qquad \dfrac{\dfrac{everyone : \forall S.\,(e_e \multimap S) \multimap S}{everyone : (e_e \multimap g_t) \multimap g_t}\,{\scriptstyle \forall E} \qquad \dfrac{\dfrac{\dfrac{greet : (e_e \multimap (s_e \multimap g_t)) \qquad [X_2 : e_e]_1}{greet(X_2) : (s_e \multimap g_t)}\,{\scriptstyle \to E} \qquad [X_{10} : s_e]_2}{greet(X_2)(X_{10}) : g_t}\,{\scriptstyle \to E}}{\lambda X_2.greet(X_2)(X_{10}) : (e_e \multimap g_t)}\,{\scriptstyle \to I_1}}{everyone(\lambda X_2.greet(X_2)(X_{10})) : g_t}\,{\scriptstyle \to E}}{\lambda X_{10}.everyone(\lambda X_2.greet(X_2)(X_{10})) : (s_e \multimap g_t)}\,{\scriptstyle \to I_2}}{someone(\lambda X_{10}.everyone(\lambda X_2.greet(X_2)(X_{10}))) : g_t}\,{\scriptstyle \to E}$$

For examples such as this, in which there are no constraints on the relative scope of the quantifiers, this is a good result: both scopes are derived, with no need for an unmotivated corresponding syntactic ambiguity. However, as we have seen, this is not always desirable; in some cases, quantifier scope is constrained, and not all scope orders are available. Thus, we require a way to impose constraints on relative scopes of quantifiers in such cases.

2.5 Scope (non-)islands

Barker (2022) points out that not every finite subordinate clause is a scope island for every quantifier. He provides the following example (first discussed by Farkas & Giannakidou 1996, 36), in which the predicate *make sure* allows *every invited speaker* to scope out of the subordinate clause and over the indefinite subject of the matrix clause:

(36) A student made sure that [every invited speaker had a ride]. (ambiguous, wide scope possible for *every invited speaker*)

Both readings are derived in a Glue setting, since the universal quantification over possible scope points allows *every invited speaker* to take either the main clause or the subordinate clause as its scope. To simplify the presentation, we provide the proof for the similarly-structured example *Someone made sure that [everyone arrived]*. The f-structure and instantiated meaning constructors are:

(37) F-structure and meaning constructors for *Someone made sure that everyone arrived*:

$$m\begin{bmatrix}\text{PRED} & \text{`MAKE.SURE}\langle\text{SUBJ, COMP}\rangle\text{'} \\ \text{SUBJ} & s\big[\text{PRED `SOMEONE'}\big] \\ \text{COMP} & a\begin{bmatrix}\text{PRED} & \text{`ARRIVE}\langle\text{SUBJ}\rangle\text{'} \\ \text{SUBJ} & e\big[\text{PRED `EVERYONE'}\big]\end{bmatrix}\end{bmatrix}$$

$$\begin{aligned}\lambda P.\, someone(P) &: \forall S.(s_e \multimap S) \multimap S \\ \lambda Q.\, everyone(Q) &: \forall T.(e_e \multimap T) \multimap T \\ \lambda Y.\lambda X.\, make.sure(X,Y) &: a_t \multimap (s_e \multimap m_t) \\ \lambda Z.\, arrive(Z) &: e_e \multimap a_t\end{aligned}$$

For this example there are three proofs: in the first proof, *everyone* scopes inside the subordinate clause, and there are two additional proofs in which *everyone* scopes outside the subordinate clause, with either scoping of *someone* and *everyone* available in that case.

(38) Three proofs for *Someone made sure that everyone arrived*:

a. *Everyone* scopes inside the subordinate clause:

$$\dfrac{\dfrac{someone : \forall S.\,(s_e \multimap S) \multimap S}{someone : (s_e \multimap m_t) \multimap m_t}\,{}_{\forall E} \quad \dfrac{make.sure : (a_t \multimap (s_e \multimap m_t)) \qquad \dfrac{\dfrac{everyone : \forall T.\,(e_e \multimap T) \multimap T}{everyone : (e_e \multimap a_t) \multimap a_t}\,{}_{\forall E} \qquad arrive : (e_e \multimap a_t)}{everyone(arrive) : a_t}\,{}_{\to E}}{make.sure(everyone(arrive)) : (s_e \multimap m_t)}\,{}_{\to E}}{someone(make.sure(everyone(arrive))) : m_t}\,{}_{\to E}$$

b. *Everyone* scopes in the main clause and takes narrow scope with respect to *someone*:

$$\dfrac{\dfrac{someone : \forall S.\,(s_e \multimap S) \multimap S}{someone : (s_e \multimap m_t) \multimap m_t}\,{}_{\forall E} \quad \dfrac{\dfrac{everyone : \forall T.\,(e_e \multimap T) \multimap T}{everyone : (e_e \multimap m_t) \multimap m_t}\,{}_{\forall E} \quad \dfrac{\dfrac{make.sure : (a_t \multimap (s_e \multimap m_t)) \qquad \dfrac{arrive : (e_e \multimap a_t) \qquad [X_{10} : e_e]_1}{arrive(X_{10}) : a_t}\,{}_{\to E}}{make.sure(arrive(X_{10})) : (s_e \multimap m_t)}\,{}_{\to E} \qquad [X_2 : s_e]_2}{make.sure(arrive(X_{10}))(X_2) : m_t}\,{}_{\to E}}{\lambda X_{10}.make.sure(arrive(X_{10}))(X_2) : (e_e \multimap m_t)}\,{}_{\to I_1}}{everyone(\lambda X_{10}.make.sure(arrive(X_{10}))(X_2)) : m_t}\,{}_{\to E}}{\lambda X_2.everyone(\lambda X_{10}.make.sure(arrive(X_{10}))(X_2)) : (s_e \multimap m_t)}\,{}_{\to I_2}}{someone(\lambda X_2.everyone(\lambda X_{10}.make.sure(arrive(X_{10}))(X_2))) : m_t}\,{}_{\to E}$$

c. *Everyone* scopes in the main clause and takes wide scope with respect to *someone*:

$$\dfrac{\dfrac{everyone : \forall T.\,(e_e \multimap T) \multimap T}{everyone : (e_e \multimap m_t) \multimap m_t}\,{}_{\forall E} \quad \dfrac{\dfrac{someone : \forall S.\,(s_e \multimap S) \multimap S}{someone : (s_e \multimap m_t) \multimap m_t}\,{}_{\forall E} \quad \dfrac{make.sure : (a_t \multimap (s_e \multimap m_t)) \qquad \dfrac{arrive : (e_e \multimap a_t) \qquad [X_{10} : e_e]_1}{arrive(X_{10}) : a_t}\,{}_{\to E}}{make.sure(arrive(X_{10})) : (s_e \multimap m_t)}\,{}_{\to E}}{someone(make.sure(arrive(X_{10}))) : m_t}\,{}_{\to I_1}}{\lambda X_{10}.someone(make.sure(arrive(X_{10}))) : (e_e \multimap m_t)}\,{}_{\to E}}{everyone(\lambda X_{10}.someone(make.sure(arrive(X_{10})))) : m_t}\,{}_{\to E}$$

This is again a good result, but as noted above, there are other cases in which (some) quantifiers are prevented from escaping (some) islands, and we need a way of imposing scope constraints in such cases.

2.6 Modifiers in Glue

Syntactically, the f-structure for a modifier such as *twice* is a member of the adjunct (ADJ) set of the modified head. For an example like (39), we have the (simplified) f-structure and target meaning in (40):

(39) John kicked the door twice.

(40) F-structure and target meaning for *John kicked the door twice*:

$$\begin{bmatrix} \text{PRED} & \text{`KICK}\langle\text{SUBJ, OBJ}\rangle\text{'} \\ \text{SUBJ} & {}_j\big[\text{PRED `JOHN'}\big] \\ \text{OBJ} & {}_d\big[\text{PRED `DOOR'}\big] \\ \text{ADJ} & \left\{\big[\text{PRED `TWICE'}\big]\right\} \end{bmatrix}_k$$
$$twice(kick(john, door)) : k_t$$

We obtain this meaning from the premises in (41):

(41) Premises in the derivation of the meaning of *John kicked the door twice*:

$$\begin{aligned} john &: j_e \\ door &: d_e \\ \lambda X.\lambda Y.\, kick(X,Y) &: j_e \multimap (d_e \multimap k_t) \\ \lambda P.\, twice(P) &: k_t \multimap k_t \end{aligned}$$

In a Glue setting, a modifier consumes the meaning constructor of the head which it modifies, and produces a new meaning constructor of the same type, assigning a modified meaning to the head. Here, the glue side of the adverbial modifier *twice* is $k_t \multimap k_t$: the resource k_t is consumed and then produced again. This is the hallmark of a modifier. On the meaning side, the meaning *twice* is applied to the meaning of the sentence *John kicked the door*, so that the meaning of the modified phrase is updated to incorporate the meaning of the modifier.

(42) Proof for *John kicked the door twice*:

$$\dfrac{twice : (k_t \multimap k_t) \qquad \dfrac{\dfrac{kick : (j_e \multimap (d_e \multimap k_t)) \qquad john : j_e}{kick(john) : (d_e \multimap k_t)}\,{\to}E \qquad door : d_e}{kick(john)(door) : k_t}\,{\to}E}{twice(kick(john)(door)) : k_t}\,{\to}E$$

In an example like *John intentionally kicked the door twice*, with two adverbs, the adverbs can apply in either order in the proof, producing an ambiguity. For this example, this is the correct result.

(43) F-structure for *John intentionally kicked the door twice*:

$$\begin{bmatrix} \text{PRED} & \text{`KICK}\langle\text{SUBJ, OBJ}\rangle\text{'} \\ \text{SUBJ} & {}_j\big[\text{PRED `JOHN'}\big] \\ \text{OBJ} & {}_d\big[\text{PRED `DOOR'}\big] \\ \text{ADJ} & \left\{ \begin{matrix} {}_i\big[\text{PRED `INTENTIONALLY'}\big] \\ {}_t\big[\text{PRED `TWICE'}\big] \end{matrix} \right\} \end{bmatrix}_k$$

(44) Premises in the derivation of the meaning of *John intentionally kicked the door twice*:

$$
\begin{aligned}
\mathit{john} &: j_e \\
\mathit{door} &: d_e \\
\lambda X.\lambda Y.\, \mathit{kick}(X,Y) &: j_e \multimap (d_e \multimap k_t) \\
\lambda P.\, \mathit{twice}(P) &: k_t \multimap k_t \\
\lambda P.\, \mathit{intentionally}(P) &: k_t \multimap k_t
\end{aligned}
$$

From these premises there are two proofs:

(45) Two proofs for *John intentionally kicked the door twice*

a. Wide scope for *twice*:

$$
\dfrac{\mathit{twice} : (k_t \multimap k_t) \qquad \dfrac{\mathit{intentionally} : (k_t \multimap k_t) \qquad \dfrac{\dfrac{\mathit{kick} : (j_e \multimap (d_e \multimap k_t)) \qquad \mathit{john} : j_e}{\mathit{kick}(\mathit{john}) : (d_e \multimap k_t)}\,{\scriptstyle \to E} \qquad \mathit{door} : d_e}{\mathit{kick}(\mathit{john})(\mathit{door}) : k_t}\,{\scriptstyle \to E}}{\mathit{intentionally}(\mathit{kick}(\mathit{john})(\mathit{door})) : k_t}\,{\scriptstyle \to E}}{\mathit{twice}(\mathit{intentionally}(\mathit{kick}(\mathit{john})(\mathit{door}))) : k_t}\,{\scriptstyle \to E}
$$

b. Wide scope for *intentionally*:

$$
\dfrac{\mathit{intentionally} : (k_t \multimap k_t) \qquad \dfrac{\mathit{twice} : (k_t \multimap k_t) \qquad \dfrac{\dfrac{\mathit{kick} : (j_e \multimap (d_e \multimap k_t)) \qquad \mathit{john} : j_e}{\mathit{kick}(\mathit{john}) : (d_e \multimap k_t)}\,{\scriptstyle \to E} \qquad \mathit{door} : d_e}{\mathit{kick}(\mathit{john})(\mathit{door}) : k_t}\,{\scriptstyle \to E}}{\mathit{twice}(\mathit{kick}(\mathit{john})(\mathit{door})) : k_t}\,{\scriptstyle \to E}}{\mathit{intentionally}(\mathit{twice}(\mathit{kick}(\mathit{john})(\mathit{door}))) : k_t}\,{\scriptstyle \to E}
$$

However, as we have seen, such an ambiguity is not always available. For examples in which one of the modifiers is closer to the head, the relative scope of the modifiers is constrained, with the closer modifier taking narrow scope. Thus, we need a means for constraining modifier scope in such cases.

2.7 Summary

Standard Glue analyses (unaugmented with licensing) do not constrain quantifier and modifier scope. It is often desirable to obtain all possible quantifier or modifier scopes, but there are also cases in which the scope of a quantifier or modifier is constrained by lexical or grammatical factors. This shortcoming has been noted before: for detailed discussion of the need for lexical and syntactic constraints on quantifier and modifier scope in a Glue setting, see Crouch & van Genabith (1999), Gotham (2019, 2021), Findlay & Haug (2022), Zymla (2024), and Dalrymple et al. (2025).

3 Imposing scope constraints: Licensing

3.1 Inspiration: Fry (1997, 1999) on negative polarity items

Ladusaw (1979) and much subsequent work discusses negative polarity items such as *ever* or *yet*, which are required to appear within the scope of a downward-entailing operator such as *not* or *nobody*.

(46) a. George didn't arrive yet.

b. Nobody arrived yet.
c. *George arrived yet.

(47) a. George didn't ever arrive.
b. Nobody ever arrived.
c. *George ever arrived.

Fry (1997, 1999) proposes a Glue-based analysis of downward-entailing negative polarity licensors such as *not* or *nobody*, which, on his analysis, contribute a *license* whose presence is required by negative polarity items such as *ever* or *yet*. Fry's original analysis adopted a now-outdated version of Glue (sometimes called 'old Glue': see Asudeh & Dalrymple in press for a discussion of 'old Glue' and other variants). Dalrymple et al. (2025) recast Fry's analysis in the current standard version of Glue, and we present their revised and updated version of Fry's approach in the following.

Dalrymple et al. (2025) propose the f-structure and Fry-style meaning constructors in (48) for the sentence *Nobody arrived yet*, where ℓ is Fry's license. Importantly, there is only one proof from these premises,[4] and in that proof *yet* takes narrow scope with respect to its licensor *nobody*. This is because the meaning constructor for *yet* requires the presence of the license ℓ, which is available only within the scope of the licensor *nobody*. If there is no negative polarity licensor, as in an example like **Fred arrived yet*, the requirement imposed by *yet* for a licensor to be present is not met, and there is no proof.

(48) F-structure and instantiated meaning constructors for *Nobody arrived yet*, with licenses:

$$a\begin{bmatrix}\text{PRED} & \text{`ARRIVE}\langle\text{SUBJ}\rangle\text{'} \\ \text{SUBJ} & n\big[\text{PRED `NOBODY'}\big] \\ \text{ADJ} & \Big\{\big[\text{PRED `YET'}\big]\Big\}\end{bmatrix}$$

$$\lambda P.\, nobody(P) : \forall S.((n_e \otimes \ell) \multimap (S \otimes \ell)) \multimap S$$
$$\lambda X.\, arrive(X) : n_e \multimap a_t$$
$$\lambda Q.\, yet(Q) : (a_t \otimes \ell) \multimap (a_t \otimes \ell)$$

(49) Sole proof for *Nobody arrived yet*:

$$\dfrac{\dfrac{\dfrac{\dfrac{arrive : (n_e \multimap a_t) \qquad [X_3 : n_e]_1}{arrive(X_3) : a_t}\,{\scriptstyle \to E} \qquad [\ell]_2}{arrive(X_3) : (a_t \otimes \ell)}\,{\scriptstyle \otimes I} \qquad yet : ((a_t \otimes \ell) \multimap (a_t \otimes \ell))}{yet(arrive(X_3)) : (a_t \otimes \ell)}\,{\scriptstyle \to E}}{\lambda X_3.yet(arrive(X_3)) : ((n_e \otimes \ell) \multimap (a_t \otimes \ell))}\,{\scriptstyle \to I_{1,2}} \qquad \dfrac{nobody : \forall F.(((n_e \otimes \ell) \multimap (F \otimes \ell)) \multimap F)}{nobody : (((n_e \otimes \ell) \multimap (a_t \otimes \ell)) \multimap a_t)}\,{\scriptstyle \forall E}}{nobody(\lambda X_3.yet(arrive(X_3))) : a_t}\,{\scriptstyle \to E}$$

More generally, Fry assumes the following schematic form for negative polarity

[4] We adopt the simplifications to the representation of the proof outlined by Dalrymple et al. (2025, 4.2), disregarding the inert meaning contributions of licenses in the proof. As Dalrymple et al. (2025) observe, this means that (what appears to be) the same meaning expression can correspond to different glue types at different points in the proof.

items and their licensors:

(50) NPI licensor: $((...\ell) \multimap (...\ell)) \multimap p$
NPI: $(...\ell) \multimap (...\ell)$

Fry's licenses have the property of being 'self-cleaning': the license ℓ is introduced within the scope of *nobody* and then consumed, so that the license is available for any negative polarity items within the scope of *nobody*, but not available in the final result. This means that a single negative polarity licensor such as *nobody* can appear in a sentence with one negative polarity item (example (49)), no negative polarity items (example (51)), or more than one negative polarity item (example (53)). Fry (1997, 1999) provides further discussion of this point.

(51) F-structure and instantiated meaning constructors for *Nobody arrived* (potential NPI licensing but no NPIs):

$$_a\begin{bmatrix}\text{PRED} & \text{`ARRIVE}\langle\text{SUBJ}\rangle\text{'} \\ \text{SUBJ} & {}_n\big[\text{PRED } \text{`NOBODY'}\big]\end{bmatrix}$$

$\lambda P.\, nobody(P) : \forall S.((n_e \otimes \ell) \multimap (S \otimes \ell)) \multimap S$
$\lambda X.\, arrive(X) : n_e \multimap a_t$

(52) Proof for *Nobody arrived*:

$$\dfrac{\dfrac{\dfrac{\dfrac{arrive : (n_e \multimap a_t) \qquad [X_3 : n_e]_1}{arrive(X_3) : a_t}\,{\scriptstyle \to E} \qquad [\ell]_2}{arrive(X_3) : (a_t \otimes \ell)}\,{\scriptstyle \otimes I}}{\lambda X_3.arrive(X_3) : ((n_e \otimes \ell) \multimap (a_t \otimes \ell))}\,{\scriptstyle \to I_{1,2}} \qquad \dfrac{nobody : \forall F.(((n_e \otimes \ell) \multimap (F \otimes \ell)) \multimap F)}{nobody : (((n_e \otimes \ell) \multimap (a_t \otimes \ell)) \multimap a_t)}\,{\scriptstyle \forall E}}{nobody(\lambda X_3.arrive(X_3)) : a_t}\,{\scriptstyle \to E}$$

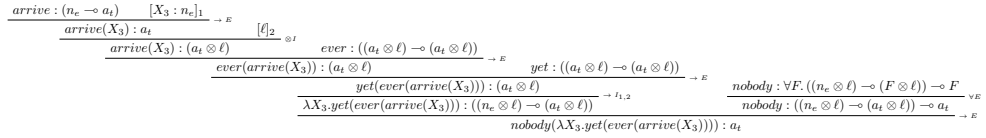
(53) F-structure and instantiated meaning constructors for *Nobody ever arrived yet*, with licenses (NPI licensing with two NPIs):

$$_a\begin{bmatrix}\text{PRED} & \text{`ARRIVE}\langle\text{SUBJ}\rangle\text{'} \\ \text{SUBJ} & {}_n\big[\text{PRED } \text{`NOBODY'}\big] \\ \text{ADJ} & \left\{\begin{bmatrix}\text{PRED } \text{`EVER'}\end{bmatrix}\right. \\ & \left.\begin{bmatrix}\text{PRED } \text{`YET'}\end{bmatrix}\right\}\end{bmatrix}$$

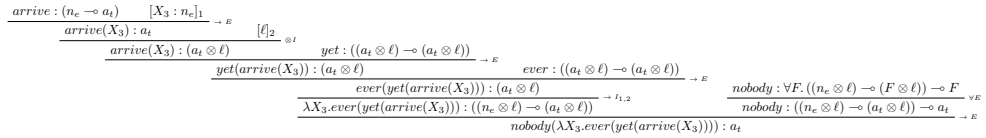
$\lambda P.\, nobody(P) : \forall S.((n_e \otimes \ell) \multimap (S \otimes \ell)) \multimap S$
$\lambda X.\, arrive(X) : n_e \multimap a_t$
$\lambda Q.\, yet(Q) : (a_t \otimes \ell) \multimap (a_t \otimes \ell)$
$\lambda R.\, ever(R) : (a_t \otimes \ell) \multimap (a_t \otimes \ell)$

(54) Two proofs for *Nobody ever arrived yet*:

a. Wide scope for *yet* (within the scope of *nobody*):



b. Wide scope for *ever* (within the scope of *nobody*):



Adopting Fry's intuition that scope constraints can be imposed via licenses analogous to Fry's ℓ license, Dalrymple et al. (2025) develop a means of constraining the relative scope of quantifiers via licenses. In the following, we review and update Dalrymple et al.'s treatment of quantifier scope constraints, and provide an expanded theory of licensing which encompasses constraints on the relative scope of modifiers as well as island constraints.

3.2 Licenses and constraints on semantic composition

We incorporate licenses as a component of the meaning constructors for items which participate in scope constraints, following Fry (1997, 1999) and Dalrymple et al. (2025). Meaning constructors encoding scope constraints govern the participation of license-bearing items in the proof.

3.2.1 Quantifiers

The intuition underpinning Fry's treatment of negative polarity licensors is that the license ℓ is available only within the scope of a negative polarity licensor. Fry's meaning constructor for the negative polarity licensor *nobody* is repeated in (55):

(55) $\lambda P.\, nobody(P) : \forall S.((n_e \otimes \ell) \multimap (S \otimes \ell)) \multimap S$

We adopt this format for quantifiers in general: a constraint requiring a quantifier to take wide scope, and therefore to act as a licensor, makes reference to the license for that quantifier.[5] Unlike Fry's analysis, and following Dalrymple et al. (2025), we assume that each quantifier contributes a *unique* license: that

[5] Dalrymple et al. (2025) propose that quantifiers make two separate contributions: the first contribution requires the presence of a license, and the second contribution provides the license.

$$\lambda P.\, nobody(P) : \forall S.((n \multimap S) \otimes \ell) \multimap S$$
$$\_ : \ell$$

The two formulations are not equivalent: in particular, when a scope constraint is added, Dalrymple et al.'s proposal also constrains scope correctly, but produces multiple proofs of the same meaning. We propose the form in (55) for quantifiers because it explicitly makes the license for a quantifier available only within its scope, in line with Fry's proposal for NPI licensors. We return to this issue in our discussion of modifier scope in Section 3.4.

is, quantifier licenses are separately instantiated for each use of the quantifier.[6] Scope constraints are imposed via introduction of a meaning constructor which references the license of the quantifier that is required to take wide scope. We differentiate licenses by appending a mnemonic letter: ℓe for the license for a particular use of the quantifier *everyone*, ℓs for *someone*, and so on.

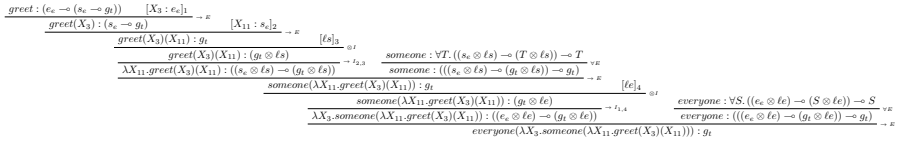
The meaning constructors in (56) include quantifier licenses but no scope constraints, and so no constraints on quantifier scope are imposed. We obtain the same readings as without licenses: either quantifier can take wide scope (the f-structure is as in (34)).

(56) Premises in the derivation of the meaning of *Everyone greeted someone*, including licenses:

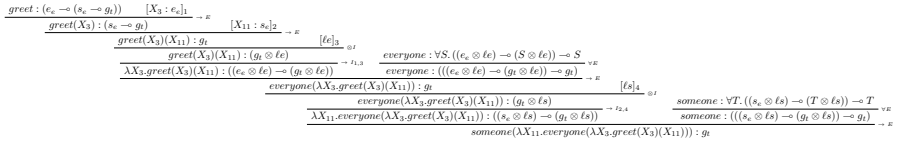
$$\lambda P.\, everyone(P) : \forall S.((e_e \otimes \ell e) \multimap (S \otimes \ell e)) \multimap S$$
$$\lambda Q.\, someone(Q) : \forall T.((s_e \otimes \ell s) \multimap (T \otimes \ell s)) \multimap T$$
$$\lambda X.\lambda Y.\, greet(X, Y) : e_e \multimap (s_e \multimap g_t)$$

(57) Two proofs for *Everyone greeted someone* with no scope constraint

a. Wide scope for *everyone*:



b. Wide scope for *someone*:



3.2.2 Scope requirements

As demonstrated by Dalrymple et al. (2025), quantifier scope can be constrained by the addition of a meaning constructor imposing a scope constraint, as schematically illustrated in (58). The scope constraint in the third line of (58) eliminates any scope possibility where $Q2$ outscopes $Q1$ by linking the individual-type resource for the wide scope quantifier (here, $v1$) with the license for the narrow scope quantifier (here, $\ell 2$).[7] Intuitively, this ensures that the resource for the individual-type variable $v1$ for quantifier $Q1$ is active in the proof while the license $\ell 2$ for $Q2$ is active. For further discussion of how licenses impose quantifier scope constraints, see Dalrymple et al. (2025).

[6] In an LFG setting, this can be accomplished by introducing the license as an instantiated symbol value of an attribute at semantic structure (Dalrymple et al., 2019, 163–164); see Section 4.1.

[7] The scope constraint in (58) is a notational variant of the corresponding constraint proposed by Dalrymple et al. (2025).

(58)

Quantifier taking wide scope:	$Q1 : \forall S.((v1 \otimes \ell 1) \multimap (S \otimes \ell 1)) \multimap S$
Quantifier taking narrow scope:	$Q2 : \forall T.((v2 \otimes \ell 2) \multimap (T \otimes \ell 2)) \multimap T$
Scope constraint:	$(v1 \otimes \ell 2) \multimap (v1 \otimes \ell 2)$

We now consider the f-structure and premises for example (59), *Everyone saw few students*, which is unambiguous, with narrow scope for the object *few students* (Beghelli & Stowell, 1997; Szabolcsi, 1997a). We simplify by providing a single premise for *few students* (assembled from the premises contributed by *few* and *students*, not provided here), and we include a scope constraint in the final line, requiring *everyone* to outscope *few students*.

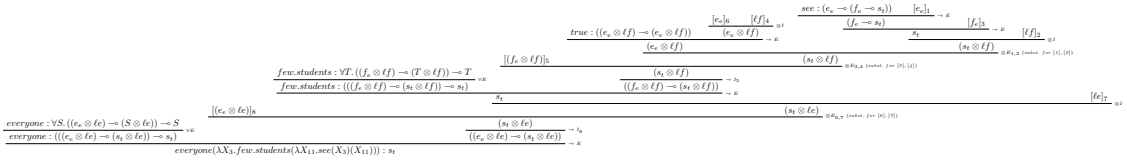
(59) F-structure and instantiated meaning constructors for *Everyone saw few students*, with scope constraint added:

$$l\begin{bmatrix} \text{PRED} & \text{`SEE}\langle\text{SUBJ,OBJ}\rangle\text{'} \\ \text{SUBJ} & e\begin{bmatrix}\text{PRED} & \text{`EVERYONE'}\end{bmatrix} \\ \text{OBJ} & f\begin{bmatrix}\text{SPEC} & \text{`FEW'} \\ \text{PRED} & \text{`STUDENT'}\end{bmatrix} \end{bmatrix}$$

$$\begin{aligned} \lambda P.\, everyone(P) &: \forall S.((e_e \otimes \ell e) \multimap (S \otimes \ell e)) \multimap S \\ \lambda Q.\, few.students(Q) &: \forall T.((f_e \otimes \ell f) \multimap (T \otimes \ell f)) \multimap T \\ \lambda X.\lambda Y.\, see(X,Y) &: e_e \multimap (f_e \multimap s_t) \\ &\quad\ (e_e \otimes \ell f) \multimap (e_e \otimes \ell f) \end{aligned}$$

With the scope constraint, there is only one proof.

(60) Sole proof for *Everyone saw few students* including scope constraint requiring *everyone* to outscope *few students*; full proof in the Appendix[8]



The scope constraint $(e_e \otimes \ell f) \multimap (e_e \otimes \ell f)$ affects only the relative scope of *everyone* and *few students*. If there is a third quantifier, it can scope freely relative to the other two quantifiers. We illustrate this point by reference to the three-quantifier example *Everyone gave someone something*, with a constraint requiring *everyone* to outscope *someone*. From these premises, we get three proofs, with the relative scope of *everyone* and *someone* fixed, and *something* scoping freely.

(61) F-structure and instantiated meaning constructors for *Everyone gave someone something*, with scope constraint requiring *everyone* to outscope *someone*:

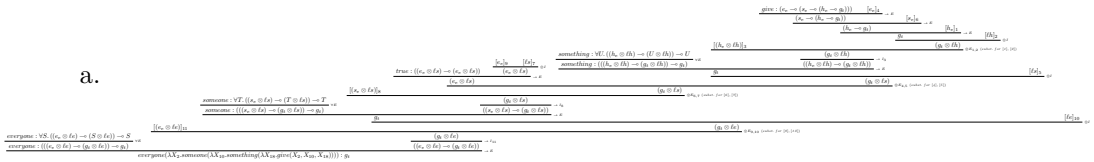
[8] In the proof, the steps labeled $\otimes E_{i,j}$ represent substitution of an i, j pair of assumptions. See the Appendix for the complete proof.

$$l\begin{bmatrix} \text{PRED} & \text{`GIVE}\langle\text{SUBJ,OBJ,OBJ}_{\text{THEME}}\rangle\text{'} \\ \text{SUBJ} & _e\big[\text{PRED `EVERYONE'}\big] \\ \text{OBJ} & _s\big[\text{PRED `SOMEONE'}\big] \\ \text{OBJ}_\theta & _h\big[\text{PRED `SOMETHING'}\big] \end{bmatrix}$$

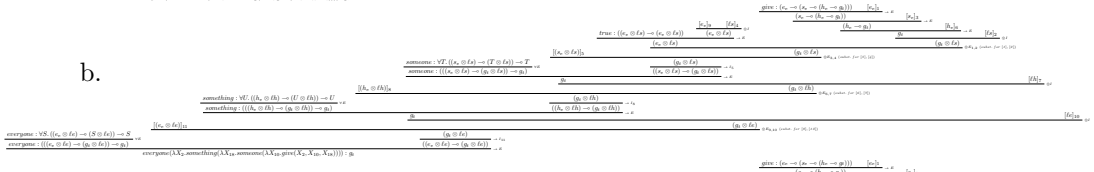
$$\begin{aligned} \lambda P.\, everyone(P) &: \forall S.((e_e \otimes \ell e) \multimap (S \otimes \ell e)) \multimap S \\ \lambda Q.\, someone(Q) &: \forall T.((s_e \otimes \ell s) \multimap (T \otimes \ell s)) \multimap T \\ \lambda R.\, something(R) &: \forall U.((h_e \otimes \ell h) \multimap (U \otimes \ell h)) \multimap U \\ \lambda X.\lambda Y.\lambda Z.\, give(X,Y,Z) &: e_e \multimap (s_e \multimap (h_e \multimap g_t)) \\ &\quad\ (e_e \otimes \ell s) \multimap (e_e \otimes \ell s) \end{aligned}$$

(62) Three proofs for *Everyone gave someone something* including scope constraint requiring *everyone* to outscope *someone*; full proofs in the Appendix

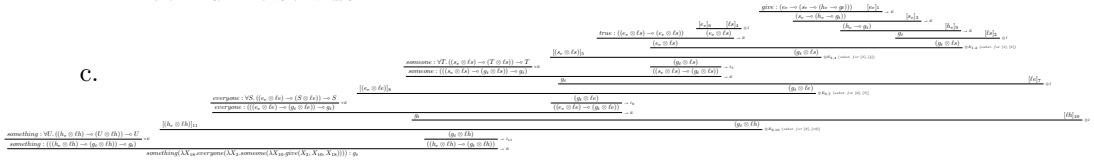
a.



b.



c.



Thus, by the use of licenses, it is possible to impose a scope constraint regulating the relative scope of two quantifiers, as demonstrated by Dalrymple et al. (2025). We will return to the question of how scope constraints are introduced in Section 4.

3.3 Scope islands

As discussed in Section 1.2, we would also like to prevent (certain) quantifiers from escaping (certain) scope islands. Barker (2022) shows that the complement of the verb *claim* is an island for the quantifier *everyone*:

(63) Someone claimed that everyone arrived.

a. $someone(\lambda x.claim(x, everyone(\lambda y.arrive(y))))$
[Available: Someone made the claim: 'everyone arrived']

b. NOT: $everyone(\lambda y.someone(\lambda x.claim(x, arrive(y))))$
[Unavailable: For each person, someone claimed that that person

arrived]

We assume the following f-structure and meaning constructor contributions; note the presence of the licenses associated with the complement of *claim*. We do not yet include a scope constraint requiring *everyone* to scope in the subordinate clause of *claim*, so the analysis is incomplete at this point.

(64) F-structure and meaning constructors for *Someone claimed that everyone arrived*, no scope constraint (provisional, scope constraint to be added):

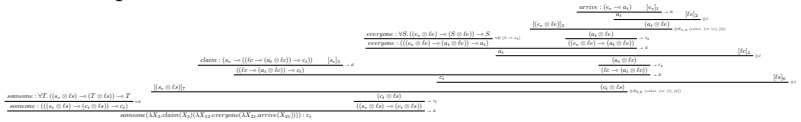
$$_c\begin{bmatrix} \text{PRED} & \text{`CLAIM}\langle\text{SUBJ, COMP}\rangle\text{'} \\ \text{SUBJ} & _s\big[\text{PRED `SOMEONE'}\big] \\ \text{COMP} & _a\begin{bmatrix} \text{PRED} & \text{`ARRIVE}\langle\text{SUBJ}\rangle\text{'} \\ \text{SUBJ} & _e\big[\text{PRED `EVERYONE'}\big] \end{bmatrix} \end{bmatrix}$$

$$\lambda P.\, someone(P) : \forall T.((s_e \otimes \ell s) \multimap (T \otimes \ell s)) \multimap T$$
$$\lambda Q.\, everyone(Q) : \forall S.((e_e \otimes \ell e) \multimap (S \otimes \ell e)) \multimap S$$
$$\lambda X.\lambda Y.\, claim(X, Y) : s_e \multimap ((\ell c \multimap (a_t \otimes \ell c)) \multimap c_t)$$
$$\lambda Z.\, arrive(Z) : e_e \multimap a_t$$

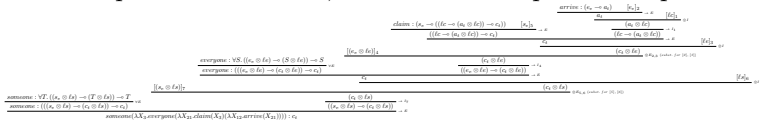
There are four proofs from these premises, depending on how the scope variable S for the quantifier *everyone* is instantiated.

(65) Four proofs for *Someone claimed that everyone arrived* with no scope constraint; full proofs in the Appendix

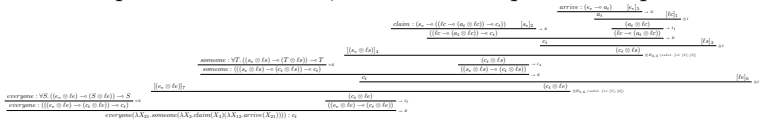
a. The scope variable S for *everyone* is instantiated to a, *everyone* scopes in subordinate clause



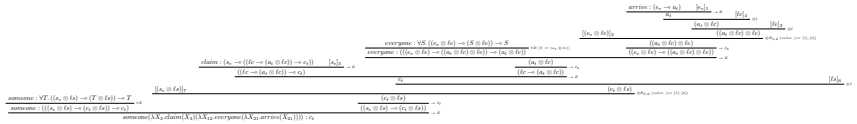
b. The scope variable S for *everyone* is instantiated to c, *everyone* scopes in main clause, takes narrow scope with respect to *someone*



c. The scope variable S for *everyone* is instantiated to c, *everyone* scopes in main clause, takes wide scope with respect to *someone*



d. The scope variable S for *everyone* is instantiated to $a \otimes \ell c$, *everyone* scopes in subordinate clause



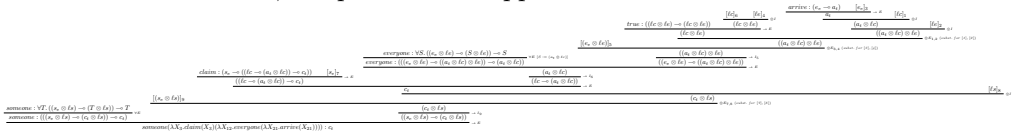
In fact, however, *claim* is an island for *everyone*, meaning that there is no reading on which *everyone* scopes in the main clause. Thus, we must eliminate those readings. To accomplish this, we introduce the scope constraint in the final line in (66), which links the license for *claim* with the license for *everyone*:

(66) Meaning constructors for *Someone claimed that everyone arrived*, including a scope constraint in the fifth line ensuring that *everyone* scopes inside the complement of *claim*:

$$\begin{aligned}
\lambda P.\, someone(P) &: \forall T.((s_e \otimes \ell s) \multimap (T \otimes \ell s)) \multimap T \\
\lambda Q.\, everyone(Q) &: \forall S.((e_e \otimes \ell e) \multimap (S \otimes \ell e)) \multimap S \\
\lambda Y.\lambda X.\, claim(X, Y) &: s_e \multimap ((\ell c \multimap (a_t \otimes \ell c)) \multimap c_t) \\
\lambda Z.\, arrive(Z) &: e_e \multimap a_t \\
&\quad (\ell c \otimes \ell e) \multimap (\ell c \otimes \ell e)
\end{aligned}$$

With this constraint, we get only one proof:

(67) Sole proof for *Someone claimed that everyone arrived* including scope constraint; full proof in the Appendix



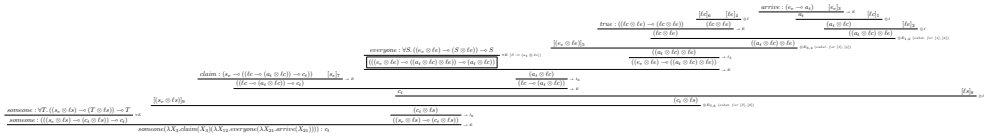
The schematic form for island-creating predicates and the scope constraint enforcing islandhood for particular quantifiers is, then:

(68)

Quantifier:	$Q : \forall S.((q_e \otimes \ell q) \multimap (S \otimes \ell q)) \multimap S$
Island-creating predicate requiring an argument c_t to produce a main-clause resource p_t:	$\lambda C.\, pred(C) : (\ell i \multimap (c_t \otimes \ell i)) \multimap p_t$
Scope constraint:	$(\ell q \otimes \ell i) \multimap (\ell q \otimes \ell i)$

Notably, the proof in (67) is possible because of the universal quantification over possible scopes in the quantifier meaning constructor: in (68), the scope S of *everyone* is instantiated not to a particular scope structure, but to a scope structure tensored with a license: $(a_t \otimes \ell c)$. The relevant step in the proof is boxed in (69). This is why we omitted the type on the variables over potential scopes in the meaning constructors for quantifiers: the universally quantified scope resource can be instantiated to a t-type resource such as a_t, or to a t-type resource tensored with one or more licenses, here $a_t \otimes \ell c$.

(69) Proof in (67) with the instantiation step boxed



Similar issues involving the absence and presence of licenses arise in the analysis of modifier licensing. That situation is different, however, because modifiers have determinate scope and do not introduce universal quantification over potential scopes. This means that their licensing apparatus must be different from that of quantifiers and island-imposing predicates, as we see in the next section.

3.4 Constraints on modifier scope

As noted in Section 1.3, the order and position of modifiers can affect scope possibilities: in some configurations, modifier scope is fixed. Recall that if two modifiers are equally close to the head which they modify (with one modifier preceding the head and the other modifier following it), both scopes are available; otherwise, the modifier that is closer to the head takes narrow scope.

(70) Andrews (1982): Both readings in (72) available for both examples
   a. John intentionally kicked the door twice.
   b. John twice kicked the door intentionally.

(71) Only one reading available:
   a. John kicked the door intentionally twice. (only reading (72-a))
   b. John kicked the door twice intentionally. (only reading (72-b))

(72) a. $twice(intentionally(kick(j, d)))$
   b. $intentionally(twice(kick(j, d)))$

The f-structure for *John kicked the door intentionally twice* is the same as for *John intentionally kicked the door twice*, given in (43).

(73) F-structure for *John kicked the door intentionally twice*:

$$\begin{bmatrix} \text{PRED} & \text{`KICK}\langle\text{SUBJ, OBJ}\rangle\text{'} \\ \text{SUBJ} & _j\begin{bmatrix}\text{PRED} & \text{`JOHN'}\end{bmatrix} \\ \text{OBJ} & _d\begin{bmatrix}\text{PRED} & \text{`DOOR'}\end{bmatrix} \\ \text{ADJ} & \left\{\begin{bmatrix}\text{PRED} & \text{`INTENTIONALLY'}\end{bmatrix}, \begin{bmatrix}\text{PRED} & \text{`TWICE'}\end{bmatrix}\right\} \end{bmatrix}_k$$

To allow for scope constraints, we augment the meaning constructors for the modifiers *twice* and *intentionally* with licenses. Unlike the situation with quantifiers, we need two kinds of license for each modifier: one is relevant for constraints requiring the modifier to take narrow scope, and the other is relevant for constraints requiring it to take wide scope. Thus, modifiers with licenses make a contribution of the form in (74), where the wide-scope license ends with w and the narrow-scope license ends with n.

(74) Meaning constructor for a modifier:

$$M \ : \ (\ell w \multimap (m_t \otimes (\ell w \otimes \ell n))) \multimap m_t \quad \ell n$$

Consider an example with three scope-taking modifiers, such as *John allegedly twice intentionally kicked the door*. Here, we require *allegedly* to outscope *twice* by means of a constraint referring to the wide-scope license of *allegedly* and the narrow-scope license of *twice*, and we also require *twice* to outscope *intentionally* by means of a constraint referring to the wide-scope license of *twice* and the narrow-scope license of *intentionally*. In this case, both the wide-scope and narrow-scope licenses of *twice* play a role in the proof.

Furthermore, unlike the situation with quantifiers, the narrow-scope license must be contributed as a separate premise, along the lines of Dalrymple et al.'s (2025) treatment of quantifier licenses. This is because we need both a positive and a negative occurrence of a narrow-scope license, but we cannot accomplish this by tensoring the narrow scope license with the resource on the right-hand side of the meaning constructor; in other words, the meaning constructor in (75) does not work:

(75) Incorrect meaning constructor for a modifier:

$$M \ : \ (\ell w \multimap (m_t \otimes (\ell w \otimes \ell n))) \multimap (m_t \otimes \ell n)$$

This meaning constructor expects to find a resource of the form $(m_t \otimes \ell n)$ (disregarding the wide scope license ℓw) and produces a result of the same type. However, the licensing resource ℓn can then never be discarded, and we would never be able to achieve a meaning for the sentence of type m_t. Quantifiers avoid this problem because they introduce universal quantification over the output type: they can take as their scope either a bare resource such as m_t, or a tensor product where m_t is tensored with a license, as in (67). In either case, licenses within the scope of a quantifier must be cleaned up, and the licenses in the input and output of the quantifier must match.

Thus, for an example with two modifiers where modifier $M1$ is required to outscope $M2$, we have premises of the following form, with each modifier contributing two premises:

(76)

Modifier taking wide scope:	$M1$	:	$(\ell 1w \multimap (m_t \otimes (\ell 1w \otimes \ell 1n))) \multimap m_t \quad \ell 1n$
Modifier taking narrow scope:	$M2$	:	$(\ell 2w \multimap (m_t \otimes (\ell 2w \otimes \ell 2n))) \multimap m_t \quad \ell 2n$
Scope constraint:			$(\ell 1w \otimes \ell 2n) \multimap (\ell 1w \otimes \ell 2n)$

Let us see how these licenses work to constrain the scope of modifiers. If the modifiers are at an equal distance to the head, as in *John intentionally kicked the door twice*, there is no scope constraint, and both scopes are available. The premises are:

(77) Premises in the derivation of the meaning of *John intentionally kicked*

the door twice, with licensing but no scope constraint:

$$\begin{aligned} \textit{john} &: j_e \\ \textit{door} &: d_e \\ \lambda X.\lambda Y.\, \textit{kick}(X,Y) &: j_e \multimap (d_e \multimap k_t) \\ \lambda P.\, \textit{twice}(P) &: (\ell tw \multimap (k_t \otimes (\ell tw \otimes \ell tn))) \multimap k_t \\ & \quad\ \ \ell tn \\ \lambda P.\, \textit{intentionally}(P) &: (\ell iw \multimap (k_t \otimes (\ell iw \otimes \ell in))) \multimap k_t \\ & \quad\ \ \ell in \end{aligned}$$

(78) Two proofs for *John intentionally kicked the door twice*, no scope constraint; full proofs in the Appendix

a.

$$\dfrac{\textit{twice} : ((\ell tw \multimap (k_t \otimes (\ell tw \otimes \ell tn))) \multimap k_t) \qquad \dfrac{\textit{intentionally} : ((\ell iw \multimap (k_t \otimes (\ell iw \otimes \ell in))) \multimap k_t) \qquad \dfrac{\dfrac{\textit{kick} : (j_e \multimap (d_e \multimap k_t)) \qquad \textit{john} : j_e}{(d_e \multimap k_t)}{\scriptstyle \to E} \qquad \textit{door} : d_e}{k_t}{\scriptstyle \to E} \qquad \dfrac{[\ell iw]_1 \qquad \textit{true} : \ell in}{(\ell iw \otimes \ell in)}{\scriptstyle \otimes I}}{(k_t \otimes (\ell iw \otimes \ell in))}{\scriptstyle \to I_1}}{(\ell iw \multimap (k_t \otimes (\ell iw \otimes \ell in)))}{\scriptstyle \to E} \qquad \dfrac{[\ell tw]_2 \qquad \textit{true} : \ell tn}{(\ell tw \otimes \ell tn)}{\scriptstyle \otimes I}}{k_t}{\scriptstyle \to E}}{(k_t \otimes (\ell tw \otimes \ell tn))}{\scriptstyle \to I_2}}{(\ell tw \multimap (k_t \otimes (\ell tw \otimes \ell tn)))}{\scriptstyle \to E}$$

$$\textit{twice}(\lambda X_6.\textit{intentionally}(\lambda X_{14}.\textit{kick}(\textit{john})(\textit{door}))) : k_t$$

b.

$$\dfrac{\textit{intentionally} : ((\ell iw \multimap (k_t \otimes (\ell iw \otimes \ell in))) \multimap k_t) \qquad \dfrac{\dfrac{\textit{twice} : ((\ell tw \multimap (k_t \otimes (\ell tw \otimes \ell tn))) \multimap k_t) \qquad \dfrac{\dfrac{\textit{kick} : (j_e \multimap (d_e \multimap k_t)) \qquad \textit{john} : j_e}{(d_e \multimap k_t)}{\scriptstyle \to E} \qquad \textit{door} : d_e}{k_t}{\scriptstyle \to E} \qquad \dfrac{[\ell tw]_1 \qquad \textit{true} : \ell tn}{(\ell tw \otimes \ell tn)}{\scriptstyle \otimes I}}{(k_t \otimes (\ell tw \otimes \ell tn))}{\scriptstyle \to I_1}}{(\ell tw \multimap (k_t \otimes (\ell tw \otimes \ell tn)))}{\scriptstyle \to E} \qquad \dfrac{[\ell iw]_2 \qquad \textit{true} : \ell in}{(\ell iw \otimes \ell in)}{\scriptstyle \otimes I}}{k_t}{\scriptstyle \to E}}{(k_t \otimes (\ell iw \otimes \ell in))}{\scriptstyle \to I_2}}{(\ell iw \multimap (k_t \otimes (\ell iw \otimes \ell in)))}{\scriptstyle \to E}$$

$$\textit{intentionally}(\lambda X_{14}.\textit{twice}(\lambda X_6.\textit{kick}(\textit{john})(\textit{door}))) : k_t$$

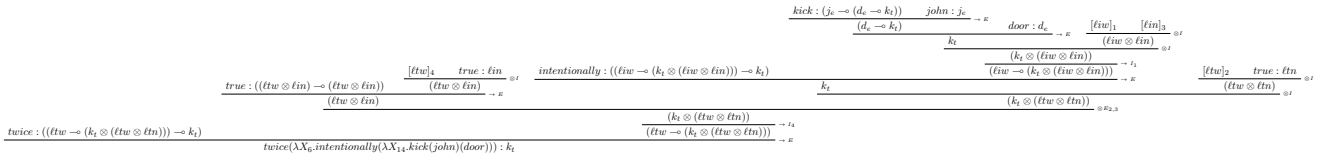
If one modifier is farther from the head, only one reading is available. For the example *John kicked the door intentionally twice*, the modifier *twice* is farther from the head and must take wide scope, so we introduce the scope constraint in the final line of (79).

(79) Premises in the derivation of the meaning of *John kicked the door intentionally twice*, with scope constraint in the final line:

$$\begin{aligned} \textit{john} &: j_e \\ \textit{door} &: d_e \\ \lambda X.\lambda Y.\, \textit{kick}(X,Y) &: j_e \multimap (d_e \multimap k_t) \\ \lambda P.\, \textit{twice}(P) &: (\ell tw \multimap (k_t \otimes (\ell tw \otimes \ell tn))) \multimap k_t \\ & \quad\ \ \ell tn \\ \lambda P.\, \textit{intentionally}(P) &: (\ell iw \multimap (k_t \otimes (\ell iw \otimes \ell in))) \multimap k_t \\ & \quad\ \ \ell in \\ & \quad\ \ (\ell tw \otimes \ell in) \multimap (\ell tw \otimes \ell in) \end{aligned}$$

There is only one proof from these premises, in which *twice* takes wide scope relative to *intentionally*:

(80) Sole proof for *John kicked the door intentionally twice*, with licensing; full proof in the Appendix:



4 Introducing the constraints

We have seen that scope constraints involving licenses can constrain the scope possibilities for quantifiers and modifiers. We now turn to the question of how the scope constraints are introduced into the derivation. We provide an LFG perspective on license introduction: we assume that the license components of a scope constraint are uniquely instantiated atoms which are the value of an attribute at semantic structure.

The question of exactly what triggers the introduction of scope constraints is complicated, and we will not provide a comprehensive treatment here; rather, we will introduce some potential mechanisms for the introduction of scope constraints, leaving many issues open for future research.

4.1 Licenses and their syntactic visibility

In an LFG setting, licenses are the values of attributes at semantic structure. This allows reference to the license of the quantifier, modifier, or predicate creating a scope island in scope constraints.

4.1.1 Quantifiers

The full lexical entry for a quantifier like *everyone* is as follows, where $\ell\_$ is an instantiated symbol (as indicated by the trailing underscore) that is instantiated to a unique value on each occasion of its use:

(81) Lexical entry for *everyone*:

everyone N $(\uparrow \textsc{pred}) = \text{`}\textsc{everyone}\text{'}$
$(\uparrow_\sigma \textsc{license}) = \ell\_$
$\lambda P.\, everyone(P) : \forall S.((\uparrow_e \otimes (\uparrow_\sigma \textsc{license})) \multimap (S \otimes (\uparrow_\sigma \textsc{license}))) \multimap S$

The f-structure and semantic structure for a particular use of the quantifier *everyone* are given in (82):

(82) F-structure, semantic structure, and meaning constructor for *everyone*:

Functional structure: Semantic structure:

$_e\big[\textsc{pred}\ \text{`}\textsc{everyone}\text{'}\big]$ $\quad$ $_{e_\sigma}\big[\textsc{license}\ \ell e\big]$

$\lambda P.\, everyone(P) : \forall S.((e_e \otimes \ell e) \multimap (S \otimes \ell e)) \multimap S$

4.1.2 Modifiers

A modifier has two licenses, one involved in constraints requiring the modifier to take wide scope, and the other involved in constraints requiring it to take narrow scope. The lexical entry for a modifier like *twice* is as in (83); note that $\ell 1\_$ and $\ell 2\_$ are instantiated to different unique values, and that this lexical entry contributes two meaning constructors, the complex expression in the next to last line and the narrow-scope license contribution in the last line:[9]

(83) Lexical entry for *twice*:

twice Adv $(\uparrow \text{PRED}) = \text{`TWICE'}$
$(\uparrow_\sigma \text{WIDE-LICENSE}) = \ell 1\_$
$(\uparrow_\sigma \text{NARROW-LICENSE}) = \ell 2\_$

$$\lambda P.\, twice(P) : ((\uparrow_\sigma \text{WIDE-LICENSE}) \multimap ((\text{ADJ} \in \uparrow)_t \otimes ((\uparrow_\sigma \text{WIDE-LICENSE}) \otimes (\uparrow_\sigma \text{NARROW-LICENSE})))) \multimap (\text{ADJ} \in \uparrow)_t$$
$$(\uparrow_\sigma \text{NARROW-LICENSE})$$

The f-structure and semantic structure for a particular use of *twice* as a member of the modifier set of an f-structure m are given in (84):

(84) F-structure, semantic structure, and meaning constructor for *twice*:

Functional structure: Semantic structure:

$$m\left[\text{ADJ } \left\{ {}_t\left[\text{PRED } \text{`TWICE'}\right] \right\}\right] \qquad {}_{t_\sigma}\begin{bmatrix} \text{WIDE-LICENSE} & \ell w \\ \text{NARROW-LICENSE} & \ell n \end{bmatrix}$$

$\lambda P.\, twice(P) : (\ell w \multimap (m_t \otimes (\ell w \otimes \ell n))) \multimap m_t$
ℓn

4.1.3 Verbs inducing scope islands

Finally, the lexical entry for a verb whose complement is a scope island, such as *claim*, is:

(85) Lexical entry for *claim*:

claim V $(\uparrow \text{PRED}) = \text{`CLAIM}\langle\text{SUBJ, COMP}\rangle\text{'}$
$(\uparrow_\sigma \text{LICENSE}) = \ell\_$

$$\lambda X.\lambda Y.\, claim(X, Y) : (\uparrow \text{SUBJ})_e \multimap (((\uparrow_\sigma \text{LICENSE}) \multimap ((\uparrow \text{COMP})_t \otimes (\uparrow_\sigma \text{LICENSE}))) \multimap \uparrow_t)$$

The f-structure and semantic structure for a particular use of *claim* are given in (86):

(86) F-structure, semantic structure, and meaning constructor for *claim*:

[9] Notationally, the final line in (83) appears to be an existential constraint; however, it is intended to introduce the narrow-scope license as a resource in the meaning derivation. It may be helpful to introduce some notational means to distinguish the meaning constructor contributions in a lexical entry from the other constraints, to avoid confusion.

Functional structure: Semantic structure:

$$_{c}\begin{bmatrix}\text{PRED} & \text{`CLAIM}\langle\text{SUBJ, COMP}\rangle\text{'}\\ \text{SUBJ} & s\\ \text{COMP} & p\end{bmatrix} \qquad _{c_\sigma}\big[\text{LICENSE } \ell c\big]$$

$$\lambda Y.\lambda X.\, claim(X,Y) : s_e \multimap ((\ell c \multimap (p_t \otimes \ell c)) \multimap c_t)$$

4.2 Quantifier strength

In order to enforce scope constraints relying on relative quantifier strength, we need a way of representing the strength of a quantifier. This is a complicated issue, explored by Szabolcsi (1997b), references cited there, and much subsequent work. For concreteness, we choose a simple representation of quantifier strength that we will provisionally assume to be relevant both for constraints on relative quantifier scope and for ability to escape quantifier islands, based loosely on Barker (2022). It is likely that this simple representation will need to be modified and enriched to handle the full range of constraints that are attested.

We represent quantifier strength by a set of numerals, with the largest numeral also represented as the value of the attribute QMAX. If we assume for the sake of argument that *everyone* has strength 1 and *few students* has strength 0 (specified by the determiner *few*), their semantic structures will include the attributes and values in (87).[10]

(87) a. *everyone*:
$$_{q1}\begin{bmatrix}\text{QSTRENGTH} & \{0,1\}\\ \text{QMAX} & 1\end{bmatrix}$$

b. *few*:
$$_{q2}\begin{bmatrix}\text{QSTRENGTH} & \{0\}\\ \text{QMAX} & 0\end{bmatrix}$$

To check the relative strength of two quantifiers $Q1$ and $Q2$, we can check whether $Q1$'s QMAX value is included in the QSTRENGTH of $Q2$, provided that $Q1$ and $Q2$ each have a value for the QMAX feature (that is, they are both quantifiers). If this fails, $Q1$ is stronger than $Q2$. In particular, assuming the values in (87), *everyone* is stronger than *few*, since *everyone*'s QMAX value is not included in the QSTRENGTH of *few*. We can use $Q1 >_q Q2$ as an abbreviation for '$Q1$ is stronger than $Q2$':

(88) $Q1 >_q Q2$ ($Q1$ is stronger than $Q2$):
($q1$ QMAX)
($q2$ QMAX)
($q1$ QMAX) $\notin$ ($q2$ QSTRENGTH)

[10] The value of QSTRENGTH is specified as a closed set, which does not allow additional values to be added:

(i) *everyone* (↑ PRED)='EVERYONE'
(↑$_\sigma$ QSTRENGTH) = $\{0,1\}$

We use this relation in enforcing relative quantifier constraints.

4.3 Introduction of constraints enforcing relative quantifier scope

Often, in the literature on constraints on quantifier scoping, the discussion centers on the relative scope of arguments of the same predicate, and in fact the most commonly cited examples involve the relative scope of subject and object coarguments of the same predicate. With Ioup (1975), Liu (1997), Beghelli & Stowell (1997), Szabolcsi (1997a), and others, we assume that grammatical function and quantifier strength are important in determining quantifier scope possibilities.

In this literature, particularly in approaches set in a movement-based theory such as Principles and Parameters theory or Minimalism, the focus is often on how to expand the range of quantifier scopes that is predicted by the theory: in other words, it is assumed that quantifier scope is tightly constrained (by grammatical function or syntactic configuration), and the aim is to provide a means for relaxing these constraints in particular circumstances to allow quantifier scopes that would otherwise not be available. In a glue setting, the situation is the reverse: glue derivations allow all quantifier scope possibilities, and our goal is to eliminate those possibilities that are unattested.

Example (89), an expanded version of (59) including semantic structures e_σ and f_σ, presents the f-structure, semantic structure, and meaning constructors for *Everyone saw few students*, which (as noted in Section 3.2.2) has been claimed to be unambiguous, with *everyone* taking widest scope. Given that we have provisionally assigned *everyone* a strength of 1 and *few students* a strength of 0, the strength of the quantifier *everyone* is greater than the strength of *few students*.

(89) F-structure, semantic structure, and instantiated meaning constructors for *Everyone saw few students*, with scope constraint added:

$$l\begin{bmatrix}\text{PRED} & \text{`SEE}\langle\text{SUBJ,OBJ}\rangle\text{'} \\ \text{SUBJ} & e\big[\text{PRED `EVERYONE'}\big] \\ \text{OBJ} & f\begin{bmatrix}\text{SPEC} & \text{`FEW'} \\ \text{PRED} & \text{`STUDENT'}\end{bmatrix}\end{bmatrix} \qquad e_\sigma\begin{bmatrix}\text{LICENSE} & \ell e \\ \text{QSTRENGTH} & \{0,1\} \\ \text{QMAX} & 1\end{bmatrix}$$

$$f_\sigma\begin{bmatrix}\text{LICENSE} & \ell f \\ \text{QSTRENGTH} & \{0\} \\ \text{QMAX} & 0\end{bmatrix}$$

$$\begin{aligned}\lambda P.\, everyone(P) &: \forall S.((e_e \otimes \ell e) \multimap (S \otimes \ell e)) \multimap S \\ \lambda Q.\, few.students(Q) &: \forall T.((f_e \otimes \ell f) \multimap (T \otimes \ell f)) \multimap T \\ \lambda X.\lambda Y.\, see(X,Y) &: e_e \multimap (f_e \multimap s_t) \\ &\quad\ (e_e \otimes \ell f) \multimap (e_e \otimes \ell f)\end{aligned}$$

Here, we explore the possibility that the scope constraint is introduced by the verb: if the strength of a transitive verb's subject is greater than the strength of its object, a scope constraint is introduced requiring the subject to outscope the

object. The third line in the lexical entry in (90) enforces this constraint; it says: 'If my SUBJ is stronger than my OBJ, introduce a scope constraint requiring my SUBJ to outscope my OBJ.' Note that if either the subject or the object is not a quantifier, the $>_q$ relation does not hold, and the constraint fails to apply.

(90) saw V $(\uparrow \text{PRED}) = \text{`SEE}\langle\text{SUBJ, OBJ}\rangle\text{'}$
$\lambda X.\lambda Y.\, see(X,Y) \;:\; (\uparrow \text{SUBJ})_e \multimap (\uparrow \text{OBJ})_e \multimap \uparrow_t$
$[(\uparrow \text{SUBJ})_\sigma >_q (\uparrow \text{OBJ})_\sigma] \Rightarrow [[(\uparrow \text{SUBJ})_\sigma \otimes ((\uparrow \text{OBJ})_\sigma \text{ LICENSE})] \multimap [(\uparrow \text{SUBJ})_\sigma \otimes ((\uparrow \text{OBJ})_\sigma \text{ LICENSE})]]$

It is well known that constraints on quantifier scope also depend in a complex way on linear order and other factors. Relative quantifier scope constraints could also be introduced in phrase structure rules or via f-precedence relations, but we leave an exploration of these possibilities to future research.

4.4 Introduction of constraints enforcing scope islands

In Section 3.3, we saw that the general format for a premise enforcing an island constraint is as in the final line of (91) (=(68)), where the license ℓi for the island-creating predicate is tensored with the license ℓq for the quantifier within the island:

(91)

Quantifier:	Q	$: \forall S.((q_e \otimes \ell q) \multimap (S \otimes \ell q)) \multimap S$
Island-creating predicate:	$\lambda C.\, p(C)$	$: (\ell i \multimap (c_t \otimes \ell i)) \multimap p_t$
Scope constraint:		$(\ell q \otimes \ell i) \multimap (\ell q \otimes \ell i)$

In order to enforce islandhood, a scope constraint must be introduced for each quantifier (of a particular maximum strength) within the island. This means that the process of introducing the scope constraint must be managed by the quantifier, since there is no way for the predicate enforcing a scope island to know how many quantifiers are contained in its island. Islands can be of various strengths, as explored by Barker (2022) and illustrated in the chart in (12). Thus, each quantifier must check whether it is in an island that it is not allowed to escape, and if it is, it must introduce a scope constraint preventing itself from scoping outside the island.

We assume with Barker that islands are specified with strengths, in a similar manner to quantifiers. Barker assigns a strength of 2 to the complement of *claim*, labeled p in (92):

(92) F-structure, semantic structure, and meaning constructor for *claim*:

Functional structure: Semantic structure:

$$_c\begin{bmatrix} \text{PRED} & \text{`CLAIM}\langle\text{SUBJ, COMP}\rangle\text{'} \\ \text{SUBJ} & s \\ \text{COMP} & p \end{bmatrix} \qquad _{p_\sigma}\begin{bmatrix} \text{QSTRENGTH} & \{0,1,2\} \\ \text{QMAX} & 2 \\ \text{LICENSE} & \ell c \end{bmatrix}$$

$\lambda Y.\lambda X.\, claim(X,Y) : s_e \multimap ((\ell c \multimap (p_t \otimes \ell c)) \multimap c_t)$

Quantifiers must check their environment for an enclosing COMP whose strength is higher than their own, and introduce a scope constraint if such a COMP is

found. The following constraint accomplishes this for the quantifier *everyone*: we are assuming that *everyone* has a strength of 1, and so the scoping constraint checks for a COMP with a strength of at least 2.

(93) Island scope constraint for *everyone*:

$$\begin{array}{l}(\quad \textsc{comp} \qquad\qquad \textsc{comp}^{*} \qquad \textsc{gf}\ \uparrow) \qquad \Rightarrow \\ \quad 2 \in (\rightarrow_{\sigma} \textsc{qstrength}) \quad 2 \notin (\rightarrow_{\sigma} \textsc{qstrength}) \\ \quad (\rightarrow_{\sigma}\ \textsc{license}) = \%L \\ \qquad\qquad\qquad\qquad ((\%L \otimes (\uparrow_{\sigma}\ \textsc{license})) \multimap (\%L \otimes (\uparrow_{\sigma}\ \textsc{license})))\end{array}$$

This quantifier looks outwards for a COMP that has a QSTRENGTH of at least 2 (making it stronger than the QSTRENGTH of *everyone*), ignoring any COMPs that do not have a QSTRENGTH of at least 2. If such a COMP is found, it is an island for *everyone*, and a scope constraint linking the licenses of *everyone* and the COMP island is introduced. Other quantifiers may specify a greater or lesser ability to escape islands by specifying a number greater or smaller than 2 in their scope constraint.

4.5 Introduction of constraints enforcing modifier scope

In Section 3.4, we saw that the general format for a premise enforcing scope relations between modifiers is as in the final line of (94) (=(76)), where the wide-scope license $\ell 1w$ for the wide-scoping modifier is tensored with the narrow-scope license $\ell 2n$ of the narrow-scoping modifier:

(94)

Modifier taking wide scope:	$M1$	:	$(\ell 1w \multimap (m_t \otimes (\ell 1w \otimes \ell 1n))) \multimap m_t\ \ell 1n$
Modifier taking narrow scope:	$M2$	:	$(\ell 2w \multimap (m_t \otimes (\ell 2w \otimes \ell 2n))) \multimap m_t\ \ell 2n$
Scope constraint:			$(\ell 1w \otimes \ell 2n) \multimap (\ell 1w \otimes \ell 2n)$

We explore the possibility that the scope constraint is introduced by the phrase structure rule.[11] We assume the phrase structure rule in (95), which handles right adjoined modifiers, and gives us the c-structure in (96) for *John kicked the door twice intentionally*.[12]

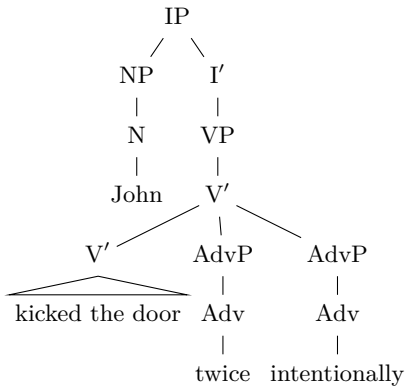
[11] In fact, a basic modifier contribution could also be transformed into a modifier contribution with licenses by the phrase structure rule, so that only modifiers which are sisters in the phrase structure would acquire the licensing apparatus. We do not explore this possibility here.

[12] This rule has the category V′ on both its left and right side, so in principle it can apply recursively, which would allow for right adjoined modifiers at different levels, and would incorrectly allow scope ambiguities for adjacent right-adjoined modifiers if the scope constraint is introduced only for modifiers that are sisters at c-structure. This problem could be solved by treating V′ as a complex category, and introducing a feature ±MOD to distinguish the mother V′[+MOD] from the daughter V′[−MOD]: this would disallow recursive application of the rule, and would force all post-V′ modifiers of a head to be sisters in the phrase structure tree. Alternatively, it would be possible to use a recursive rule to combine modifiers with the head one at a time, but then a complex rule appealing to f-precedence would be required to introduce

(95) V′ rule for right adjunction of modifiers, preliminary:

$$\begin{array}{lcll} \mathrm{V}' & \longrightarrow & \mathrm{V}' & \mathrm{XP}^{+} \\ & & \uparrow = \downarrow & \downarrow \in (\uparrow \textsc{adj}) \end{array}$$

(96)



We now augment the phrase structure rule to introduce scope constraints, rewriting the sequence of one or more modifiers XP+ as XP XP*, and introducing scoping constraints for the modifiers after the first XP. These scope constraints require adverbial phrases closer to the head to take narrow scope relative to adverbial phrases farther from the head. Here, we are concerned with post-V′ modifiers, so we want each modifier after the first modifier to scope over the previous modifier.

(97) Revised V′ rule for right adjunction of modifiers, non-final:

$$\begin{array}{lcllll} \mathrm{V}' & \longrightarrow & \mathrm{V}' & \mathrm{XP} & \mathrm{XP}^{*} \\ & & \uparrow = \downarrow & \downarrow \in (\uparrow \textsc{adj}) & \downarrow \in (\uparrow \textsc{adj}) \end{array}$$

To do this, we require reference to the f-structure of the left or right sister node and their corresponding f-structures: the node immediately to the left is referred to as $<*$ and its f-structure is $<*_{\phi}$, and the node immediately to the right is referred to as $*>$ and its f-structure is $*>_{\phi}$ (Dalrymple et al., 2019, 6.4). Here, we want to say that each non-initial postmodifier introduces a scope constraint linking its wide scope license with the narrow scope license of the immediately preceding modifier. The following additional annotation accomplishes this:

(98) Revised V′ rule for right adjunction of modifiers, final:

the modifier scope constraint. We present this rule as a simple way of introducing the scope constraint, and leave exploration of other methods and refinements to future research. One important empirical question to be answered is whether modifier scope constraints apply only to adjacent modifiers, as we would assume if scope constraints are introduced by particular phrase structure rules, or also to nonadjacent modifiers in languages allowing discontinuous nominal phrases.

$$\begin{array}{lcccc} \mathrm{V}' & \longrightarrow & \mathrm{V}' & \mathrm{XP} & \mathrm{XP}^{*} \\ & & \uparrow = \downarrow & \downarrow \in (\uparrow \textsc{adj}) & \downarrow \in (\uparrow \textsc{adj}) \\ & & & & ((\downarrow_{\sigma} \textsc{wide-license}) \otimes (<*_{\phi\sigma} \textsc{narrow-license})) \multimap \\ & & & & ((\downarrow_{\sigma} \textsc{wide-license}) \otimes (<*_{\phi\sigma} \textsc{narrow-license})) \end{array}$$

A similar constraint is needed for premodifiers, where all modifiers but the final modifier introduce a constraint referring to the immediately following modifier.

5 Remaining issues

We have proposed a general licensing mechanism for constraining quantifier and modifier scope and enforcing scope islands. Much work remains to be done, and we hope that the current proposals will provide a basis for this work.

5.1 Theoretical issues to be explored

Among the theoretical issues which remain to be explored:

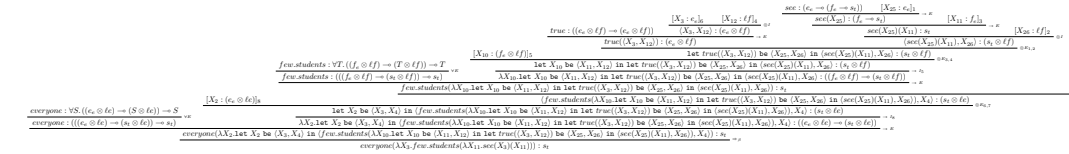
- What is the domain within which relative quantifier scope constraints are defined? Do such constraints operate only on coarguments, on clausemates, or within a wider domain? The answer to this question will determine the range of formal possibilities for introduction of quantifier scope constraints.
- Are quantifiers associated with a single strength ranking that is relevant for both relative quantifier scoping and ability to escape islands, or do we need different specifications of quantifier strength for different kinds of constraints?
- How do modifier scope constraints work in languages with very free word order and discontinuity? Should modifier scope constraints be defined in terms of f-precedence rather than by a phrase structure rule?

5.2 Computational issues

Zymla (2024) notes that Glue suffers from two kinds of problems with ambiguity: spurious ambiguity and unwanted ambiguity. The current proposal constitutes a step towards solving the unwanted ambiguity problem, but the spurious ambiguity problem remains, and may even become more serious because of the complications to many of the premises that this approach introduces. Perhaps we are currently at a stage comparable to the period during which phrase structure trees were used to model linguistic structure, but before chart parsing and efficient parsing algorithms had been developed. Clearly, further research is needed.

A Full proofs

(99) Full proof for example (60), *Everyone saw few students*, including scope constraint requiring wide scope for *Everyone*. In each of the following proofs, $\Rightarrow_\beta$ marks the term obtained by reduction from the final line, treating premises such as *true* as the identity function.

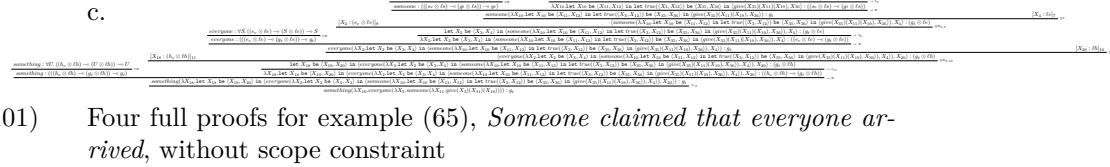


(100) Three full proofs for example (62), *Everyone gave someone something*, including scope constraint requiring *everyone* to outscope *someone*, producing a three-way ambiguity

a.

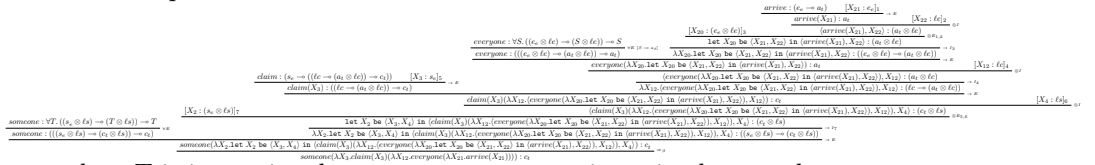
b.

c.

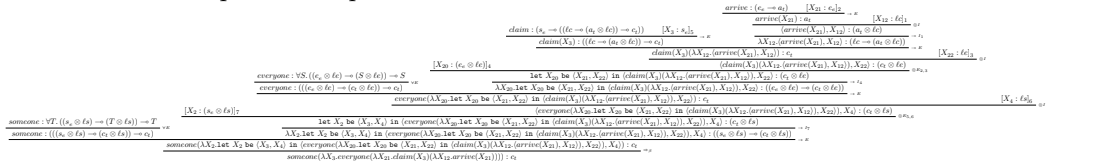


(101) Four full proofs for example (65), *Someone claimed that everyone arrived*, without scope constraint

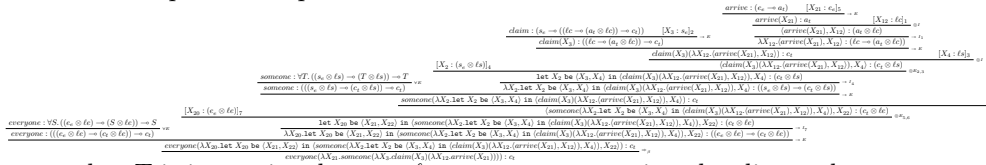
a. The scope variable T for *everyone* is instantiated to a, *everyone* scopes in subordinate clause



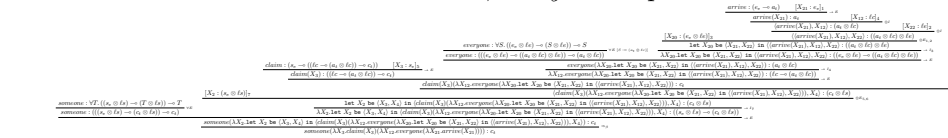
b. T is instantiated to c, *everyone* scopes in main clause, takes narrow scope with respect to *someone*



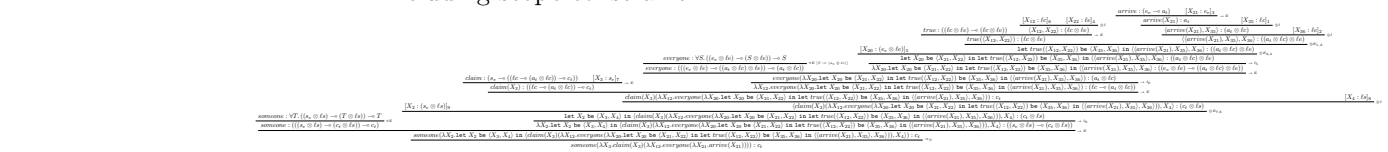
c. T is instantiated to c, *everyone* scopes in main clause, takes wide scope with respect to *someone*



d. T is instantiated to $a \otimes \ell c$, *everyone* scopes in subordinate clause

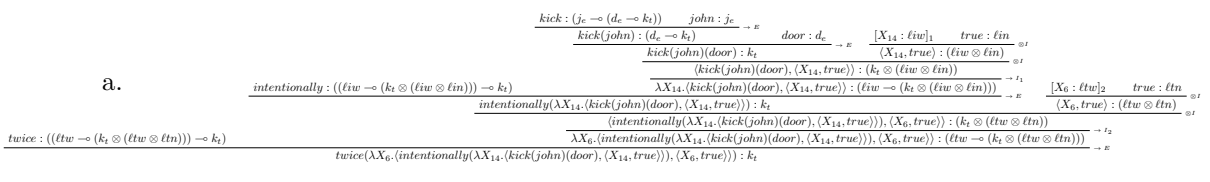


(102) Sole proof for example (67), *Someone claimed that everyone arrived*, including scope constraint

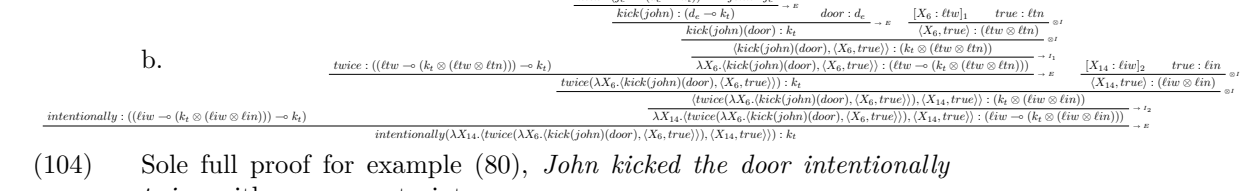


(103) Two full proofs for example (78), *John intentionally kicked the door twice*, no scope constraint

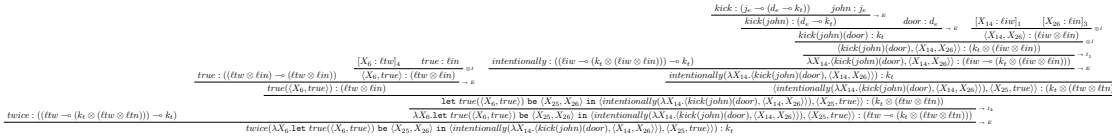
a.



b.



(104) Sole full proof for example (80), *John kicked the door intentionally twice*, with scope constraint



References

Andrews, Avery D., III. 1982. A note on the constituent structure of adverbials and auxiliaries. *Linguistic Inquiry* 13. 313–317.

Andrews, Avery D., III. 2018. Sets, heads, and spreading in LFG. *Journal of Language Modelling* 6(1). 131–174. doi:10.15398/jlm.v6i1.175.

Andrews, Avery D., III & Christopher D. Manning. 1999. *Complex predicates and information spreading in LFG*. Stanford: CSLI Publications.

Asudeh, Ash. 2022. Glue Semantics. *Annual Review of Linguistics* 8. 321–341. doi:10.1146/annurev-linguistics-032521-053835.

Asudeh, Ash. 2023. Glue semantics. In Dalrymple (2023). `https://langsci-press.org/catalog/book/312`.

Asudeh, Ash & Richard Crouch. 2001. Glue semantics for HPSG. In Frank Van Eynde, Lars Hellan & Dorothee Beermann (eds.), *Proceedings of the 8th International Conference on Head-Driven Phrase Structure Grammar*, 1–19. Stanford: CSLI Publications. `http://cslipublications.stanford.edu/HPSG/2/hpsg01.html`.

Asudeh, Ash & Mary Dalrymple. in press. Glue semantics. In *Handbook of linguistic semantics: Bridging theory, philosophy and cognition*, Springer Nature.

Barker, Chris. 2022. Rethinking scope islands. *Linguistic Inquiry* 53(4). 633–661. doi:10.1162/ling_a_00419.

Beghelli, Filippo & Tim Stowell. 1997. Distributivity and negation: The syntax of each and every. In Anna Szabolcsi (ed.), *Ways of scope taking*, 71–107. Dordrecht: Springer.

Belyaev, Oleg. 2023a. Core concepts of LFG. In Dalrymple (2023). `https://langsci-press.org/catalog/book/312`.

Belyaev, Oleg. 2023b. Introduction to LFG. In Dalrymple (2023). `https://langsci-press.org/catalog/book/312`.

Crouch, Richard & Josef van Genabith. 1999. Context change, underspecification and the structure of Glue language derivations. In *Semantics and syntax in Lexical Functional Grammar*, 117–189. MIT Press.

Dalrymple, Mary (ed.). 2023. *The handbook of Lexical Functional Grammar*. Language Science Press. `https://langsci-press.org/catalog/book/312`.

Dalrymple, Mary, Jamie Y. Findlay & Dag T. T. Haug. 2025. Constraining scope in Glue Semantics: The linear logic approach. In Miriam Butt, Jamie Y. Findlay & Ida Toivonen (eds.), *Proceedings of the LFG'25 Conference*, 110–131. Stanford, CA: CSLI Publications. `https://lfg-proceedings.org/lfg/index.php/main/article/view/75/68`.

Dalrymple, Mary, John Lamping, Fernando C. N. Pereira & Vijay A. Saraswat. 1995. Linear logic for meaning assembly. In Suresh Manandhar, Gabriel Pereira Lopes & Werner Nutt (eds.), *Proceedings of computational logic for natural language processing*, 75–93. Edinburgh. `https://arxiv.org/abs/cmp-lg/9504012`.

Dalrymple, Mary, John Lamping, Fernando C. N. Pereira & Vijay A. Saraswat. 1997. Quantifiers, anaphora, and intensionality. *Journal of Logic, Language and Information* 6(3). 219–273. doi:10.1023/A:1008224124336.

Dalrymple, Mary, John Lamping & Vijay Saraswat. 1993. LFG semantics via constraints. In Steven Krauwer, Michael Moortgat & Louis des Tombe (eds.), *Proceedings of the Sixth Conference of the European Chapter of the Association for Computational Linguistics (EACL 1993)*, 97–105. Association for Computational Linguistics. doi:10.3115/976744.976757.

Dalrymple, Mary, John J. Lowe & Louise Mycock. 2019. *The Oxford Reference Guide to Lexical Functional Grammar*. Oxford: Oxford University Press. doi:10.1093/oso/9780198733300.001.0001.

Farkas, Donka & Anastasia Giannakidou. 1996. How clause-bounded is the scope of universals? In Teresa Galloway & Justin Spence (eds.), *Proceedings of SALT 6*, 35–52. `https://journals.linguisticsociety.org/proceedings/index.php/SALT/article/view/2764`.

Findlay, Jamie Y. & Dag T. T. Haug. 2022. Managing scope ambiguities in Glue via multistage proving. In Miriam Butt, Jamie Y. Findlay & Ida Toivonen (eds.), *Proceedings of the LFG'22 Conference*, 143–163. Stanford, CA: CSLI Publications. `https://lfg-proceedings.org/lfg/index.php/main/article/view/18`.

Fry, John. 1997. Negative polarity licensing at the syntax–semantics interface. In *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics (ACL-EACL 97)*, 144–150. Madrid: Association for Computational Linguistics. doi:10.3115/976909.979636.

Fry, John. 1999. Proof nets and negative polarity licensing. In *Semantics and syntax in Lexical Functional Grammar*, 91–116. MIT Press.

Girard, Jean-Yves. 1987. Linear logic. *Theoretical Computer Science* 50(1). 1–102. doi:10.1016/0304-3975(87)90045-4.

Gotham, Matthew. 2018. Making Logical Form type-logical: Glue semantics for Minimalist syntax. *Linguistics & Philosophy* 41(5). 511–556. doi:10.1007/s10988-018-9229-z.

Gotham, Matthew. 2019. Constraining scope ambiguity in LFG+Glue. In Miriam Butt, Tracy Holloway King & Ida Toivonen (eds.), *Proceedings of the LFG'19 Conference*, Stanford, CA: CSLI Publications. https://typo.uni-konstanz.de/lfg-proceedings/LFGprocCSLI/LFG2019/lfg2019-gotham.pdf.

Gotham, Matthew. 2021. Approaches to scope islands in LFG+Glue. In Miriam Butt, Jamie Y. Findlay & Ida Toivonen (eds.), *Proceedings of the LFG'21 Conference*. Stanford, CA: CSLI Publications. https://typo.uni-konstanz.de/lfg-proceedings/LFGprocCSLI/LFG2021/lfg2021-gotham.pdf.

Ioup, Georgette Lee. 1975. *The treatment of quantifier scope in a transformational grammar*: City University of New York dissertation.

Jackendoff, Ray S. 1972. *Semantic interpretation in generative grammar*, Current Studies in Linguistics. Cambridge, MA: MIT Press.

Janssen, Theo M. V. & Thomas Ede Zimmermann. 2025. Montague semantics. The Stanford Encyclopedia of Philosophy. https://plato.stanford.edu/archives/spr2025/entries/montague-semantics/.

Ladusaw, William A. 1979. *Polarity sensitivity as inherent scope relations*: The University of Texas at Austin dissertation.

Lakoff, George. 1971. On generative semantics. In Danny D. Steinberg & Leon A. Jakobovits (eds.), *Semantics: An interdisciplinary reader in philosophy, linguistics and psychology*, Cambridge, UK: Cambridge University Press.

Larson, Richard & Sungeun Cho. 2003. Temporal adjectives and the structure of possessive dps. *Natural Language Semantics* 11(3). 217–247.

Larson, Richard K. 1990. Double objects revisited: reply to Jackendoff. *Linguistic Inquiry* 21(4). 589–632. https://www.jstor.org/stable/4178697.

Liu, Feng-Hsi. 1997. *Scope and specificity*. John Benjamins.

Partee, Barbara H. & Vladimir Borschev. 1998. Integrating lexical and formal semantics: Genitives, relational nouns, and type-shifting. In Robin Cooper & Thomas Gamkrelidze (eds.), *Proceedings of the second tbilisi symposium on language, logic, and computation*, 229–241. Tbilisi: Center on Language, Logic, Speech, Tbilisi State University. http://www-unix.oit.umass.edu/~partee/docs/tbilisi17.pdf.

Partee, Barbara H. & Vladimir Borschev. 2003. Genitives, relational nouns, and argument-modifier ambiguity. In Ewald Lang, Claudia Maienborn & Cathrine Fabricius-Hansen (eds.), *Modifying adjuncts*, vol. 4, Interface Explorations, 67–112. Berlin: Mouton de Gruyter.

Reinhart, Tanya. 1997. Quantifier scope: How labor is divided between QR and choice functions. *Linguistics and Philosophy* 20(4). 335–397.

Rodman, Robert. 1976. Scope phenomena, "movement transformations", and relative clauses. In Barbara H. Partee (ed.), *Montague grammar*, New York: Academic Press. doi:10.1016/c2013-0-11289-5.

Schneider-Zioga, P. 1988. Double objects and small clauses. USC, Los Angeles.

Szabolcsi, Anna. 1997a. Strategies for scope taking. In Anna Szabolcsi (ed.), *Ways of scope taking*, 109–154. Dordrecht: Kluwer.

Szabolcsi, Anna (ed.). 1997b. *Ways of scope taking*. Dordrecht: Kluwer.

Zymla, Mark-Matthias. 2024. Ambiguity management in computational Glue semantics. In Miriam Butt, Jamie Y. Findlay & Ida Toivonen (eds.), *Proceedings of the LFG'24 Conference*, 285–310. Konstanz: PubliKon. `https://lfg-proceedings.org/lfg/index.php/main/article/view/59`.