Approaches to scope islands in LFG+Glue

Matthew Gotham, University of Oxford

In this paper I examine two possible approaches to scope islands in LFG with Glue semantics. For each approach, I show how it could be made to account for novel empirical arguments made in [1], and go on to argue that there are at least some conceptual reasons to prefer the less conservative approach.

Background It is a commonplace in the literature on quantification that certain syntactic structures are 'islands' for scope, in the sense that no quantifier within the structure may take scope over any element outside the structure. The parade example of such a structure is the finite clause. For example, (1) has only one interpretation, as indicated below.

(1) A warden thinks that every prisoner escaped.
$\Rightarrow$ $\textbf{a}(\textbf{warden}, \lambda x.\textbf{think}(x, \textbf{every}(\textbf{prisoner}, \textbf{escape})))$ *a warden > every prisoner*
$\not\Rightarrow$ $\textbf{every}(\textbf{prisoner}, \lambda y.\textbf{a}(\textbf{warden}, \lambda x.\textbf{think}(x, \textbf{escape}(y))))$ **every prisoner > a warden*

However, the generalization that finite clauses are scope islands may well be too coarse. On the one hand, indefinites have become known as 'exceptional scope takers' because, despite being quantificational, they are not constrained by scope islands in this way. For example, (2) allows both the *a warden > every prisoner* and *every prisoner > a warden* interpretations.

(2) Every warden thinks that a prisoner escaped.

Moreover, several authors have noted that in certain circumstances finite clauses are not islands for universal quantifiers either. For example, (3) permits both the *an accomplice > every prisoner* and *every prisoner > an accomplice* interpretations.

(3) An accomplice ensured that every prisoner escaped.

[1] argues that whether or not a clause is an island for scope depends both on the predicate embedding the clause and the relevant quantifier embedded in it. So for example, while the clause embedded by *ensure* is not a scope island for *every N*, it is for a monotone-decreasing quantifier such as *no N*, as in (4)—which is anomalous because the only available interpretation contradicts world knowledge regarding what it means to be an accomplice.

(4) #An accomplice ensured that no prisoner escaped.
$\Rightarrow$ $\textbf{a}(\textbf{accomplice}, \lambda x.\textbf{ensure}(x, \textbf{no}(\textbf{prisoner}, \textbf{escape})))$ *an accomplice > no prisoner*
$\not\Rightarrow$ $\textbf{no}(\textbf{prisoner}, \lambda y.\textbf{a}(\textbf{accomplice}, \lambda x.\textbf{ensure}(x, \textbf{escape}(y))))$ **no prisoner > an accomplice*

As [1] notes, there is a further application of this insight when it comes to negative polarity items (NPIs). To be licensed, an NPI must be interpreted within the scope of an appropriate 'negative' licensor—[3] shows a method for ensuring this in LFG+Glue. However, as is acknowledged in [3], this method has the shortcoming that it does not ensure that an NPI be interpreted in the scope of its *closest* relevant licensor. For example, in (5) there are two potential licensors for the NPI *anyone*: *surprised* and *didn't*; but the NPI has to be interpreted as scoping under both of them, as shown. A natural explanation for this would be that, in addition to being *licensors* for NPIs, at least some such expressions also induce *scope islands* for NPIs.

(5) Martha is surprised that Mary didn't help anyone.
$\Rightarrow$ $\textbf{surprise}(\textbf{not}(\textbf{someone}(\lambda x.\textbf{help}(\textbf{mary}, x))), \textbf{martha})$ *surprised > didn't > anyone*
$\not\Rightarrow$ $\textbf{surprise}(\textbf{someone}(\lambda x.\textbf{not}(\textbf{help}(\textbf{mary}, x))), \textbf{martha})$ **surprised > anyone > didn't*

As [1] also notes, there appears to be a systematicity to this interaction between scope island inducers and escapers, dubbed the **Scope Island Subset Constraint** (SISC): 'given any two scope takers, the set of scope islands that trap one is a subset of the set of scope islands that trap the other'. The interaction of scope islands and scope takers is summarized in Figure 1. (For this hierarchy to work, we have to take *didn't* to be of type $(e \to t) \to e \to t$ so as not to block a *subject* quantifer from taking scope over sentential negation.)

Blocking features and off-path constraints Following a suggestion in [4], we could attempt to account for the differential scope-taking abilities of different quantifiers by means of constraints on the path between the f-structure position of a quantifier and its scope-taking position. As a baseline, a potential lexical entry for an indefinite might be as shown in (6)a., where 'exceptional scope' is permitted by dint of the path being unconstrained. By contrast, the received view according to which finite clauses are scope islands can be expressed by the toy lexical entry in (6)b., as now the path cannot pass through an f-structure with a TENSE value.

(6) a. *someone* N
$\%A = (\text{GF}^* \ \text{GF} \uparrow)$
$\textbf{someone} : (\uparrow \multimap \%A) \multimap \%A$

b. *everyone* N
$\%A = \left(\begin{array}{cc} \text{GF}^* & \text{GF} \uparrow \\ \neg(\rightarrow \text{TENSE}) & \end{array} \right)$
$\textbf{everyone} : (\uparrow \multimap \%A) \multimap \%A$

Evidently, this baseline is inadequate to account for the interaction between quantifiers and clause-embedders described above and in Figure 1. What we would need is for clause-embedders to project some feature into f-structure that quantifiers could be sensitive to. For example, we could differentiate the island strengths projected by *think* and *ensure* by means of the partial lexical entries in (7). Then the different island-escaping abilities of *every N* and *no N* can be expressed in (8).

(7) a. *thinks* V
$(\uparrow \textsc{comp ScopeIsland}) = 2$
$\lambda p.\lambda x.\textbf{think}(x,p) : (\uparrow \textsc{comp}) \multimap (\uparrow \textsc{subj}) \multimap \uparrow$

b. *ensured* V
$(\uparrow \textsc{comp ScopeIsland}) = 1$
$\lambda p.\lambda x.\textbf{ensure}(x,p) : (\uparrow \textsc{comp}) \multimap (\uparrow \textsc{subj}) \multimap \uparrow$

(8) a. *no-one* N
$\%A = \left(\begin{array}{cc} \textsc{gf}^* & \textsc{gf} \uparrow \\ (\rightarrow \textsc{ScopeIsland}) \neq \{1 \mid 2 \mid 3\} & \end{array} \right)$
$\lambda P.\textbf{not}(\textbf{someone}(P)) : (\uparrow \multimap \%A) \multimap \%A$

b. *everyone* N
$\%A = \left(\begin{array}{cc} \textsc{gf}^* & \textsc{gf} \uparrow \\ (\rightarrow \textsc{ScopeIsland}) \neq \{2 \mid 3\} & \end{array} \right)$
$\textbf{everyone} : (\uparrow \multimap \%A) \multimap \%A$

This approach makes use only of established LFG+Glue technology, and the ScopeIsland attribute functions like e.g. the LDD attribute in the analysis of the effect of 'bridge verbs' on long-distance dependencies in [2]. However, we seem to be missing a generalization: what we would really like to be able to say in the off-path constraints is something like $\neg((\rightarrow \textsc{ScopeIsland}) > 0)$ in (8)a. and $\neg((\rightarrow \textsc{ScopeIsland}) > 1)$ in (8)b. But of course this is impossible because, despite the suggestive names, the feature values are atoms and not integers that can be compared by inequality symbols. Relatedly, to the extent that the SISC is a reliable generalization our means of accounting for it are poor. For example, there is nothing in the formal system that stops a quantifier from having the off-path constraint $(\rightarrow \textsc{ScopeIsland}) \neq \{1 \mid 3\}$—i.e. being able to escape from the islands induced by *ensure* and *didn't* but not *think*.

Multi-modal Glue semantics An alternative approach would be to impose scope islands within Glue semantics. The base fragment (i.e. excluding quantification) of linear logic normally used in Glue, multiplicative intuitionistic linear logic or MILL, is equivalent to the Lambek calculus with permutation or **LP**, and relates to the logics **L** and **NL** (the associative and non-associative Lambek calculi) commonly used in categorial grammar as shown in Figure 2 ([6]).

So far, this has been a good choice of logic for Glue: unlike in categorial grammar, the logic is not meant to account for word order and so it makes sense for it to be commutative. So far it has also made sense for the logic to be associative, but scope islands may actually give us a reason to care about how premises are grouped, and so restrict associativity. We can do so selectively by combining elements of **LP** (as before) and **NLP** (which is non-associative) in a multimodal system, where the modes correspond to the island/escaper strengths outlined in Figure 1. The rules of inference for the fragment given in Figure 3 show how the modes interact. Note that now, because we no longer assume generalized associativity, there is bracketing on the left hand side of sequents. The mode indices on those brackets correspond to mode indices on occurrences of $\multimap$. Commutativity is ensured by the structural rule P (for *permutation*), and we have restricted associativity thanks to the rule MA (*mixed associativity*). This, in combination with the lexicon shown in Figure 5, permits just the right scope takers to escape from just the right islands.

For example, we have the following theorems, corresponding respectively to the inverse scope readings of (2) and (3) respectively. We show the proof of (10) in Figure 6 for reference, based on the simplified f-structure shown in Figure 4.

(9) $((([\text{escaped}], [\text{thinks}])^3, [\text{every warden}])^0, [\text{a prisoner}])^0 \vdash \textbf{a}(\textbf{prisoner}, \lambda x.\textbf{every}(\textbf{warden}, \lambda y.\textbf{think}(y, \textbf{escape}(x))))$

(10) $((([\text{escaped}], [\text{ensured}])^1, [\text{an accomplice}])^0, [\text{every prisoner}])^0$
$\vdash \textbf{every}(\textbf{prisoner}, \lambda x.\textbf{a}(\textbf{accomplice}, \lambda y.\textbf{ensure}(y, \textbf{escape}(x))))$

By contrast, there is no grouping of the relevant premises in (1) or (4) that will permit an inverse scope interpretation to be derived. The point to note is that MA is required for an auxiliary assumption to get into a position from which it can be abstracted in the appropriate step of $\multimap$ I. In Figure 6 the first instance of MA is the crucial one: MA can apply because the inner and outer bracket indices are both 1. If the embedding verb had instead been *thinks* then the outer bracket index would have been 2; and if the quantifier intended to take wide scope had instead been *no prisoner* then the inner index would have had to have been 0. In either case MA would have been inapplicable. These results have all been checked in the Grail theorem prover for type-logical grammars ([5]).

Discussion The multi-modal Glue approach requires a significant complication of the syntax/semantics interface, but the benefit is that the SISC follows from the rules of inference for the linear logic fragment, whereas it is hard to see how the SISC could be enforced in the blocking features-based approach, for the reasons given above. Moving forward, the discussion in [4] strongly suggests that some kind of complication of the linear logic fragment used in Glue will turn out to be necessary in order to account for *intra*-clausal constraints on quantifier scope, anyway. It remains to be seen whether the approach advocated in [4], or here, can be assimilated to the other, thereby maximizing the empirical benefit for the amount of additional complexity introduced.

References [1] C. Barker. Rethinking Scope Islands. *Linguistic Inquiry* (2021). Advance publication. [2] M. Dalrymple et al. *The Oxford Reference Guide to Lexical Functional Grammar*. OUP, 2019. [3] J. Fry. Proof Nets and Negative Polarity Licensing. In: *Semantics and Syntax in Lexical Functional Grammar*. Ed. by M. Dalrymple. MIT Press, 1999, 91–116. [4] M. Gotham. Constraining Scope Ambiguity in LFG+Glue. *Proceedings of the LFG Conference* 24 (2019), 111–129. [5] R. Moot. The Grail Theorem Prover. In: *Modern Perspectives in Type-Theoretical Semantics*. Ed. by S. Chatzikyriakidis and Z. Luo. Springer, 2017, 247–277. [6] R. Moot and C. Retoré. *The Logic of Categorial Grammars*. Springer, 2012.

	a(n) N	any N	every N	no N	**Island strength**
doesn't		*	*	*	3
think			*	*	2
ensure				*	1
Escaper strength	3	2	1	0	

Figure 1: Relative strength of islands and escapers, adapted from [1]

$$\begin{tikzcd}
 & \textbf{LP} & \\
\textbf{L} \arrow[ur, "\text{commut}"] & & \textbf{NLP} \arrow[ul, "\text{assoc}"'] \\
 & \textbf{NL} \arrow[ul, "\text{assoc}"] \arrow[ur, "\text{commut}"'] &
\end{tikzcd}$$

Figure 2: Substructural type logics

For modes $i, j \in \{0, 1, 2, 3\}$:

$$\dfrac{\Gamma \vdash x : A \quad \Delta \vdash f : A \multimap_i B}{(\Gamma, \Delta)^i \vdash f(x) : B} \multimap_i \text{E} \qquad \dfrac{(x : A, \Gamma)^i \vdash y : B}{\Gamma \vdash \lambda x.y : A \multimap_i B} \multimap_i \text{I} \qquad \dfrac{\text{axiom}}{x : A \vdash x : A}$$

$$\dfrac{(\Gamma, \Delta)^i \vdash x : A}{(\Delta, \Gamma)^i \vdash x : A} \text{P} \qquad \dfrac{((\Gamma, \Delta)^i, \Sigma)^j \vdash x : A}{(\Gamma, (\Delta, \Sigma)^j)^i \vdash x : A} \text{MA} \qquad \leftarrow \text{provided that } i \geq j$$

Figure 3: Rules of inference for multi-modal Glue

$$f : \left[\begin{array}{ll} \text{PRED} & \text{‘ensure}\langle g, h\rangle\text{’} \\ \text{SUBJ} & g : \left[\text{“an accomplice”}\right] \\ \text{COMP} & h : \left[\begin{array}{ll} \text{PRED} & \text{‘escape}\langle i\rangle\text{’} \\ \text{SUBJ} & i : \left[\text{“every prisoner”}\right] \end{array}\right] \end{array}\right]$$

Figure 4: Simplified f-structure of (3)

$$\begin{aligned}
\textit{thinks} &\rightsquigarrow [\text{thinks}] := \lambda p.\lambda x.\mathbf{think}(x, p) : (\uparrow \text{COMP}) \multimap_2 ((\uparrow \text{SUBJ}) \multimap_i \uparrow) \\
\textit{ensured} &\rightsquigarrow [\text{ensured}] := \lambda p.\lambda x.\mathbf{ensure}(x, p) : (\uparrow \text{COMP}) \multimap_1 ((\uparrow \text{SUBJ}) \multimap_i \uparrow) \\
\textit{a warden} &\rightsquigarrow [\text{a warden}] := \lambda P.\mathbf{a}(\mathbf{warden}, P) : (\uparrow \multimap_3 \%A) \multimap_0 \%A \\
\textit{every prisoner} &\rightsquigarrow [\text{every prisoner}] := \lambda P.\mathbf{every}(\mathbf{prisoner}, P) : (\uparrow \multimap_1 \%A) \multimap_0 \%A \\
\textit{no prisoner} &\rightsquigarrow [\text{no prisoner}] := \lambda P.\mathbf{not}(\mathbf{a}(\mathbf{prisoner}, P)) : (\uparrow \multimap_0 \%A) \multimap_0 \%A \\
\textit{escaped} &\rightsquigarrow [\text{escaped}] := \mathbf{escape} : (\uparrow \text{SUBJ}) \multimap_i \uparrow
\end{aligned}$$

Index i on an occurrence of $\multimap$ indicates that any of the modes is possible (so really we have lexical schemata). The local name $\%A$ is defined as in (6)a. I could alternatively have used linear logic quantification instead of a local name.

Figure 5: Lexicon

$$\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{}{x : i \vdash x : i}
\quad
\dfrac{}{[\text{escaped}] \vdash \mathbf{escape} : i \multimap_1 h}
}{(x, [\text{escaped}])^1 \vdash \mathbf{escape}(x) : h} \multimap_1 \text{E}
\quad
\dfrac{}{[\text{ensured}] \vdash \lambda p.\lambda y.\mathbf{ensure}(y, p) : h \multimap_1 (g \multimap_3 f)}
}{((x, [\text{escaped}])^1, [\text{ensured}])^1 \vdash \lambda y.\mathbf{ensure}(y, \mathbf{escape}(x)) : g \multimap_3 f} \multimap_1 \text{E}
}{(x, ([\text{escaped}], [\text{ensured}])^1)^1 \vdash \lambda y.\mathbf{ensure}(y, \mathbf{escape}(x)) : g \multimap_3 f} \text{MA}
\quad
\dfrac{}{[\text{an accomplice}] \vdash \lambda P.\mathbf{a}(\mathbf{accomplice}, P) : (g \multimap_3 f) \multimap_0 f}
}{((x, ([\text{escaped}], [\text{ensured}])^1)^1, [\text{an accomplice}])^0 \vdash \mathbf{a}(\mathbf{accomplice}, \lambda y.\mathbf{ensure}(y, \mathbf{escape}(x))) : f} \multimap_0 \text{E}
}{(x, (([\text{escaped}], [\text{ensured}])^1, [\text{an accomplice}])^0)^1 \vdash \mathbf{a}(\mathbf{accomplice}, \lambda y.\mathbf{ensure}(y, \mathbf{escape}(x))) : i \multimap_1 f} \text{MA}
}{(([\text{escaped}], [\text{ensured}])^1, [\text{an accomplice}])^0 \vdash \lambda x.\mathbf{a}(\mathbf{accomplice}, \lambda y.\mathbf{ensure}(y, \mathbf{escape}(x))) : f} \multimap_1 \text{I}
\quad
\dfrac{}{[\text{every prisoner}] \vdash \lambda P.\mathbf{every}(\mathbf{prisoner}, P) : (i \multimap_1 f) \multimap_0 f}
}{((([\text{escaped}], [\text{ensured}])^1, [\text{an accomplice}])^0, \text{every prisoner})^0 \vdash \mathbf{every}(\mathbf{prisoner}, \lambda x.\mathbf{a}(\mathbf{accomplice}, \lambda y.\mathbf{ensure}(y, \mathbf{escape}(x)))) : f} \multimap_0 \text{E}$$

Figure 6: Derivation of the inverse scope interpretation of (3)