Polarity Sensitivity and Scope in 'Glue Language' Semantics

John Fry

Stanford University/Xerox PARC

Proceedings of the LFG97 Conference

University of California, San Diego

Miriam Butt and Tracy Holloway King (Editors)

1997
CSLI Publications
`http://www-csli.stanford.edu/publications/`

1 Introduction

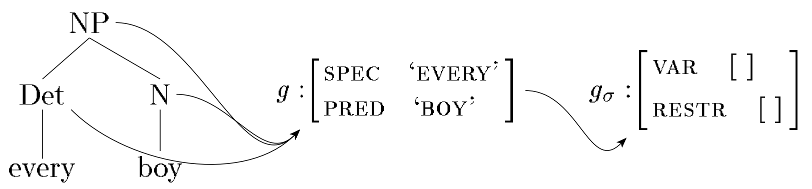
A recent strain of research on the interface between syntax and semantics in LFG, starting with [DLS93], uses a fragment of linear logic [Gir87] as a 'glue language' for assembling the meaning of a sentence from the meanings of its individual words and constituents. Among the advantages of this deductive approach is an elegant account of quantifier scoping in which each possible scope reading has a unique corresponding proof, obviating the need for quantifier storage [DLPS94].

In addition to its role in assembling scope readings, the linear logic glue language can also be harnessed to constrain the combinatorics of scope readings at the syntax-semantics interface in a variety of interesting ways. For example, Dalrymple *et al.* [DLPSar] showed how properties of the glue language prevent certain scopings which are ill-formed due to their interaction with bound anaphora. More recently, Fry [Fry97] proposed a glue language implementation of the constraint, originally formulated by Ladusaw [Lad79], that negative polarity items (NPI's) be licensed withing the semantic scope of a negative or 'monotonically decreasing' context.

The present paper gives a brief review of the basic glue language approach to quantifier scope, followed by an expanded version of the NPI licensing proposal originally introduced in [Fry97]. Whereas [Fry97] focused on linear logic deduction using proof nets (a topic which is ignored here), the present account fleshes out the original NPI proposal in greater detail and extends the treatment to new areas such as right-decreasing quantifiers and 'weak' vs. 'strong' licensers.

2 Background: quantification in glue language semantics

In the account of quantification in [DLPSar], sentences and their phrasal constituents are assigned a c-structure and f-structure, and also a semantic or s-structure which is related to the f-structure via a projection σ. For example, the quantified NP *every boy* is assigned the following structures:



The semantic projection g_σ has two attributes, $(g_\sigma\ \textsc{var})$ and $(g_\sigma\ \textsc{restr})$. The value of VAR is the first-order e-type variable, e.g. x, that is bound by the quantifier. The value of RESTR is the restriction of the generalized quantifier; in this case, the meaning of *boy*.

The premises of a meaning deduction are glue language *meaning constructors* which are associated with each entry in the lexicon. The meaning constructor for the determiner *every* after it has been instantiated with the f-structure label g is given in (1).

(1) **every:** $\forall H, R, S.\ (\forall x.\ (g_\sigma\ \textsc{var}) \rightsquigarrow_e x \multimap (g_\sigma\ \textsc{restr}) \rightsquigarrow_t R(x))$
$\quad \otimes (\forall y.\ g_\sigma \rightsquigarrow_e y \multimap H \rightsquigarrow_t S(y))$
$\quad \multimap H \rightsquigarrow_t \mathit{every}(R, S)$

Formula (1) can be interpreted as follows. The higher-order variables R and S range over the meanings of the restriction and scope, respectively, of the quantifier. H ranges over possible scopes for the entire sentence in which the quantifier occurs. Lower-case x and y act simply as traditional first-order variable "placeholders" within the restriction and scope.[1] The subformula

$$\forall x.\ (g_\sigma\ \text{VAR}) \rightsquigarrow_e x \multimap (g_\sigma\ \text{RESTR}) \rightsquigarrow_t R(x)$$

specifies that if $(g_\sigma\ \text{VAR})$ means x, then that meaning can be "consumed" (in the linear logic version of modus ponens) in order to produce the meaning $R(x)$ of the quantifier's restriction $(g_\sigma\ \text{RESTR})$. Similarly, in the subformula

$$\forall y.\ g_\sigma \rightsquigarrow_e y \multimap H \rightsquigarrow_t S(y)$$

if the meaning $g_\sigma \rightsquigarrow_e y$ can be consumed then the scope H of the entire generalized quantifier can take on the value $S(y)$.

In a typical proof, the meaning constructor for a noun will supply appropriate values for the VAR and RESTR attributes of the NP, and these resources will be consumed by the determiner's meaning constructor. In this example, the noun is *boy* and its meaning constructor is given in (2).

(2) **boy:** $\forall X.\ (g_\sigma\ \text{VAR}) \rightsquigarrow_e X \multimap (g_\sigma\ \text{RESTR}) \rightsquigarrow_t \mathit{boy}(X)$

Together, the meaning constructors (1) and (2) act as premises for the deduction of the meaning constructor for the NP *every boy*. Applying the variable substitutions $X \mapsto x$ and $R \mapsto \mathit{boy}$ followed by modus ponens on (1) and (2) yields formula (3).

(3) **every-boy:** $\forall H, S.\ (\forall y.\ g_\sigma \rightsquigarrow_e y \multimap H \rightsquigarrow_t S(y)) \multimap H \rightsquigarrow_t \mathit{every}(\mathit{boy}, S)$

We now demonstrate how the quantified NP contributes to the meaning of a sentence in which it appears, using sentence (4) as an example.

(4) Every boy left.

$$f\!:\left[\begin{array}{ll} \text{PRED} & \text{`LEAVE'} \\ \text{SUBJ} & g\!:\left[\begin{array}{ll} \text{SPEC} & \text{`EVERY'} \\ \text{PRED} & \text{`BOY'} \end{array}\right] \end{array}\right]$$

The meaning constructor for the intransitive verb *left*, after being instantiated with the values in (4), is shown in (5).

(5) **left:** $\forall Y.\ g_\sigma \rightsquigarrow_e Y \multimap f_\sigma \rightsquigarrow_t \mathit{leave}(Y)$

To derive a meaning for (4), we apply the substitutions $H \mapsto f_\sigma$, $Y \mapsto y$ and $S \mapsto \lambda y.\mathit{leave}(y)$ and apply modus ponens to (3) and (5). This produces what is in this simple case the only possible reading:

$$\textbf{every-boy-left:}\quad f_\sigma \rightsquigarrow_t \mathit{every}(\mathit{boy}, \lambda y.\mathit{leave}(y)).$$

A limited higher-order unification scheme produces the unifier $\lambda x.p(x)$ from a second-order term $S(x)$ and a first-order term $p(x)$. As noted in [DLPSar], all the apparatus that is required for these examples falls within the decidable $l\lambda$ fragment of Miller [Mil90], and therefore can be implemented as an extension of a first-order unification scheme such as that of Prolog.

[1]Dalrymple *et al.* [DLPSar] make a distinction between quantification of upper-case variables like H, R, S and lower-case variables such as x. Lower-case variables stand for arbitrary constants rather than particular terms, and generally are given limited scope within the antecedent of the outermost implication. Upper-case variables are Prolog-like variables that become instantiated to specific terms within the proof, and generally scope over the entire meaning constructor.

2.1 Scope ambiguities

In the case of a sentence with more than one quantifier, its multiple scope readings emerge naturally with each unique derivation. Consider the sentence

(6) Every boy saw a girl.

$$f\!: \begin{bmatrix} \text{PRED} & \text{`SEE'} \\ \text{SUBJ} & g\!: \begin{bmatrix} \text{SPEC} & \text{`EVERY'} \\ \text{PRED} & \text{`BOY'} \end{bmatrix} \\ \text{OBJ} & h\!: \begin{bmatrix} \text{SPEC} & \text{`A'} \\ \text{PRED} & \text{`GIRL'} \end{bmatrix} \end{bmatrix}$$

In this case the new premise **a-girl** is made available:

$$\textbf{a-girl}\!: \;\; \forall H, S.\; (\forall z.\, h_\sigma \rightsquigarrow_e z \multimap H \rightsquigarrow_t S(z)) \multimap H \rightsquigarrow_t a(girl, S)$$

The two possible readings depend on the relative order of application of the generalized quantifiers **a-girl** and **every-boy**. If **every-boy** is applied to the predicate before **a-girl**, we obtain the wide-scope $\forall\exists$ reading

$$f_\sigma \rightsquigarrow_t every(boy, \lambda u.a(girl, \lambda v.see(u, v))).$$

Otherwise the narrow-scope $\exists\forall$ reading is produced:

$$f_\sigma \rightsquigarrow_t a(girl, \lambda v.every(boy, \lambda u.see(u, v))).$$

2.2 Constraints on scope taking

In the above account, all quantifiers were implicitly assumed to have equal 'raising' or scope-taking potential. Under such an assumption,[2] a sentence with n quantifiers will in principle have $n!$ different scope possibilities. Fortunately, the linear logic glue language, in addition to its role in assembling scope readings, can also be harnessed to *constrain* the combinatorics of scope readings. An example of this is found in [DLPSar], where Dalrymple *et al.* showed how properties of the glue language prevent certain scopings which are ill-formed due to their interaction with bound anaphora, e.g. forcing *every* to take wide scope in the sentence *Every candidate appointed a friend of his*, thus producing the single legitimate scope reading.

The implementation of negative polarity licensing constraints given here can be considered another example of using the glue language to constrain the combinatorics of scope readings at the syntax-semantics interface. We will see, for example, how this account constrains the potential ambiguities of a sentence like *Nobody thinks few doctors treated anyone* and blocks the semantic interpretation of infelicitous examples like $*$*Kim ever left*.

3 Review of negative polarity licensing

Ladusaw [Lad79] established what is now a well-known generalization in semantics, namely that negative polarity lexical items (NPI's) are licensed within the scope of downward-entailing operators. Examples of NPI's are found across several categories, including adverbs (*ever*, *yet*, *anymore*, *at all*, *even*), determiners (*any*), and certain idiomatic expressions (*give a damn*, *lift a finger*). As illustrated in sentence pairs (7)-(9), NPI's cannot appear felicitously in a sentence without being licensed by some sort of negative or decreasing context.

[2] A dubious, if pedagogically convenient assumption—see [Sza97] for counterexamples.

(7) No boy has ever been to Moscow.
$*$A boy has ever been to Moscow.

(8) Neither Sue nor Bill lifted a finger to help.
$*$Sue lifted a finger to help.

(9) Kim rarely had any fun.
$*$Kim had any fun.

NPI licensers include certain adverbs (*rarely*, *hardly*) and certain affective predicates (*be amazed that*, *too* [ADJ] *to*...). NPI's are also licensed by so-called 'monotone decreasing' quantifiers, which are built from determiners like *no*, *not all*, *at most four*, *less than half*, and *neither ... nor ....* A determiner Q is monotone decreasing if it validates inferences from bigger toward smaller sets—i.e., iff $Q(A, B) \models Q(A, B \cap C)$. Examples of such decreasing entailments are given in (10) and (11).

(10) no boy came home $\models$ no boy came home late

(11) neither Sue nor Bill sang $\models$ neither Sue nor Bill sang *Fidelio*

The negative or decreasing context created by a licenser can apparently license more than one NPI within its scope. For example, we can add an additional NPI to each example in (7)-(9), producing the felicitous sentences in (12)-(14).

(12) No boy has ever been to Moscow even once.

(13) Neither Sue nor Bill ever lifted a finger to help.

(14) Kim rarely had any fun at all.

Although NPI's require a negative or decreasing licenser, the converse is not the case. That is, licensers are just as free as non-licensers to occur in sentences which contain no NPI's. This is demonstrated by the grammaticality of (15)-(17).

(15) No boy went to Moscow.
A boy went to Moscow.

(16) Neither Sue nor Bill helped us.
Sue helped us.

(17) Kim rarely had fun.
Kim had fun.

4 A glue language treatment of NPI licensing

This section reviews the account of NPI licensing using glue language which was proposed in [Fry97]. Readers interested in details of the implementation using linear logic proof nets should consult [Fry97]; in this section, the linguistic aspects will be emphasized and new issues will be explored.

4.1 Meaning constructors for NPI's

There is a resource-based interpretation of the NPI phenomena described in the previous section. The negative or decreasing licensing operator must make available a resource, call it ℓ, which will license the NPI's, if any, within its scope. If no such resource is made available the NPI's are unlicensed and the sentence must be rejected.

We first show how NPI's can be made to require the ℓ resource. Generally, the way one implements such a requirement in linear logic is to put the required resource on the left side of the implication operator $\multimap$. This is precisely our approach. However, since the NPI is just 'borrowing' the license (as we saw, more than one NPI may be licensed, as in *No one ever saw anyone*), we also add the resource to the right hand side of the implication to make it available for other NPI's if necessary. That is, for a meaning constructor of the form $A \multimap B$, we construct a corresponding NPI meaning constructor of the form

$$(A \otimes \ell) \multimap (B \otimes \ell).$$

For example, the meaning constructor proposed in [DLS93] for the sentential modifier *obviously* is

$$\textbf{obviously:} \quad \forall P.\ f_\sigma \rightsquigarrow_t P \multimap f_\sigma \rightsquigarrow_t \mathit{obviously}(P)$$

Under this analysis of sentential modification, NPI adverbs such as *yet* or *ever* would take the same form, but with the licensing apparatus added:

$$\textbf{ever:} \quad \forall P.\ (f_\sigma \rightsquigarrow_t P \otimes \ell) \multimap (f_\sigma \rightsquigarrow_t \mathit{ever}(P) \otimes \ell).$$

This technique can be readily applied to the other categories of NPI as well. In the case of determiners, we assign them meaning constructors from the same template as *every* ((1), repeated here as (18)).

$$(18)\quad \textbf{every:} \quad \begin{array}{l} \forall H, R, S.\ (\forall x.\ (g_\sigma\ \text{VAR}) \rightsquigarrow_e x \multimap (g_\sigma\ \text{RESTR}) \rightsquigarrow_t R(x)) \\ \quad \otimes (\forall y.\ g_\sigma \rightsquigarrow_e y \multimap H \rightsquigarrow_t S(y)) \\ \quad \multimap H \rightsquigarrow_t \mathit{every}(R, S) \end{array}$$

For the NPI determiner *any*[3] the licensing apparatus is added to produce the meaning constructor in (19).[4]

$$(19)\quad \textbf{any:} \quad \begin{array}{l} \forall H, R, S.\ (\forall x.\ (g_\sigma\ \text{VAR}) \rightsquigarrow_e x \multimap (g_\sigma\ \text{RESTR}) \rightsquigarrow_t R(x)) \\ \quad \otimes (\forall y.\ g_\sigma \rightsquigarrow_e y \multimap H \rightsquigarrow_t S(y) \otimes \ell) \\ \quad \multimap (H \rightsquigarrow_t \mathit{any}(R, S) \otimes \ell) \end{array}$$

The only function of the $\ell \multimap \ell$ pattern inside an NPI is to consume the resource ℓ and then produce it again. However, for this to happen, the resource ℓ will have to be generated by some licenser whose scope includes the NPI. Thus an ill-formed sentence containing an unlicensed NPI such as (20), for example, receives no semantic interpretation.

[3] *Any* also has another, so-called 'free choice' interpretation (as in e.g. *Any pen will do*) [Lad79, KL93], which we ignore here.

[4] Note that we have kept the left-side licensing resource ℓ *inside* the scope constructor of *any*, attached to $S(y)$. Although this particular placement is not necessary for ensuring that *any* gets licensed, it is necessary for ensuring that other NPI's can be licensed after *any*. Such a case is found in e.g. *No girl gave any present to any boy.*

(20) ∗Kim ever left.

$$f{:}\left[\begin{array}{ll} \text{PRED} & \text{`LEAVE'} \\ \text{SUBJ} & g{:}\left[\text{PRED} \quad \text{`KIM'}\right] \\ \text{MODS} & \left\{\left[\text{PRED} \quad \text{`EVER'}\right]\right\} \end{array}\right]$$

The relevant meaning constructors for (20) being

$$\begin{array}{ll} \textbf{Kim:} & g_\sigma \rightsquigarrow_e Kim \\ \textbf{ever:} & \forall P.\ (f_\sigma \rightsquigarrow_t P \otimes \ell) \multimap (f_\sigma \rightsquigarrow_t ever(P) \otimes \ell) \\ \textbf{left:} & \forall Y.\ g_\sigma \rightsquigarrow_e Y \multimap f_\sigma \rightsquigarrow_t leave(Y). \end{array}$$

In this case, no outside ℓ resource is made available by a decreasing operator (licenser) which would allow it to consume the NPI **ever**. As a result, the extraneous, unconsumed ℓ material in **ever** guarantees that no proof can be generated, since the resource-conscious property of linear logic requires that all premises be used up in the proof.

4.2 Meaning constructors for NPI licensers

It is clear from the proposal so far that lexical items which license NPI's must make available a ℓ resource within their scope which can be consumed by the NPI. However, that is not enough; as demonstrated in (15)-(17), a licenser can still occur if no NPI is present. The resource accounting of linear logic requires that we 'clean up' by consuming any excess ℓ resources in order for the meaning deduction to go through. Fortunately, this problem can be solved within the licenser's meaning constructor itself.

The general technique is as follows. For a lexical category whose meaning constructor is of the form $A \multimap B$, we assign to the NPI licensers of that category the meaning constructor

$$(\ell \multimap (A \otimes \ell)) \multimap B.$$

Because it is embedded inside another implication, the inner implication here functions as a hypothetical. Using this meaning constructor for the licenser, all of the NPI licensing occurs within the hypothetical (left) side of the outermost implication. Because the ℓ resource is made available to the NPI within the hypothetical, it is guaranteed that the NPI is assembled within, and therefore falls under, the scope A of the licenser.

Furthermore, the formula is 'self cleaning', in that the ℓ resource, even if not used by an NPI, does not survive the hypothetical and so cannot affect the meaning of the licenser in some other way. The licensing constructor $(\ell \multimap (A \otimes \ell)) \multimap B$ can derive the same meanings as the nonlicensing version $A \multimap B$, since $(\ell \multimap (A \otimes \ell)) \multimap B \vdash A \multimap B$ trivially holds (see [Fry97]).

This self-cleaning property means that a licensing resource ℓ is exactly that—a license. Within its scope, it is available to be used once, several times, or not at all, as required.

A simple example is provided by the NPI-licensing adverb *rarely*. We modify our sentential adverb template to create a meaning constructor for *rarely* which licenses an NPI within the sentence it modifies.

$$\textbf{rarely:} \quad \forall P.\ (\ell \multimap (f_\sigma \rightsquigarrow_t P \otimes \ell)) \multimap f_\sigma \rightsquigarrow_t rarely(P)$$

The case of the monotonic decreasing determiners such as *no* and *few* is slightly more complex but follows the same pattern. A preliminary meaning constructor for *no* is given in (21).

(21) **no**: $\forall H, R, S.\ (\forall x.\ (g_\sigma\ \text{VAR}) \rightsquigarrow_e x \multimap (g_\sigma\ \text{RESTR}) \rightsquigarrow_t R(x))$
$\otimes (\forall y.\ (g_\sigma \rightsquigarrow_e y \otimes \ell) \multimap (H \rightsquigarrow_t S(y) \otimes \ell))$
$\multimap H \rightsquigarrow_t no(R, S)$

We will demonstrate (21) in two sentences, one without and another with an NPI. The first example is (22).

(22) No boy left.

$$f{:}\begin{bmatrix} \text{PRED} & \text{`LEAVE'} \\ \text{SUBJ} & g{:}\begin{bmatrix} \text{SPEC} & \text{`NO'} \\ \text{PRED} & \text{`BOY'} \end{bmatrix} \end{bmatrix}$$

The premises are:

$$\begin{array}{ll} \textbf{no}: & \forall H,R,S.\ (\forall x.\ (g_\sigma\ \text{VAR})\rightsquigarrow_e x \multimap (g_\sigma\ \text{RESTR})\rightsquigarrow_t R(x)) \\ & \quad \otimes (\forall y.\ (g_\sigma\rightsquigarrow_e y \otimes \ell) \multimap (H\rightsquigarrow_t S(y) \otimes \ell)) \\ & \quad \multimap H\rightsquigarrow_t no(R,S) \\ \textbf{boy}: & \forall X.\ (g_\sigma\ \text{VAR})\rightsquigarrow_e X \multimap (g_\sigma\ \text{RESTR})\rightsquigarrow_t boy(X) \\ \textbf{left}: & \forall Y.\ g_\sigma\rightsquigarrow_e Y \multimap f_\sigma\rightsquigarrow_t leave(Y) \end{array}$$

By the 'self cleaning' property $(\ell \multimap (A \otimes \ell)) \multimap B \vdash A \multimap B$, these premises reduce to just

$$\begin{array}{ll} \textbf{no}: & \forall H,R,S.\ (\forall x.\ (g_\sigma\ \text{VAR})\rightsquigarrow_e x \multimap (g_\sigma\ \text{RESTR})\rightsquigarrow_t R(x)) \\ & \quad \otimes (\forall y.\ g_\sigma\rightsquigarrow_e y \multimap H\rightsquigarrow_t S(y)) \\ & \quad \multimap H\rightsquigarrow_t no(R,S) \\ \textbf{boy}: & \forall X.\ (g_\sigma\ \text{VAR})\rightsquigarrow_e X \multimap (g_\sigma\ \text{RESTR})\rightsquigarrow_t boy(X) \\ \textbf{left}: & \forall Y.\ g_\sigma\rightsquigarrow_e Y \multimap f_\sigma\rightsquigarrow_t leave(Y) \end{array}$$

At this point the proof proceeds exactly as per example (4) in Section 2.

A second example, this time of a sentence containing an NPI, is given in (23) (repeated here from [Fry97]), in which the determiners *no* and *any* play the role of licenser and NPI, respectively.

(23) No boy saw any girl.

$$f{:}\begin{bmatrix} \text{PRED} & \text{`SEE'} \\ \text{SUBJ} & g{:}\begin{bmatrix} \text{SPEC} & \text{`NO'} \\ \text{PRED} & \text{`BOY'} \end{bmatrix} \\ \text{OBJ} & h{:}\begin{bmatrix} \text{SPEC} & \text{`ANY'} \\ \text{PRED} & \text{`GIRL'} \end{bmatrix} \end{bmatrix}$$

Normally, a sentence with two quantifiers would generate two different scope readings—in this case, (24) and (25).

(24) $f_\sigma\rightsquigarrow_t no(boy, \lambda x.any(girl, \lambda y.see(x,y)))$

(25) $f_\sigma\rightsquigarrow_t any(girl, \lambda y.no(boy, \lambda x.see(x,y)))$

To save space we will pre-compose the meaning constructors of the determiners (e.g. **no**) and nouns (e.g. **boy**) into single generalized quantifiers (e.g. **no-boy**). That is, we instantiate the restrictor of each quantifier so that only the scope needs to be solved. The premises, then, are:

(26)
$$\begin{array}{ll} \textbf{no-boy}: & \forall H,S.\ (\forall x.\ (g_\sigma\rightsquigarrow_e x \otimes \ell) \multimap (H\rightsquigarrow_t S(x) \otimes \ell)) \multimap H\rightsquigarrow_t no(boy,S) \\ \textbf{saw}: & \forall X,Y.\ (g_\sigma\rightsquigarrow_e X \otimes h_\sigma\rightsquigarrow_e Y) \multimap f_\sigma\rightsquigarrow_t see(X,Y) \\ \textbf{any-girl}: & \forall I,T.\ (\forall y.\ h_\sigma\rightsquigarrow_e y \multimap I\rightsquigarrow_t T(y) \otimes \ell) \multimap (I\rightsquigarrow_t any(girl,T) \otimes \ell) \end{array}$$

Recall that Ladusaw's [Lad79] generalization is that negative polarity items are licensed *within the scope* of their licensers. In fact, Ladusaw [Lad79, p. 96-101] gives several arguments why *any* cannot take wide scope in such a case. The premises in (26) implement exactly this constraint, since **no-boy** makes available the license ℓ exclusively within its scope. Since a license is *required* by the NPI **any-girl**, the only alternative is for **no-boy** to take wide scope over **any-girl**. The only available proof then (given in detail in [Fry97]) corresponds to reading (24), and reading (25) is blocked, as desired.

4.3 Leftward licensing

To keep things simple, we have restricted discussion of 'decreasingness' to the so-called 'right-decreasing' entailment pattern. However, leftward polarity licensing is captured just as straightforwardly using the technique just presented. A determiner Q is monotonic *left decreasing* if it validates inferences of the form $Q(A, B) \models Q(A \cap C, B)$. In other words, the restriction, rather than the scope, is the decreasing set. Examples of such left-decreasing entailments are given in (27) and (28).

(27) every boy left $\models$ every boy I admire left

(28) no boy came home $\models$ no good boy came home

Obviously the determiners *every* and *no* are both left-decreasing, although *every* is not right-decreasing. The complete meaning constructors, then, for *every* and *no* can be given as follows.

$$\begin{array}{ll}\textbf{every}: & \forall H, R, S.\ (\forall x.\ ((g_\sigma\ \text{VAR}) \rightsquigarrow_e x \otimes \ell) \multimap ((g_\sigma\ \text{RESTR}) \rightsquigarrow_t R(x) \otimes \ell)) \\ & \quad \otimes (\forall y.\ g_\sigma \rightsquigarrow_e y \multimap H \rightsquigarrow_t S(y)) \\ & \quad \multimap\ H \rightsquigarrow_t \mathit{every}(R, S)\end{array}$$

$$\begin{array}{ll}\textbf{no}: & \forall H, R, S.\ (\forall x.\ ((g_\sigma\ \text{VAR}) \rightsquigarrow_e x \otimes \ell) \multimap ((g_\sigma\ \text{RESTR}) \rightsquigarrow_t R(x) \otimes \ell)) \\ & \quad \otimes (\forall y.\ (g_\sigma \rightsquigarrow_e y \otimes \ell) \multimap (H \rightsquigarrow_t S(y) \otimes \ell)) \\ & \quad \multimap\ H \rightsquigarrow_t \mathit{no}(R, S)\end{array}$$

In the case of *every*, the ℓ resource is only made available within the embedded implication that comprises the restriction, so only NPI's within the restriction will be licensed. *No*, on the other hand, licenses within the restrictor, within the scope, or within both, if necessary. Thus the meaning constructors above account for all of the data in (29) and (30) below.

(29) Every boy had fun.
Every boy who ever had the measles had fun.
*Every boy had any fun.
*Every boy who ever had the measles had any fun.

(30) No boy had fun.
No boy who ever had the measles had fun.
No boy had any fun.
No boy who ever had the measles had any fun.

4.4 Weak vs. strong licensing

The analysis presented here and in [Fry97] is too rough-grained in that it overgenerates licensing possibilities when multiple licensers are present. While judgments about sentences with multiple licensers or multiple negations are notoriously slippery, there seem to be several clear cases where the present analysis permits spurious licensings. For example, in (31), the negative morpheme *–n't* licenses the NPI *lift a finger*.

(31) John didn't$_\ell$ lift a finger$_\ell$ to help.

Clause (31) could then conceivably be embedded into the larger sentence (32):

(32) I'd be surprised if John didn't$_\ell$ lift a finger$_\ell$ to help.

However, the proposed analysis would also permit (33), an alternative derivation of (32) where the NPI is licensed by the wider-scoping licenser.

(33) I'd be surprised$_\ell$ if John didn't lift a finger$_\ell$ to help.

The licensing in (33) is made possible by the self-cleaning property of the licenser *–n't*, i.e. the same property that permits a sentence with no NPI's like *John didn't help*. However, it is clear that only (32) represents a legitimate licensing in this case.

This state of affairs might lead one to propose a constraint limiting licensing to the licenser with narrowest possible scope. Indeed, explicit negations (i.e. "n-words") seem to behave in this way.[5] However, there seem to be exceptions among less overt licensers. For example, (34) has two licensers, *–n't* and *refuse*.

(34) Doctors shouldn't be allowed to refuse to treat anybody.

In this case, both the narrow and wide licensing possibilities seem to be available. In reading (35), the negation *n't* licenses the NPI *anybody*.

(35) $\neg_\ell$ allowed: doctor(x), anybody(y)$_\ell$: x refuses to treat y.

The second, less plausible reading in (36) is actually the narrower one where *refuse* scopes over and licenses *anybody*.

(36) $\neg$ allowed: doctor(x): x refuses$_\ell$ to treat anybody$_\ell$.

Assuming then that explicitly negative "n-word" licensers like *no* always take the narrowest licensing scope, we can characterize such licensers as *strong*, in contrast to *weak* licensers such as *refuse* and *few*.[6] The weak licensers, in other words, have the option of passing up the opportunity to license an NPI, allowing it to be licensed by a strong licenser in a higher clause; the strong licensers, on the other hand, have no such option.

As a consequence, sentence (37) has only one reading, given in (38):

(37) Few people think no doctor treated anyone.

[5] I'm grateful to Bill Ladusaw (p.c.) for the observations in this paragraph.

[6] A somewhat different taxonomy of NPI's according to strength is proposed by Zwarts [Zwaar].

(38) few people think: no$_\ell$ doctors(x), anybody(y)$_\ell$, x treats y

In sentence (39), on the other hand, the weak licenser *few* can optionally pass up the licensing of *anybody*, producing the two readings (40) and (41).

(39) Nobody thinks few doctors treated anyone.

(40) nobody thinks: few$_\ell$ doctors(x), anybody(y)$_\ell$, x treats y

(41) nobody$_\ell$ thinks: anybody(y)$_\ell$, few doctors(x), x treats y

John Lamping (p.c.) suggests a modification to the NPI licensing scheme which implements the weak/strong distinction described above. Lamping's idea is to (1) index the ℓ resources with Prolog-style variables, and (2) introduce new ℓ resources at the start and end of each quantifier, in order to pass through the licensing context. For example, the scope distinctions observed above for sentences (37) and (39) can be accounted for using the following meaning constructors for the relevant quantifiers:

nobody:

$$\begin{array}{l} \ell(\_,\_) \multimap \\ \quad ((g_\sigma \rightsquigarrow_e x \otimes \ell(y, Gen)) \multimap (H \rightsquigarrow_t S(x) \otimes \ell(y, Z))) \\ \qquad \multimap H \rightsquigarrow_t no(person, S) \otimes \ell(\_, Z) \end{array}$$

few-doctors:

$$\begin{array}{l} \ell(\_, W) \multimap \\ \quad ((g_\sigma \rightsquigarrow_e x \otimes \ell(y, W)) \multimap (H \rightsquigarrow_t S(x) \otimes \ell(y, Y))) \\ \qquad \multimap H \rightsquigarrow_t few(doctor, S) \otimes \ell(\_, Y) \end{array}$$

anyone:

$$\begin{array}{l} \ell(y, X) \multimap \\ \quad ((g_\sigma \rightsquigarrow_e x \otimes \ell(y, X)) \multimap (H \rightsquigarrow_t S(x) \otimes \ell(y, X))) \\ \qquad \multimap H \rightsquigarrow_t any(person, S) \otimes \ell(y, X) \end{array}$$

The above scheme treats the license resource as a two-place predicate $\ell(T, V)$, where the toggle T is set to y if the current context is a monotone decreasing one and n otherwise, and V is a Prolog-style variable used to keep track of which operator is acting as the licenser. The underscore _ is the standard Prolog anonymous variable, and the symbol *Gen* is a unique ground term (gensym). Under Lamping's scheme, NPI's like *anyone* require a y (decreasing) licensing context, and they make sure that the same token X passed in by the licenser is passed out again unchanged. Weak licensers like *few* pass a token W to their scope, but the token can optionally be changed (i.e. to Y) within the scope, for example by a strong licenser 'taking over' the licensing from the weak one. Strong licensers like 'nobody' always pass a unique ground token (gensym) to their scope, which prevents them from passing along the token of another licenser, ensuring that strong licensers can never pass up their licensing to some outer licenser.

The above extension of the linear logic NPI licensing treatment is admittedly a bit cumbersome; however, it can be taken as evidence that certain subtleties observed in negative polarity phenomena are not beyond formalization and implementation within this framework.

5 Loose ends and future work

5.1 The lexicon

In this paper the polarity licensing apparatus was simply added to the appropriate individual entries in the lexicon. This bears closer scrutiny. Obviously, a lexicon composed solely of individual words would be insufficient—an idiomatic expression like *lift a finger* requires NPI marking, even though none of its words in isolation does. And although both *no* and *fewer than four* are downward-entailing, and therefore licensers, the combination *no fewer than four* is increasing and therefore should be free of the licensing apparatus. However, it seems clear that such issues would pose no problem for a lexicon of reasonable sophistication.

A more substantial question is: How does the licensing apparatus enter the lexicon in the first place? In this presentation, the NPI's and licensers were known *a priori* and marked appropriately in the lexicon. The licensing was then accomplished within the glue language (linear logic). Focusing on the glue language, we made minimal assumptions about the nature of the actual meaning language that is the object of the assembly process. However, if the meaning language could be used to express inferential and other relevant semantic properties of the lexical items, perhaps these properties could then be harnessed within the lexicon to *derive* the appropriate licensing properties. The question of whether this could be done in an elegant fashion we leave for future research.[7] The concern of this paper is the question of *how* licensing works at the syntax-semantics interface, rather than *why* particular items act as licensers or NPI's, or whether these items should be identified by lexical, semantic or syntactic properties. The technique presented in this paper is independent of, and in fact compatible with a variety of, theories of the 'why' of polarity licensing. In any case, the approach of deriving licensing properties directly from the inferential properties of lexical items would seem to be the most promising and interesting approach to take in this regard.

6 Conclusion

We have elaborated on and extended slightly the 'glue language' approach to semantics of Dalrymple *et al.* We then presented a glue language treatment of negative polarity licensing which ensures that NPI's are licensed within the semantic scope of their licensers in both left- and right-decreasing contexts.

The glue language treatment of licensing presented here has two notable properties. First, it requires no new global rules or features; we introduced only minor modifications of the meaning constructors of the relevant items within the lexicon, without introducing new ambiguity. The second feature relates to the following observations: The status of a lexical item as an NPI or licenser depends crucially on its meaning; i.e. on semantic rather than syntactic properties. On the other hand, the requirement that NPI's be licensed in order to appear felicitously in a sentence is clearly a fact about syntax. So the domain of NPI licensing is really the interface between syntax and semantics, where meanings are composed under syntactic guidance. The treatment presented here operates precisely at that level, because it is implemented entirely within the glue language of the syntax-semantics interface.

References

[DLPS94] Mary Dalrymple, John Lamping, Fernando C. N. Pereira, and Vijay Saraswat. A deduc-

[7]This question is posed by Ed Stabler [Sta97] for glue language semantics in general.

tive account of quantification in LFG. In Makoto Kanazawa, Christopher J. Piñón, and Henriette de Swart, editors, *Quantifiers, Deduction, and Context.* CSLI Publications, Stanford, CA, 1994.

[DLPSar] Mary Dalrymple, John Lamping, Fernando C. N. Pereira, and Vijay Saraswat. Quantifiers, anaphora, and intensionality. *Journal of Logic, Language and Information*, to appear.

[DLS93] Mary Dalrymple, John Lamping, and Vijay Saraswat. LFG semantics via constraints. In *Proceedings of the 6th Meeting of the European Association for Computational Linguistics*, University of Utrecht, April 1993.

[Fry97] John Fry. Negative polarity licensing at the syntax-semantics interface. In *Proceedings of the Thirty-Fifth Annual Meeting of the ACL and Eighth Conference of the EACL*, pages 144–150, Madrid, July 1997. Association for Computational Linguistics.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50, 1987.

[KL93] Nirit Kadmon and Fred Landman. Any. *Linguistics and Philosophy 16*, pages 353–422, 1993.

[Lad79] William A. Ladusaw. *Polarity Sensitivity as Inherent Scope Relations.* PhD thesis, University of Texas, Austin, 1979. Reprinted in Jorge Hankamer, editor, *Outstanding Dissertations in Linguistics*. Garland, 1980.

[Mil90] Dale A. Miller. A logic programming language with lambda abstraction, function variables and simple unification. In Peter Schroeder-Heister, editor, *Extensions of Logic Programming*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 1990.

[Sta97] Edward P. Stabler. Computing quantifier scope. In Anna Szabolcsi, editor, *Ways of Scope Taking.* Kluwer, Dordrecht, 1997.

[Sza97] Anna Szabolcsi, editor. *Ways of Scope Taking.* Kluwer, Dordrecht, 1997.

[Zwaar] Franz Zwarts. Three kinds of negative polarity. In F. Hamm and E. Hinrichs, editors, *Plural Quantification.* Reidel, Dordrecht, to appear.

`fry@csli.stanford.edu`
Linguistics Department
Stanford University
Stanford, CA 94305-2150 USA