Inverse Linking via Function Composition

Gregory M. Kobele

University of Chicago

Abstract

The phenomenon of Inverse Linking has proven challenging for theories of the syntax-semantics interface; a noun phrase within another behaves with respect to binding as though it were structurally independent. In this paper I show that, using an LF-movement style approach to the syntax-semantics interface, we can derive all and only the appropriate meanings for such constructions using no semantic operations other than function application and composition. The solution relies neither on a proliferation of lexical ambiguity nor on abandoning the idea that pronouns denote variables, but rather on a straightforward (and standard) reification of assignment functions, which allows us to define abstraction operators within our models.

1 Introduction

Inverse linking (see May and Bale (2005) for an historical overview) refers to the phenomenon of a quantificational noun phrase (QNP) embedded as the argument of a prepositional phrase attached to another QNP taking semantic scope over that noun phrase, as in reading b of example 1, the gross surface structure of which is as sketched in figure 1.

(1) At least two senators on every committee voted against the bill.

   a. At least two senators who are on every committee voted against the bill.

   b. For every committee, there are at least two senators on that committee who voted against the bill.

The challenge the simple existence of inversely linked readings poses for a theory of grammar is to account for how the embedded PP-internal QNP

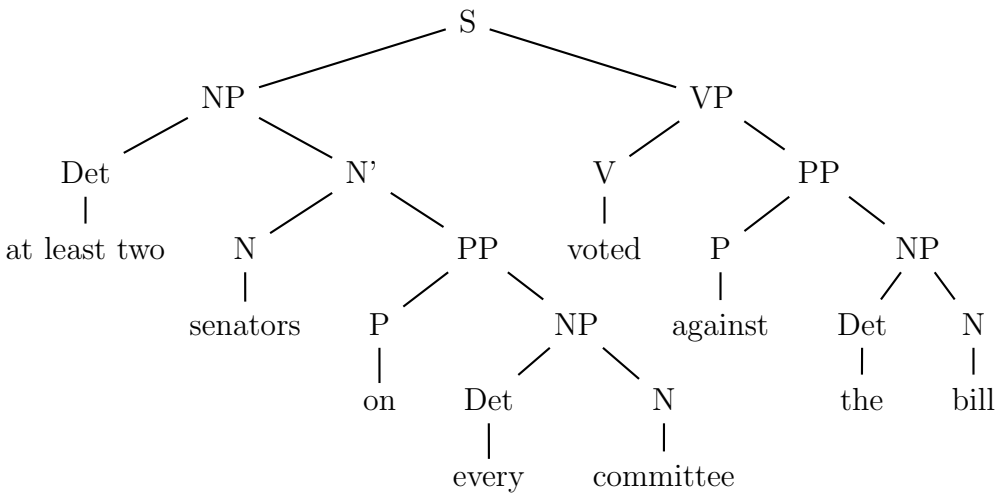

Figure 1: The gross surface structure of example 1

is able to semantically outscope the noun phrase which contains it. There are two major classes of analyses.[1] The first option (May, 1977; Sauerland, 2005), sketched in figure 2, is to interpret the embedded QNP as taking sentential scope (i.e. as scoping entirely outside of the QNP which contains it). The second option (May, 1985; Heim and Kratzer, 1998), as in figure 3, is to interpret the embedded QNP as forming a complex quantificational element with the QNP which contains it. There are two additional empirical constraints on a theory of inverse linking. First is the fact, called *Larson's generalization* in May and Bale (2005), that quantificational noun phrases external to the containing QNP cannot intervene semantically between the embedded and containing QNPs, as shown in example 2.

(2) Two politicians spy on someone from every city.

If the sentence in 2 had a reading in which *every city* took widest scope, and *someone* narrowest, with *two politicians* in between, then it should be true of a situation in which each city has different politicians ($\forall < 2$), but where no two politicians spy on the same individual ($2 < \exists$). However, it doesn't seem to be.

[1]Another alternative is offered by continuations (see, for example,Barker (2002)). Syntactic movement has at the very least intuitive relations to continuations (of the bounded variety (of which there are many)), but the analysis of quantification presented in Barker (2002) is not a natural fit with the syntactic analyses in the minimalist program, even if and when this latter is reanalyzed in terms of (string-)continuations. The same is true of analyses written in the context of syntactic frameworks, such as categorial grammar, in which type changing operations are more natural than in the minimalist framework used here.

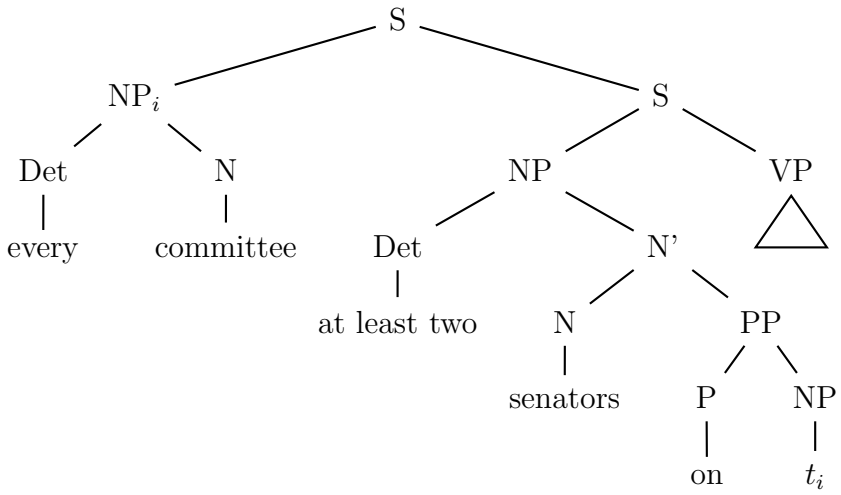

Figure 2: Explaining inverse linking: Sentential scope

The second empirical constraint on a theory of inverse linking comes from the fact that pronouns external to the containing QNP can be bound by the embedded QNP on the inversely linked reading, as in example 3.

(3) Some man from every city secretly despises it.

Example 3 can be true of a situation in which the city despised varies across individuals, i.e. Garrison from Lake Wobegon secretly despises Lake Wobegon, Garth from Waco secretly despises Waco, Gary from Wasilla secretly despises Wasilla, etc.

The sentential scope approach to inverse linking (as in figure 2) immediately accounts for the ability of embedded QNPs to bind variables external to their containing QNP, but requires additional stipulations to correctly rule out cases of scope-splitting. The complex quantifier approach to inverse linking (as in figure 3) on the other hand, while seeming as though it may provide a simple explanation of the scope-splitting prohibition, requires that some method of interpreting 'complex quantifiers' be specified before it is able to make predictions. What would seem to be needed is the following, where $\mathcal{Q}$ is the embedded QNP denotation, and $D(N(x))$ is the containing QNP denotation, where x is the interpretation of the trace of the embedded QNP.[2]

$$\lambda A_{et}.\mathcal{Q}(\lambda x_e.D(N(x))(A))$$

[2]Thus, the denotation of the NP in figure 3 should be the following.

$$\lambda A_{et}.\mathbf{every}(\mathbf{committee})(\lambda x_e.\mathbf{two}(\mathbf{senator} \wedge \mathbf{on}(x))(A))$$

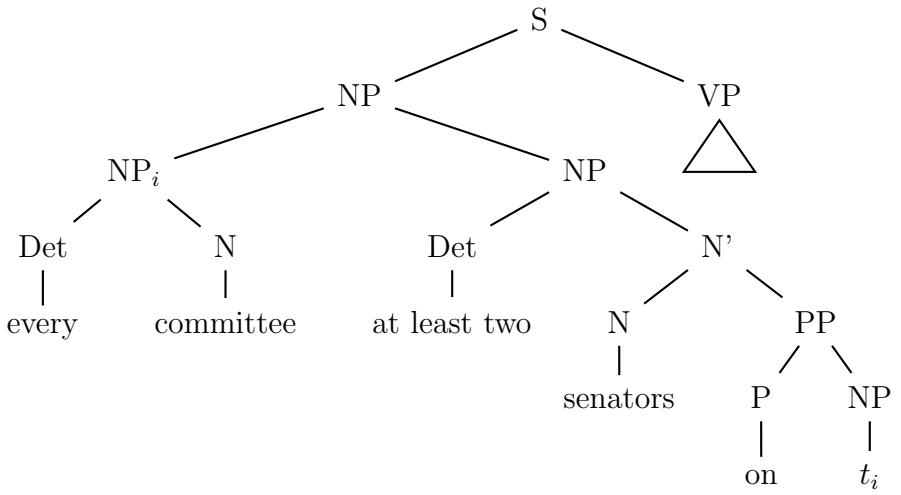

Figure 3: Explaining inverse linking: Complex quantifier formation

As can be verified by inspecting the term above, this denotation correctly rules out the scope splitting cases discussed above (see example 2). One problem with fleshing the complex quantifier approach to inverse linking out in this way is that it seems to require construction specific semantic machinery for its description: how does the meaning of the embedded QNP, $\mathcal{Q}$ combine with the meaning of the containing QNP, $D(N(x))$, so as to result in the (or something like the) meaning representation above? A more fundamental problem is that, although it allows for the embedded quantifier (denoting $\mathcal{Q}$) to take scope over the rest of the sentence (replacing the variable A), which we want in order to account for the binding facts, the representation here does *not* allow pronouns in the sentence to be bound by $\mathcal{Q}$, as the λ-calculus prohibits 'variable capture.'[3] This problem can be seen as undermining the position that pronouns should be interpreted as variables, and thus as an argument for an E-type treatment of pronouns (see Büring (2004) and §4.2).

In contrast to previous accounts of this phenomenon, which were able to maintain the assumption that pronoun binding is variable binding only by introducing purely syntactic distinctions into the semantics (see Larson (1985)

[3]Λ-terms provide an overly fine grained representation of functions, in the sense that infinitely many distinct terms can represent the same function. An example: $\lambda x.x$ is a different term than $\lambda y.y$, although they stand for the very same function. β-conversion is defined so as to make sure that, if α and α' are terms denoting the same function, then $\alpha(\beta)$ and $\alpha'(\beta)$ also denote the same object. For example, $\lambda x.\lambda y.x$ and $\lambda x.\lambda z.x$ both denote the function $\mathbf{K}(a)(b) = a$, and so $(\lambda x.\lambda y.x)(y)$ must denote the same function as $(\lambda x.\lambda z.x)(y)$, namely the function denoted by $\lambda x.y$. For this reason, variable capture is ruled out, as it, being sensitive to accidental properties of our representations, would disrupt the intended equivalences between terms.

and §4.1), the solution I will propose in §3 is that the mechanism underlying complex quantifier formation is simply function composition. Mixing meta- and object-language for the moment, the complex quantifier *a friend of every senator* will have the following form, where $\circ$ denotes function composition.

$$(\mathbf{every}(\mathbf{senator}) \circ \lambda y) \circ \mathbf{some}(\lambda x.\mathbf{friend}(x, y))$$

Applied to a predicate denotation, A, this will yield the following.

$$\begin{aligned}((&\mathbf{every}(\mathbf{senator}) \circ \lambda y) \circ \mathbf{some}(\lambda x.\mathbf{friend}(x, y)))(A) \\ &= (\mathbf{every}(\mathbf{senator}) \circ \lambda y)(\mathbf{some}(\lambda x.\mathbf{friend}(x, y))(A)) \\ &= \mathbf{every}(\mathbf{senator})(\lambda y.\mathbf{some}(\lambda x.\mathbf{friend}(x, y))(A))\end{aligned}$$

This provides a simple solution to the ability of inversely scoped quantifiers to bind into the scope argument of their containing QNP; at the point at which the scope argument is introduced, the variable binding of the inversely scoping QNP has not yet been performed.

Strictly speaking, however, the formula written above is nonsense: λ-abstraction is not a function in our model, and thus it cannot be 'composed' with something that is. This, however, is due to the fact that it has become common to think of assignment functions as parameterizing model-theoretic interpretation, rather than, equivalently, as part of the models themselves. In §2 I review how to reify assignment functions. This allows me to find a 'lambda abstraction' function in our models, which I use in §3 to make legitimate the solution outlined above. In contrast to the solution put forth in Sternefeld (1997), which also makes use of cylindrified models, mine does not treat the binding relation in inverse-linking constructions dynamically. I review previous approaches in §4. Section 5 is the conclusion.

2 The Status of Assignment Functions

It is standard in presentations of the semantics of first-order logic to treat formulae as having interpretations in the models only with respect to assignment functions. In other words, there is no single monolithic interpretation function $[\![\cdot]\!]_{\mathcal{M}}$, but instead a family of interpretation functions $([\![\cdot]\!]^g_{\mathcal{M}})_{g\in G}$ parameterized by assignments. An equivalent perspective reifies the family G of assignment functions, allowing there to be a single interpretation function $[\![\cdot]\!]_{\mathcal{M}}$, the relation of which to the family $([\![\cdot]\!]^g_{\mathcal{M}})_{g\in G}$ can be schematized as per the below.

$$[\![\phi]\!]_{\mathcal{M}} := \{g : [\![\phi]\!]^g_{\mathcal{M}} = \texttt{true}\}$$

In other words, this alternative perspective takes the meaning of a sentence ϕ to be the set of all assignment functions with respect to which ϕ is true on the standard perspective. Despite being equivalent from the perspective of entailment, moving the assignment functions into the model gives access to first rate model theoretic objects which behave like lambda abstraction, but which can be composed with, among other things, generalized quantifier denotations to yield semantic objects of the kind alluded to at the end of the previous section.

Our models are built from the following objects:

- E is the set of *entities*
- T is the set of *propositions*
- G is the set of *contexts of use*

Here, I will take T to be the set $\{0, 1\}$ of truth values, and G to be the set $E^{\mathbb{N}}$ of assignment functions, where $\mathbb{N} = \{0, 1, 2, \ldots\}$ is the set of natural numbers and B^A the set of functions with domain A and codomain B.[4] Given $g, h \in G$, I write g_i for $g(i)$, and $g \approx_i h$ is true just in case if g and h differ, then only in the value they take at i (i.e. for any j, if $g_j \neq h_j$ then $j = i$). The notation $g^{[i:=a]}$ denotes the assignment h, such that $h_i = a$, and $g \approx_i h$. I write $x \in y$ as an abbreviation for $y(x) = 1$.

Natural language expressions denote in domains built from these sets in the following straightforward way.

- $D_e := E^G$
- $D_t := T^G$
- $D_{\alpha\beta} := {D_\beta}^{D_\alpha}$

My use of G is thus similar to Montague's type s, as used in Montague (1970). Viewed from this perspective, my type e is Montague's type se (individual concepts), and my type t Montague's type st (propositions). The difference between my use of G and Montague's s is that the 'denotation domain' of type s was for Montague not just the set of assignment functions (as it is here), but rather the set of pairs of assignment functions and possible worlds.[5]

[4] To deal with intensionality, we can instead take $T = \{0, 1\}^W$, for W an arbitrary set (of *possible worlds*) (Keenan and Faltz, 1985). Furthermore, we can take $G = E^{\mathbb{N}} \times E^{\mathbb{N}}$ to deal with dynamic binding (Groenendijk and Stokhof, 1991).

[5] In Montague (1973), the 'denotation domain' of type s was understood as pairs of worlds and times, with assignment functions relegated to parameters on the interpretation function.

A model $\mathcal{M} = \langle E, I \rangle$ provides an interpretation function I mapping elements of individual constants to elements of E, and i-ary relation symbols to i-ary relations over E. A trace $\mathsf{t_i}$ denotes a function $[\![\mathsf{t_i}]\!]_{\mathcal{M}} \in D_e$ from G to E, such that $[\![\mathsf{t_i}]\!]_{\mathcal{M}}(g) = g_i$. I will call such functions *variables*, and write them $\mathbf{x_i}$ (so in particular $[\![\mathsf{t_i}]\!]_{\mathcal{M}} = \mathbf{x_i}$). We extend $[\![\cdot]\!]_{\mathcal{M}}$ to our other constants in the following way:

- for c an individual constant, $[\![c]\!]_{\mathcal{M}} \in D_e$

$$[\![c]\!]_{\mathcal{M}}(g) = I(c)$$

- for r^i an i-place predicate constant, $[\![r^i]\!]_{\mathcal{M}} \in D_{e^i t}$

$$g \in [\![r^i]\!]_{\mathcal{M}}(a_1) \ldots (a_i) \text{ iff } \langle a_1(g), \ldots, a_i(g) \rangle \in I(r^i)$$

While there are a great many functions from, say, D_e to D_t, the objects we are interested in denote in a very restricted subset of this space. This is captured in the present system by means of the interpretation function I, which assigns 'lowered' meanings to lexical items (so $I(\mathsf{laugh}) \in 2^E$), in terms of which the denotations of these items are defined (so $[\![\mathsf{laugh}]\!] \in D_{et} = [G \to E] \to G \to T$).

The primary difference between this system and the one presented in Montague (1974) is lies in the denotation of predicates, which in Montague's system are functions with domain $E^i \to G \to T$, and which are here functions with domain $\left(E^G\right)^i \to G \to T$.

The payoff for setting things up in this manner is that our models contain functions which act on sentence denotations to yield predicate denotations in the same way the lambda-operator can be prefixed to a sentence to yield a one-place predicate. For every $i \in \mathbb{N}$, we have a function $\lambda_i \in D_{tet}$, which we define as follows.[6] For any $H \in D_t$, and $a \in D_e$,

$$\lambda_i(H)(a) = \{g : g^{[i:=a(g)]} \in H\}$$

With these definitions, it can be proven that, for example,

$$\lambda_1([\![\mathsf{laugh}]\!]_{\mathcal{M}}(\mathbf{x_1}))(a) = [\![\mathsf{laugh}]\!]_{\mathcal{M}}(a)$$

[6]It is worth remarking in how far this defined lambda-abstraction operator faithfully models the behaviour of the lambda in the lambda-calculus. It is clear that they are not identical: the lambda-operator in the lambda calculus 'combines' with a variable of any type σ, and an expression of any type τ to form an expression of type $(\sigma\tau)$. This formal situation recalls the empirical restriction on quantification in natural language argued for by Landman (2005), according to which variables are only of type e.

$$\begin{aligned}\lambda_1([\![\mathsf{laugh}]\!]_{\mathcal{M}}(\mathbf{x_1}))(a) &= \{g : g^{[1:=a(g)]} \in [\![\mathsf{laugh}]\!]_{\mathcal{M}}(\mathbf{x_1})\}\} \\ &= \{g : \mathbf{x_1}(g^{[1:=a(g)]}) \in I(\mathsf{laugh})\} \\ &= \{g : (g^{[1:=a(g)]})_1 \in I(\mathsf{laugh})\} \\ &= \{g : a(g) \in I(\mathsf{laugh})\} \\ [\![\mathsf{laugh}]\!]_{\mathcal{M}}(a) &= \{g : g \in [\![\mathsf{laugh}]\!]_{\mathcal{M}}(a)\}\end{aligned}$$

3 Inverse linking via Function composition

Here I will illustrate the complex quantifier approach to inverse linking, as described in section §1, above.

I will adopt a Heim and Kratzer (1998)-style perspective on semantic interpretation, in the sense that the structures which serve as input to the denotation function $[\![\cdot]\!]_{\mathcal{M}}$ will be syntactic structures to which quantifier raising transformations have already applied. Aside from putting assignments into the model as in the previous section, a major difference in the system here is that indices will be represented on the moved expression (so NP_i is an NP which binds a trace t_i). Such an object will be interpreted as usual, except that its denotation will be composed with a binder for x_i, λ_i:

$$[\![XP_i]\!]_{\mathcal{M}} := [\![XP]\!]_{\mathcal{M}} \circ \lambda_i$$

I adopt a type-driven approach to semantic combination (Klein and Sag, 1985), with basic combinatory operators not only forward and backward function application, but also forward and backward function composition.[7] For α a binary branching node with daughters β and γ, I write $[\![\alpha]\!]_{\mathcal{M}} = \textsc{combine}([\![\beta]\!]_{\mathcal{M}}, [\![\gamma]\!]_{\mathcal{M}})$, where COMBINE is a catch-all for whichever of the above named combinatory operators fits the bill. As mentioned above, a movement subscript contributes a semantic binder; so if the mother is α_i, and the daughters are β and γ, $[\![\alpha_i]\!]_{\mathcal{M}} = \textsc{combine}([\![\beta]\!]_{\mathcal{M}}, [\![\gamma]\!]_{\mathcal{M}}) \circ \lambda_i$.

3.1 Inverse Scope

We can calculate the meaning of the inversely linked reading in 4 below (with gross syntactic structure as in figure 4) in the following manner.

[7] Adding function composition as an available semantic mode of combination means that object quantifier phrases are semantically combinable with predicates in situ. Allowing object DPs to combine with their verbs via function composition is not what we want, as then [John [loves [every girl]]] would translate into (**every**(**girl**) ∘ **love**)(**john**), which means that every girl loves John! Thus, by allowing function composition as a legitimate mode of semantic combination, I am forced to the position that quantifier raising is not 'type driven', but is rather syntactic feature driven.

(4) Some relative of every lawyer despises him.

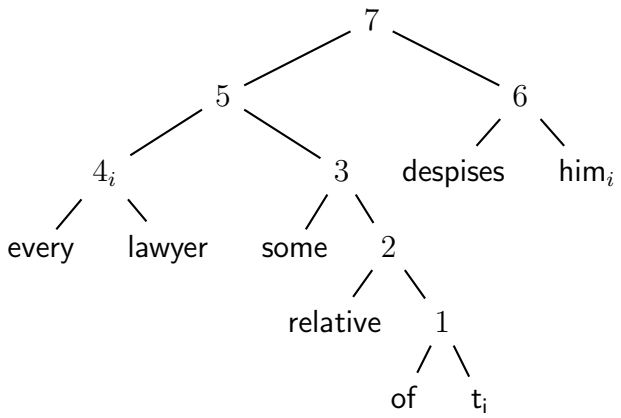

Figure 4: The LF-structure for the inverse scope reading of example 4

The lexical items (the leaves of figure 4) denote as described below.

$$[\![\mathsf{some}]\!]_{\mathcal{M}} = \mathbf{some} \in D_{(et)(et)t}$$
$$[\![\mathsf{every}]\!]_{\mathcal{M}} = \mathbf{every} \in D_{(et)(et)t}$$
$$[\![\mathsf{of}]\!]_{\mathcal{M}} = \mathbf{of} \in D_{ee}$$
$$[\![\mathsf{t_i}]\!]_{\mathcal{M}} = [\![\mathsf{him_i}]\!]_{\mathcal{M}} = \mathbf{x_i} \in D_e$$
$$[\![\mathsf{lawyer}]\!]_{\mathcal{M}} = \mathbf{lawyer} \in D_{et}$$
$$[\![\mathsf{relative}]\!]_{\mathcal{M}} = \mathbf{relative} \in D_{eet}$$
$$[\![\mathsf{despises}]\!]_{\mathcal{M}} = \mathbf{despise} \in D_{eet}$$

The denotations of the internal nodes (which are numbered in the figure), are given below.

1. $[\![1]\!]_{\mathcal{M}} = \textsc{combine}([\![\mathsf{of}]\!]_{\mathcal{M}}, [\![\mathsf{t_i}]\!]_{\mathcal{M}})$. As the type of $[\![\mathsf{of}]\!]_{\mathcal{M}}$ is ee, and the type of $[\![\mathsf{t_i}]\!]_{\mathcal{M}}$ is e, we apply the first argument to the second and obtain a value of type e:
$$[\![\mathsf{of}]\!]_{\mathcal{M}}([\![\mathsf{t_i}]\!]_{\mathcal{M}}) = \mathbf{of}(\mathbf{x_i})$$

2. $[\![2]\!]_{\mathcal{M}} = \textsc{combine}([\![\mathsf{relative}]\!]_{\mathcal{M}}, [\![1]\!]_{\mathcal{M}})$. As the type of $[\![\mathsf{relative}]\!]_{\mathcal{M}}$ is eet, and the type of $[\![1]\!]_{\mathcal{M}}$ is e, we again apply the first argument to the second and obtain a value of type et:
$$[\![\mathsf{relative}]\!]_{\mathcal{M}}([\![1]\!]_{\mathcal{M}}) = \mathbf{relative}(\mathbf{of}(\mathbf{x_i}))$$

3. $[\![3]\!]_{\mathcal{M}} = \textsc{combine}([\![\mathsf{some}]\!]_{\mathcal{M}}, [\![2]\!]_{\mathcal{M}})$. As $[\![\mathsf{some}]\!]_{\mathcal{M}}$ is of type $(et)(et)t$, and the type of $[\![2]\!]_{\mathcal{M}}$ is et, we apply once more the first argument to the second and obtain a value of type $(et)t$:
$$[\![\mathsf{some}]\!]_{\mathcal{M}}([\![2]\!]_{\mathcal{M}}) = \mathbf{some}(\mathbf{relative}(\mathbf{of}(\mathbf{x_i})))$$

4. $[\![4_i]\!]_{\mathcal{M}} = \textsc{combine}([\![\mathsf{every}]\!]_{\mathcal{M}}, [\![\mathsf{lawyer}]\!]_{\mathcal{M}}) \circ \lambda_i$. As the type of $[\![\mathsf{every}]\!]_{\mathcal{M}}$ is $(et)(et)t$, and the type of $[\![\mathsf{lawyer}]\!]_{\mathcal{M}}$ is et, we apply yet again the first argument to the second and obtain a value of type $(et)t$, which composes with λ_i of type tet to yield a value of type tt:

$$[\![\mathsf{every}]\!]_{\mathcal{M}}([\![\mathsf{lawyer}]\!]_{\mathcal{M}}) \circ \lambda_i = \mathbf{every}(\mathbf{lawyer}) \circ \lambda_i$$

5. $[\![5]\!]_{\mathcal{M}} = \textsc{combine}([\![4_i]\!]_{\mathcal{M}}, [\![3]\!]_{\mathcal{M}})$. As the type of $[\![4_i]\!]_{\mathcal{M}}$ is tt, and the type of $[\![3]\!]_{\mathcal{M}}$ is $(et)t$, we compose the first argument with the second and obtain a value of type $(et)t$:

$$(\mathbf{every}(\mathbf{lawyer}) \circ \lambda_i) \circ (\mathbf{some}(\mathbf{relative}(\mathbf{of}(\mathbf{x_i}))))$$

6. $[\![6]\!]_{\mathcal{M}} = \textsc{combine}([\![\mathsf{despises}]\!]_{\mathcal{M}}, [\![\mathsf{him}_i]\!]_{\mathcal{M}})$, as the type of $[\![\mathsf{despises}]\!]_{\mathcal{M}}$ is eet, and the type of $[\![\mathsf{him}_i]\!]_{\mathcal{M}}$ is e, we apply the first argument to the second and obtain a value of type et:

$$[\![\mathsf{despises}]\!]_{\mathcal{M}}([\![\mathsf{him}_i]\!]_{\mathcal{M}}) = \mathbf{despise}(\mathbf{x_i})$$

7. $[\![7]\!]_{\mathcal{M}} = \textsc{combine}([\![5]\!]_{\mathcal{M}}, [\![6]\!]_{\mathcal{M}})$. As the type of $[\![5]\!]_{\mathcal{M}}$ is $(et)t$, and the type of $[\![6]\!]_{\mathcal{M}}$ is et, we apply one last time the first argument to the second and obtain a value of type t:

$$((\mathbf{every}(\mathbf{lawyer}) \circ \lambda_i) \circ (\mathbf{some}(\mathbf{relative}(\mathbf{of}(\mathbf{x_i})))))(\mathbf{despise}(\mathbf{x_i}))$$

By the definition of function composition, this set of assignments is identical to the below:

$$\mathbf{every}(\mathbf{lawyer})(\lambda_i(\mathbf{some}(\mathbf{relative}(\mathbf{of}(\mathbf{x_i})))(\mathbf{despise}(\mathbf{x_i}))))$$

Although not much more will be said about the denotations of common nouns and verbs, the quantifiers and prepositions are intended to be logical constants in our models, and we can calculate on that basis a more refined description of the denotation of this reading.

The denotations of our constants are given in figure 5. As **of** is here taken to denote the identity function, the set of assignments derived above is identical to the below.

$$\mathbf{every}(\mathbf{lawyer})(\lambda_i(\mathbf{some}(\mathbf{relative}(\mathbf{x_i}))(\mathbf{despise}(\mathbf{x_i}))))$$

Applying the definition of the function **some** in the above, we obtain the following.

$$\{g : \forall a.\ g \in \mathbf{lawyer}(a) \rightarrow g \in \lambda_i(\mathbf{some}(\mathbf{relative}(\mathbf{x_i}))(\mathbf{despise}(\mathbf{x_i})))(a)\}$$

$$\mathbf{some}(A)(B) := \{g : \text{for some } a \in D_e \; g \in A(a) \text{ and } g \in B(a)\}$$

$$\mathbf{of}(a) = a \qquad\qquad \mathbf{x_i}(g) = g_i$$

$$\mathbf{lawyer}(a) = \{g : a(g) \in \texttt{lawyer}\}$$

$$\mathbf{relative}(a)(b) = \{g : \langle a(g), b(g)\rangle \in \texttt{relative}\}$$

$$\mathbf{despise}(a)(b) = \{g : \langle a(g), b(g)\rangle \in \texttt{despise}\}$$

$$\mathbf{every}(A)(B) := \{g : \text{for every } a \in D_e \text{ if } g \in A(a) \text{ then } g \in B(a)\}$$

Figure 5: The denotations of the constants

Unpacking the definition of λ_i from section §2, we arrive at the below.

$$\{g : \forall a.\; g \in \mathbf{lawyer}(a) \rightarrow g \in \{h :\; h^{[i:=a(g)]} \in \mathbf{some}(\mathbf{relative}(\mathbf{x_i}))(\mathbf{despise}(\mathbf{x_i}))\}\}$$

Rewriting $g \in \{h : \Phi(h)\}$ as $\Phi(g)$ nets us the following.

$$\{g : \forall a.\; g \in \mathbf{lawyer}(a) \rightarrow g^{[i:=a(g)]} \in \mathbf{some}(\mathbf{relative}(\mathbf{x_i}))(\mathbf{despise}(\mathbf{x_i}))\}$$

By the definition of **some**, this set is identical to the one below.

$$\{g : \forall a.\; g \in \mathbf{lawyer}(a) \rightarrow g^{[i:=a(g)]} \in \{h : \exists b.\; h \in \mathbf{relative}(\mathbf{x_i})(b) \wedge h \in \mathbf{despise}(\mathbf{x_i})(b)\}\}$$

Again, we rewrite $g \in \{h : \Phi(h)\}$ as $\Phi(g)$:

$$\{g : \forall a.\; g \in \mathbf{lawyer}(a) \rightarrow \exists b.\; g^{[i:=a(g)]} \in \mathbf{relative}(\mathbf{x_i})(b) \wedge g^{[i:=a(g)]} \in \mathbf{despise}(\mathbf{x_i})(b)\}$$

Finally, unpacking the definitions of the non-logical constants, we arrive at the description of this set below.

$$\{g : \forall a.\; a(g) \in \texttt{lawyer} \rightarrow \exists b.\; \langle b(g^{[i:=a(g)]}), a(g)\rangle \in \texttt{relative} \wedge \langle b(g^{[i:=a(g)]}), a(g)\rangle \in \texttt{despise}\}$$

I claim that this is the right meaning for the sentence. In particular, that this set is equivalent to the more familiar description below, which we see

immediately to describe G if every lawyer has a relative who despises him, and $\emptyset$ otherwise:

$$\{g : \forall\alpha \in E.\ \alpha \in \texttt{lawyer} \rightarrow \exists\beta \in E.\ \langle\beta, \alpha\rangle \in \texttt{relative} \wedge \langle\beta, \alpha\rangle \in \texttt{despise}\}$$

We can confirm this in the following manner. First, although quantification in the set we derived is over $D_e = [G \rightarrow E]$, the objects quantified over are always applied to assignment functions, resulting in elements of E. In particular, even though we universally quantify over a, we are in fact looking at all possible images of g under a, which is simply E (D_e contains among others functions $\mathbf{e}$ for every $e \in E$ such that $\mathbf{e}(g) = e$ for all g). Similarly for the existential quantification over b; even though b might be 'crazy', at the end of the day, we are only viewing its image in the set of entities E. Thus, if there is some individual $\beta \in E$ such that β is a relative of α, then, regardless of the assignment function g, if we look at enough $b \in D_e$, we are guaranteed to find one such that $b(g) = \beta$. Similarly, if every individual $\alpha \in E$ is such that if he is a lawyer, then he has a relative who despises him, then regardless of which $a \in D_e$ we look at, and which $g \in G$ we apply a to, $a(g)$ is always going to be such an α. The situation here is parallel to quantifying over intentions, but always talking about their extensions at a particular world.

3.2 Direct Scope

Using function composition, we are also able to render the direct scope reading of example 4 without modifying the denotations assigned to our lexical items, simply by locating the landing site of QR beneath the determiner some, as in figure 6. The denotation of which we calculate in the above man-

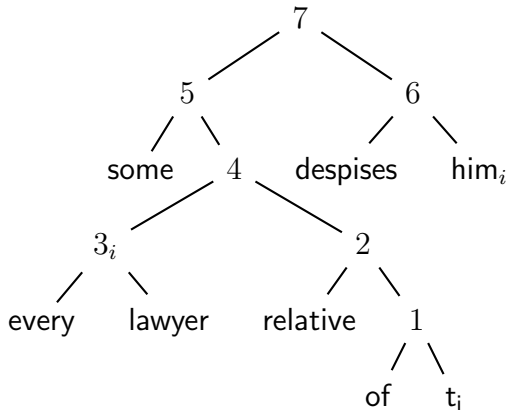

Figure 6: The LF-structure for the direct scope reading of example 4

ner to be $\mathbf{some}((\mathbf{every}(\mathbf{lawyer}) \circ \lambda_i) \circ (\mathbf{relative}(\mathbf{x_i})))(\mathbf{despise}(\mathbf{x_i}))$, which cashes out to

$$\begin{aligned}\{g : \exists b.\ (\forall a.\ g \in \mathbf{lawyer}(a) \rightarrow \\ g^{[i:=a(g)]} \in (\mathbf{relative}(\mathbf{x_i})(b))) \\ \wedge g \in \mathbf{despise}(\mathbf{x_i})(b)\}\end{aligned}$$

As before, we can express this in more familiar terms: this reading is true with respect to an assignment g just in case there is an individual b of whom it is true that he despises g_i, and that for every lawyer a, b is a relative of $(g^{[i:=a]})_i = a$. Note that the QP every lawyer is *not* able to bind pronouns in the scope argument of the QP some relative of every lawyer in this reading.

4 A Comparison with Other Approaches

In what follows I present two alternative analyses of the phenomena of inverse linking, and, in so far as possible, compare them to the one presented herein. The first is the analysis presented by Larson (1985), which provides a mechanism for complex quantifier formation, and makes the standard assumption that pronouns denote variables. Next, I consider the analysis of Büring (2001), who abandons the idea that pronouns denote variables.

4.1 Larson (1985)

Larson (1985) presents a cooper-storage account of inverse-linking (see also Keller (1988)). Modifying Cooper's (1983) original proposal, Larson allows stored expressions to themselves contain stored expressions, et cetera; the denotation of syntactic structures are pairs of expressions, and lists of denotations of syntactic structures (the base case is when the lists are empty). For example, one denotation of the NP *some relative of every lawyer* is as shown below:

$$\langle \mathsf{some}(\mathsf{relative}(\mathsf{of}(\mathsf{x_i}))), \langle \mathsf{every}(\mathsf{lawyer}), \mathsf{x_i} \rangle \rangle$$

When this NP combines with the VP *despises him*, its entire denotation is stored:

$$\langle \mathsf{despises}(\mathsf{x_i})(\mathsf{x_k}), \langle \mathsf{some}(\mathsf{relative}(\mathsf{of}(\mathsf{x_i}))), \langle \mathsf{every}(\mathsf{lawyer}), \mathsf{x_i} \rangle, \mathsf{x_k} \rangle \rangle$$

Having introduced a new 'type' of stored object, Larson is at liberty to define how it is retrieved from the store. In order to capture Larson's generalization, he requires that upon retrieval of a complex expression from

the store, its pieces be applied to the main meaning, in order (from least to most deeply embedded). Continuing with our example above, when $\langle \mathsf{some}(\mathsf{relative}(\mathsf{of}(\mathsf{x_i}))), \langle \mathsf{every}(\mathsf{lawyer}), \mathsf{x_i}\rangle, \mathsf{x_k}\rangle$ is removed from the store, both quantifiers are applied sequentially to the main meaning, starting with the least embedded one:

$$\langle \mathsf{every}(\mathsf{lawyer})(\lambda \mathsf{x_i}.\mathsf{some}(\mathsf{relative}(\mathsf{of}(\mathsf{x_i})))(\lambda \mathsf{x_k}.\mathsf{despises}(\mathsf{x_i})(\mathsf{x_k})))\rangle$$

The main difference between Larson's account and the one presented herein lies in the nature of the elements on the store (viewing the present account from the perspective of cooper-storage, as in Kobele (2006)). In the account here, stored elements are uniformly model-theoretic objects (functions of type $(et)t$), whereas Larson is forced to view stored elements as (at best) heterogeneous triples consisting of model-theoretic objects, a list of stored elements, and syntactic objects (variable names). Whereas Larson is forced to stipulate his eponymous generalization (by means of his definition of retrieval of complex stored elements), in the present system it is a necessary consequence of the fact that complex quantifiers are formed by function composition.

4.2 Büring (2001)

Büring (2001) (see also Büring (2004)) adopts the complex quantifier perspective on inverse linking, and thus assigns an LF structure to inverse linking sentences similar to that in figure 3. To allow the embedded QNP to combine with the container NP, he introduces a 'composition' combinator, which applies to $\mathcal{Q}$ of type $(et)t$, and $\lambda x.D(N(x))$ of type $e(et)t$ to give $\lambda A_{et}.\mathcal{Q}(\lambda y.D(N(y))(A))$ of type $(et)t$. As per the discussion in footnote 3, free variables in the scope argument of this complex quantifier are unbindable. To overcome this problem, Büring rejects the view of pronouns as denoting variables, and adopts an E-type view of pronouns (see Elbourne (2002) and references therein), according to which the denotation of a pronoun *it* is $\textsc{the}(P)$, where P is a contextually determined property. Thus, according to Büring, the following two sentences have identical LFs:

(5) Some man from every city secretly despises it

(6) Some man from every city secretly despises the city

The semantic co-variance between the pronoun and the embededded quantifier is mediated by virtue of the fact that quantifiers (as shown in figure 7) introduce a universal quantification over minimal situations q satisfying their restrictors (in the case of 5, q would contain a man, a city, and the man

$$[\![Q_\sigma]\!]^g(A)(B) :=$$
$$\{t : \mathcal{Q}(\lambda x.cp(g(\sigma))(t) \in A(x))(\lambda x.\forall q \in min(A(x)).q \in B(x))\}$$

$cp(s)(t)$ denotes the counterpart situation to s in the world of situation t

Figure 7: Quantifiers in an E-type semantics

being from that city). The pronoun (covert definite description of the form the_τ city) in the scope argument then picks out the unique city in each such minimal situation. This is achieved by modifying the assignment function which parameterizes the interpretation function on the quantifier's scope argument to interpret the situation parameter on the definite determiner (τ) as the situation q which was introduced in the semantic scope of the quantifier every, and which is a minimal satisfier of *x is a city that some man is from.*[8]

5 Conclusion

We have seen that making assignment functions first class denizens of our models allows for straightforward implementation of obvious ideas about the interpretation of inverse linking constructions. The ideas outlined above in section §3 show how complex quantifier formation can be done using the functions λ_i and function composition. Of independent interest is that these very same functions and operations allow us to treat syntactic indices on moving expressions in the way syntacticians are used to, assigning a denotation directly to a DP with index i ($[\![DP]\!]_\mathcal{M} \circ \lambda_i$), instead of having to re-arrange structures so as to introduce the indicies in separate syntactic positions (as done by Heim and Kratzer (1998)).

It might be claimed that the incorporation of assignment functions into our models is too high a price to pay, as it makes the denotations of things ugly. Important to keep in mind when evaluating this claim is that the assignment functions have been there all along, as meta-linguistic parameters on an uncountably infinite family of interpretation functions. In contrast to the E-type account of Büring (2004), which is also "nerve-wrackingly

[8]Also necessary is a 'downward closure' operator, $\leq$, which applies to a set of situations s to give back the set of all subsituations of all situation in s ($\leq(s) := \{s' : s' \leq s\}$). This allows, for example, a sentence like *no man's mother despises [the man]* to be understood as satisfying a situation s just in case the set of men's mothers in s is disjoint from the set of individuals for whom every minimal situation q of them being a man's mother can be extended to ($\leq$) a situation of them despising the unique man in q.

complex," the pronouns-as-variables approach explored here can be presented at a high-enough level so as to allow us to do semantics as we are used to, without needing to delve into the details of the semantic objects manipulated.

The debate about whether pronouns should be interpreted as variables or as definite descriptions (or something else) is a complex one, with sophisticated analyses having been developed on all sides. What I hope to have shown here is that, if our semantic meta-language allows for variable capture, an elegant account of inverse linking constructions simply falls out if we treat pronouns as variables.

References

Barker, C. (2002). Continuations and the nature of quantification. *Natural Language Semantics 10*, 211–242.

Büring, D. (2001). A situation semantics for binding out of DP. In R. Hastings, B. Jackson, and Z. Zvolenski (Eds.), *Proceedings from Semantics and Linguistic Theory XI*, Ithaca, pp. 56–75. CLC.

Büring, D. (2004). Crossover situations. *Natural Language Semantics 12*(1), 23–62.

Cooper, R. (1983). *Quantification and Syntactic Theory*. Dordrecht: D. Reidel.

Elbourne, P. (2002). *Situations and Individuals*. Ph. D. thesis, Massachusetts Institute of Technology.

Groenendijk, J. and M. Stokhof (1991). Dynamic predicate logic. *Linguistics and Philosophy 14*, 39–100.

Heim, I. and A. Kratzer (1998). *Semantics in Generative Grammar*. Blackwell Publishers.

Keenan, E. L. and L. M. Faltz (1985). *Boolean Semantics for Natural Language*. Dordrecht: D. Reidel.

Keller, W. R. (1988). Nested cooper storage: The proper treatment of quantification in ordinary noun phrases. In U. Reyle and C. Rohrer (Eds.), *Natural Language Parsing and Linguistic Theories*, Number 35 in Studies in Linguistics and Philosophy, pp. 432–447. Dordrecht: D. Reidel.

Klein, E. and I. A. Sag (1985). Type-driven translation. *Linguistics and Philosophy 8*, 163–201.

Kobele, G. M. (2006). *Generating Copies: An investigation into structural identity in language and grammar*. Ph. D. thesis, University of California, Los Angeles.

Landman, M. (2005). *Variables in Natural Language*. Ph. D. thesis, University of Massachusetts, Amherst.

Larson, R. K. (1985). Quantifying into NP. unpublished ms.

May, R. (1977). *The Grammar of Quantification*. Ph. D. thesis, MIT, Cambridge, Massachusetts.

May, R. (1985). *Logical Form: Its Structure and Derivation*. Cambridge, Massachusetts: MIT Press.

May, R. and A. Bale (2005). Inverse linking. In M. Everaert and H. van Riemsdijk (Eds.), *The Blackwell Companion to Syntax*, Volume 2, Chapter 36, pp. 639–667. Oxford: Blackwell.

Montague, R. (1970). Universal grammar. *Theoria 36*(3), 373–398.

Montague, R. (1973). The proper treatment of quantification in ordinary english. In J. Hintikka, J. Moravcsik, and P. Suppes (Eds.), *Approaches to Natural Language*, pp. 221–242. Dordrecht: D. Reidel.

Montague, R. (1974). English as a formal language. In *Formal Philosophy: Selected Papers of Richard Montague*, Chapter 6, pp. 188–221. New Haven: Yale University Press. edited and with an introduction by R. H. Thomason.

Sauerland, U. (2005). DP is not a scope island. *Linguistic Inquiry 36*(2), 303–314.

Sternefeld, W. (1997). The semantics of reconstruction and connectivity. *Arbeitspapiere des SFB 340 97*, 1–58. Tübingen.