Jack Isaac Rabinovitch jrabinovitch@g.harvard.edu

Ling 112 Section

September 30

Reflexivity (Reinhart and Reuland 1993)

(1) a. Condition A: A reflexive-marked syntactic predicate is reflexive.
Reflexive marking → Reflexive meaning
b. Condition B: A reflexive semantic predicate is reflexive marked. Reflexive meaning → Reflexive marking

(2)

	Reflexive marking	No reflexive marking
Reflexive meaning	Use of anaphora	Condition B violation
No reflexive meaning	Condition A violation	Use of pronouns

Reference and Variables

A referent is some individual (in the real world, or an imagined world). Reference denotes the referent which some syntactic object represents. We may sometimes represent a referent with the use of indices: numbers or tags which represent some pointer to a reference.

(3) He$_i$ sees her$_j$.

We can think of predicates as kinds of functions; in this way, every time we add an argument to a sentence, we need to think of it as applying to a predicate.

(4) a. VP: sees her$_j$
b. VP meaning: $\lambda x_k.x_k$ sees her$_j$
c. vP: He$_i$ sees her$_j$
d. vP meaning: He$_i$ $\lambda x_k.x_k$ sees her$_j$
→ He$_i$ sees her$_j$

An index represents some variable within the semantics. Variables can be free or bound. Free variables get their reference from the context.

(5) Free variables:
Syntax: He$_i$ sees her$_j$.
Context: $g(i)$ is Liam, $g(j)$ is Ayaka.
⇒ Liam sees Ayaka.

A bound variable is a variable which acts like a ‘slot’ in some function; it gets its value or reference from whatever the function is applied to. We can represent these sentences in two ways: one way includes the lambdas, and one way applies the variables to each lambda.

(6) a. With lambdas:
He$_i$ $\lambda x_k.x_k$ sees her$_j$
i, j are free. k is bound.

b. Without lambdas:
He$_i$ $\lambda x_k.x_k$ sees her$_j$
i, j are free. k is bound.

Anaphora are necessarily bound, meaning that they necessarily act like these 'slots' in a function.

(7) He sees himself:

a. With lambdas:
He$_i$ $\lambda x_j.x_j$ sees himself$_j$ i is free. j is bound.

b. Without lambdas:
He$_i$ sees himself$_i$ He$_i$ is free, himself$_i$ is bound.

Note that when we use lambdas, we can represent bound variables and (variables which combine with) their binders with different indices. When we don't use lambdas, we apply each referent to their lambda function, and thus bound variables and their binders share an index. When we don't use lambdas, a single index may be free or bound depending on which syntactic object is referring to.

Another way of getting coreference is through having two items have different indices which refer to the same individual. This relies on the context to provide the equality.

(8) His mother sees him:

a. With lambdas:
[His$_i$ mother]$_j$ $\lambda x_l.x_l$ sees him$_k$
i, j, k are free. l is bound.

b. Without lambdas:
[His$_i$ mother]$_j$ sees him$_k$
i, j, k are free.

We can say that the indices i and j both point to the same individual in the context, thus getting coreference without binding.

Deriving Strict and Sloppy

Sometimes, binding and coreference change the interpretation of a sentence.

(9) Sloppy = Binding: Only John cuts his hair.

a. With lambdas:
Only John$_i$ $\lambda x_j . x_j$ cuts his$_j$ hair
Other than John, no one else is an individual x_j such that [x_j cuts x_j's hair]

b. Without lambdas:
Only John$_i$ cuts his$_i$ hair
$i = i$

(10) Strict = Non-binding Coreference: Only John cuts his hair.

a. With lambdas:
Only John$_i$ $\lambda x_j . x_j$ cuts his$_k$ hair
Other than John, no one else is an individual x_j such that [x_j cuts x_k's hair]

b. Without lambdas:
Only John$_i$ cuts his$_k$ hair
$i = k$

(11) Third reading: no coreference.

a. With lambdas:
Only John$_i$ $\lambda x_j . x_j$ cuts his$_k$ hair
Other than John, no one else is an individual x_j such that [x_j cuts x_k's hair]

b. Without lambdas:
Only John$_i$ cuts his$_k$ hair
$i \neq k$