Lecture 3: A Fragment of English. More Applications of the Lambda Calculus.

1. English Fragment 1 ... 1
1.0 Introduction ... 1
1.1. Syntactic categories and their semantic types. ... 2
1.2. Syntactic Rules and Semantic Rules. ... 2
1.2.1. Basic syntactic rules ... 3
1.2.2. Semantic interpretation of the basic rules. ... 3
1.2.3. Rules of Relative clauses, Quantification, Phrasal Negation. See Section 6. ... 4
1.2.4. Type multiplicity and type shifting. ... 5
1.3. Lexicon. ... 5
2. Examples ... 7
3. Rules of Relative clause formation, Quantifying In, Phrasal Negation. ... 8
3.1. (Restrictive) Relative clause formation. ... 8
3.2. Quantifying In. ... 8
3.3. Conjunction. ... 10
3.4. Phrasal and lexical negation. ... 11
REFERENCES. ... 11

1. English Fragment 1.

1.0 Introduction

In this handout we present a small sample English grammar (a "fragment", in MG terminology), that is, an explicit description of the syntax and semantics of a small part of English. This fragment is intended to serve several purposes: making certain aspects of formal semantics more explicit, including (and illustrating) more of the basics of the lambda-calculus. The fragment is of interest in its own right and will also serve as background for the next lecture. The fragment, with its very minimal lexicon, also illustrates the typically minimal treatment of the lexicon in classical Montague grammar.

The semantics of the fragment will be given via translation into Montague's Intensional Logic (IL) (the alternatives would be to give a direct model-theoretic interpretation, or an interpretation via translation into some other model-theoretically interpreted intermediate language). In Lecture 2 we presented Montague's IL: Its type structure and the model structures in which it is interpreted, and its syntax and model-theoretic semantics.

Now we introduce the fragment of English: first the syntactic categories and the category-type correspondence, then the basic syntactic rules and the principles of semantic interpretation, and then a small lexicon and some meaning postulates. In Section 2 we present some examples. Certain rules of the fragment are postponed to Section 3 where they receive separate discussion; these are rules that go beyond the simple phrase structure rule schemata of Section 1.

1.1. Syntactic categories and their semantic types.

Syntactic category	**Semantic type (extensionalized)**	**Expressions**
ProperN	e	names (*John*)
S	t	sentences
CN(P)	$e \to t$	common noun phrases (*cat*)
NP	(i) e	"e-type" or "referential" NPs (*John, the king*)
	(ii)$(e\to t) \to t$	noun phrases as generalized quantifiers (*every man, the king, a man, John*)
	(iii) $e \to t$	NPs as predicates (*a man, the king*)
ADJ(P)	(i) $e \to t$	predicative adjectives (*carnivorous, happy*)
	(ii)$(e \to t) \to (e \to t)$	adjectives as predicate modifiers (*skillful*)
REL	$e \to t$	relative clauses (*who(m) Mary loves*)
VP, IV	$e \to t$	verb phrases, intransitive verbs (*loves Mary, is tall, walks*)
TV(P)	type(NP) $\to$ type(VP)	transitive verb (phrase) (*loves*)
is	$(e \to t) \to (e \to t)$	*is*
DET	type(CN) $\to$ type(NP)	*a, some, the, every, no*

1.2. Syntactic Rules and Semantic Rules.

Two different approaches to semantic interpretation of natural language syntax (both compositional, both formalized, and illustrated, by Montague):

A. Direct Model-theoretic interpretation: Semantic values of natural language expressions (or their "underlying structure" counterparts) are given directly in model-theoretic terms; no intermediate language like Montague's intensional logic (but for some linguists there is a syntactic level of "logical form" to which this model-theoretic interpretation applies, so the distinction between the two strategies is not always sharp.) This is the direct "English as a formal language" strategy. For illustration, see Heim and Kratzer (1998). Also see the discussion in Larson's chapter 12.

B. Interpretation via translation: Stage 1: compositional translation from natural language to a language of semantic representation, such as Montague's intensional logic. For an expression of category C formed from expressions of category A and of category B, determine **TR**(γ) as a function of **TR**(α) and **TR**(β). Stage 2: Apply the compositional model-theoretic interpretation rules to the intermediate language.

We will follow the strategy of interpretation via translation, using Montague's IL as the intermediate language. But everything we do could also be done by direct interpretation.

Some abbreviations and notational conventions:

We will sometimes write α' as a shorthand for **TR**(α). And sometimes we use the category name in place of a variable over expressions of that category, writing **TR**(A), or A', in place of **TR**(α) when α is an expression of category A. And we will write some of our syntactic rules like simple phrase structure rules. Here is an example of a syntactic rule and corresponding translation rule, and their abbreviations as they will appear below.

<u>Official Syntactic Rule</u>: If α is an expression of category DET and β is an expression of category CNP, then $F_0(\alpha,\beta)$ is an expression of category NP, where $F_0(\alpha,\beta) = \alpha\beta$.
<u>Official Semantic Rule</u>**:** If $\mathbf{TR}(\alpha) = \alpha'$ and $\mathbf{TR}(\beta) = \beta'$, then $\mathbf{TR}(F_0(\alpha,\beta)) = \alpha'(\beta')$.

<u>Abbreviated Syntactic Rule</u>: NP → DET CNP
<u>Abbreviated Semantic Rule</u>: NP' = DET'(CNP')

1.2.1. Basic syntactic rules

Basic rules, phrasal:

S	→	NP VP
NP	→	DET CNP
CNP	→	ADJP CNP
CNP	→	CNP REL
VP	→	TVP NP
VP	→	is ADJP
VP	→	is NP

Basic rules, non-branching rules introducing lexical categories:

NP	→	ProperN
CNP	→	CN
TVP	→	TV
ADJP	→	ADJ
VP	→	IV

1.2.2. Semantic interpretation of the basic rules.

The basic principle for all semantic interpretation in formal semantics is the principle of compositionality; the meaning of the whole must be a function of the meanings of the parts. In the most "stipulative" case, we write a semantic interpretation rule (translation or direct model-theoretic interpretation) for each syntactic formation rule, as in classical MG. In more contemporary approaches, we look for general principles governing the form of the rules and their correspondence (possibly mediated by some syntactic level of "Logical Form".) Here we are using an artificially simple fragment, and we have presented the syntactic rules in a form which is explicit but not particularly general; but we have the tools to illustrate a few basic generalizations concerning syntax-semantics correspondence.

1.2.2.1.Type-driven translation. (Partee 1976, Partee and Rooth 1983, Klein and Sag 1985)

To a great extent, possibly completely, we can formulate general principles for the interpretation of the basic syntactic constructions based on the semantic types of the constituent parts.

So suppose we are given a rule A → B C, and we want to know how to determine A' as a function of B'and C' (equivalently, **TR**(A) as a function of **TR**(B) and **TR**(C); and ultimately, ||A|| as a function of ||B|| and ||C||.) Similarly for non-branching rules A → B.

General principles: function-argument application, predicate conjunction, identity.

The following versions of general type-driven interpretation principles are taken from Heim and Kratzer (1995).
In the original they are written for direct model-theoretic interpretation.

(1) *Terminal Nodes* (TN): If α is a terminal node, then $[[\alpha]]$ is specified in the lexicon.
(2) *Non-Branching Nodes* (NN): If α is a non-branching node, and β is its daughter node, then $[[\alpha]] = [[\beta]]$.
(3) *Functional Application* (FA): If α is a branching node, $\{\beta,\gamma\}$ is the set of α's daughters, and $[[\beta]]$ is a *function* whose domain contains $[[\gamma]]$, then $[[\alpha]] = [[\beta]]([[\gamma]])$.
(4) *Predicate Modification* (PM) : If α is a branching node, $\{\beta, \gamma\}$ is the set of α's daughters, and $[[\beta]]$ and $[[\gamma]]$ are both in $D_{<e,t>}$, then $[[\alpha]] = \lambda x \in D_e .\ [[\beta]](x) = 1$ and $[[\gamma]](x) = 1$.
(A further principle is needed for *intensional functional application*, which we will mention only later.)

Exactly analogous principles can be written for **type-driven *translation***:

(1) *Terminal Nodes* (TN): If α is a terminal node, then **TR**(A) is specified in the lexicon.
(2) *Non-Branching Nodes* (*NN*): If A → B is a unary rule and A,B are of the same type, then **TR**(A) = **TR**(B).
(3) *Functional Application* (FA): If A is a branching node, {B,C} is the set of A's daughters, and B' is of a functional type a → b and C' is of type a, then **TR**(A) = **TR**(B)(**TR**(C)).
(4) *Predicate Modification* (*PM*) : If A is a branching node, {B,C} is the set of A's daughters, and if B' and C' are of (same) predicative type a → t, and the syntactic category A can also correspond to type a → t, then $\mathbf{TR}(A) = \lambda \boldsymbol{x}[\mathbf{TR}(B)(\boldsymbol{x})\ \&\ \mathbf{TR}(C)(\boldsymbol{x})]$. (i.e. ||A|| = ||B|| ∩ ||C||.)

1.2.2.2. Result of those principles for the translation of the basic rules.

Function-argument application: S→ NP VP, NP → DET CNP, VP → TVP NP,
VP → *is* ADJP, VP → *is* NP, and those instances of CNP → ADJP CNP in which ADJP is of type (e→t)→(e→t).

Example: Consider the rule S→NP VP. If NP is of type (e→t)→t and VP is of type e→t, then the translation of S will be NP'(VP') (e.g., *Every man walks*). If NP is of type e and VP is of type e→t, then the translation of S will be VP'(NP') (e.g., *John walks*).

Predicate modification: CNP → CNP REL, and those instances of CNP → ADJP CNP in which ADJP is of type e→t.

Non-branching nodes: NP → ProperN, CNP → CN, TVP → TV, ADJP → ADJ.

1.2.3. Rules of Relative clauses, Quantification, Phrasal Negation. See Section 3.

1.2.4. Type multiplicity and type shifting.

We noted in Lecture 1 that classical model-theoretic semantics in the Montague tradition requires that there be a single semantic type for each syntactic category. But in Fragment 1, several syntactic types have more than one corresponding semantic type. The possibility of type multiplicity and type shifting has been increasingly recognized in the last decade or so, and there are a variety of formal approaches that accommodate type multiplicity without giving up compositionality. We will not go into details about formal issues here, but we do want to include a number of categories with multiple semantic types; several were introduced in Fragment 1, and more will be introduced in later lectures.

Montague tradition: uniform treatment of NP's as generalized quantifiers, type $(e \to t) \to t$.

John	$\lambda \boldsymbol{P}[\boldsymbol{P}(\mathbf{John})]$	(the set of all of John's properties)
a fool	$\lambda \boldsymbol{P} \exists \boldsymbol{x}[\mathbf{fool}(\boldsymbol{x})\ \&\ \boldsymbol{P}(\boldsymbol{x})]$	
every man	$\lambda \boldsymbol{P} \forall \boldsymbol{x}[\mathbf{man}(\boldsymbol{x}) \to \mathbf{P}(\boldsymbol{x})]$	

Intuitive type multiplicity of NP's (and see Heim 1982, Kamp 1981):

John	"referential use":	**John**	type e
a fool	"predicative use":	**fool**	type $e \to t$
every man	"quantifier use":	(above)	type $(e\to t)\to t$

Resolution: All NP's have meaning of type $(e\to t)\to t$; some also have meanings of types e and/or $e \to t$. General principles for predicting (Partee 1986). Predicates may semantically take arguments of type e, $e \to t$, or $(e\to t)\to t$, among others. (More on type-shifting in Lecture 6 (or see RGGU 2005 Lecture 8 on my website).

Type choice determined by a combination of factors including coercion by demands of predicates, "try simplest types first" strategy, and default preferences of particular determiners.

Note the effects of this type multiplicity on type-driven translation. The $S \to NP\ VP$ rule, for instance, will have two different translations. The VP, we have assumed, is always of type $e \to t$. If the NP is of type e, the translation will be VP'(NP'), whereas if the NP is of type $(e\to t)\to t$, the translation will be NP'(VP'), as noted above in Section 1.2.2.2. **[See Homework problem #3 of Homework 1.]**

1.3. Lexicon.

Here we illustrate the treatment of the lexicon in Montague (1973) ("PTQ"). Montague, not unreasonably, saw a great difference between the study of the principles of compositional semantics, which are very similar to the principles of compositional semantics for logical languages as studied in logic and model theory, and the study of lexical semantics, which he perceived as much more "empirical". For Montague, it was important to figure out the difference in logical type between *easy* and *eager*, or between *seem* and *try*, but he did not try to say anything about the difference in meaning between two elements with the same "structural" or type-theoretic behavior, such as *easy* and *difficult* or *run* and *walk*. For Montague, most lexical items were considered atomic expressions of a given type, and simply translated into constants of IL of the given type.

First we simply list some lexical items of various syntactic categories; aside from the category DET, these are all open classes. Then we discuss their semantics.

In later lectures we will be concerned with how best to enrich the semantic information associated with the lexicon in ways compatible with a compositional semantics.

ProperN: *John, Mary, Bill, ...*
DET: *some, a, the, every*
ADJ: *carnivorous, happy, skillful, tall, former, alleged, old, ...*
CN: *man, king, violinist, surgeon, fish, senator, ...*
TV: *sees, loves, catches, eats, ...*
IV: *walks, talks, runs, ...*

Semantics of Lexicon (MG):
Open class lexical items (nouns, adjectives, verbs) translated into constants of appropriate type (notation: English expressions *man, tall* translated into IL constants **man**, **tall**, etc.). Interpretation of these constants a central task of lexical semantics. A few open class words (e.g. *be*, *entity*, *former*) sometimes treated as part of the "logical vocabulary".

Closed class lexical items: some treated like open class items (e.g. most prepositions), others (esp. "logical" words) given explicit interpretations, as illustrated below.

Determiners:
We have three types of NPs and correspondingly three types of DETs. Not all DETs occur in all types; *the* is one of the few that does. For DETs that occur in more than one type, we will subscript the "homonyms" with mnemonic subscripts: **e** for those that combine with a CNP to form an e-type NP, **pred** for those that form predicate nominals, and **GQ** for those that form generalized quantifier-type NPs. (Note that these are not the types of the DETs themselves, but their own types have unpleasantly long and hard-to-read names.) There are systematic relations among these "homonyms" (Partee 1987), but we are not discussing them here.

(i) e-forming DETs.
For the translation of $the_{\boldsymbol{e}}$, we need to add the <u>*iota-operator*</u> ι to IL.

Syntax: If $\varphi \in ME_t$ and $\boldsymbol{u}$ is a variable of type *e*, then $\iota \boldsymbol{u}[\varphi] \in ME_e$.

Semantics: $\|\iota \boldsymbol{u}[\varphi]\|^{M,w,g} = d$ iff there is one and only one $d \in D$ such that $\|\varphi\|^{M,w,g[d/u]} = 1$.
$\|\iota \boldsymbol{u}[\varphi]\|^{M,w,g}$ is undefined otherwise.

So $\iota \boldsymbol{x}(\mathbf{king}(\boldsymbol{x}))$ denotes the unique individual who is king, if there is one, and is undefined if there is either no king or more than one king.

The type for determiners as functors forming e-type ("referential") terms is $(e \to t) \to e$; the only determiner of this type we will introduce is the_e. For ease of reading, we will give the translations of representative NPs, rather than for the DET itself; the DET translation can be formed in each case by λ-abstraction on the CNP (see below, **TR**(a_{pred})).

$\mathbf{TR}(the_e\ king) = \iota \boldsymbol{x}(\mathbf{king}(\boldsymbol{x}))$

(ii) predicate-forming DETs.
DETs as functors forming predicate nominals are of type $(e \to t) \to (e \to t)$.

$\mathbf{TR}(a_{pred}\ man) = \mathbf{man}$
$\mathbf{TR}(the_{pred}\ man) = \lambda \boldsymbol{x}[\mathbf{man}(\boldsymbol{x})\ \&\ \forall \boldsymbol{y}[\mathbf{man}(\boldsymbol{y}) \to \boldsymbol{y}=\boldsymbol{x})]]$

We illustrate the translation of the DET itself with the translation of a_{pred}.

TR(a_{pred}) = $\lambda \boldsymbol{P}[\boldsymbol{P}]$

(iii) generalized quantifier-forming DETs.

DETs as functors forming generalized quantifiers are of type $(e\rightarrow t)\rightarrow((e\rightarrow t)\rightarrow t)$

TR(a_{GQ} *man*) = $\lambda \boldsymbol{P}\exists \boldsymbol{x}[\mathbf{man}(\boldsymbol{x})\ \&\ \boldsymbol{P}(\boldsymbol{x})]$

TR($every_{GQ}$ *man*) = $\lambda \boldsymbol{P}\forall \boldsymbol{x}[\mathbf{man}(\boldsymbol{x}) \rightarrow \boldsymbol{P}(\boldsymbol{x})]$ **(see Homework problem 4)**

TR(the_{GQ} *man*) = $\lambda \boldsymbol{P}\exists \boldsymbol{x}[\mathbf{man}(\boldsymbol{x})\ \&\ \forall \boldsymbol{y}[\mathbf{man}(\boldsymbol{y}) \rightarrow \boldsymbol{y}=\boldsymbol{x})]\&\ \boldsymbol{P}(\boldsymbol{x})]$

The copula *be*:

TR(*is*) = $\lambda \boldsymbol{P}\lambda \boldsymbol{x}[\boldsymbol{P}(\boldsymbol{x})]$ ("Predicate!")

Results (see also Section 5, and Homework problems 7, 8):

TR(*is green*) = **green**

TR(*is* a_{pred} *man*) = **man**

TR(*is* the_{pred} *king*) = $\lambda \boldsymbol{x}[\mathbf{king}(\boldsymbol{x})\ \&\ \forall \boldsymbol{y}[\mathbf{king}(\boldsymbol{y}) \rightarrow \boldsymbol{y}=\boldsymbol{x})]]$

2. Examples

(1) *is happy*

TR(*is*) = $\lambda \boldsymbol{P}\lambda \boldsymbol{x}[\boldsymbol{P}(\boldsymbol{x})]$

TR(*happy*) = **happy**

TR(*is happy*) = $\lambda \boldsymbol{P}\lambda \boldsymbol{x}[\boldsymbol{P}(\boldsymbol{x})]$ **(happy)**

= $\lambda \boldsymbol{x}[\mathbf{happy}(\boldsymbol{x})]$ = **happy**

(2) *The violinist is happy* (with e-type interpretation of subject)

TR($[_{NP}$ *the violinist*$]$) = $\iota \boldsymbol{x}[\ \mathbf{violinist}(\boldsymbol{x})]$ type: e

TR($[_{VP}$ *is happy*$]$) = **happy** type: $e \rightarrow t$

TR($[_{S}$ *the violinist is happy*$]$ = $\mathbf{happy}(\iota \boldsymbol{x}[\mathbf{violinist}(\boldsymbol{x})])$ type: t

(VP meaning applies to NP meaning)

(3) *Every violinist is happy* (with GQ-type subject)

TR(*every violinist*) = $\lambda \boldsymbol{P}\forall \boldsymbol{x}[\mathbf{violinist}\ (\boldsymbol{x}) \rightarrow \boldsymbol{P}(\boldsymbol{x})]$ type $(e \rightarrow t) \rightarrow t$

TR(*is happy*) = **happy** type $e \rightarrow t$

TR(*every violinist is happy*) = $\lambda \boldsymbol{P}\forall \boldsymbol{x}[\mathbf{violinist}\ (\boldsymbol{x}) \rightarrow \boldsymbol{P}(\boldsymbol{x})](\mathbf{happy})$

= $\forall \boldsymbol{x}[\mathbf{violinist}\ (\boldsymbol{x}) \rightarrow \mathbf{happy}(\boldsymbol{x})]$

(4) *Every surgeon is a skillful violinist*

(The type of *every surgeon* must be $(e \rightarrow t) \rightarrow t$; the type of *a skillful violinist* must be $e \rightarrow t$. Assume the type of *skillful* is $(e \rightarrow t) \rightarrow (e \rightarrow t)$.)

TR(*every surgeon*) = $\lambda \boldsymbol{P}\forall \boldsymbol{x}[\mathbf{surgeon}\ (\boldsymbol{x}) \rightarrow \boldsymbol{P}(\boldsymbol{x})]$

TR(*skillful violinist*) = **skillful(violinist)**

TR(*is a skillful violinist*) = **TR** (*a skillful violinist*)= **TR**(*skillful violinist*)

TR(*every surgeon is a skillful violinist*) = $\forall \boldsymbol{x}[\mathbf{surgeon}(\boldsymbol{x}) \rightarrow \mathbf{skillful}(\mathbf{violinist})(\boldsymbol{x})]$

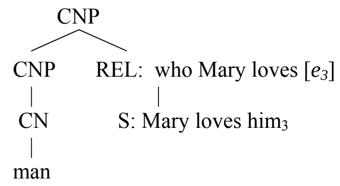
Formal Semantics, Lecture 3
Barbara H. Partee, MGU, February 27, 2007

3. Rules of Relative clause formation, Quantifying In, Phrasal Negation.

3.1. (Restrictive) Relative clause formation.

We begin with an illustration of what the rule does before stating it (in a sketchy form). Consider the CNP *man who Mary loves*:

Syntactic derivation (very sketchy):



The types for CN, CNP, and REL are all $e \rightarrow t$; so the principle for combining CNP and REL gives: $\lambda \boldsymbol{y}$[CNP'($\boldsymbol{y}$) & REL'($\boldsymbol{y}$)] (Predicate conjunction)

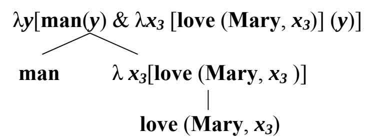
The relative clause itself is a predicate formed by λ-abstraction on the variable corresponding to the WH-word. (Partee 1976 suggests a general principle that all "unbounded movement rules" are interpreted as involving variable-binding; and λ-abstraction can be taken as the most basic variable-binding operation.)

A syntactically very crude and informal version of the relative clause rule, with its semantic interpretation, can be stated as follows:

Rel Clause Rule, syntax: If φ is an S and φ contains an indexed pronoun he_i / him_i in relativizable position, then the result of adjoining *who(m)* to S and leaving a trace e_i in place of he_i / him_i is a REL.

Rel Clause Rule, semantics: If φ translates as φ', then REL translates as $\lambda \boldsymbol{x_i}$[φ'].

Semantic derivation corresponding to the syntactic derivation above; compositional translation into IL: (read bottom-to-top) **(and see Homework problem 5a)**



By λ-conversion, the top line is equivalent to: $\lambda \boldsymbol{y}[\mathbf{man}(\boldsymbol{y})\ \&\ \mathbf{love}\ (\mathbf{Mary}, \boldsymbol{y})]$

3.2. Quantifying In.

This is (an informal statement of) Montague's Quantifying In rule; it is similar to the Quantifier-Lowering rule of Generative Semantics and Quantifier Raising (QR) of May (1977); various alternative treatments of quantifier scope ambiguity exist, including Cooperstorage (Cooper 1975) and Herman Hendriks's flexible typing approach (Hendriks 1988, 1993).

Quantifying In Rule, Syntax: (informally stated): An NP combines with a sentence with respect to a choice of variable ("he_i" in MG). Substitute the NP for the first occurrence of the variable; change any further occurrences of the variable into pronouns of the appropriate number and gender.

Semantic rule: NP'($\lambda \boldsymbol{x_i}$ [S']) (The set of properties denoted by the NP includes the property denoted by the λ-expression derived from the sentence.)

We illustrate with two derivations for the ambiguous sentence *Every student read a book.*

<u>**Syntactic derivation (i)**</u> (rough sketch; read from bottom to top. **Bold** is used here to show which variables are substituted for at each step.)

S: every student read a book
NP: every student | S: $\mathbf{he_3}$ read a book
NP: a book | S: he_3 read $\mathbf{him_2}$

Compositional Translation: (every student)'($\lambda \boldsymbol{x_3}$ [(a book)'($\lambda \boldsymbol{x_2}$ [read ($\boldsymbol{x_3}$, $\boldsymbol{x_2}$)])])

Rough paraphrase: Every student has the property that there is a book that he read.

If you write out the interpretations of the NPs and apply Lambda-Conversion as many times as possible, the result will be (some alphabetic variant of) the first-order PC formula $\forall \boldsymbol{x}(\mathbf{student}(\boldsymbol{x}) \rightarrow \exists \boldsymbol{y}(\mathbf{book}(\boldsymbol{y})\ \&\ \mathbf{read}(\boldsymbol{x,y})))$.

<u>**Syntactic derivation (ii)**</u>

S: every student read a book
NP: a book | S: every student read $\mathbf{him_2}$
NP: every student | S: $\mathbf{he_3}$ read him_2

Compositional Translation: (a book)'($\lambda \boldsymbol{x_2}$[(every student)'($\lambda \boldsymbol{x_3}$ [read ($\boldsymbol{x_3}$, $\boldsymbol{x_2}$)])]) [**See Homework problem 6**.]
Paraphrase: Some book has the property that every student read it.
After applying Lambda-Conversion as many times as possible, the result will be (some alphabetic variant of) the first-order PC formula $\exists \boldsymbol{y}(\mathbf{book}(\boldsymbol{y})\ \&\ \forall \boldsymbol{x}(\mathbf{student}(\boldsymbol{x}) \rightarrow \mathbf{read}(\boldsymbol{x,y})))$.

Observation: Compositional semantics requires that every ambiguous sentence be explainable on the basis of ambiguous lexical items and/or multiple syntactic derivations. Semantic structure mirrors syntactic part-whole structure, which in Montague Grammar is

represented by syntactic derivational structure, not surface structure. There are different theories of the semantically relevant syntactic structure: "Derivation trees" or "analysis trees" (MG), LF (Chomskian GB or Minimalist theory), Tectogrammatic Dependency Trees (Prague), Deep Syntactic Structure (Mel'čuk) Underlying Structure (Generative Semantics), ... GPSG, HPSG, and various contemporary versions of Categorial Grammar are attempts to represent all the necessary syntactic information directly in a single "level" of syntax.

3.3. Conjunction.

One simple and elegant application of lambda abstraction which Montague used in PTQ is its use in defining the interpretation of "Boolean" phrasal conjunction, disjunction, and negation in terms of the corresponding sentential operations.

"Boolean" phrasal conjunction, illustrated in all the examples below, is distinguished from "part-whole" or "group" conjunction, illustrated by "John and Mary are a happy couple" and "The flag is red and white", which are not equivalent, respectively, to "John is a happy couple and Mary is a happy couple" and "The flag is red and the flag is white".

To illustrate this application, we add a few syntactic and semantic rules to our fragment. Note: in the semantic rules, we use S_1 and S_2, etc., to refer to the first and second S in the syntactic rule.

Syntactic rules for conjunction:	**Corresponding semantic rules:**
S $\rightarrow$ S and S	S' = S_1' & S_2'
S $\rightarrow$ S or S	S' = S_1' $\vee$ S_2'
VP $\rightarrow$ VP and VP	VP' = λx [VP_1'(x)& VP_2'(x)]
VP $\rightarrow$ VP or VP	VP' = λ x [VP_1'(x) $\vee$ VP_2'(x)]
NP $\rightarrow$ NP and NP	NP' = λP[NP_1'(P) & NP_2'(P)]
NP $\rightarrow$ NP or NP	NP' = λ P [NP_1'(P) $\vee$ NP_2'(P)]

The NP conjunction and disjunction rules presuppose that the NPs are interpreted as generalized quantifiers, type <<e,t>,t>; ***P*** is a variable of type <e,t>. (Conjoined NPs of type e can be interpreted as groups, but not as conjoined by Boolean conjunction.)

Examples:
Some animals swim and some animals fly. (S-conjunction)
Some animals swim and fly. (VP-conjunction)
Every fish and some birds swim. (NP-conjunction)
Every painting and every statue was photographed or (*was*) *videotaped.* (NP-conjunction and VP-conjunction (or conjunction of participles, if we omit the second 'was', but it's equivalent to VP conjunction). The rules do correctly "predict" which conjunction has wider scope.
(**Optional unlisted extra homework question: work out the last example.** Treat "was photographed" and "was videotaped" as simple 1-place predicates for this exercise.)

We could extend the rules above, and generalize them (as is done in Partee and Rooth 1983), so as to include further types of phrasal conjunction such as the following:
John bought and read a new book. (TV conjunction)
No number is even and odd. (Predicate ADJP conjunction)
Mary saw an old and interesting manuscript. (Pre-nominal ADJP conjunction.)

In fact, we do not have to "stipulate" the rules one-by-one as we have done above; it is possible to predict them in a general way from the *types* of the expressions being conjoined. But that goes beyond the scope of these lectures; see Partee and Rooth 1983.

3.4. Phrasal and lexical negation.

As an additional augmentation of our grammar which adds further illustration of the application of the lambda calculus, let us consider the relations among sentence negation, phrasal negation, and lexical (prefixal) negation.

The syntax of sentential negation in English is slightly complicated because of its interaction with the system of verbal auxiliaries, which we have not included in our simple grammar. Let us ignore the syntactic complexities here and work with a "Logical Form" grammar in which we have a simple phrase structure rule S $\rightarrow$ NEG S. And let us add to the lexicon an element "NOT" of syntactic category NEG, of semantic type t $\rightarrow$ t, whose translation into IL is simply ¬. Then the interpretation of NEG S, by function-argument application, will just be ¬S'.

Now what about phrasal negation, like *not every boy*, *not today, Mary but not John, not very intelligent, not love Bill*? Negation, like conjunction, is a "Boolean" operation, and it is easy to define its interpretation with expressions of various types on the basis of its interpretation with sentences. For example:

Syntax: VP $\rightarrow$ NEG_{VP} VP

Semantics: Type: Since the type of VP is e$\rightarrow$t, the type of NEG_{VP} must be (e $\rightarrow$ t)$\rightarrow$(e $\rightarrow$ t).

$$\mathbf{TR}(\text{VP2}) = \mathbf{TR}(\text{NEG}_{VP})(\mathbf{TR}(\text{VP1})$$
$$= \lambda \boldsymbol{P}[\lambda \boldsymbol{x}[\neg\, \boldsymbol{P}(\boldsymbol{x})]](\mathbf{TR}(\text{VP1}))$$
$$= \lambda \boldsymbol{x}[\neg \text{VP1}'\ (\boldsymbol{x})]$$

Similar rules for negation of other phrasal categories can be derived in a uniform way. (**See Homework Problem 5b, which asks for NEG_{NP} and optionally NEG_{DET}**.)

Both negation and conjunction (and disjunction) thus have natural extensions to a wide range of types (see Partee and Rooth 1983), with meanings systematically derivable from their basic meanings, which apply to sentences.

Lexical negation, as in *unhappy, impossible, unbroken, inedible*, can also be defined with the use of lambdas:

$$\mathbf{TR}(\textit{unhappy}) = \lambda \boldsymbol{x}[\neg \mathbf{TR}(\textit{happy})(\boldsymbol{x})] = \lambda \boldsymbol{x}[\neg \mathbf{happy}(\boldsymbol{x})]$$

===

REFERENCES.

*****Note: See website for links to some of these in downloadable form. *****

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

Klein, Ewan and Ivan Sag (1985), "Type-driven translation", *Linguistics and Philosophy* 8, 163-201.

Lewis, David (1970) "General semantics" *Synthese* 22, 18-67; reprinted in D.Davidson and G.Harman (eds.), *Semantics of Natural Language*. Dordrecht: Reidel (1972), 169-218. Also reprinted in Partee, Barbara H. ed. 1976. *Montague Grammar*. New York: Academic Press.

Montague, R. (1973) "The Proper Treatment of Quantification in Ordinary English," in K.J.J. Hintikka, J.M.E. Moravcsik, and P. Suppes, eds., *Approaches to Natural Language*, Reidel, Dordrecht, 221-242, reprinted in Montague (1974) 247-270. Also reprinted in Portner, Paul, and Partee, Barbara H. eds. 2002. *Formal Semantics: The Essential Readings*. Oxford: Blackwell Publishers.

Montague, Richard (1974) *Formal Philosophy: Selected Papers of Richard Montague.* Edited and with an introduction by Richmond Thomason, New Haven: Yale Univ. Press.

Partee, Barbara H. (1976) "Semantics and syntax: the search for constraints" in C.Rameh (ed.) *Georgetown University Round Table on Languages and Linguistics 1976*, Georgetown: Georgetown University School of Languages and Linguistics (99-110).

Partee, Barbara H. 1986. Noun phrase interpretation and type-shifting principles. In *Studies in Discourse Representation Theory and the Theory of Generalized Quantifiers*, eds. J. Groenendijk et al., 115-143. Dordrecht: Foris. [Reprinted in Portner and Partee, eds., 2002, 357-381. Reprinted in Partee, Barbara H. 2004. *Compositionality in Formal Semantics: Selected Papers by Barbara H. Partee*. Oxford: Blackwell Publishing, 203-230].

Partee, Barbara and Mats Rooth (1983) "Generalized conjunction and type ambiguity", in R. Bauerle, C. Schwarze, and A. von Stechow (eds.) *Meaning, Use and Interpretation of Language*, Walter de Gruyter, Berlin: 361-383. Reprinted in: Portner, Paul, and Partee, Barbara H. eds. 2002. *Formal Semantics: The Essential Readings*. Oxford: Blackwell Publishers.

Partee, B., A. ter Meulen, and R.E. Wall (1990) *Mathematical Methods in Linguistics*, Dordrecht: Kluwer Academic Publishers.

Portner, Paul, and Partee, Barbara H. eds. 2002. *Formal Semantics: The Essential Readings*. Oxford: Blackwell Publishers.