

Lecture 2: A Fragment of English. More Applications of the Lambda Calculus.

1. English Fragment 1.	1
1.1. Syntactic categories and their semantic types.....	1
1.2. Syntactic Rules and Semantic Rules.....	1
1.2.1. Basic syntactic rules.....	2
1.2.2. Semantic interpretation of the basic rules.....	3
1.2.3. Rules of Relative clauses, Quantification, Phrasal Negation. See Section 3.....	4
1.2.4. Type multiplicity and type shifting.....	4
1.3. Lexicon.....	5
2. Examples.....	7
3. Rules of Relative clause formation, Quantifying In, Phrasal Negation.....	7
3.1. (Restrictive) Relative clause formation.....	7
3.2. Quantifying In.....	8
3.3 Phrasal and lexical negation.....	9
REFERENCES.....	10
HOMEWORK #2.....	10

1. English Fragment 1.

1.1. Syntactic categories and their semantic types.

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

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).

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 γ of category C formed from expressions α of category A and β of category B, determine $\mathbf{TR}(\gamma)$ as a function of $\mathbf{TR}(\alpha)$ and $\mathbf{TR}(\beta)$. Stage 2: Apply the compositional model-theoretic interpretation rules to the intermediate language.

We will follow strategy 2, using Montague’s IL as the intermediate language.
Some abbreviations and notational conventions:

We will sometimes write α' as a shorthand for $\mathbf{TR}(\alpha)$. And sometimes we use the category name in place of a variable over expressions of that category, writing $\mathbf{TR}(A)$, or A' , in place of $\mathbf{TR}(\alpha)$ 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.

Official Syntactic Rule: 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 $F_0(\alpha, \beta) = \alpha\beta$.

Official Semantic Rule: If $\mathbf{TR}(\alpha) = \alpha'$ and $\mathbf{TR}(\beta) = \beta'$, then $\mathbf{TR}(F_0(\alpha, \beta)) = \alpha'(\beta')$.

Abbreviated Syntactic Rule: NP \rightarrow DET CNP

Abbreviated Semantic Rule: NP' = DET'(CNP')

1.2.1. Basic syntactic rules

Basic rules, phrasal:

S \rightarrow NP VP

NP \rightarrow DET CNP

CNP \rightarrow ADJP CNP

VP \rightarrow TVP NP

VP \rightarrow is ADJP

VP \rightarrow is NP

Basic rules, non-branching rules introducing lexical categories:

NP \rightarrow ProperN

CNP \rightarrow CN

TVP \rightarrow TV

ADJP \rightarrow ADJ

VP \rightarrow 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 $A \rightarrow B C$, and we want to know how to determine A' as a function of B' and C' (equivalently, $\mathbf{TR}(A)$ as a function of $\mathbf{TR}(B)$ and $\mathbf{TR}(C)$; and ultimately, $\mathbf{5}A\mathbf{5}$ as a function of $\mathbf{5}B\mathbf{5}$ and $\mathbf{5}C\mathbf{5}$.) Similarly for non-branching rules $A \rightarrow B$.

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

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

- (1) *Terminal Nodes* (TN): If α is a terminal node, then $\llbracket \alpha \rrbracket$ is specified in the lexicon.
- (2) *Non-Branching Nodes* (NN): If α is a non-branching node, and β is its daughter node, then $\llbracket \alpha \rrbracket = \llbracket \beta \rrbracket$.
- (3) *Functional Application* (FA): If α is a branching node, $\{\beta, \gamma\}$ is the set of α 's daughters, and $\llbracket \beta \rrbracket$ is a function whose domain contains $\llbracket \gamma \rrbracket$, then $\llbracket \alpha \rrbracket = \llbracket \beta \rrbracket(\llbracket \gamma \rrbracket)$.
- (4) *Predicate Modification* (PM): If α is a branching node, $\{\beta, \gamma\}$ is the set of α 's daughters, and $\llbracket \beta \rrbracket$ and $\llbracket \gamma \rrbracket$ are both in $D_{\langle e, t \rangle}$, then $\llbracket \alpha \rrbracket = \lambda x \in D_e . \llbracket \beta \rrbracket(x) = 1$ and $\llbracket \gamma \rrbracket(x) = 1$. (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 $\mathbf{TR}(A)$ is specified in the lexicon.
- (2) *Non-Branching Nodes* (NN): If $A \rightarrow B$ is a unary rule and A, B are of the same type, then $\mathbf{TR}(A) = \mathbf{TR}(B)$.
- (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 $a \rightarrow b$ and C' is of type a , then $\mathbf{TR}(A) = \mathbf{TR}(B)(\mathbf{TR}(C))$.
- (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 $a \rightarrow t$, and the syntactic category A can also correspond to type $a \rightarrow t$, then $\mathbf{TR}(A) = \lambda x[\mathbf{TR}(B)(x) \ \& \ \mathbf{TR}(C)(x)]$. (i.e. $\mathbf{5}A\mathbf{5} = \mathbf{5}B\mathbf{5} \cap \mathbf{5}C\mathbf{5}$.)

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

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

Example: Consider the rule $S \rightarrow NP VP$. If NP is of type $(e \rightarrow t) \rightarrow t$ and VP is of type $e \rightarrow t$,
 then the translation of S will be $NP'(VP')$. If NP is of type e and VP is of type $e \rightarrow t$, then
 the translation of S will be $VP'(NP')$.

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

Non-branching nodes: $NP \rightarrow ProperN$, $CNP \rightarrow CN$, $TVP \rightarrow TV$, $ADJP \rightarrow ADJ$.

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 $(e \rightarrow t) \rightarrow t$.

<i>John</i>	$\lambda P[P(\mathbf{John})]$	(the set of all of John's properties)
<i>a fool</i>	$\lambda P \exists x[\mathbf{fool}(x) \ \& \ P(x)]$	
<i>every man</i>	$\lambda P \forall x[\mathbf{man}(x) \rightarrow P(x)]$	

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

<i>John</i>	"referential use":	John	type e
<i>a fool</i>	"predicative use":	fool	type $e \rightarrow t$
<i>every man</i>	"quantifier use":	(above)	type $(e \rightarrow t) \rightarrow t$

Resolution: All NP's have meaning of type $(e \rightarrow t) \rightarrow t$; some also have meanings of types
 e and/or $e \rightarrow t$. General principles for predicting (Partee 1987). Predicates may semantically
 take arguments of type e , $e \rightarrow t$, or $(e \rightarrow t) \rightarrow t$, among others. (More on type-shifting in Lecture
 4, and more on type multiplicity of adjectives in Lecture 3.)

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 $S \rightarrow NP VP$ rule,
 for instance, will have two different translations. The VP, we have assumed, is always of type

$e \rightarrow t$. If the NP is of type e , the translation will be $VP'(NP')$, whereas if the NP is of type $(e \rightarrow t) \rightarrow t$, the translation will be $NP'(VP')$, as noted above in Section 1.2.2.2. [See Homework problem #3.]

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 very much 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_e , we need to add the iota-operator to IL; we will also use it in the treatment of genitives in Lecture 3.

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

Semantics: $5u[\varphi]5^{M,w,g} = d$ iff there is one and only one $d \in D$ such that $5\varphi5^{M,w,g[d/u]} = 1$.
 $5u[\varphi]5^{M,w,g}$ is undefined otherwise.

So $\iota x(\mathbf{king}(x))$ 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 $(e \rightarrow t) \rightarrow e$; 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, $\mathbf{TR}(a_{pred})$).

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

(ii) predicate-forming DETs.

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

$$\begin{aligned} \mathbf{TR}(a_{pred} \text{ man}) &= \mathbf{man} \\ \mathbf{TR}(the_{pred} \text{ man}) &= \lambda x[\mathbf{man}(x) \ \& \ \forall y[\mathbf{man}(y) \rightarrow y=x]] \end{aligned}$$

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

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

(iii) generalized quantifier-forming DETs.

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

$$\begin{aligned} \mathbf{TR}(a_{GQ} \text{ man}) &= \lambda P \exists x[\mathbf{man}(x) \ \& \ P(x)] \\ \mathbf{TR}(every_{GQ} \text{ man}) &= \lambda P \forall x[\mathbf{man}(x) \rightarrow P(x)] \\ \mathbf{TR}(the_{GQ} \text{ man}) &= \lambda P \exists x[\mathbf{man}(x) \ \& \ \forall y[\mathbf{man}(y) \rightarrow y=x]] \ \& \ P(x) \end{aligned}$$

The copula *be*:

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

Results (see also Section 2., and Homework problems 2, 4, 5):

$$\begin{aligned} \mathbf{TR}(is \text{ green}) &= \mathbf{green} \\ \mathbf{TR}(is \ a_{pred} \ \text{man}) &= \mathbf{man} \\ \mathbf{TR}(is \ the_{pred} \ \text{king}) &= \lambda x[\mathbf{king}(x) \ \& \ \forall y[\mathbf{king}(y) \rightarrow y=x]] \end{aligned}$$

2. Examples

(1) *is happy*

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

$$\mathbf{TR}(happy) = \mathbf{happy}$$

$$\begin{aligned} \mathbf{TR}(is\ happy) &= \lambda P \lambda x [P(x)] (\mathbf{happy}) \\ &= \lambda x [\mathbf{happy}(x)] = \mathbf{happy} \end{aligned}$$

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

$$\mathbf{TR}([\text{NP } the\ violinist]) = \mathbf{ix}[\mathbf{violinist}(x)] \quad \text{type: } e$$

$$\mathbf{TR}([\text{VP } is\ happy]) = \mathbf{happy} \quad \text{type: } e \rightarrow t$$

$$\begin{aligned} \mathbf{TR}([\text{S } the\ violinist\ is\ happy]) &= \mathbf{happy}(\mathbf{ix}[\mathbf{violinist}(x)]) \quad \text{type: } t \\ (\text{VP meaning applies to NP meaning}) \end{aligned}$$

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

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

$$\mathbf{TR}(is\ happy) = \mathbf{happy} \quad \text{type } e \rightarrow t$$

$$\begin{aligned} \mathbf{TR}(every\ violinist\ is\ happy) &= \lambda P \forall x [\mathbf{violinist}(x) \rightarrow P(x)] (\mathbf{happy}) \\ &= \forall x [\mathbf{violinist}(x) \rightarrow \mathbf{happy}(x)] \end{aligned}$$

(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 $(e \rightarrow t) \rightarrow (e \rightarrow t)$.

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

$$\mathbf{TR}(skillful\ violinist) = \mathbf{skillful}(\mathbf{violinist})$$

$$\mathbf{TR}(is\ a\ skillful\ violinist) = \mathbf{TR}(a\ skillful\ violinist) = \mathbf{TR}(skillful\ violinist)$$

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

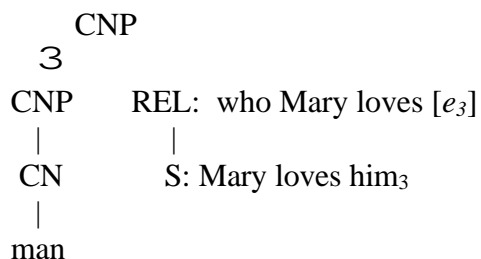
3. Rules of Relative clause formation, Quantifying In, Phrasal Negation.

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:



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

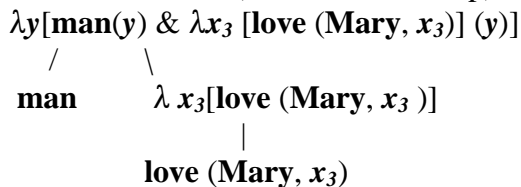
The relative clause itself is a predicate formed by λ -abstraction on the variable corresponding to the WH-word. (Partee 1976 suggests a general principle that all “unbounded movement rules” are interpreted as involving variable-binding; and λ -abstraction can be taken as the most basic variable-binding 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 $\lambda x_i[\phi']$.

Semantic derivation corresponding to the syntactic derivation above; compositional translation into IL: (read bottom-to-top)



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

3.2. Quantifying In.

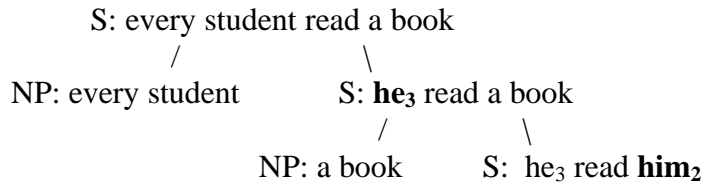
This is (an informal statement of) Montague’s Quantifying In rule; it is similar to the Quantifier-Lowering rule of Generative Semantics and Quantifier Raising (QR) of May (1977); various alternative treatments of quantifier scope ambiguity exist, including Cooper-storage (Cooper 1975) and Herman Hendriks’s 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: $\text{NP}'(\lambda x_i [S'])$ (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*.

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

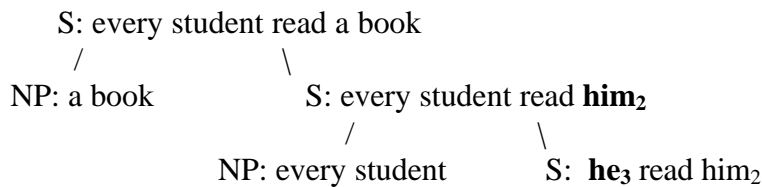


Compositional Translation: $(\text{every student})'(\lambda x_3 [(a \text{ book})'(\lambda x_2 [\text{read}(x_3, x_2)])])$

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 $\forall x(\text{student}(x) \rightarrow \exists y(\text{book}(y) \ \& \ \text{read}(x,y)))$.

Syntactic derivation (ii)



Compositional Translation: $(a \text{ book})'(\lambda x_2[(\text{every student})'(\lambda x_3 [\text{read}(x_3, x_2)])])$ [See **Homework problem 1.**]

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 $\exists y(\text{book}(y) \ \& \ \forall x(\text{student}(x) \rightarrow \text{read}(x,y)))$.

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’chuk) Underlying Structure (Generative Semantics), GPSG, HPSG, and various contemporary versions of Categorical Grammar are attempts to represent all the necessary syntactic information directly in a single “level” of syntax.

3.3 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 $S \rightarrow \text{NEG } S$. 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 \neg . Then the interpretation of NEG S, by function-argument application, will just be $\neg S'$.

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: $VP \rightarrow \text{NEG}_{VP} VP$

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

$$\begin{aligned} \mathbf{TR}(VP2) &= \mathbf{TR}(\text{NEG}_{VP})(\mathbf{TR}(VP1)) \\ &= \lambda P[\lambda x[\neg P(x)]](\mathbf{TR}(VP1)) \\ &= \lambda x[\neg VP1'(x)] \end{aligned}$$

Similar rules for negation of other phrasal categories can be derived in a uniform way. 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:

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

REFERENCES.

- 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.
- 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 (1987) "Noun phrase interpretation and type-shifting principles", in J. Groenendijk, D. de Jongh, and M. Stokhof, eds., *Studies in Discourse Representation Theory and the Theory of Generalized Quantifiers*, GRASS 8, Foris, Dordrecht, 115-143.
- Partee, Barbara and Mats Rooth, "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 (1983) 361-383.

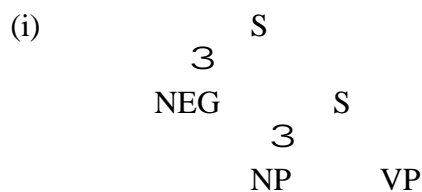
HOMEWORK #2.

(Choose two or three.)

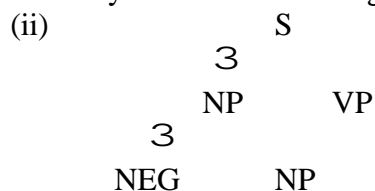
- Fill in the missing steps in the derivation of reduced forms of translations of the example *every student reads a book*, on derivation (ii) in 3.2. Use Montague's generalized quantifier interpretations of *every student* and *a book*, as given in Section 6 of Lecture 1, repeated under “generalized quantifier-forming DETS” in Section 1.3. of this lecture. This is an exercise in compositional interpretation and lambda-conversion.
- Fill in all the steps to show why $\mathbf{TR}(\text{is } a_{pred} \text{ man}) = \mathbf{man}$.

3. Write out the semantic derivation of *John walks* two ways, once using Montague's generalized quantifier interpretation of *John*, once using the type-e interpretation of *John*.
4. Write the translation for *every*, by abstracting on the CNP in the given translation of *every man*. Hint: the translation of *every*, like that of every DET, should begin with λQ , where Q is a variable of type $e \rightarrow t$, the type of the CNP with which the DET "wants to" combine.
5. Write the translations of *the* in each of its three types.
6. a. Figure out the translation for *no man* in two types: generalized quantifier type, and predicative type; the latter should be the same as the interpretation of *not a man*, which is not an entirely natural use of English *no man*, but is natural for Dutch *geen man*.
 b. Abstract on **man** in your answer to 6a to figure out the translations for *no* in two types.
7. Work out the translations, using lambda-conversion for simplification of results, of the following. Always apply lambda-conversion as soon as it is applicable, so that the formulas do not become more complex than necessary.
 - a. *Every violinist who loves Prokofiev is happy.*
 - b. *Not every violinist is unhappy.*

Note: Work this out two ways, which should come out equivalent. First pretend that the *not* is sentential negation, although according to the rules of English syntax, this is not a possible position for a sentential *not*. So the first syntactic structure should begin as follows:



Then figure out what the type and translation should be for a *not* which can apply to NPs of type $(e \rightarrow t) \rightarrow t$, and work out the translation for the sentence under an NP-negation analysis, where the syntactic structure begins as follows:



There is a third possibility, which is to apply *not* to *every*; if you have figured out how to do (i) and (ii), you'll be able to figure out how to do the third; it's just more lambdas. Don't do it unless you really want to. What might be more interesting would be to work on linguistic arguments to try to decide how many of the three are real possibilities for English.