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 ............................3
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, Conjunction, Anaphora. See Section 3 ..................................................4
1.2.4. Type multiplicity and type shifting ................................5
1.3. Lexicon ..........................................................5
2. Examples ..........................................................7
3. Relative clauses, Quantifying In, Conjunction, Anaphora. ........8
3.1. (Restrictive) Relative clause formation. ...........................8
3.2. Quantifying In. ......................................................8
3.3. Conjunction. .......................................................10
3.4. Bound variable anaphora vs. pragmatic anaphora – preview (New 2009) ..................................................11

Seminar “Homework” ..................................................12
REFERENCES .........................................................12

No specific assigned readings on this topic; but see References at the end, and see the useful
“Links to Readings 2009” document that’s linked from the course website.
(http://drive.iit.umn.edu/partee/Semantics_Reading%20Handouts%202009.doc) In the three weeks before our next lecture, March 27, finish Homework #1 and start reading about
Pragmatics – the handout for Lecture 4 will be online by tonight, I hope, and relevant readings will be suggested there and available for download. There are also some reading suggestions for these 3 weeks in the “Links to Readings 2009”.

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 rest of the course. 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. In section 3 we also raise a few preliminary issues concerning pragmatics; we
turn more seriously to pragmatics starting in Lecture 4.

1.1. Syntactic categories and their semantic types.

<table>
<thead>
<tr>
<th>Syntactic category</th>
<th>Semantic type (extensionalized)</th>
<th>Expressions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProperN</td>
<td>e</td>
<td>names (John)</td>
</tr>
<tr>
<td>S</td>
<td>t</td>
<td>sentences</td>
</tr>
<tr>
<td>CN(P)</td>
<td>e → t</td>
<td>“e-type” or “referential” NPs (John, the king, he)</td>
</tr>
<tr>
<td>NP</td>
<td>(i) e → t</td>
<td>noun phrases as generalized quantifiers (every man, the king, a man, John)</td>
</tr>
<tr>
<td>ADJ(P)</td>
<td>(i) e → t</td>
<td>NPs as predicates (a man, the king)</td>
</tr>
<tr>
<td>REL</td>
<td>e → t</td>
<td>predicative adjectives (carnivorous, happy)</td>
</tr>
<tr>
<td>VP, IV</td>
<td>e → t</td>
<td>adjectives as predicate modifiers (skillful)</td>
</tr>
<tr>
<td>TV(P)</td>
<td>type(NP) → type(VP)</td>
<td>relative clauses (who(m) Mary loves)</td>
</tr>
<tr>
<td>it</td>
<td>(e → t) → (e → t)</td>
<td>verb phrases, intransitive verbs (loves Mary, is tall, walks)</td>
</tr>
<tr>
<td>DET</td>
<td>type(CN) → type(NP)</td>
<td>transitive verb (phrase) (loves)</td>
</tr>
<tr>
<td>is</td>
<td>(e → t) → (e → t)</td>
<td>is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>a, some, the, every, no</td>
</tr>
</tbody>
</table>

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 t of category C formed from expressions “of” category A and 5 of category B,
determine TR(t) 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.
Official Syntactic Rule: If $\alpha$ is an expression of category DET and $\beta$ is an expression of category CNP, then $F_0(\alpha', \beta S)$ is an expression of category NP, where $F_0(\alpha', \beta S) = ^{S}\alpha^{0}$.  

Official Semantic Rule: If $TR(\alpha') = ^{a}\alpha^{'}$ and $TR(\beta S) = ^{b}\beta^{'}S$; then $TR(F_0(\alpha', \beta S)) = ^{a}( ^{b}\beta^{'}S)$.

Abbreviated Syntactic Rule: $NP \rightarrow DET\ C\ NP$

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

1.2.1. Basic syntactic rules

Basic rules, phrasal:

$S \rightarrow NP\ VP$

$NP \rightarrow DET\ CNP$

$CNP \rightarrow ADJP\ CNP$

$CNP \rightarrow CNP\ REL$

$VP \rightarrow TVP\ NP$

$VP \rightarrow is\ ADJP$

$VP \rightarrow is\ NP$

Basic rules, non-branching rules introducing lexical categories:

$NP \rightarrow ProperN$

$NP \rightarrow Pronoun$

$CNP \rightarrow CN$

$TVP \rightarrow TV$

$ADJP \rightarrow ADJ$

$VP \rightarrow IV$

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, $TR(A)$ as a function of $TR(B)$ and $TR(C)$; and ultimately, $[\alpha]A$ as a function of $[\beta]B$ and $[\gamma]C$.) 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).

In the original they are written for direct model-theoretic interpretation.

1. Terminal Nodes (TN): If $\alpha$ is a terminal node, then $[[\alpha]]$ is specified in the lexicon.

2. Non-Branching Nodes (NN): If $\alpha$ is a non-branching node, and $\beta$ is its daughter node, then $[[\alpha]] = [[\beta]]$.

3. Functional Application (FA): If $\alpha$ is a branching node, $[[\beta\gamma]]$ is the set of $\gamma$'s daughters, and $[[\beta]]$ is a function whose domain contains $[[\gamma]]$, then $[[\alpha]] = [[\beta]]([[[\gamma]]])$.

4. Predicate Modification (PM): If $\alpha$ is a branching node, $[[\beta\gamma]]$ is the set of $\gamma$'s daughters, and $[[\beta]]$ and $[[\gamma]]$ are both in $D_{\leq 3}$, then $[[\alpha]] = \lambda x \in D_{\leq 3} [ [[\beta]] (x) = 1 \land [[\gamma]] (x) ]$.

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 $\alpha$ is a terminal node, then $TR(A)$ 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 $TR(A) = TR(B)$.

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 $a \rightarrow b$ 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 $a \rightarrow t$, and the syntactic category $A$ can also correspond to type $a \rightarrow t$, then $TR(A) = \lambda x ( TR(B' \& TR(C'(x)) )$ (i.e. $[[A]] = [[B]] \cap [C] )$.

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 $e \rightarrow t \rightarrow e$ and $VP$ is of type $e \rightarrow t$, 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 $e \rightarrow t$, then the translation of $S$ will be $VP'(NP')$ (e.g., John walks).

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, Conjunction, Anaphora. 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
First we simply list some lexical items of various syntactic categories; several were introduced in Fragment 1, and more will be introduced in later lectures.

Montague tradition: uniform treatment of NPs as generalized quantifiers, type (e → t) → t.

\[
\begin{align*}
\text{John} & : \lambda p([p, \text{John}]) \quad \text{(the set of all John’s properties)} \\
\text{he} & : \lambda p([p, x]) \quad \text{(the set of all of (g(x))’s properties)} \\
\text{a fool} & : \lambda P([\text{fool}(x) \& P(x)]) \\
\text{every man} & : \lambda P([\text{man}(x) \rightarrow P(x)])
\end{align*}
\]

Intuitive type multiplicity of NPs (and see Heim 1982, Kamp 1981):

\[
\begin{align*}
\text{John} & \quad \text{"referential use"} : \quad \text{John} \quad \text{type e} \\
\text{be,} & \quad \text{"e-type variable"} : \quad x_0 \quad \text{type e} \\
\text{a fool} & \quad \text{"predicative use"} : \quad \text{fool} \quad \text{type e} \rightarrow t \\
\text{every man} & \quad \text{"quantifier use"} : \quad \text{(above)} \quad \text{type (e→t)→t}
\end{align*}
\]

Resolution: All NP’s have meaning of type (e→t)→t; some also have meanings of types e and/or e→t. General principles for predicting (Partee 1986). Predicates may semantically take arguments of type e, e→t, or (e→t)→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 → NP VP rule, for instance, will have two different translations. The VP, we have assumed, is always of type e→t. If the NP is of type e, the translation will be VP*(NP*), whereas if the NP is of type (e→t)→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 categories DET and Pronoun, 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.
(ii) predicate-forming DETs.
DETs as functors forming predicate nominals are of type \(e \rightarrow t\) → \((e \rightarrow t)\).
\[
\begin{align*}
\text{TR}(\text{a\_good man}) & = \text{man} \\
\text{TR}(\text{the\_good man}) & = \lambda x[\text{man}(x) \land \forall y[\text{man}(y) \rightarrow y=x]]
\end{align*}
\]
We illustrate the translation of the DET itself with the translation of \(\text{a\_good\_man}\).
\[
\text{TR}(\text{a\_good\_man}) = \lambda P[P]
\]
(iii) generalized quantifier-forming DETs.
DETs as functors forming generalized quantifiers are of type \((e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow t)\).
\[
\begin{align*}
\text{TR}(\text{a\_good\_man}) & = \lambda P[\text{man}(x) \land P(x)] \\
\text{TR}(\text{the\_good\_man}) & = \lambda P[\text{man}(x) \rightarrow P(x)] (\text{see Homework 1, problem 4})
\end{align*}
\]
\[
\text{TR}(\text{every\_good\_man}) = \lambda P[\text{man}(x) \land \forall y[\text{man}(y) \rightarrow y=x] & \land P(x)]
\]

The copula be:
\[
\text{TR}(\text{is}) = \lambda P\!x[P(x)] ("Predicate")
\]

Results (see also Section 5, and “Seminar Homework”, problem 2):
\[
\begin{align*}
\text{TR}(\text{is\_green}) & = \text{green} \\
\text{TR}(\text{is\_a\_good\_man}) & = \text{man} \\
\text{TR}(\text{is\_the\_good\_man}) & = \lambda x[\text{king}(x) \land \forall y[\text{king}(y) \rightarrow y=x]]
\end{align*}
\]

2. Examples

(1) is happy
\[
\begin{align*}
\text{TR}(\text{is\_happy}) & = \lambda P\!x[P(x)] \\
\text{TR}(\text{happy\_is}) & = \text{happy} \\
\text{TR}(\text{is\_happy}) & = \lambda P\!x[P(x)] (\text{happy}) \\
& = \lambda x[\text{happy}(x)] = \text{happy}
\end{align*}
\]

(2) The violinist is happy
(with e-type interpretation of subject)
\[
\begin{align*}
\text{TR}[\text{the\_violinist}] & = \lambda x[\text{violinist}(x)] \quad \text{type: e} \\
\text{TR}[\text{the\_violinist\_is\_happy}] & = \text{happy}(x[\text{violinist}(x)]) \quad \text{type: e} \rightarrow t \\
\text{TR}[\text{the\_violinist\_is\_happy\_is\_happy}] & = \text{happy}(x[\text{violinist}(x)]) \quad \text{(VP meaning applies to NP meaning)}
\end{align*}
\]

(3) Every violinist is happy
(with GQ-type subject)
\[
\begin{align*}
\text{TR}[\text{every\_violinist}] & = \lambda P\!x[\text{violinist}(x) \rightarrow P(x)] \quad \text{type (e \rightarrow t) \rightarrow t} \\
\text{TR}[\text{every\_violinist\_is\_happy}] & = \lambda P\!x[\text{violinist}(x) \rightarrow P(x)] (\text{happy}) \\
& = \forall x[\text{violinist}(x) \rightarrow \text{happy}(x)]
\end{align*}
\]

(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 \((e \rightarrow t) \rightarrow (e \rightarrow t)\).)
\[
\begin{align*}
\text{TR}[\text{every\_surgeon}] & = \lambda P\!x[\text{surgeon}(x) \rightarrow P(x)] \\
\text{TR}[\text{skillful\_surgeon}] & = \lambda x[\text{skillful}(x)] \\
\text{TR}[\text{every\_surgeon\_is\_a\_skillful\_surgeon}] & = \lambda P\!x[\text{surgeon}(x) \rightarrow \text{skillful}(x)] (\text{happy}) \\
& = \forall x[\text{surgeon}(x) \rightarrow \text{skillful}(x)] (\text{happy})
\end{align*}
\]

3. Relative clauses, Quantifying In, Conjunction, Anaphora.

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):

\[
\begin{align*}
\text{CNP} & \\
\text{REL:} & \text{who Mary loves [x\_2]} \\
& \text{CN S: Mary loves him, man}
\end{align*}
\]

The types for CN, CNP, and REL are all \(e \rightarrow t\); so the principle for combining CNP and REL
gives: \(\lambda y[\text{CNP}(y) \land \text{REL}(y)]\) (Predicate conjunction)

The relative clause itself is a predicate formed by \(\lambda\)-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 \(\lambda\)-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 \(q\) is an S and \(q\) contains an indexed pronoun \(he, / him, in\)
relativizable position, then the result of adjoining who(m) to S and leaving a trace \(e_i\) in place
of \(he, / him, is a REL.

Rel Clause Rule, semantics: If \(q\) translates as \(q'\), then REL translates as \(\lambda x[q']\).

Semantic derivation corresponding to the syntactic derivation above; compositional
translation into IL: (read bottom-to-top) (and see Homework 1, Problem 5)

\[
\begin{align*}
\lambda y[\text{man}(y)] & \land \lambda x[\text{love} (\text{Mary, } x)] (\text{y}) \\
\text{man} & \lambda x[\text{love} (\text{Mary, } x)] \\
& \text{love (Mary, } x)
\end{align*}
\]

By \(\lambda\)-conversion, the top line is equivalent to: \(\lambda y[\text{man}(y) \land \text{love} (\text{Mary, } y)]\)

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” 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: \[\lambda x \langle S' \rangle \text{ (The set of properties denoted by the NP includes the property denoted by the }\lambda\text{-expression derived from the sentence.)}\]

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.)

\[
\begin{align*}
S & : \text{every student read a book} \\
NP & : \text{every student read him}_2 \\
NP & : \text{a book} \\
\text{Compositional Translation:} & (\text{every student})'\lambda x_1 \{ \langle x_1 \rangle \lambda x_2 [\text{read}(x_1, x_2)] \}
\end{align*}
\]

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 \(3y(\text{book}(y) \& \text{read}(x,y)))

**Syntactic derivation (ii)**

\[
\begin{align*}
S & : \text{every student read a book} \\
NP & : \text{every student read him}_2 \\
NP & : \text{a book} \\
\text{Compositional Translation:} & (\text{a book})'\lambda x_2 [\text{every student}]'(\lambda x_1 [\text{read}(x_1, x_2)])
\end{align*}
\]

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 \(3y(\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 is 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**:

<table>
<thead>
<tr>
<th>Corresponding semantic rules:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S \rightarrow S_1 &amp; S_2)</td>
</tr>
<tr>
<td>(S \rightarrow S &amp; S_2)</td>
</tr>
<tr>
<td>(VP \rightarrow VP &amp; VP)</td>
</tr>
<tr>
<td>(VP \rightarrow VP &amp; VP)</td>
</tr>
<tr>
<td>(NP \rightarrow NP &amp; NP)</td>
</tr>
<tr>
<td>(NP \rightarrow NP &amp; NP)</td>
</tr>
</tbody>
</table>

The NP conjunction and disjunction rules presuppose that the NPs are interpreted as generalized quantifiers, type \(\langle e,t,t\rangle\). \(P\) is a variable of type \(\langle e,t\rangle\). (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 partitives, if we omit the second ‘was’, but it’s equivalent to VP conjunction). The rules do correctly “predict” which conjunction has wider scope.

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. Bound variable anaphora vs. pragmatic anaphora – preview (New 2009)

The Relative Clause rule and the Quantifying In rule are the only rules we’ve seen so far that mention the pronouns $he$, translated as $x$. Those rules create variable binding.

(i) who, which in relative clauses: expressions of lambda abstraction. (See Homework 1, Problem 5)

(ii) Relative clauses may contain bound variable pronouns:

$a$ man [who loves a woman who loves him]: the bracketed relative clause is derived from a sentence such as $he_1$ loves a woman who loves him or (for any other choice of variable). The relative clause formation rule “abstracts on” $x$, adding who and deleting the first occurrence of $he_2$, and replaces the other occurrence of $he_2$ by a pronoun of suitable gender. (Puzzle: at what stage of the derivation is the gender determinable?)

(iii) The Quantifying In rule causes an underlying $he$ to be bound by a lambda operator. See Syntactic Derivations (i) and (ii) above. Exercise 1 of today’s “Seminar homework” involves filling in the missing steps in Derivation (ii).

(iv) The Quantifying In rule can also create sentences that contain bound variable pronouns, for instance if we combine every professor with the (open) sentence $He$ knows a student who admires him; the result will be Every professor knows a student who admires him. (See “Seminar homework”, Exercises 5a and 6.)

Bound variable anaphora is a clear instance of semantic anaphora: it is a sentence-internal phenomenon in which the antecedent must syntactically c-command the pronoun and semantically bind it. (Actually, it is not literally the antecedent that semantically binds the pronoun, but a lambda-operator associated with the antecedent by the compositional rules.) But some kinds of anaphora seem to be more pragmatic in nature.

(i) “Free variable pronouns”: Some pronouns are like free variables with values assigned by context.

What should we say about “sentences” like $He$ knows a student who admires him? One option is to call any “sentence” containing pseudo-pronouns like he unngrammatical. Another option is to let such sentences be the sources for sentences with non-bound-variable pronouns like He knows a student who admires him or She knows a student who admires her. On this option, we assume that the discourse context must include an assignment function $g$ indicating the “intended referent” of he or she, and we also assume that gender-marking reflects a presupposition about the gender of this intended referent. We will assume this latter hypothesis, at least for now. The truth conditions of such a sentence may very well depend on the choice of assignment function. Later we’ll discuss the difference between “demonstrative” uses of pronouns and what is sometimes called “discourse anaphora,” both of which are different from “bound variable anaphora” (Partee 1978).

(ii) Non-restrictive relative clauses. Potts (2002a, 2002b, 2005) develops a semantic theory of non-restrictive relative clauses like the one in (7) below, as well as a variety of other appositive and “supplementary” modifiers. We will study Potte’s work later, since it is an important contribution to formal semantics and pragmatics.

(7) Chris Potts, who taught twice in the St. Petersburg summer school, is a semanticist.

For now, let us just note that this who is very different from the who in a restrictive relative clause; this one is much more like a pronoun, co-referential with Chris Potts. How does this work? We’ll come back to it later this semester.

Seminar “Homework”

Some exercises that were homework exercises in other years, but we can do (some of) them together in seminar this year.

1. 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. (Lecture 3). Use Montague’s generalized quantifier interpretations of every student and a book, as given in Section 3.2 of Lecture 2, repeated under “generalized quantifier-forming DEPS” in Section 3.3 of Lecture 3. This is an exercise in compositional interpretation and lambda- conversion.

2. Fill in all the steps to show why $TR(is_a_{man}) = man$. (Lecture 3)

3. Write the translations of the in each of its three types. (Lecture 3)

4. Translate the following two sentences into first-order logic. You don’t have to do this compositionally – just write down the formulas.

a. Every candidate voted for every candidate.

b. Every candidate voted for himself.

c. Every professor knows a student who admires him. (where the antecedent of him is every professor)

b. Every professor knows a student who admires himself. (where the antecedent of himself is a student)

6. Making use of the rules in the fragment in Lecture 3, work out the translation of the sentence in 5a compositionally. The problem is a combination of Exercise 5 from Homework 1 and the way bound variable anaphora is introduced in the Quantifying In rule.

REFERENCES.

**Note: See website (Links to Readings 2009) for full references and downloadable links for the most accessible of these, marked ** below. **


***Potts, Christopher. 2002a. The lexical semantics of parentheticals and appositive-which.

***Potts, Christopher. 2002a. The lexical semantics of parentheticals and appositive-which.

***Potts, Christopher. 2002b. The syntax and semantics of A-sentential.

***Potts, Christopher. 2002b. The syntax and semantics of A-sentential.


**Partee, Barbara H. (1990) "The lexical semantics of parentheticals and appositive-which."

**Partee, Barbara H. (1990) "The lexical semantics of parentheticals and appositive-which."

**Partee, Barbara H. (1990) "The lexical semantics of parentheticals and appositive-which."